
GaussianFusion: Gaussian-Based Multi-Sensor Fusion for End-to-End Autonomous Driving

Shuai Liu, Quanmin Liang, Zefeng Li, Boyang Li*, Kai Huang

School of Computer Science and Engineering, Sun Yat-sen University
 {liush376@mail2, liby83@mail, huangk36@mail}.sysu.edu.cn

Abstract

Multi-sensor fusion is crucial for improving the performance and robustness of end-to-end autonomous driving systems. Existing methods predominantly adopt either attention-based flatten fusion or bird’s eye view fusion through geometric transformations. However, these approaches often suffer from limited interpretability or dense computational overhead. In this paper, we introduce **GaussianFusion**, a **Gaussian**-based multi-sensor **fusion** framework for end-to-end autonomous driving. Our method employs explicit and compact Gaussian representations as intermediate carriers to aggregate information from diverse sensors. Specifically, we initialize a set of 2D Gaussians uniformly across the driving scene, where each Gaussian is parameterized by physical attributes and equipped with explicit and implicit features. These Gaussians are progressively refined by integrating multi-modal features. The explicit features capture rich semantic and spatial information about the traffic scene, while the implicit features provide complementary cues beneficial for trajectory planning. To fully exploit rich spatial and semantic information in Gaussians, we design a cascade planning head that iteratively refines trajectory predictions through interactions with Gaussians. Extensive experiments on the NAVSIM and Bench2Drive benchmarks demonstrate the effectiveness and robustness of the proposed GaussianFusion framework. The source code will be released at <https://github.com/Say2L/GaussianFusion>.

1 Introduction

End-to-end (E2E) autonomous driving [16, 25, 4, 7, 32] has attracted growing attention for its potential to streamline the traditional modular pipeline by directly mapping sensor inputs to driving actions through deep learning. This paradigm reduces system complexity and enables joint optimization across tasks. However, relying on a single sensor often limits the system’s ability to handle diverse and challenging driving scenarios. To address this limitation, multi-sensor fusion has become essential, as it allows the model to leverage complementary information from different sensors such as cameras, LiDARs, and radars. This integration enhances perception reliability and provides richer input for learning robust driving policies.

Existing multi-modal fusion strategies in E2E autonomous driving can be broadly divided into two categories: flatten fusion and bird’s eye view (BEV) fusion. Flatten fusion methods [7, 36, 1, 5] typically compress sensor features, such as those from images and LiDAR point clouds, into a shared latent space, where feature interaction is performed using attention mechanisms, as shown in Figure 1 (a). These approaches are appealing due to their flexibility and efficiency, often requiring minimal geometric calibration. However, the lack of explicit spatial grounding in 3D space limits their interpretability and makes them less effective in scenarios requiring precise spatial reasoning. In contrast, BEV fusion methods [22, 35, 30, 45] project multi-modal features into a common BEV

*Corresponding author.

coordinate, leveraging geometric priors to align data from different sensors, as shown in Figure 1 (b). This facilitates structured spatial understanding and improves performance in downstream perception tasks such as 3D object detection and map construction. Nevertheless, BEV fusion incurs significant computational and memory overhead due to the dense nature of the BEV representation, especially when high-resolution inputs or fine-grained features are involved. As a result, there remains an ongoing challenge to develop fusion frameworks that can balance spatial awareness, efficiency, and scalability in complex driving environments.

3D Gaussians have recently gained attention in camera-based 3D scene representation and reconstruction [26, 18, 17] due to their physical interpretability, compactness, and inherent sparsity. These properties make them promising candidates for multi-sensor fusion in autonomous driving, where efficiency and structured spatial understanding are critical. However, applying Gaussian representations in this context introduces several challenges. First, it is difficult to effectively supervise Gaussian parameters due to the lack of fine-grained 3D scene annotations in existing E2E driving datasets [11, 23, 49]. Second, existing approaches [18, 17] mainly focus on 3D scene representation, leaving their suitability for the motion planning task under-explored. Third, efficiently leveraging Gaussian representation for accurate trajectory generation requires careful architecture design. Addressing these issues is key to enabling Gaussian-based representations in E2E autonomous driving frameworks.

With the aforementioned innovations and considerations, we propose GaussianFusion, a Gaussian-based multi-sensor fusion framework for E2E autonomous driving. Our approach leverages 2D Gaussians to represent the traffic scene, offering improved efficiency over 3D Gaussians. Notably, 2D Gaussians require supervision only from BEV semantic maps, which are widely available in E2E datasets [11, 23, 49]. To tailor the fusion process to the motion planning task, we design a dual-branch fusion pipeline. The first branch captures local features from multi-sensor inputs for each Gaussian and is primarily responsible for traffic scene reconstruction. The second branch aggregates global planning cues from the same inputs and is dedicated exclusively to motion planning. Moreover, to fully exploit the representational capacity of Gaussians, we introduce a cascade planning module that refines the anchor trajectories by querying the Gaussian representations in a cascaded manner.

We evaluate GaussianFusion on the planning-oriented NAVSIM dataset [11]. Utilizing the V2-99 backbone [27], our approach achieves 92.0 PDMS [11], significantly surpassing current state-of-the-art methods. To further evaluate the generalization and robustness of our framework, we conduct experiments on the closed-loop benchmark Bench2Drive [23], where the results consistently demonstrate the effectiveness of GaussianFusion. The key contributions of this work are summarized as follows:

- We first introduce Gaussian representations into the domain of multi-sensor fusion for E2E autonomous driving, and we propose a dual-branch fusion pipeline tailored to the planning-centric task.
- We design a cascade planning head specifically adapted to the Gaussian representation, which iteratively refines trajectories through hierarchical Gaussian queries.
- Extensive evaluations on both open-loop (NAVSIM) and closed-loop (Bench2Drive) benchmarks demonstrate the superior performance and robustness of GaussianFusion.

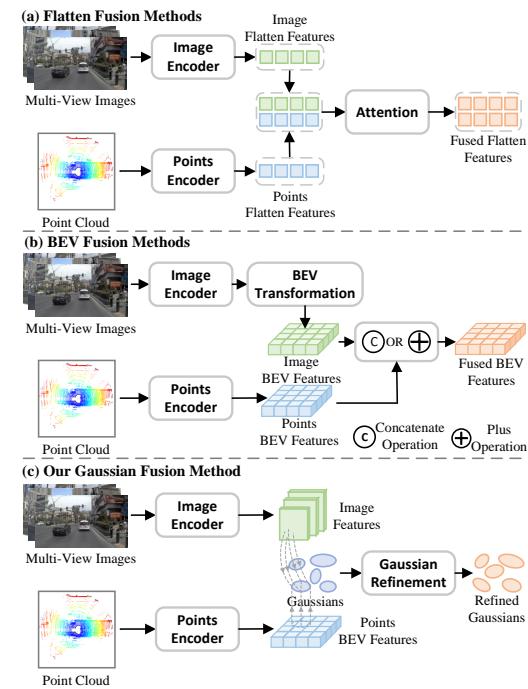


Figure 1: Pipelines of different multi-sensor fusion methods.

2 Related Work

End-to-End Autonomous Driving. E2E autonomous driving approaches can be broadly categorized into single-task and multi-task methods. Single-task methods [3, 8, 9] primarily focus on the planning task. An early work [3] employs a convolutional neural network (CNN) to directly map raw pixel data from a front-facing camera to steering commands. Building on this idea, [8] proposes conditional imitation learning, which extends traditional imitation learning by integrating high-level navigational commands to better capture driver intent. CILRS [9] further explores behavior cloning within the E2E framework, aiming to improve performance through more structured supervision.

