VLM-AD: End-to-End Autonomous Driving through Vision-Language Model Supervision

Yi Xu[1,2*] **Yuxin Hu**[3*] **Zaiwei Zhang**[4*] **Gregory P. Meyer**[4*]
Siva Karthik Mustikovela[1] **Siddhartha Srinivasa**[5*] **Eric M. Wolff**[1] **Xin Huang**[6*]
[1] Cruise LLC (GM) [2] Northeastern University [3] OpenAI
[4] Meta [5] University of Washington [6] Waymo LLC
xu.yi@northeastern.edu eric.wolff@gm.com xhuang@csail.mit.edu

Abstract: Human drivers rely on commonsense reasoning to navigate diverse and dynamic real-world scenarios. Existing end-to-end (E2E) autonomous driving (AD) models are typically optimized to mimic driving patterns observed in data, without capturing the underlying reasoning processes. This limitation constrains their ability to handle challenging driving scenarios. To close this gap, we propose VLM-AD, a method that leverages vision-language models (VLMs) as teachers to enhance training by providing additional supervision that incorporates unstructured reasoning information and structured action labels. Such supervision enhances the model's ability to learn richer feature representations that capture the rationale behind driving patterns. Importantly, our method does not require a VLM during inference, making it practical for real-time deployment. When integrated with state-of-the-art methods, VLM-AD achieves significant improvements in planning accuracy and reduced collision rates on the nuScenes dataset. It further improves route completion and driving scores under closed-loop evaluation, demonstrating its effectiveness in long-horizon, interactive driving scenarios and its potential for safe and reliable real-world deployment.

Keywords: End-to-End Autonomous Driving, Vision-Language Model, Multimodal Large Language Model, Foundation Models for Driving

1 Introduction

End-to-end autonomous driving (AD) unifies perception, prediction, and planning into a single framework. This integration aims to coordinate multiple complex tasks, including detection, tracking, mapping, prediction, and planning. Recent approaches [1, 2, 3] have tackled these challenges by using sensor data to generate planned ego trajectories with a single, holistic model. Although these methods have shown promising results, their performance degrades in challenging, long-tail events [4, 5]. On the other hand, human drivers often handle such scenarios effectively by reasoning through the driving environment and adapting their actions accordingly. This highlights a training gap in current E2E models, which rely solely on the trajectory supervision as sequences of points, lacking the reasoning information necessary for learning rich and robust feature representations to achieve better driving performance.

Manual annotation of reasoning information is often costly, time-consuming, and prone to inconsistent and subjective results, making it difficult to obtain high-quality and scalable annotations. Large foundation models offer an alternative by providing their reasoning capabilities for complex tasks such as driving. Recent methods [6, 7, 8, 9, 10, 11, 12, 13, 14, 15] have directly integrated large foundation models, such as large language models (LLMs) [16, 17, 18] and vision-language models (VLMs) [19, 20, 21, 22], into AD systems to leverage their reasoning capabilities. However, these

*Work done at Cruise LLC (GM).

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

methods require extensive fine-tuning to translate language-based outputs into precise numerical results, such as planned trajectories or control signals. In addition, these methods rely on large foundation models during inference, which significantly increases both training costs and inference time, making these methods impractical for real-world applications. Given the limitations of manual annotation and the challenges of directly integrating large foundation models into driving systems, we pose the following question: Can large foundation models, such as VLMs, generate reasoning-based text information to enhance autonomous driving models without requiring integration at inference time?

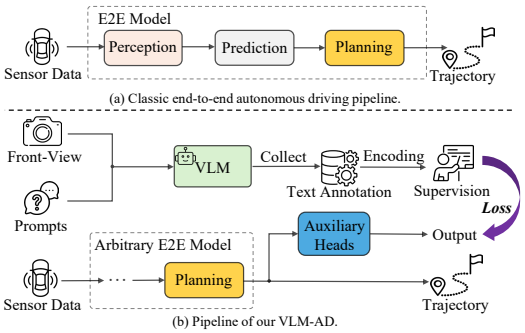


Figure 1: VLM-AD augments an arbitrary end-to-end driving model using auxiliary text prediction tasks during training. These tasks distill driving reasoning knowledge from a VLM to encourage the model to learn richer representations, without fine-tuning a VLM at training time or requiring a VLM at inference time.

Motivated by this question, we propose VLM-AD, illustrated in Fig. 1, a novel method that leverages VLMs as teachers to automatically generate reasoning-based text annotations. These annotations then serve as supplementary supervisory signals to train end-to-end pipelines, extending beyond standard trajectory labels. Specifically, given a sequence of multi-view images and the future trajectory of the ego vehicle, we project the future trajectory onto the initial front-view image to incorporate critical temporal movement information. We then prompt the VLM model with questions regarding the vehicle's current status, intended future actions, and reasoning process to generate both freeform and structured responses, thus infusing critical VLM knowledge into the training pipeline.

This scalable approach enables us to build a dataset enriched with VLM-generated annotations, effectively addressing the absence of reasoning cues in existing driving datasets. We design auxiliary tasks based on these annotations and integrate them seamlessly into existing end-to-end models for joint training. These tasks encourage the model to learn richer feature representations for improved driving performance, without requiring VLM involvement at inference time. Our contributions can be summarized as follows:

- We propose VLM-AD, a simple yet effective approach that distills driving reasoning knowledge from VLMs into end-to-end AD pipelines through a high-quality dataset of reasoning-based behavioral text annotations, generated through carefully crafted prompts directed to VLMs.
- We design two plug-and-play auxiliary tasks to supervise existing end-to-end AD pipelines through both unstructured freeform text and structured action labels. These tasks enable effective distillation of VLM knowledge, guiding the model to learn richer feature representations for improved planning performance, without requiring VLM fine-tuning or inference-time usage.
- Extensive experiments demonstrate significant improvements in L2 planning accuracy and collision rate over UniAD, VAD, and SparseDrive on the nuScenes dataset, as well as higher route completion and driving scores on the closed-loop CARLA Town05 benchmark.