Multi-task methods extend the E2E autonomous driving paradigm by incorporating auxiliary tasks such as 3D object detection, mapping, and motion forecasting. Compared to single-task approaches, the multi-task paradigm enables the learning of scene representations with improved generalization and interpretability [19]. This has led to growing interest from the research community. UniAD [16] introduces a planning-oriented framework that sequentially integrates perception, prediction, and planning modules. VAD [25] and its improved version VADv2 [4] adopt a vectorized scene representation to enhance computational efficiency. More recently, diffusion-based strategies [6] have gained attention in E2E autonomous driving. GoalFlow [43] introduces goal-conditioned denoising, leveraging goal points to guide the generation of multi-modal trajectories. DiffusionDrive [32] proposes a truncated diffusion policy to accelerate the denoising process, improving inference speed without sacrificing trajectory quality.

Sensor Fusion for Autonomous Driving. Existing sensor fusion approaches can be generally classified into flatten fusion and BEV fusion methods. Flatten fusion methods [36, 7, 20, 1, 5] typically compress image and LiDAR features into flattened representations, which are then fused using attention mechanisms [39, 40]. For example, InterFuser [36] concatenates flattened features from multiple sensors and applies self-attention [39] to enable information exchange. TransFuser [7, 20] follows the same paradigm but incorporates multi-scale features to enhance fusion. To mitigate the influence of irrelevant image regions, TransFusion [1] introduces a 2D circular Gaussian mask that modulates the cross-attention weights. AutoAlign [5] further improves upon this by introducing a cross-attention-based feature alignment module that adaptively aggregates pixel-level image features for each voxel. While these methods eliminate the need for explicit calibration, they often suffer from limited interpretability and lack the capability for explicit 3D scene modeling.

BEV fusion methods [31, 22, 35, 30, 45] typically fuse multi-modal features within the BEV plane based on geometric projection relationships. MILE [15] lifts image features into 3D using predicted depth distributions, then aggregates voxel features into BEV via grid-based sum-pooling. ContFuse [31] employs 3D point clouds as intermediaries to project pixel-level image features onto the BEV plane for fusion. ThinkTwice [22] and DriveAdapter [21] adopt the LSS framework [35] to transform image features into BEV space, which are then concatenated with LiDAR-derived BEV features. BEVFormer [30] and FusionAD [45] further extend the BEV representation by incorporating a height dimension into the BEV queries, enabling spatially-aware feature extraction from image inputs. This type of method addresses the interpretability and 3D modeling limitations inherent in flatten fusion methods. However, since operating on a dense BEV grid, these methods often incur significant computational and memory overhead.

In contrast to existing fusion paradigms, we propose GaussianFusion, which leverages Gaussian representations as intermediaries to extract and aggregate multi-modal features. Gaussians are well-regarded for their strong scene reconstruction [26] and explicit representation properties. Moreover, Gaussian representations are sparsely distributed in space, in stark contrast to the dense grids used in BEV fusion methods. As a result, our GaussianFusion effectively circumvents the limitations of both flatten fusion (lack of structure and interpretability) and BEV fusion (computational and memory inefficiency), offering a more efficient and interpretable alternative for multi-sensor integration.

Gaussian Representation. 3D Gaussian Splatting (3D-GS) [26] first introduced 3D Gaussians as a scene representation for radiance field rendering, achieving high visual fidelity and real-time performance. Deformable-GS [44] extends this technique to dynamic scene reconstruction and rendering. Other works [38, 46] adapt Gaussian splatting for 3D content creation. The GaussianFormer series [18, 17] utilizes 3D Gaussians to perform occupancy prediction.

The approach most closely related to ours is GaussianAD [48], which models future scene evolution using Gaussian flows. While both methods leverage Gaussians for representing traffic scenes, our

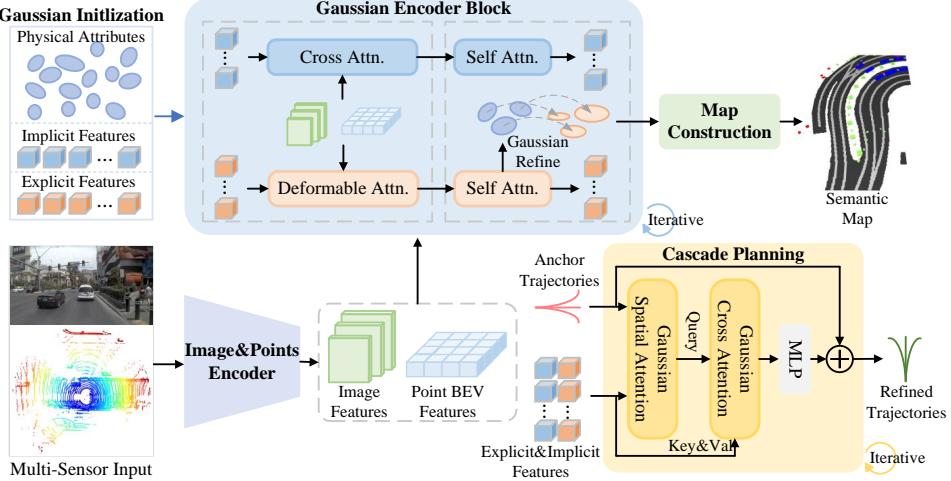


Figure 2: **The overall framework of our GaussianFusion.** Given raw multi-sensor data as input, GaussianFusion first extracts image and point features using a backbone network. It then initializes a set of Gaussians, which are iteratively refined through Gaussian encoder blocks. Finally, the refined Gaussians are used to construct the semantic map and to iteratively adjust the anchor trajectories.

GaussianFusion differs from GaussianAD in several key aspects: 1) GaussianFusion is designed for multi-sensor fusion in E2E autonomous driving, whereas GaussianAD focuses solely on vision-based systems. 2) Our method employs 2D Gaussians as scene representations, in contrast to the 3D Gaussians used in GaussianAD. This design choice enables supervision using only BEV-level annotations (e.g., semantic maps and agent states), eliminating the need for dense 3D occupancy labels and significantly improving computational efficiency. 3) Additionally, GaussianFusion incorporates a dual-branch feature fusion design tailored for the planning-centric task.

3 Method

The objective of E2E autonomous driving is to predict the future trajectory of the ego vehicle directly from raw sensor inputs. Formally, given multi-view images $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{3 \times H \times W} | i = 1, \dots, N\}$, LiDAR points $P \in \mathbb{R}^{n \times 3}$ and transformation matrices $\mathcal{T} = \{\mathbf{T}_i \in \mathbb{R}^{3 \times 3} | i = 1, \dots, N\}$, the goal is to predict the ego vehicle trajectory $\tau = \{(x_t, y_t)\}_{t=1}^T$, where N , (H, W) , n , (x_t, y_t) and T denotes the number of views, the image resolution, the number of points, the waypoint coordinate at time t and the planning horizon, respectively.

The overview of our GaussianFusion is illustrated in Figure 2. It can be divided into three phases as follows: (1) Gaussian initialization (Sec. 3.1); (2) Gaussian encoder: Gaussians from multi-sensor (Sec. 3.2); (3) Gaussian decoder: Gaussians to scene reasoning (Sec. 3.3).

3.1 Gaussian Initialization

Given that autonomous vehicles primarily operate on planar surfaces, 2D Gaussians suffice to model the traffic scene. Specifically, we randomly generate a set of 2D Gaussians $\mathcal{G} = \{\mathbf{G}_i | i = 1, \dots, P\}$, where P denotes the predefined number of Gaussians. Each Gaussian is characterized by both physical attributes and hidden features. The physical attributes include mean $\mathbf{m} \in \mathbb{R}^2$, scale $\mathbf{s} \in \mathbb{R}^2$, rotation $\mathbf{r} \in \mathbb{R}^2$, and semantic logits $\mathbf{c} \in \mathbb{R}^C$, where C is the number of semantic categories. The rotation \mathbf{r} is represented using its sine and cosine components. The hidden features consist of both explicit and implicit features, which are then refined through two separate branches. Further details regarding the explicit and implicit features are provided in Sec. 3.2.

3.2 Gaussian Encoder: Gaussians from Multi-Sensor Fusion

To learn meaningful Gaussian representations, we first utilize two independent backbones to extract multi-scale features from images and LiDAR points. These features are then used to iteratively

refine both the physical attributes and hidden features of Gaussians. Each iteration consists of a point cross-attention module, an image cross-attention module, a Gaussian self-attention module, and a refinement module. The hidden features of each Gaussian are divided into explicit and implicit components, each serving distinct roles. Explicit features are derived from local regions of multi-sensor inputs via explicit geometric transformations and are responsible for updating the physical properties of Gaussians. In contrast, implicit features interact with global multi-sensor features without relying on geometric transformations, and are used exclusively for trajectory planning.

Point Cross-Attention: Gaussians from Points. A point cross-attention (PCA) module is employed to extract information from point features. Specifically, for each Gaussian $\mathbf{G} = \{\mathbf{m}, \mathbf{s}, \mathbf{r}, \mathbf{c}, \mathbf{f}^{\text{exp}}, \mathbf{f}^{\text{imp}}\}$, we generate a set of query points $\mathcal{Q} = \{(x_i, y_i) \mid i = 1, \dots, n_q\}$, where (x_i, y_i) denotes the location of the i -th query point, and n_q is the total number of queries. Following [18], the query set includes both fixed and learnable points: the fixed queries are distributed around each Gaussian based on its covariance matrix, while the learnable queries are constrained within the interior of the Gaussian. Given multi-scale point feature maps $\mathcal{M}^p = \{\mathbf{M}_i^p \in \mathbb{R}^{d \times H_i^p \times W_i^p} \mid i = 1, \dots, n_s\}$, where (H_i^p, W_i^p) denotes the resolution of the i -th scale feature map and n_s is the number of scales, we apply a deformable attention layer [42] to aggregate information from these features and update the explicit features of Gaussians.

$$\mathbf{f}^{\text{exp}\dagger} = \sum_{i=1}^{n_q} \text{DeAttn}(\mathbf{f}^{\text{exp}}, \mathcal{Q}[i], \mathcal{M}^p), \quad (1)$$

where $\mathbf{f}^{\text{exp}\dagger}$ denotes the updated explicit features using multi-scale point features, $\mathcal{Q}[i]$ indicates the i -th point in \mathcal{Q} , and $\text{DeAttn}(\cdot)$ represents the deformable attention. For the implicit features, we use a vanilla cross-attention [40] to build interactions with the last scale point features $\mathbf{F}_{n_s}^p$:

$$\mathbf{f}^{\text{imp}\dagger} = \text{CrossAttn}(\mathbf{f}^{\text{imp}}, \mathbf{M}_{n_s}^p), \quad (2)$$

where $\mathbf{f}^{\text{imp}\dagger}$ denotes the updated implicit features, and CrossAttn refers to the cross-attention layer. For brevity, we omit the residual connection and feed-forward network (FFN) components in the formulation. The resulting updated Gaussian is represented as $\mathbf{G}^\dagger = \{\mathbf{m}, \mathbf{s}, \mathbf{r}, \mathbf{c}, \mathbf{f}^{\text{exp}\dagger}, \mathbf{f}^{\text{imp}\dagger}\}$. Note that, for clarity, we illustrate the process using a single Gaussian as an example.

Image Cross-Attention: Gaussians from Images. To incorporate visual information from multi-view images, we employ an Image Cross-Attention (ICA) module. Similar to the PCA module, ICA generates both fixed and learnable query points for each Gaussian. However, these queries additionally incorporate height information to enable projection into the image plane. Specifically, we first generate 2D query points $\mathcal{Q}_{2d} = \{(x_i, y_i) \mid i = 1, \dots, n_q\}$, identical to those used in the PCA module. For each 2D query location, we then uniformly sample n_p pillar points along the vertical axis. The bottom of each pillar is fixed at $z = z_{\min}$, while the top is parameterized by a learnable variable $z_g \in [z_{\min}, z_{\max}]$, where z_{\min} and z_{\max} define the vertical bounds of the traffic scene. This results in a set of 3D query points $\mathcal{Q}_{3d} = \{(x_i, y_i, z_i) \mid i = 1, \dots, n_q \times n_p\}$. Given the multi-scale image feature maps $\mathcal{M}^I = \{\mathbf{M}_i^I \in \mathbb{R}^{d \times N \times H_i^I \times W_i^I} \mid i = 1, \dots, n_s\}$ extracted by the image backbone, where (H_i^I, W_i^I) denote the resolution of the i -th scale feature map and N is the number of camera views, the explicit and implicit features of Gaussians are computed as follows:

$$\mathbf{f}^{\text{exp}\ddagger} = \sum_{i=1}^{n_q \times n_p} \text{DeAttn}(\mathbf{f}^{\text{exp}\dagger}, \mathcal{Q}_{3d}[i], \mathcal{M}^I), \quad \mathbf{f}^{\text{imp}\ddagger} = \text{CrossAttn}(\mathbf{f}^{\text{imp}\dagger}, \mathbf{M}_{n_s}^I), \quad (3)$$

where $\mathcal{Q}_{3d}[i]$ denotes the i -th 3D query point in \mathcal{Q}_{3d} . Following the PCA module, we obtain the updated Gaussian representation $\mathbf{G}^\ddagger = \{\mathbf{m}, \mathbf{s}, \mathbf{r}, \mathbf{c}, \mathbf{f}^{\text{exp}\ddagger}, \mathbf{f}^{\text{imp}\ddagger}\}$.

Gaussians Refinement Module. After aggregating information from multi-modal features, we further refine the Gaussian representations. Specifically, we employ two separate self-attention layers [39] to build interactions among all Gaussians—one for the explicit features and the other for the implicit features:

$$\begin{aligned} \{\mathbf{f}_1^{exp'}, \dots, \mathbf{f}_P^{exp'}\} &= \text{SelfAttn}(\{\mathbf{f}_1^{exp\dagger}, \dots, \mathbf{f}_P^{exp\dagger}\}, \{\mathbf{e}_1, \dots, \mathbf{e}_P\}), \\ \{\mathbf{f}_1^{imp'}, \dots, \mathbf{f}_P^{imp'}\} &= \text{SelfAttn}(\{\mathbf{f}_1^{imp\dagger}, \dots, \mathbf{f}_P^{imp\dagger}\}, \{\mathbf{e}_1, \dots, \mathbf{e}_P\}), \\ \{\mathbf{e}_1, \dots, \mathbf{e}_P\} &= \text{PosEmbed}(\{\mathbf{m}_1, \dots, \mathbf{m}_P\}), \end{aligned} \quad (4)$$

where \mathbf{e}_i denotes the positional embedding of the i -th Gaussian, $\text{SelfAttn}(\cdot)$ and $\text{PosEmbed}(\cdot)$ refer to the self-attention and positional embedding layers [33], respectively. Subsequently, following [18], a multi-layer perceptron (MLP) is employed to refine the physical attributes of Gaussians based on their explicit features:

$$\mathbf{G}' = \{\mathbf{m}' + \mathbf{m}, \mathbf{s}', \mathbf{r}', \mathbf{c}', \mathbf{f}^{exp'}, \mathbf{f}^{imp'}\}, \quad (\mathbf{m}', \mathbf{s}', \mathbf{r}', \mathbf{c}') = \text{MLP}(\mathbf{f}^{exp'}). \quad (5)$$

The Gaussian encoder described above is applied iteratively to refine the Gaussian representations. The final updated Gaussians are then passed to the Gaussian decoder, which performs downstream tasks such as mapping and planning.