2 Related Work

End-to-End Autonomous Driving. End-to-end autonomous driving systems jointly train all modules toward a unified goal, resulting in reduced information loss throughout the pipeline. Unified frameworks such as ST-P3 [1] and UniAD [2] propose vision-based end-to-end AD systems that unify perception, prediction, and planning. These models achieve state-of-the-art results on the open-loop nuScenes dataset [23]. Following works, such as VAD [3] and VADv2 [24], introduce a vectorized encoding approach for efficient scene representation and extend to closed-loop simulation on CARLA [25]. Recent methods like Ego-MLP [26], BEV-Planner [27], and PARA-Drive [28] have been developed to explore ego-status and novel design spaces within modular stacks to further

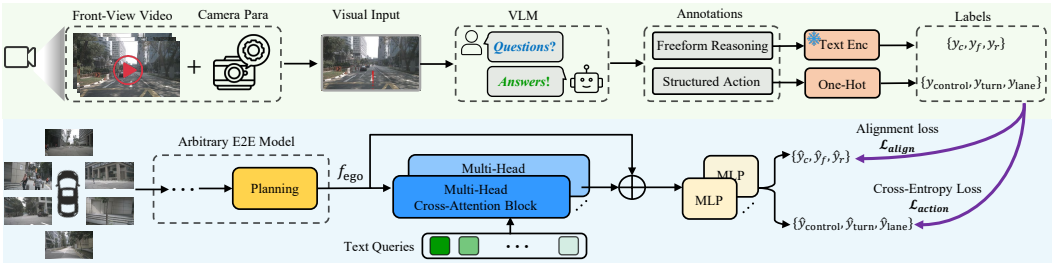


Figure 2: Framework of our proposed VLM-AD. We leverage a VLM as a teacher to generate both freeform reasoning and structured action annotations, which are converted into supervisory signals to enable the model to learn richer representations through auxiliary text alignment and action classification heads. As a result, our method offers better planning results and interpretable action predictions, without requiring a VLM at inference time.

enhance driving performance. In parallel, methods like GenAD [29], GraphAD [30], SparseAD [31], and SparseDrive [32] continue to push the boundaries of end-to-end driving systems through improved representations and task formulations. While E2E driving models show promising results in the development of E2E driving methods, they are primarily optimized to mimic driving patterns in the data, without capturing the underlying reasoning processes. This limitation is largely due to the lack of reasoning information in existing datasets. Consequently, these methods are unable to acquire deeper reasoning knowledge, which could limit their performance in challenging scenarios.

Foundation Models for Autonomous Driving. Foundation models, including large-language models (LLMs) and vision-language models (VLMs), are being increasingly applied in autonomous driving to leverage their advanced reasoning capabilities. GPT-Driver [6] and Driving-with-LLMs [7] use LLMs to provide action recommendations with explanations, thus enhancing decision transparency. A recent approach [33] leverages LLMs to evaluate lane occupancy and safety, enabling more human-like intuitive scene understanding. However, LLM-based methods primarily rely on language inputs, which limits their potential to incorporate rich visual features essential for driving.

VLMs address this gap by integrating language and vision for multimodal reasoning, supporting tasks like scene understanding [34, 35, 36, 37] and data generation [38, 39, 40]. VLMs have also been used for unified navigation and planning [41, 10, 42, 43] as well as end-to-end autonomous driving [8, 9, 11, 12, 44]. However, existing VLM-based methods often require extensive domain-specific fine-tuning, which significantly increases computational cost and inference latency. Closely related to our method in end-to-end autonomous driving, VLP [12] transforms ground-truth trajectory and bounding box labels into text features for contrastive learning, but it does not introduce information beyond existing supervision labels. In contrast, our method leverages VLMs to provide additional reasoning information to further enhance driving performance.

3 Method

Fig. 2 presents an overview of our proposed VLM-AD framework, which consists of two main components. The first is the annotation component, where a VLM generates rich auxiliary information, forming a supplementary dataset for supervision. The second component is our designed auxiliary heads designed to align with this supervision, which can be seamlessly integrated into any E2E model following the planning module.

3.1 VLM Text Annotation

We utilize a VLM as the teacher to enrich the dataset with additional information, leveraging its reasoning capabilities from visual inputs to deepen an E2E model's understanding of driving behaviors. The annotation process can be defined as:

$$\mathcal{A} = \mathbf{M}(\mathcal{P}, \mathcal{V}), \tag{1}$$

where $\mathbf{M}(\cdot)$ represents the VLM model, $\mathcal{P}$ denotes the language prompts, $\mathcal{V}$ is the visual input, and $\mathcal{A}$ is the model's natural language output, serving as annotations for the dataset. Our goal is to provide images captured from the ego vehicle's camera, along with specifically crafted prompts, to obtain detailed informative responses from the VLM, leveraging its extensive world knowledge.

In our work, we employ GPT-4o [45], a high-performance VLM trained on internet-scale data, to automatically annotate our dataset. GPT-4o is used to interpret the scenario, generate suitable reasoning-based responses, and identify the actions of the ego vehicle in complex scenarios.

Visual Input. When determining the visual input, we encounter two challenges. The first challenge is selecting the appropriate image(s) from multiple cameras that provide 360-degree coverage around the ego vehicle. We explore two approaches: creating a composite large image from all views or using only the front-view image, which typically contains the most relevant information needed for most driving tasks. Our annotation results show that both methods yield comparable output quality, so we opt for the front-view image alone to reduce overall complexity.

The second challenge involves integrating temporal information, which is essential for effective planning and decision-making. We also consider two approaches. One straightforward approach is to input several consecutive frames as a sequence, with prompts that indicate the future timestamps. However, we observe that VLMs struggle with temporal continuity and often confuse the ego vehicle's identity, likely due to limitations in temporal grounding [46, 47]. Instead, we project the ego vehicle's future trajectory onto a single front-view image, leveraging the camera's intrinsic and extrinsic parameters along with sensor specifications. We specify in the prompts that the projected trajectory reflects the vehicle's future path. This cost-effective design allows the VLM to interpret temporal information more reliably than using an image sequence.

Freeform Reasoning Annotation. As a key input to the VLM, a well-designed question is essential for enhancing reasoning capabilities [48] and improving the explainability of the VLM's responses. In our approach, we focus on the planning task, by designing prompts specifically to obtain reasoning from the VLM. We create two types of questions, beginning with open-ended questions intended to generate free-form, unstructured responses that contain rich, high-dimensional language information. We refer to these responses as unstructured reasoning annotations.