3.3 Gaussian Decoder: Gaussians to Scene Reasoning

To effectively regulate the 2D Gaussians, we design a Gaussian decoder comprising two components: map construction and cascade planning. The map construction module explicitly reconstructs the traffic scene, providing backpropagation gradients that guide the Gaussian encoder in refining the physical attributes and explicit features. Following [17], we implement this module using probabilistic Gaussian superposition; further details are provided in Appendix A. The cascade planning module generates trajectory predictions in a cascaded manner, where each subsequent output is refined based on the preceding one. In addition to leveraging the explicit features, it also incorporates the implicit features obtained from the Gaussian implicit fusion branch.

Cascade Planning. We adopt an anchor-based planning strategy [4, 29, 28], which constructs an anchor trajectory vocabulary based on the trajectory distribution observed in the dataset. Given the set of Gaussians obtained from the Gaussian encoder, we refine anchor trajectories $\mathcal{A} = \{\mathbf{A}_i \in \mathbb{R}^{T \times 2} \mid i = 1, \dots, k\}$ in a cascaded manner, where T denotes the planning horizon and the number of trajectory points. Taking a single anchor trajectory $\mathbf{A} \in \mathcal{A}$ as an example, we first compute the distance between each of its trajectory points and all Gaussians. For each point, we select its top- m nearest Gaussians, forming a Gaussian subset $\mathcal{G}_A = \{\mathbf{G}_i \mid i = 1, \dots, mT\}$. The anchor feature \mathbf{F}_A is then obtained by querying this Gaussian set:

$$\begin{aligned} \mathbf{F}_A &= \text{CrossAttn}(\mathbf{F}_{query}, \mathbf{F}_{\mathcal{G}_A}), \\ \mathbf{F}_{query} &= \text{Embedding}(\mathbf{A}), \\ \mathbf{F}_{\mathcal{G}_A} &= \{[\mathbf{f}_i^{exp}; \mathbf{f}_i^{imp}] \mid i = 1, \dots, mT\}, \end{aligned} \quad (6)$$

where CrossAttn denotes a cross-attention layer, and Embedding represents an embedding layer that encodes the anchor trajectory \mathbf{A} into an initial query feature \mathbf{F}_{query} . The terms \mathbf{f}_i^{exp} and \mathbf{f}_i^{imp} refer to the explicit and implicit features of the Gaussian $\mathbf{G}_i \in \mathcal{G}_A$, respectively, and $[\mathbf{f}_i^{exp}; \mathbf{f}_i^{imp}]$ denotes their concatenation. The traffic map and surrounding agents can be decoded from the most recent Gaussians \mathcal{G} , enabling \mathcal{G} to serve as a comprehensive representation of the traffic scene. Therefore, we employ a cross-attention layer to build interactions between the anchor feature \mathbf{F}_A and hidden features of \mathcal{G} . The refined trajectory $\tau = \{(x_t, y_t)\}_{t=1}^T$ is obtained as follows:

$$\begin{aligned} \tau &= \text{MLP}(\mathbf{F}_o) + \mathbf{A}, \\ \mathbf{F}_o &= \text{CrossAttn}(\mathbf{F}_A, \mathbf{F}_{\mathcal{G}}), \\ \mathbf{F}_{\mathcal{G}} &= \{[\mathbf{f}_i^{exp}; \mathbf{f}_i^{imp}] \mid i = 1, \dots, P\}. \end{aligned} \quad (7)$$

The trajectory is refined in a cascaded manner, where the output trajectory τ from the current stage is used as the anchor input for the subsequent stage, iteratively repeating the steps described in Eq.6 and Eq. 7 (referred to as Gaussian Spatial Attention and Gaussian Cross Attention in Figure 2, respectively). We adopt the same trajectory loss function as proposed in [32] to supervise the cascade refinement process.

Method	Input	Img. Backbone	NC \uparrow	DAC \uparrow	TTC \uparrow	EP \uparrow	PDMS \uparrow
LTF	Camera	ResNet34	97.4	92.8	92.4	79.0	83.8
TransFuser	C & L	ResNet34	97.7	92.8	93.0	79.2	84.0
GoalFlow	C & L	ResNet34	98.3	93.3	94.8	79.8	85.7
Hydra-MDP	C & L	ResNet34	98.3	96.0	94.6	78.7	86.5
Hydra-MDP++	C	ResNet34	97.6	96.0	93.1	80.4	86.6
ARTEMIS	C & L	ResNet34	98.3	95.1	94.3	81.4	87.0
DiffusionDrive	C & L	ResNet34	98.2	96.2	94.7	82.2	88.1
GaussianFusion (Ours)	C & L	ResNet34	98.3	97.2	94.6	83.0	88.8
GoalFlow	C & L	V2-99	98.4	98.3	94.6	85.0	90.3
Hydra-MDP-C	C	V2-99	98.7	98.2	95.0	86.5	91.0
Hydra-MDP++	C	V2-99	98.6	98.6	95.1	85.7	91.0
GaussianFusion (Ours)	C & L	V2-99	98.7	98.1	95.7	88.2	92.0

Table 1: **Performance on the Navtest Benchmark with Original Metrics.** Definitions of sub-metrics are provided in Appendix B. The best results of different backbones are highlighted in bold, separately.

Method	NC \uparrow	DAC \uparrow	EP \uparrow	TTC \uparrow	DDC \uparrow	LK \uparrow	EC \uparrow	EPDMS \uparrow
TransFuser* [7]	97.7	92.8	79.2	92.8	98.3	67.6	95.3	77.8
VADv2* [4]	97.3	91.7	77.6	92.7	98.2	66.0	97.4	76.6
Hydra-MDP* [29]	97.5	96.3	80.1	93.0	98.3	65.5	97.4	79.8
Hydra-MDP++* [28]	97.9	96.5	79.2	93.4	98.9	67.2	97.7	80.6
ARTEMIS [13]	98.3	95.1	81.5	97.4	98.6	96.5	98.3	83.1
DiffusionDrive [32]	98.2	96.2	87.6	97.3	98.6	97.0	98.4	84.0
GaussianFusion (Ours)	98.3	97.3	87.5	97.4	99.0	97.4	98.3	85.0

Table 2: **Performance on the NAVSIM *navtest* split with extended metrics.** “*” represents that the results are sourced from Hydra-MDP++ [28]. The best results are highlighted in bold.

4 Experiments

4.1 Benchmark and Metric

We evaluate models on NAVSIM [11] and Bench2Drive [23] benchmarks. NAVSIM, built upon the OpenScene dataset [10], provides 120 hours of challenging driving data with high-resolution camera images and LiDAR inputs spanning up to 1.5 seconds. It filters out trivial driving scenarios to emphasize complex decision-making. We use the official Predictive Driver Model Score (PDMS) and its extended version, EPDMS [28], as evaluation metrics. Bench2Drive [23], based on the CARLA simulator [12], evaluates E2E autonomous driving across 220 routes that cover 44 interactive scenarios under diverse conditions. We adopt the CARLA Driving Score (DS) and multi-ability metrics for a fine-grained assessment. For more details about these metrics, please refer to Appendix B.

5 Implementation Details

For the NAVSIM benchmark, we use the NAVSIM *train* split for training. We utilize input from front, left-front, and right-front cameras, along with LiDAR point clouds. The camera images are cropped to a resolution of 448×250 . For the Bench2Drive [23] benchmark, we adopt only the front-view camera, which already provides a wide field of view. In this case, we use the resolution of 1024×384 , which is consistent with the setting used in TransFuser++ [49]. For the two benchmarks, LiDAR points are projected onto the BEV plane, following the approach in TransFuser [7]. In our main experiments, the number of Gaussians is set to 512, and each Gaussian feature has a dimensionality of 128. We adopt 4 GaussianEncoder blocks and 2 cascade planning blocks. The number of anchor trajectories is set to 20 following [32]. It is worth noting that the map construction module in the GaussianDecoder is detached during inference for efficiency. Training is performed using the AdamW optimizer [34], with 50 epochs, a weight decay of 1×10^{-4} , and a maximum learning rate of 6×10^{-4} , which follows a cosine annealing schedule for learning rate decay. Hyper-parameter analysis is in Appendix C.

Method	Overall↑		Merge	Multi-Ability↑				Mean ↑
	DS	SR		Overtake	EBrake	GiveWay	TSign	
PDM-Lite [37]	97.0	92.3	88.8	93.3	98.3	90.0	93.7	92.8
AD-MLP [47]	18.1	0.0	0.0	0.0	0.0	0.0	4.4	0.9
TCP [41]	40.7	15.0	16.2	20.0	20.0	10.0	7.0	14.6
VAD [25]	42.4	15.0	8.1	24.4	18.6	20.0	19.2	18.1
UniAD [16]	45.8	16.4	14.1	17.8	21.7	10.0	14.2	15.6
ThinkTwice [22]	62.4	33.2	27.4	18.4	35.8	50.0	54.4	37.2
DriveTransformer [24]	63.5	35.0	17.6	35.0	48.4	40.0	52.1	38.6
DriveAdapter [21]	64.2	33.1	28.8	26.4	48.8	50.0	56.4	42.1
TF++* [49]	76.9±0.9	54.0±1.0	48.8±2.2	37.6±7.0	64.2±8.4	50.0±0.0	59.7±6.1	52.0±3.0
GaussianFusion (Ours)	79.1±1.1	54.4±2.6	36.6±3.3	64.4±2.3	66.5±6.4	53.3±5.8	60.8±6.4	56.3±3.7

Table 3: **Performance on the Bench2Drive benchmark.** ‘SR’, ‘EBrake’, and ‘TSign’ denote the success rate, emergency braking, and traffic sign compliance, respectively. PDM-Lite is a rule-based planner that can access privileged information from the CARLA simulator. ‘*’ denotes the implementation of the same backbone and training dataset setting as our method.

Gaussian Exp. Fusion	Gaussian Imp. Fusion	Cascade Planning	Agent Pred.	Param.	DAC↑	TTC↑	EP↑	LK↑	EPDMS↑
✗	✗	✗	✗	55.7M	94.1	96.9	87.5	96.2	81.8
✓	✗	✗	✗	49.6M	96.5	97.1	87.8	97.0	84.2
✓	✓	✗	✗	51.6M	96.6	97.3	87.8	97.3	84.5
✓	✗	✓	✗	53.7M	97.0	97.2	87.4	97.2	84.4
✓	✓	✓	✗	55.8M	97.3	97.4	87.5	97.4	85.0
✓	✓	✓	✓	59.5M	96.2	97.3	87.6	97.0	83.8

Table 4: **Ablation study** of different architecture configurations. We utilize TransFuser [7] without the agent prediction head as the baseline model (the first row setting).

5.1 Comparison with State-of-the-Art Methods

Results on NAVSIM. We benchmark GaussianFusion against leading state-of-the-art (SOTA) methods on the NAVSIM *navtest* split. We employ ResNet34 [14] and V2-99 [27] as the image backbone of GaussianFusion, respectively. As shown in Table 1, our method significantly surpasses previous methods, particularly on critical sub-metrics like DAC and Ego Vehicle Progress (EP) under the ResNet34 setting. Our enhanced GaussianFusion, equipped with V2-99, further improves performance across all metrics—achieving state-of-the-art results on the Navtest benchmark. This demonstrates the scalability of our framework with stronger backbones. Additionally, we evaluate performance under the EPDMS metric. Note that EPDMS poses a stricter challenge than PDMS by incorporating more nuanced driving criteria. As shown in Table 2, our approach achieves 85.0 EPDMS, outperforming recent strong baselines such as [32] and [13] by 1.0 and 1.9 points, respectively. A closer look reveals that most gains come from the Drivable Area Compliance (DAC) and Lane Keeping (LK) sub-metrics, indicating that GaussianFusion enables more stable and context-aware behavior in complex environments. These results consistently affirm the robustness and effectiveness of our method across multiple evaluation protocols.

Results on Bench2Drive. We further conduct experiments on the closed-loop benchmark, Bench2Drive, to compare our method with existing SOTA E2E methods. Due to high performance variability in the CARLA simulator, we report results across three different random seeds. As shown in Table 3, our method, GaussianFusion, achieves the best overall performance (79.4 DS), outperforming all learning-based baselines. It shows balanced strength across diverse tasks, with notable gains in overtaking and traffic sign compliance. Compared to the rule-based privileged method PDM-Lite, our method still falls short to some extent, indicating that there remains substantial room for improvement in E2E autonomous driving methods.

5.2 Ablation Study

Effect of Different Components. To understand the impact of each design choice in GaussianFusion, we perform a controlled ablation study by incrementally adding Gaussian Explicit Fusion (**Gaussian Exp. Fusion**), Gaussian Implicit Fusion (**Gaussian Imp. Fusion**), the Cascade Planning head

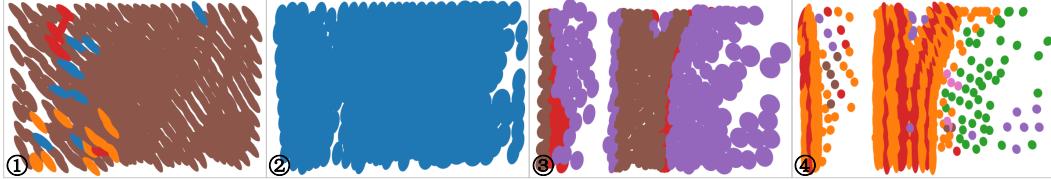


Figure 3: **Visualization of Gaussians** during the refinement process. Gaussians with different semantics are shown in different colors.



Figure 4: **Visualization of predicted and ground-truth trajectories**, shown in red and green, respectively.