To maximize the reasoning capabilities of the VLM, we provide detailed context descriptions as preliminary instructions before posing specific questions, defined as follows:

- C_1: *This is the front-view image of the ego vehicle. The red line indicates the future trajectory, no line suggests stopping or slowing down. When explaining the reasoning, please focus on the camera image and the surrounding context rather than referencing the plotted trajectory.*
- Q_{1-1}: *Please describe the ego vehicle's current actions.*
- Q_{1-2}: *Please predict the ego vehicle's future actions.*
- Q_{1-3}: *Please explain the reasoning of current and future action.*

The full input prompt is defined as $\mathcal{P}_1 = [C_1, Q_1]$, where $Q_1 = \{Q_{1-1}, Q_{1-2}, Q_{1-3}\}$ represents the set of questions. These open-ended questions yield free-form text annotations describing the ego vehicle's current status, intended future actions, and the reasoning underlying the VLM's knowledge.

Structured Action Annotation. To examine the flexibility of our method, we define a second type of question in a structured format. Specifically, we create three distinct action sets and prompt the VLM to select answers from these predefined options. This allows us to obtain a single action annotation for each question. Specifically, the context and questions are defined as follows:

- C_2: *This is the front-view image of the ego vehicle. The red line indicates the future trajectory, no line suggests stopping or slowing down.*
- Q_{2-1}: *Please describe the ego vehicle's action from the control action list:* {*go straight, move slowly, stop, reverse*}.
- Q_{2-2}: *Please describe the ego vehicle's action from the turn action list:* {*turn left, turn right, turn around, none*}.

- Q_{2-3}: *Please describe the ego vehicle's action from the lane action list: {change lane to the left, change lane to the right, merge into the left lane, merge into the right lane, none}.*

The complete input prompt is defined as $\mathcal{P}_2 = [C_2, Q_2]$, where $Q_2 = \{Q_{2-1}, Q_{2-2}, Q_{2-3}\}$ represents the set of structured action questions. In this way, we can obtain three specific actions from the VLM. Compared to freeform text annotations, one major benefit of structured annotations is that they can be used to supervise an E2E driving model to predict human-interpretable actions, as demonstrated in the experimental results of Sec. 4.

3.2 Auxiliary Heads

Typically, data-driven end-to-end autonomous driving methods [2, 3] focus on summarizing a learnable ego feature f_{ego} to produce planning results, which is essential for generating reliable and accurate planning trajectories. This learnable ego feature aggregates all relevant information about the ego vehicle from upstream modules through different networks. In our approach, we develop auxiliary heads that use this ego feature as input, enabling the model to distill knowledge from the VLM's responses.

Annotation Encoding. Using the Q_1 questions, we obtain three text responses, denoted as $\mathcal{A}_1 = \{\mathcal{A}_c, \mathcal{A}_f, \mathcal{A}_r\}$, which represent descriptions of the current action, future action prediction, and reasoning, respectively. Using the Q_2 questions, we obtain three actions from predefined sets, denoted as $\mathcal{A}_2 = \{\mathcal{A}_{\text{control}}, \mathcal{A}_{\text{turn}}, \mathcal{A}_{\text{lane}}\}$, corresponding to the control action, turn action, and lane action. To convert these annotations into supervisory signals, we apply two distinct approaches to generate two corresponding types of labels, effectively integrating them into end-to-end autonomous driving pipelines as supervision.

For the freeform text annotations from Q_1, we utilize an off-the-shelf language model, such as CLIP [19], to convert the text into feature representations. For the structured answers, each action is encoded as a one-hot label. Formally:

$$y_1 = \text{CLIP}(\mathcal{A}_1), \quad y_2 = \text{One-Hot}(\mathcal{A}_2), \tag{2}$$

where y_1 and y_2 each have three components: $y_1 = \{y_c, y_f, y_r\}$ and $y_2 = \{y_{\text{control}}, y_{\text{turn}}, y_{\text{lane}}\}$. Here, y_c, y_f, and y_r are feature vectors of size C, where C is the dimension of text embedding, while y_{control}, y_{turn}, and y_{lane} are three one-hot action labels with size N, where $N_{\text{control}} = 4$, $N_{\text{turn}} = 4$ and $N_{\text{lane}} = 5$, respectively.

Text Feature Alignment. Using the three text features $y_1 = \{y_c, y_f, y_r\}$ as supervision, we develop a feature alignment head that takes the ego feature f_{ego} as input. This setup resembles knowledge distillation, where the alignment head is trained to match the text features provided by the VLM.

In this head, we initialize three learnable text queries, $q_1 = \{q_c, q_f, q_r\}$. Each query interacts with the ego feature f_{ego} via a multi-head cross-attention (MHCA) block, where the text query acts as the attention query q, and the ego feature serves as both the key k and value v, producing updated text queries. These updated queries are then concatenated with the ego feature to form the feature representation for this text head, which is subsequently processed through an MLP layer to produce the final feature alignment output. This process is formulated as:

$$q_1' = \text{MHCA}(q, k, v), \quad q = q_1, k = v = f_{\text{ego}}, \quad \hat{f}_1 = \text{MLP}(q_1' \oplus f_{\text{ego}}), \tag{3}$$

where $\oplus$ denotes concatenation, and $\hat{f}_1 = \{\hat{f}_c, \hat{f}_f, \hat{f}_r\}$ represents three output features to be aligned with the corresponding VLM text features. Note that we use three independent MHCA blocks, one for each component, enabling each text query to focus on specific aspects of the ego feature that can be represented in text form.

Inspired by the knowledge distillation approach DINO [49] that controls the smoothness and sharpness of features, we adopt a similar strategy to normalize the text and output features with different temperature parameters, producing feature distributions rather than raw feature values as follows:

$$P(y_1) = \frac{\exp(y_1/\tau_t)}{\sum_{k=1}^{C} \exp(y_1^{(k)}/\tau_t)}, \quad P(\hat{f}_1) = \frac{\exp(\hat{f}_1/\tau_s)}{\sum_{k=1}^{C} \exp(\hat{f}_1^{(k)}/\tau_s)}, \tag{4}$$

where τ_t and τ_s are temperature parameters that control the sharpness of these distributions. This adjustment enables better alignment between the output features and supervisory labels, enhancing alignment quality for knowledge distillation. Note that we do not apply the centering operation, as we consider the supervision to be ground truth.