(**Cascade Planning**), and the Agent Prediction head (**Agent Pred.**) to the TransFuser baseline [7]. The results are shown in Table 4. Introducing **Gaussian Exp. Fusion** leads to a substantial +2.4 gain in EPDMS, while also reducing the parameter count. Adding **Gaussian Imp. Fusion** further improves performance to 84.5 EPDMS, with only a slight increase in parameters. Separately, incorporating **Cascade Planning** with only the explicit fusion also improves trajectory prediction, increasing LK and EPDMS compared to using explicit fusion alone. The best overall performance is achieved when combining dual-branch fusion and **Cascade Planning**, reaching an EPDMS of 85.0. These findings confirm that our architectural components offer strong performance gains without significantly increasing model complexity. Finally, we observe that incorporating the agent prediction head degrades performance. We attribute this to the agent prediction task failing to provide effective guidance for the Gaussian refinement process, instead introducing ambiguity—what we refer to as Gaussian confusion. Given that the semantic map already encodes sufficient agent-related information, we remove the agent prediction head from our final design.

Different Multi-Sensor Fusion Methods. Table 5 presents a comprehensive comparison of various multi-sensor fusion methods in terms of model parameters, semantic map construction, trajectory planning, and inference latency. The latency is measured by an RTX3090. To ensure a fair comparison, all methods adopt the same backbone and task heads.

Compared with the BEV fusion method, our Gaussian fusion achieves better performance (+0.04 mIoU and +0.7 EPDMS), while also significantly reducing computational cost: the FLOPs are 38.9% lower and latency is reduced by 25.6%, showing clear advantages in both accuracy and efficiency.

Fusion Method	Param.	FLOPs (G)	mIoU↑	EPDMS↑	Latency ↓ (ms)
Flatten	55.7M	27.4	0.50	81.8	18
BEV	51.9M	58.4	0.51	83.8	43
Gaussian	51.6M	35.7	0.55	84.5	32

Table 5: **Comparison of fusion methods.**

ments in deployment.

Compared with the flatten (TransFuser-style) fusion, which is a very lightweight approach, our method exhibits only a 30.3% increase in FLOPs, while achieving much stronger performance (+0.05 mIoU and +2.7 EPDMS). However, our method’s latency is 77.8% higher, which we believe is largely due to the differences in operator-level optimization. This could be further mitigated through engineering improvements in deployment.

5.3 Qualitative Comparison

To provide an intuitive understanding of the refinement process in the Gaussian encoder, we visualize the spatial distribution of Gaussians at different refinement stages, as shown in Figure 3. At the initial stage, Gaussians are evenly dispersed throughout the scene. As refinement proceeds, they progressively converge toward foreground regions. This behavior highlights the advantages of the Gaussian representation, which offers a more compact and adaptable alternative to traditional dense BEV maps. More visualizations of Gaussians are shown in Figure 5 in the Appendix. We also present

predicted ego-vehicle trajectories under a variety of traffic scenarios in Figure 4. To qualitatively evaluate the prediction accuracy, we compare these trajectories against ground-truth data. In the leftmost scenario of Figure 4, the vehicle makes an unprotected left turn without signal guidance—a challenging case. Our method still predicts a trajectory closely matching the ground truth. In addition, as shown in the two rightmost scenarios of Figure 4, our method is capable of producing accurate trajectory plans even in dense traffic conditions, further demonstrating its robustness and reliability.

6 Conclusion

In this work, we propose GaussianFusion, a Gaussian-based multi-sensor fusion framework for E2E autonomous driving. By utilizing compact and flexible 2D Gaussian representations, our method balances spatial awareness with computational efficiency. A dual-branch fusion architecture captures both local details and global planning cues from multi-modal inputs, while a cascade planning module progressively refines trajectory predictions. Experiments on NAVSIM and Bench2Drive benchmarks show that GaussianFusion significantly improves planning performance with high efficiency. These results underscore the promise of Gaussian representations for efficient and interpretable sensor fusion in the E2E autonomous driving system. The limitation of GaussianFusion lies in its customized CUDA operations, which are not fully optimized. In future work, we plan to either further optimize these operations or replace them with operations of a well-established neural network library.

Limitations. We have not conducted quantitative evaluations on tracking small or fast-moving objects. Similarly, our method has not been assessed under sensor occlusion or noise, which are critical factors for real-world deployment. Moreover, our method depends on 3D bounding box and road topology labels to optimize the 2D Gaussians. While effective, such supervision can be expensive or unavailable in real-world scenarios.

Acknowledgments

This work was supported in part by the the Guangxi Key R & D Program under Grant No.GuikeAB24010324, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515011485.

References

- [1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *CVPR*, 2022.
- [2] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018.
- [3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [4] Shaoyu Chen, Bo Jiang, Hao Gao, Bencheng Liao, Qing Xu, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243*, 2024.
- [5] Zehui Chen, Zhenyu Li, Shiquan Zhang, Liangji Fang, Qinhong Jiang, Feng Zhao, Bolei Zhou, and Hang Zhao. Autoalign: Pixel-instance feature aggregation for multi-modal 3d object detection. In *IJCAI*, 2022.
- [6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *IJRR*, 2023.
- [7] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *TPAMI*, 2022.

- [8] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*, 2018.
- [9] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *ICCV*, 2019.
- [10] OpenScene Contributors. Openscene: The largest up-to-date 3d occupancy prediction benchmark in autonomous driving. <https://github.com/OpenDriveLab/OpenScene>, 2023.
- [11] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *NeurIPS*, 2024.
- [12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, 2017.
- [13] Renju Feng, Ning Xi, Duanfeng Chu, Rukang Wang, Zejian Deng, Anzheng Wang, Liping Lu, Jinxiang Wang, and Yanjun Huang. Artemis: Autoregressive end-to-end trajectory planning with mixture of experts for autonomous driving. *arXiv preprint arXiv:2504.19580*, 2025.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [15] Anthony Hu, Gianluca Corrado, Nicolas Griffiths, Zachary Murez, Corina Gurau, Hudson Yeo, Alex Kendall, Roberto Cipolla, and Jamie Shotton. Model-based imitation learning for urban driving. In *NeurIPS*, 2022.
- [16] Yihan Hu, Jiazhui Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhui Wang, et al. Planning-oriented autonomous driving. In *CVPR*, 2023.
- [17] Yuanhui Huang, Amonnut Thammatadatrakoon, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, and Jiwen Lu. Gaussianformer-2: Probabilistic gaussian superposition for efficient 3d occupancy prediction. *CoRR*, 2024.
- [18] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Gaussianformer: Scene as gaussians for vision-based 3d semantic occupancy prediction. In *ECCV*, 2024.
- [19] Keishi Ishihara, Anssi Kanervisto, Jun Miura, and Ville Hautamaki. Multi-task learning with attention for end-to-end autonomous driving. In *CVPR Workshops*, 2021.
- [20] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *ICCV*, 2023.
- [21] Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *ICCV*, 2023.
- [22] Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *CVPR*, 2023.
- [23] Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024.
- [24] Xiaosong Jia, Junqi You, Zhiyuan Zhang, and Junchi Yan. Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. In *ICLR*, 2025.
- [25] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *ICCV*, 2023.
- [26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 2023.