Structured Action Classification. We obtain the structured action labels $y_2 = \{y_{\text{control}}, y_{\text{turn}}, y_{\text{lane}}\}$ from the VLM using question Q_2. We then construct another action classification head that takes the ego feature f_{ego} as input. Similar to the previous feature alignment stage, we initialize three learnable action queries, q_{control}, q_{turn}, and q_{lane}, which interact with f_{ego} through three MHCA blocks. In this setup, each action query serves as the attention query q, while the ego feature acts as the key k and value v, producing updated action queries. We then concatenate updated queries with the ego feature to create the representation for the action classification head, passing it through an MLP layer followed by a Softmax to generate the action predictions. This process is formulated as:

$$q_2' = \text{MHCA}(q, k, v), \quad q = q_2, k = v = f_{\text{ego}}, \quad \hat{f}_2 = \text{Softmax}\left(\text{MLP}(q_2' \oplus f_{\text{ego}})\right), \tag{5}$$

where $\hat{f}_2 = \{\hat{f}_{\text{control}}, \hat{f}_{\text{turn}}, \hat{f}_{\text{lane}}\}$ represents the predicted control action, turn action, and lane action. We use independent MHCA blocks for each action query to produce distinct action labels.

3.3 Auxiliary Loss

We define two parallel auxiliary tasks following the planning module to enable the model to distill knowledge from the VLM, and the overall training loss is a weighted sum of two components:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{align}} + \lambda_2 \mathcal{L}_{\text{action}}, \tag{6}$$

where each component corresponds to a distinct auxiliary text head to provide supervision:

$$\begin{aligned} \mathcal{L}_{\text{align}} &= -P(y_c)\log(P(\hat{f}_c)) - P(y_f)\log(P(\hat{f}_f)) - P(y_r)\log(P(\hat{f}_r)) \\ \mathcal{L}_{\text{action}} &= -\sum_{i=1}^{N_{\text{control}}} y^i_{\text{control}} \log(\hat{f}^i_{\text{control}}) - \sum_{i=1}^{N_{\text{turn}}} y^i_{\text{turn}} \log(\hat{f}^i_{\text{turn}}) - \sum_{i=1}^{N_{\text{lane}}} y^i_{\text{lane}} \log(\hat{f}^i_{\text{lane}}) \end{aligned}. \tag{7}$$

For feature alignment, we use cross-entropy loss to align the supervisory and output features, capturing the critical information conveyed by the text. For the action classification task, we also apply cross-entropy loss to ensure accurate classification.

4 Experiments

4.1 Open-Loop Settings

Baselines. Our proposed method is a general framework compatible with various end-to-end autonomous driving methods. We validate its effectiveness by applying it to three widely recognized open-source methods, UniAD [2], VAD [3], and SparseDrive [32]. Additionally, we compare it with VLP [12], which projects ego vehicle ground-truth labels into text feature space via CLIP [19] for contrastive learning.

Dataset. We use the nuScenes dataset [23] for open-loop planning evaluation. nuScenes is a large-scale autonomous driving dataset featuring 1000 scenes, each with a duration of approximately 20 seconds and annotated at 2Hz. The dataset includes detailed annotations, making it a popular benchmark for end-to-end autonomous driving research.

Metrics. We focus on the planning task and use standard metrics such as L2 displacement error and collision rate to evaluate performance for open-loop evaluation.

4.2 Closed-Loop Settings

We further evaluate our proposed method in a closed-loop setting. Following the protocol established by VAD [3], we conduct simulations using the CARLA [25] simulator and the Town05 [50]

Table 1: Planning results of our VLM-AD method and baselines. The best results are shown in bold and the second best are underlined. VLM-AD consistently outperforms the baselines, with the reasoning-focused Q_1 contributing the most significant improvements.

ID	Method	Q_1	Q_2	L2 (m) ↓				Collision Rate (%) ↓				Ckpt. Source
				1s	2s	3s	Avg.	1s	2s	3s	Avg.	
0	UniAD [2]			0.48	0.96	1.65	1.03	0.05	0.17	0.71	0.31	Official
1	UniAD*			0.46	0.96	1.67	1.03	0.11	0.22	0.74	0.36	Reproduced
2	VLP [12]			0.43	0.86	1.47	0.92	**0.03**	0.15	0.48	0.22	Official
3	VLM-AD	✓		0.40	0.83	1.44	0.89	0.05	**0.11**	0.56	0.24	-
4	VLM-AD		✓	0.41	0.87	1.46	0.91	0.06	0.13	0.68	0.29	-
5	VLM-AD	✓	✓	**0.39**	**0.82**	**1.43**	**0.88**	0.05	**0.11**	**0.43**	**0.19**	-
6	VAD-Base [3]			0.41	0.70	1.06	0.72	**0.04**	0.43	1.15	0.54	Official
7	VAD-Base#			0.33	0.59	0.94	0.62	0.19	0.30	0.53	0.34	Reproduced
8	VLP [12]			0.26	0.47	0.78	0.50	0.12	0.17	0.42	**0.23**	Official
9	VLM-AD	✓		**0.24**	0.48	0.76	0.49	0.12	**0.16**	**0.41**	**0.23**	-
10	VLM-AD		✓	0.30	0.50	0.82	0.54	0.16	0.24	0.43	0.28	-
11	VLM-AD	✓	✓	**0.24**	**0.46**	**0.75**	**0.48**	0.12	0.17	**0.41**	**0.23**	-
12	VAD-Tiny [3]			0.46	0.76	1.12	0.78	0.21	0.35	0.58	0.38	Official
13	VAD-Tiny#			0.35	0.62	0.96	0.64	0.12	0.19	0.44	0.25	Reproduced
14	VLM-AD	✓		**0.30**	**0.54**	0.82	**0.55**	**0.08**	**0.15**	**0.38**	**0.20**	-
15	VLM-AD		✓	0.31	0.55	0.88	0.58	0.10	0.18	0.41	0.23	-
16	VLM-AD	✓	✓	**0.30**	**0.54**	**0.80**	**0.55**	0.11	**0.15**	**0.38**	0.21	-
17	SparseDrive-B [32]			0.29	0.55	0.91	0.58	**0.01**	**0.02**	0.13	0.06	Official
18	VLM-AD	✓		0.25	0.50	0.86	0.54	**0.01**	**0.02**	**0.12**	**0.05**	-
19	VLM-AD		✓	0.27	0.52	0.87	0.55	**0.01**	**0.02**	0.13	**0.05**	-
20	VLM-AD	✓	✓	**0.24**	**0.49**	**0.84**	**0.52**	**0.01**	**0.02**	**0.12**	**0.05**	-

benchmark. To ensure a fair comparison, we integrate our method into the VAD framework and adopt the same evaluation definitions for closed-loop evaluation.

Metrics. We use the standard Route Completion (RC) and Driving Score (DS) metrics for closed-loop performance evaluation.

4.3 Implementation Details

We use official codes of the UniAD, VAD, and SparseDrive, adhering to the hyperparameters specified in their official implementations. For our proposed VLM-AD, we define two auxiliary task heads, each containing an MHCA block with 8 heads and 3 cross-attention layers, and we set 3 text queries for each of Q_1 and Q_2. During training, we set the temperature parameters $\tau_s = 0.1$ and $\tau_t = 0.04$ to control the sharpness of the features, and we set $\lambda_1 = 1$ and $\lambda_2 = 0.1$ to balance $\mathcal{L}_{align}$ and $\mathcal{L}_{action}$. All models are trained on 8 NVIDIA H100 GPUs using the PyTorch framework [51]. Complete implementation details, annotation quality analysis, and additional experiments, including visualizations, are provided in the appendix.

4.4 Main Results

Open-Loop Planning Results. Tab. 1 presents the results of applying our proposed VLM-AD to three baselines, UniAD, VAD, and SparseDrive, as well as comparisons with VLP. Comparing methods ID 0 and 1, we achieve nearly identical planning results using the authors' officially trained checkpoints. For methods IDs 6 and 7, and IDs 12 and 13, we observe some discrepancies between our reproduced results and the reported values, which we attribute to a correction in image configuration in the official codebase[2]. From the first section of the table, we observe that VLM-AD significantly outperforms UniAD on both average L2 planning error and average collision rate by introducing Q_1 and Q_2. It also surpasses a state-of-the-art baseline VLP in both metrics. Regarding VAD, our VLM-AD consistently outperforms both VAD-Base and VAD-Tiny, particularly on the

[2] https://github.com/hustvl/VAD/issues/9

Table 3: Ablation study on the contributions of Q_{1-1}, Q_{1-2} and Q_{1-3} to the UniAD baseline. The best is in bold and the second best is underlined.

Method	Q_{1-1}	Q_{1-2}	Q_{1-3}	Q_2	L2 (m) ↓				Collision Rate (%) ↓			
					1s	2s	3s	Avg.	1s	2s	3s	Avg.
UniAD					0.48	0.96	1.65	1.03	<u>0.05</u>	0.17	0.71	0.31
VLM-AD	✓	✓			<u>0.41</u>	0.87	1.53	0.94	0.08	0.17	0.61	0.29
VLM-AD	✓		✓		<u>0.41</u>	<u>0.84</u>	**1.44**	<u>0.90</u>	<u>0.05</u>	**0.11**	**0.48**	**0.21**
VLM-AD		✓	✓		0.42	0.87	<u>1.49</u>	0.93	**0.03**	<u>0.14</u>	<u>0.51</u>	<u>0.23</u>
VLM-AD	✓	✓	✓		**0.40**	**0.83**	**1.44**	**0.89**	<u>0.05</u>	**0.11**	0.56	0.24

L2 planning error metric, and it achieves superior performance compared to the VLP in VAD-Base. For SparseDrive-B, we observe that integrating our VLM-AD leads to improvements in L2 planning error over the baseline model, while maintaining comparable or slightly better collision rates. This is largely due to the collision-aware rescore module in the baseline, which already effectively reduces collision rates. Importantly, the integration of VLM-AD does not compromise this strength, further demonstrating the robustness of our approach. These results demonstrate the effectiveness and advantages of our VLM-AD approach. Additionally, Q_1 yields better results than Q_2, verifying the value of supervising the driving model through rich reasoning information.

Closed-Loop Planning Results. Tab. 2 presents the closed-loop results on the Town05 benchmark for both short and long routes. Our proposed method, VLM-AD, achieves the best performance across all metrics, including Driving Score (DS) and Route Completion (RC). It surpasses VAD-Base and ST-P3, two strong vision-based baselines, with noticeable gains particularly in DS on the long route. These results demonstrate the effectiveness and robustness of VLM-AD in realistic closed-loop driving scenarios, highlighting its ability to reason and act reliably under long-horizon tasks.

Table 2: Closed-loop evaluation results of our VLM-AD method and baselines. The best is in bold and the second best is underlined. †: LiDAR-based method.

Method	Town05 Short		Town05 Long	
	DS ↑	RC ↑	DS ↑	RC ↑
CILRS [52]	7.47	13.40	3.68	7.19
LBC [53]	30.97	55.01	7.05	32.09
Transfuser† [50]	54.52	78.41	<u>33.15</u>	56.36
ST-P3 [1]	55.14	86.74	11.45	<u>83.15</u>
VAD-Base [3]	<u>64.29</u>	<u>87.26</u>	30.31	75.20
+VLM-AD	**67.78**	**88.56**	**35.25**	**84.14**

4.5 Ablation Study

We further analyze the contributions of each sub-question, Q_{1-1}, Q_{1-2}, and Q_{1-3}, within Q_1. Each sub-question provides specific text information related to the ego vehicle's current status, predicted future actions, and reasoning. Tab. 3 presents an ablation study of these three sub-questions. The results indicate that each sub-question positively impacts the overall performance, demonstrating that our designed questions provide valuable information for the planning task. Notably, the reasoning feature contributes the most to reducing the L2 planning error, underscoring its importance in enhancing driving performance. We defer additional ablation studies to Sec. C.

5 Conclusion

In this work, we presented VLM-AD, a novel approach to enhancing end-to-end autonomous driving models by leveraging vision-language models (VLMs) as auxiliary teachers. By integrating VLM-based annotations through targeted questions that include unstructured reasoning text and structured action labels, we enriched the training process with additional reasoning and action supervision. Our method significantly improves planning accuracy and collision rates in the nuScenes dataset, while also achieving higher route completion and driving scores under closed-loop evaluation. Importantly, it does not require VLMs at inference time, making it plug-and-play for real-world deployment without additional inference overhead.

Limitations

One limitation of our work is its focus solely on the planning task, with questions centered on behavior and action. In future work, our approach could be extended to generate relevant annotations for perception or prediction, potentially benefiting the entire autonomous driving pipeline. Additionally, we plan to apply our method to more challenging datasets and design a broader set of questions to extract diverse and useful knowledge from VLMs.

References

[1] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *ECCV*, pages 533–549, 2022.