- [27] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *CVPR workshops*, 2019.
- [28] Kailin Li, Zhenxin Li, Shiyi Lan, Yuan Xie, Zhizhong Zhang, Jiayi Liu, Zuxuan Wu, Zhiding Yu, and Jose M Alvarez. Hydra-mdp++: Advancing end-to-end driving via expert-guided hydra-distillation. *arXiv preprint arXiv:2503.12820*, 2025.
- [29] Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, et al. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024.
- [30] Zhiqi Li, Wenhui Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: learning bird’s-eye-view representation from lidar-camera via spatiotemporal transformers. *TPAMI*, 2024.
- [31] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018.
- [32] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. *arXiv preprint arXiv:2411.15139*, 2024.
- [33] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: dynamic anchor boxes are better queries for DETR. In *ICLR*, 2022.
- [34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [35] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*, 2020.
- [36] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *CoRL*, 2023.
- [37] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengan Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. In *ECCV*, 2024.
- [38] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [40] Xi Wei, Tianzhu Zhang, Yan Li, Yongdong Zhang, and Feng Wu. Multi-modality cross attention network for image and sentence matching. In *CVPR*, 2020.
- [41] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*, 2022.
- [42] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *CVPR*, 2022.
- [43] Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. *arXiv preprint arXiv:2503.05689*, 2025.
- [44] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024.
- [45] Tengju Ye, Wei Jing, Chunyong Hu, Shikun Huang, Lingping Gao, Fangzhen Li, Jingke Wang, Ke Guo, Wencong Xiao, Weibo Mao, et al. Fusionad: Multi-modality fusion for prediction and planning tasks of autonomous driving. *arXiv preprint arXiv:2308.01006*, 2023.

- [46] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023.
- [47] Jiang-Tian Zhai, Ze Feng, Jinhao Du, Yongqiang Mao, Jiang-Jiang Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv preprint arXiv:2305.10430*, 2023.
- [48] Wenzhao Zheng, Junjie Wu, Yao Zheng, Sicheng Zuo, Zixun Xie, Longchao Yang, Yong Pan, Zhihui Hao, Peng Jia, Xianpeng Lang, et al. Gaussianad: Gaussian-centric end-to-end autonomous driving. *arXiv preprint arXiv:2412.10371*, 2024.
- [49] Julian Zimmerlin, Jens Beißwenger, Bernhard Jaeger, Andreas Geiger, and Kashyap Chitta. Hidden biases of end-to-end driving datasets. *arXiv.org*, 2412.09602, 2024.

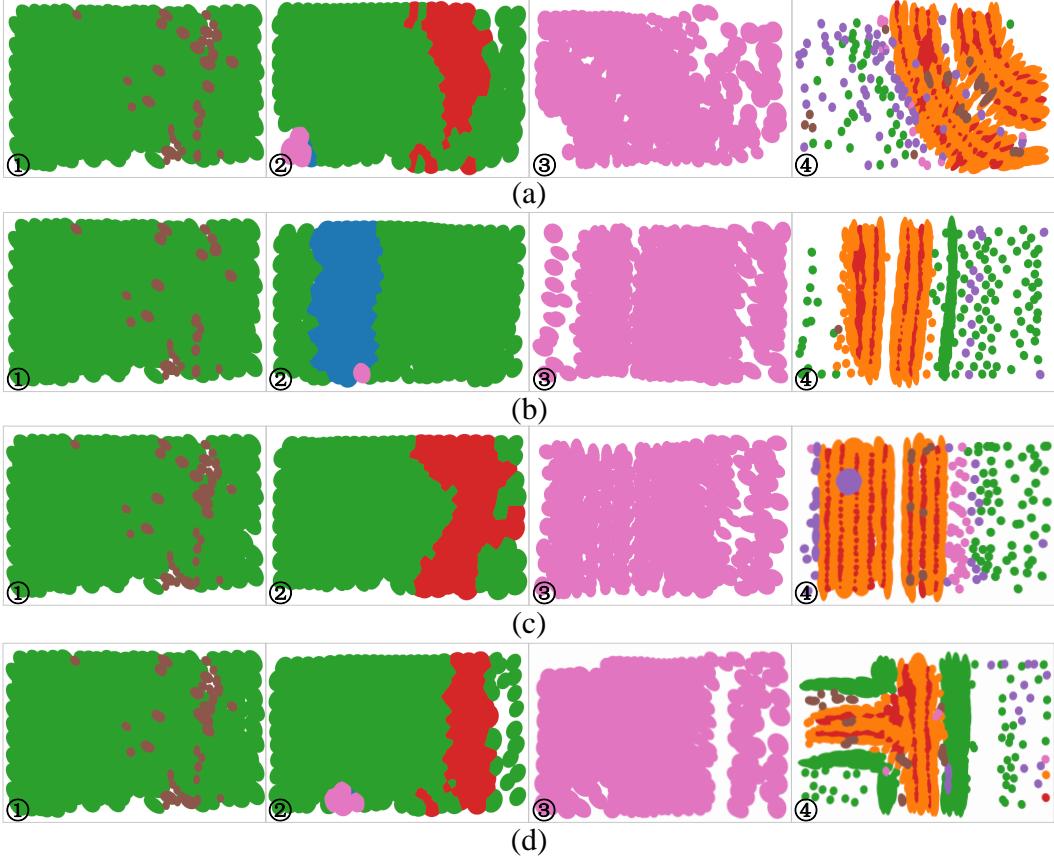


Figure 5: **Visualization of Gaussians** of different inputs during the refinement process. Gaussians with different semantics are shown in different colors.

A Map Construction

Following [17], we utilize Gaussian probability superposition to construct the semantic map. Specifically, for a map pixel position $\mathbf{x} \in \mathbb{R}^2$, we first calculate its probability of falling in different Gaussians:

$$\alpha(\mathbf{x}; \mathbf{G}_i) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i)\right), \quad (8)$$

$$\Sigma_i = \mathbf{R} \mathbf{S}^T \mathbf{R}^T, \quad \mathbf{S} = \text{diag}(\mathbf{s}_i), \quad \mathbf{R} = \text{r2m}(\mathbf{r}_i),$$

where \mathbf{m}_i , \mathbf{s}_i , and \mathbf{r}_i denote the mean, scale, and rotation vectors of the Gaussian \mathbf{G}_i , respectively. Σ_i represents the covariance matrix of \mathbf{G}_i ; $\text{diag}(\cdot)$ denotes the operator that constructs a diagonal matrix from a given vector; and $\text{r2m}(\cdot)$ denotes the function that converts a rotation vector into its corresponding rotation matrix. Then the superposition probability can be calculated as:

$$\mathbf{o}(\mathbf{x}; \mathcal{G}) = \sum_{i=1}^P p(\mathbf{G}_i | \mathbf{x}) \mathbf{c}'_i = \frac{\sum_{i=1}^P p(\mathbf{x} | \mathbf{G}_i) a_i \mathbf{c}'_i}{\sum_{j=1}^P p(\mathbf{x} | \mathbf{G}_j) a_j} \in \mathbb{R}^C, \quad (9)$$

$$p(\mathbf{x} | \mathbf{G}_i) = \frac{1}{(2\pi)^{\frac{3}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m})^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{m})\right),$$

where $\mathbf{o}(\mathbf{x}; \mathbf{G}_i)$ denotes the logits of different semantics on \mathbf{x} , a_i represents the prior probability of the existence of i -th Gaussian which is predicted by the network, and $p(\mathbf{G}_i | \mathbf{x})$ indicates the

Gaussian Numbers	DAC↑	TTC↑	EP↑	LK↑	EPDMS↑
256	96.6	97.3	87.6	97.1	84.1
512	97.3	97.4	87.5	97.4	85.0
768	97.2	97.5	87.4	97.5	84.8

Table 6: Impact of the number of Gaussians.