[2] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, et al. Planning-oriented autonomous driving. In *CVPR*, pages 17853–17862, 2023.

[3] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *ICCV*, pages 8340–8350, 2023.

[4] P. S. Chib and P. Singh. Recent advancements in end-to-end autonomous driving using deep learning: A survey. *IEEE TIV*, 2023.

[5] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE TPAMI*, 2024.

[6] J. Mao, Y. Qian, J. Ye, H. Zhao, and Y. Wang. Gpt-driver: Learning to drive with gpt. In *NeurIPS Workshop*, 2023.

[7] L. Chen, O. Sinavski, J. Hünermann, A. Karnsund, A. J. Willmott, D. Birch, D. Maund, and J. Shotton. Driving with llms: Fusing object-level vector modality for explainable autonomous driving. In *ICRA*, pages 14093–14100, 2024.

[8] B. Jin, X. Liu, Y. Zheng, P. Li, H. Zhao, T. Zhang, Y. Zheng, G. Zhou, and J. Liu. Adapt: Action-aware driving caption transformer. In *ICRA*, pages 7554–7561, 2023.

[9] W. Wang, J. Xie, C. Hu, H. Zou, J. Fan, W. Tong, Y. Wen, S. Wu, H. Deng, Z. Li, et al. Drivemlm: Aligning multi-modal large language models with behavioral planning states for autonomous driving. *arXiv preprint arXiv:2312.09245*, 2023.

[10] X. Tian, J. Gu, B. Li, Y. Liu, C. Hu, Y. Wang, K. Zhan, P. Jia, X. Lang, and H. Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

[11] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K.-Y. K. Wong, Z. Li, and H. Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE RA-L*, 2024.

[12] C. Pan, B. Yaman, T. Nesti, A. Mallik, A. G. Allievi, S. Velipasalar, and L. Ren. Vlp: Vision language planning for autonomous driving. In *CVPR*, pages 14760–14769, 2024.

[13] S. Wang, Z. Yu, X. Jiang, S. Lan, M. Shi, N. Chang, J. Kautz, Y. Li, and J. M. Alvarez. Omnidrive: A holistic llm-agent framework for autonomous driving with 3d perception, reasoning and planning. *arXiv preprint arXiv:2405.01533*, 2024.

[14] K. Ding, B. Chen, Y. Su, H.-a. Gao, B. Jin, C. Sima, W. Zhang, X. Li, P. Barsch, H. Li, et al. Hint-ad: Holistically aligned interpretability in end-to-end autonomous driving. *arXiv preprint arXiv:2409.06702*, 2024.

[15] J.-J. Hwang, R. Xu, H. Lin, W.-C. Hung, J. Ji, K. Choi, D. Huang, T. He, P. Covington, B. Sapp, et al. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024.

[16] J. Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] A. Radford. Improving language understanding by generative pre-training. 2018.

[18] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021.

[20] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pages 12888–12900, 2022.

[21] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pages 19730–19742, 2023.

[22] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *Advances in neural information processing systems*, pages 34892–34916, 2024.

[23] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020.

[24] S. Chen, B. Jiang, H. Gao, B. Liao, Q. Xu, Q. Zhang, C. Huang, W. Liu, and X. Wang. Vadv2: End-to-end vectorized autonomous driving via probabilistic planning. *arXiv preprint arXiv:2402.13243*, 2024.

[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *CoRL*, pages 1–16, 2017.

[26] J.-T. Zhai, Z. Feng, J. Du, Y. Mao, J.-J. Liu, Z. Tan, Y. Zhang, X. Ye, and J. Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv preprint arXiv:2305.10430*, 2023.

[27] Z. Li, Z. Yu, S. Lan, J. Li, J. Kautz, T. Lu, and J. M. Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *CVPR*, pages 14864–14873, 2024.

[28] X. Weng, B. Ivanovic, Y. Wang, Y. Wang, and M. Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *CVPR*, pages 15449–15458, 2024.

[29] W. Zheng, R. Song, X. Guo, C. Zhang, and L. Chen. Genad: Generative end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 87–104, 2024.

[30] Y. Zhang, D. Qian, D. Li, Y. Pan, Y. Chen, Z. Liang, Z. Zhang, S. Zhang, H. Li, M. Fu, et al. Graphad: Interaction scene graph for end-to-end autonomous driving. *arXiv preprint arXiv:2403.19098*, 2024.

[31] D. Zhang, G. Wang, R. Zhu, J. Zhao, X. Chen, S. Zhang, J. Gong, Q. Zhou, W. Zhang, N. Wang, et al. Sparsead: Sparse query-centric paradigm for efficient end-to-end autonomous driving. *arXiv preprint arXiv:2404.06892*, 2024.

[32] W. Sun, X. Lin, Y. Shi, C. Zhang, H. Wu, and S. Zheng. SparseDrive: End-to-end autonomous driving via sparse scene representation. In *ICRA*, 2025.

[33] C. Cui, Y. Ma, X. Cao, W. Ye, and Z. Wang. Receive, reason, and react: Drive as you say, with large language models in autonomous vehicles. *IEEE ITSM*, 2024.

[34] T. Qian, J. Chen, L. Zhuo, Y. Jiao, and Y.-G. Jiang. Nuscenes-qa: A multi-modal visual question answering benchmark for autonomous driving scenario. In *AAAI*, pages 4542–4550, 2024.

[35] Y. Inoue, Y. Yada, K. Tanahashi, and Y. Yamaguchi. Nuscenes-mqa: Integrated evaluation of captions and qa for autonomous driving datasets using markup annotations. In *WACV*, pages 930–938, 2024.

[36] T. Choudhary, V. Dewangan, S. Chandhok, S. Priyadarshan, A. Jain, A. K. Singh, S. Srivastava, K. M. Jatavallabhula, and K. M. Krishna. Talk2bev: Language-enhanced bird's-eye view maps for autonomous driving. In *ICRA*, pages 16345–16352, 2024.

[37] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, P. Luo, A. Geiger, and H. Li. Drivelm: Driving with graph visual question answering. In *ECCV*, 2024.

[38] F. Jia, W. Mao, Y. Liu, Y. Zhao, Y. Wen, C. Zhang, X. Zhang, and T. Wang. Adriver-i: A general world model for autonomous driving. *arXiv preprint arXiv:2311.13549*, 2023.

[39] X. Wang, Z. Zhu, G. Huang, X. Chen, J. Zhu, and J. Lu. Drivedreamer: Towards real-world-driven world models for autonomous driving. *arXiv preprint arXiv:2309.09777*, 2023.

[40] G. Zhao, X. Wang, Z. Zhu, X. Chen, G. Huang, X. Bao, and X. Wang. Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. *arXiv preprint arXiv:2403.06845*, 2024.

[41] A. Keysan, A. Look, E. Kosman, G. Gürsun, J. Wagner, Y. Yao, and B. Rakitsch. Can you text what is happening? integrating pre-trained language encoders into trajectory prediction models for autonomous driving. *arXiv preprint arXiv:2309.05282*, 2023.

[42] P. Wang, M. Zhu, H. Lu, H. Zhong, X. Chen, S. Shen, X. Wang, and Y. Wang. Bevgpt: Generative pre-trained large model for autonomous driving prediction, decision-making, and planning. *arXiv preprint arXiv:2310.10357*, 2023.

[43] D. Fu, X. Li, L. Wen, M. Dou, P. Cai, B. Shi, and Y. Qiao. Drive like a human: Rethinking autonomous driving with large language models. In *WACV*, pages 910–919, 2024.

[44] P. Liu, H. Liu, H. Liu, X. Liu, J. Ni, and J. Ma. Vlm-e2e: Enhancing end-to-end autonomous driving with multimodal driver attention fusion. *arXiv preprint arXiv:2502.18042*, 2025.

[45] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[46] Y. Qiu, Z. Zhao, Y. Ziser, A. Korhonen, E. M. Ponti, and S. B. Cohen. Are large language models temporally grounded? *arXiv preprint arXiv:2311.08398*, 2023.

[47] I. Kesen, A. Pedrotti, M. Dogan, M. Cafagna, E. C. Acikgoz, L. Parcalabescu, I. Calixto, A. Frank, A. Gatt, A. Erdem, et al. Vilma: A zero-shot benchmark for linguistic and temporal grounding in video-language models. *arXiv preprint arXiv:2311.07022*, 2023.

[48] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, pages 24824–24837, 2022.

[49] Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9650–9660, 2021.

[50] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *CVPR*, pages 7077–7087, 2021.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[52] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *ICCV*, pages 9329–9338, 2019.

[53] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *CoRL*, pages 66–75, 2020.

[54] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE TPAMI*, 45(11):12878–12895, 2022.

[55] B. Jaeger, K. Chitta, and A. Geiger. Hidden biases of end-to-end driving models. In *ICCV*, pages 8240–8249, 2023.

[56] Z. Huang, C. Lv, Y. Xing, and J. Wu. Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding. *IEEE Sensors Journal*, 21(10): 11781–11790, 2020.

[57] Z. Li, T. Motoyoshi, K. Sasaki, T. Ogata, and S. Sugano. Rethinking self-driving: Multi-task knowledge for better generalization and accident explanation ability. *arXiv preprint arXiv:1809.11100*, 2018.

[58] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *CVPR*, pages 2174–2182, 2017.

[59] K. Chitta, A. Prakash, and A. Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *ICCV*, pages 15793–15803, 2021.

[60] J. Zhang, Z. Huang, and E. Ohn-Bar. Coaching a teachable student. In *CVPR*, pages 7805–7815, 2023.

[61] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *CoRL*, pages 726–737, 2023.

[62] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *CVPR*, pages 21983–21994, 2023.

[63] H. Shao, L. Wang, R. Chen, S. L. Waslander, H. Li, and Y. Liu. Reasonnet: End-to-end driving with temporal and global reasoning. In *CVPR*, pages 13723–13733, 2023.

[64] K. Ishihara, A. Kanervisto, J. Miura, and V. Hautamaki. Multi-task learning with attention for end-to-end autonomous driving. In *CVPR*, pages 2902–2911, 2021.

[65] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*, volume 35, pages 6119–6132, 2022.

[66] Y. Xu, A. Bazarjani, H.-g. Chi, C. Choi, and Y. Fu. Uncovering the missing pattern: Unified framework towards trajectory imputation and prediction. In *CVPR*, pages 9632–9643, 2023.

[67] Y. Xu and Y. Fu. Sports-traj: A unified trajectory generation model for multi-agent movement in sports. In *ICLR*, 2025.

[68] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, pages 1–18, 2022.

[69] I. Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[70] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[71] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21 (140):1–67, 2020.

[72] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. In *NeurIPS*, pages 16857–16867, 2020.

[73] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

A Related Work

Multi-Task Learning. Multi-task learning (MTL) jointly performs several related tasks using a shared representation through separate branches or heads. This approach leverages shared domain knowledge, enhancing feature robustness and generalization, making it well-suited for end-to-end autonomous driving. In AD systems, auxiliary tasks such as semantic segmentation [54, 55, 56, 57, 58], depth estimation [57, 58], HD mapping, and BEV segmentation [59, 60, 61, 62, 63] are commonly adopted to extract meaningful perception representations for subsequent objects. Beyond vision tasks, prior methods [64, 65] have explored predicting auxiliary signals such as traffic light states and control commands, as well as unifying different trajectory modeling tasks [66, 67], to enhance driving performance. Inspired by the success of multi-task learning, we design novel auxiliary tasks to encourage the model to learn richer feature representations through high-quality reasoning annotations from VLMs, ultimately leading to more reliable planning performance.

B Implementation Details

When integrating our proposed VLM-AD method into UniAD [2][3], we follow the joint training protocol defined in UniAD. In the first stage, we initialize the model using BEVFormer [68] weights and train the perception and mapping tasks for 6 epochs. In the second stage, we freeze the image backbone and BEV encoder and perform end-to-end training using our proposed VLM-AD approach for 20 epochs. The model is trained using an initial learning rate of 2×10^{-4}, a learning rate multiplier of 0.1, and the AdamW optimizer [69] with a weight decay of 0.01.

When integrating our proposed VLM-AD method into VAD [3][4], we adopt the same hyperparameters as in their original implementation. The model is trained using the AdamW optimizer [69] and a cosine annealing scheduler [70], with a weight decay of 0.01 and an initial learning rate of 2×10^{-4}.

When integrating our proposed VLM-AD method into SparseDrive [32][5], we adopt the same hyperparameter settings as in their original implementation. Specifically, we use the SparseDrive-B base model with a ResNet-101 backbone and input image size of 512×1408. The model is trained using the AdamW optimizer [69] and a cosine annealing scheduler [70] in a two-stage training setup, with a learning rate of 3×10^{-4} and a weight decay of 1×10^{-3}.