Encoder Blocks	DAC↑	TTC↑	EP↑	LK↑	EPDMS↑
3	97.1	97.5	87.4	97.5	84.6
4	97.3	97.4	87.5	97.4	85.0
5	97.1	97.5	87.7	97.2	84.7

Table 7: Effect of the number of encoder blocks.

posterior probability of the i -th Gaussian being present given the observation at \mathbf{x} . Based on these quantities, we can derive the probabilities for the background and foreground:

$$p_b(\mathbf{x}) = \prod_{i=1}^P (1 - \alpha(\mathbf{x}; \mathbf{G}_i)) \in \mathbb{R}^1, \quad (10)$$

$$p_f(\mathbf{x}) = (1 - p_b(\mathbf{x})) \text{Softmax}(\mathbf{o}(\mathbf{x}; \mathcal{G})) \in \mathbb{R}^C,$$

where $p_b(\mathbf{x})$ and $p_f(\mathbf{x})$ denote the background and foreground probabilities of pixel \mathbf{x} , Softmax(\cdot) represents the Softmax function. Intuitively, for a background pixel \mathbf{x}_b , the $p_b(\mathbf{x}_b)$ will decrease with Gaussians gather on \mathbf{x}_b because $p_b(\mathbf{x}_b) \leq (1 - \alpha(\mathbf{x}_b; \mathbf{G}_i))$ holds for any Gaussian. Conversely, for a foreground pixel \mathbf{x}_f , the $p_f(\mathbf{x}_f)$ will increase with Gaussians gather on \mathbf{x}_f due to $1 - p_b(\mathbf{x}_f) \geq \alpha(\mathbf{x}_f; \mathbf{G}_i)$ holds for any Gaussian. Therefore, Eq. 10 will encourage Gaussians to move to foreground pixels, as demonstrated in Figure 3. Given background and foreground predictions $[p_b; p_f] \in \mathbb{R}^{C+1}$ and ground-truth bev map, we employ cross entropy loss L_{ce} and the lovasz-softmax [2] loss L_{lov} as the map construction loss functions.

B Metrics

Cascade Stages	DAC↑	TTC↑	EP↑	LK↑	EPDMS↑
1	96.6	97.4	87.4	97.1	84.3
2	97.3	97.4	87.5	97.4	85.0
3	97.0	97.4	87.8	97.3	84.7

Table 8: Results of the stage of cascade planning.

(C), and Ego Vehicle Progress (EP). EPDMS extends this framework by incorporating additional criteria: Lane Keeping (LK), Extended Comfort (EC), Driving Direction Compliance (DDC), Traffic Light Compliance (TLC), and False-Positive Penalty Filtering, thereby offering a more comprehensive assessment of driving behavior.

For the Bench2Drive [23] benchmark, we employ the CARLA Driving Score (DS), which jointly considers route completion and penalties for traffic infractions. In addition to DS, we also use multi-ability metrics that target specific driving skills of the autonomous driving system, such as merging, overtaking, yielding, traffic sign compliance, and emergency braking.

C Hyper-Parameter Analysis

In this section, we determine the optimal value for the number of Gaussians, the number of encoder blocks, and the stage number of cascade planning through experiments on the NAVSIM dataset. Table 6 shows the impact of the number of Gaussians. When increasing the number of Gaussians from 256 to 512, the performance is improved significantly. However, further enlarging the number of Gaussians does not bring gains. Table 7 presents the effect of the number of Gaussian encoder blocks. Setting the number of encoder blocks to 4 is the best choice. Since we only supervise the output of the last encoder block, too deep may affect performance. Table 8 illustrates the performance of different stage number of cascade planning. We observe that 2 cascade stages are enough to generate accurate trajectories.

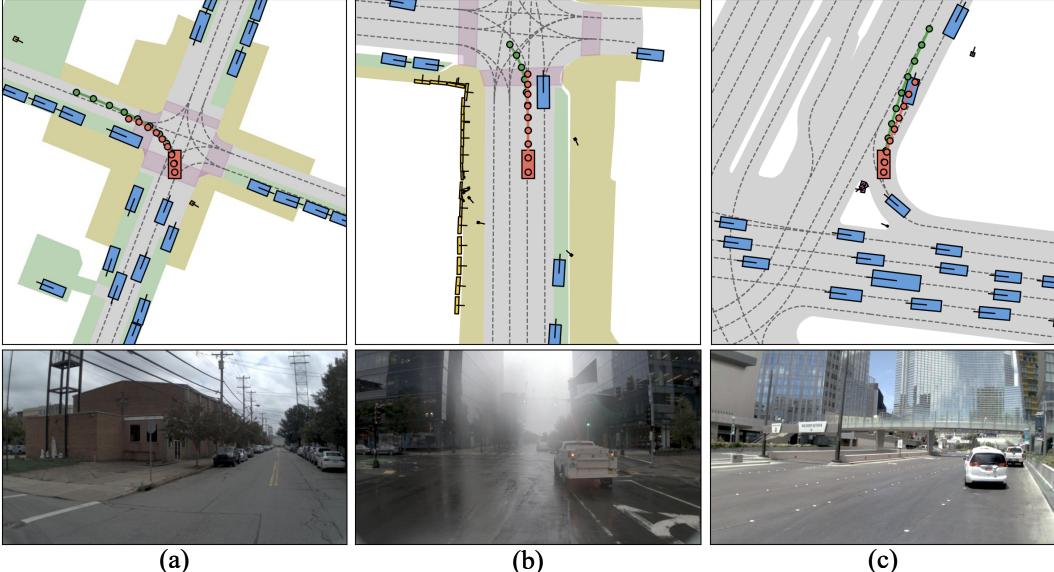


Figure 6: **Failure mode** of GaussianFusion in some scenarios.

D Failure Mode

To better understand the limitations of GaussianFusion in complex real-world scenarios, we visualize typical failure cases in Figure 6. These cases reveal distinct behavioral biases in the learned policy, reflecting the implicit priors encoded in the training dataset and the conservative nature of GaussianFusion.

In Figure 6(a), the model demonstrates overly conservative behavior when executing turning maneuvers at intersections. Compared with human experts who perform smooth and anticipatory turns, GaussianFusion tends to delay or narrow the turning trajectory. We hypothesize that this is attributed to the data distribution, where the majority of demonstrated driving behaviors follow cautious and robust strategies, leading the model to learn a risk-averse policy in ambiguous topological regions.

Figure 6(b) illustrates model behavior under adverse weather conditions such as fog. In low-visibility scenarios, the model prioritizes stability by choosing a slow and straight driving strategy rather than making proactive turning decisions. This behavior suggests that Gaussian primitives lose discriminative power in degraded sensor conditions, causing the planner to rely on the safest longitudinal control instead of predictive planning.

Figure 6(c) reflects the model’s tendency to maintain its lane and follow a leading vehicle that is changing lanes, instead of performing an overtaking maneuver. This indicates a bias toward conservative following behavior in highly dynamic multi-agent scenarios. The model may underestimate the feasibility or safety margin of overtaking due to uncertainty in surrounding vehicle intentions, resulting in a reactive rather than assertive driving strategy.

Overall, these failure cases highlight that GaussianFusion inherits conservative driving priors from the dataset and tends to favor low-risk behaviors when facing uncertainty. While such conservatism improves safety in most situations, it may limit driving efficiency or adaptability in complex environments.