For closed-loop evaluation, we integrate our method into the VAD-Base framework and follow the same experimental settings described in their paper [3]. Specifically, as in VAD-Base, the navigation information consists of a sparse goal location and a corresponding discrete navigational command, which are encoded using an MLP and passed to the planning head as part of the input features. Additionally, we retain the original traffic light classification branch used in VAD-Base, which comprises a ResNet-50 backbone followed by an MLP-based classification head. The input to this branch is the upper-middle region of the cropped front-view image, and the resulting feature map is flattened and provided to the planning head to help the model interpret traffic light states.

All experimental settings are kept consistent with the corresponding baseline models, as our proposed method is a plug-and-play module that can be seamlessly integrated into various end-to-end driving frameworks without architectural changes or hyperparameter tuning. This design ensures robustness, cross-model compatibility, and fair comparisons under consistent training conditions, while also highlighting the practicality of VLM-AD for real-world deployment.

To encode freeform annotations into text features, we use the pre-trained CLIP-ViT-B/32 [19] model with a dimension of 512. Additionally, we experiment with other text encoders such as T5-base [71] and MPNet-base [72], both encoding freeform annotations into text features with a dimension of 768, as described in Sec. 4.5.

[3] https://github.com/OpenDriveLab/UniAD

[4] https://github.com/hustvl/VAD

[5] https://github.com/swc-17/SparseDrive

Table 4: Results of different variants of VLM-KD. CLIP indicates the use of the contrastive learning loss defined in [19] to align the text features of Q_1. MSE represents the use of MSE loss for feature alignment, KL represents KL divergence, and CosSim indicates cosine similarity for aligning features. Align refers to the alignment loss defined in our method.

Method	Variant	Planning Results	
		Avg. L2 ↓	Avg. Col ↓
UniAD	-	1.03	0.31
VLM-AD	CLIP	0.94	<u>0.26</u>
VLM-AD	MSE	0.99	0.30
VLM-AD	KL	<u>0.92</u>	<u>0.26</u>
VLM-AD	CosSim	0.96	0.28
VLM-AD	Align	**0.89**	**0.24**

Table 5: Results of different designs for VLM-KD. MLP indicates that an MLP layer is used to replace the MHCA block for Q_2. T5 and MPNet indicate the use of different language models to convert reasoning annotations from Q_1 into reasoning features that serve as supervision labels.

Method	Variant	Planning Results	
		Avg. L2 ↓	Avg. Col ↓
UniAD	-	1.03	0.31
VLM-AD	MLP	0.94	0.29
VLM-AD	MHCA	**0.91**	**0.29**
VLM-AD	T5	0.94	0.29
VLM-AD	MPNet	0.91	0.26
VLM-AD	CLIP	**0.89**	**0.24**

Table 6: Results of using different hyperparameters of λ_1 and λ_2 to control the weights of $\mathcal{L}_{align}$ and $\mathcal{L}_{action}$.

Method	Variant	Planning Results	
		Avg. L2 ↓	Avg. Col ↓
UniAD	-	1.03	0.31
VLM-AD	$\lambda_1 = 1, \lambda_2 = 1$	<u>0.90</u>	<u>0.26</u>
VLM-AD	$\lambda_1 = 0.1, \lambda_2 = 1$	0.92	0.29
VLM-AD	$\lambda_1 = 1, \lambda_2 = 0.1$	**0.88**	**0.19**

C Ablation Study

Contrastive Learning with Text Annotations. In addition to our proposed auxiliary task, we employ a contrastive learning strategy similar to VLP. We follow VLP [12] to project our generated annotations from Q_1 using CLIP [19] to obtain three distinct text features. Each text feature is then contrasted with the ego feature individually before we compute the contrastive loss.

The results, presented in Tab. 4, show that contrastive learning improves performance compared to the baseline, though it is still worse than our custom auxiliary tasks. This may be because each frame contains only one ego vehicle, resulting in a single positive sample for contrastive learning that can lead to unstable alignment.

Feature Alignment Loss. We also study alternative options of feature alignment, including minimizing contrastive learning loss in CLIP [19], MSE loss, KL divergence loss [73], or maximizing negative cosine similarity to align the three features from $Q1$. The results, shown in Tab. 4, indicate that MSE loss performs slightly better than UniAD, by minimizing the Euclidean distance between features, driving outputs toward their mean, which causes information loss during training. Both CLIP loss, KL divergence, and cosine similarity outperform UniAD but are inferior to our align-

A_c: The ego vehicle is currently moving forward at a moderate speed.
A_f: The ego vehicle is likely to continue moving forward and follow the curve of the road to the left.
A_r: The road ahead is clear with no visible obstacles or traffic, allowing the vehicle to maintain its current speed and direction. The environment suggests a low-traffic area, and the road's curve indicates the vehicle will naturally follow it to the left.
A_{control}: move slowly
A_{turn}: turn left
A_{lane}: none

(a) Annotation using a single front-view image.

Context: This is an overview image covering the entire 360-degree surroundings of the ego vehicle. The red line indicates the future trajectory, no line suggests stopping or slowing down. When explaining the reasoning, please focus on the camera image and surrounding context rather than referencing the plotted trajectory.
A_c: The ego vehicle is moving forward on a clear road.
A_f: The ego vehicle is likely to turn left.
A_r: The road is clear with no visible obstacles, and the surroundings suggest a typical urban environment. The presence of lane markings and the absence of traffic support the likelihood of a left turn.
A_{control}: move slowly
A_{turn}: turn left
A_{lane}: none

(b) Annotation using all 6 images.

Figure 3: Comparison of annotation strategies: (a) only the front-view image is used; (b) all 6 camera views are included, but the trajectory is still projected onto the front view.

Context: These are 7 consecutive front-view images of the ego vehicle. The first image represents the current time step, while the following 6 images represent the next 6 future time steps.
A_c: The ego vehicle is moving forward along a straight path.
A_f: The ego vehicle will continue to move forward, approaching the speed bump and gate visible in the images.
A_r: The sequence of images shows the vehicle gradually moving closer to the speed bump and gate, indicating forward motion. The road markings and alignment suggest the vehicle is maintaining a straight path. The environment appears controlled, with clear lane markings and a speed limit sign, guiding the vehicle's movement.
A_{control}: go straight
A_{turn}: none
A_{lane}: none

Figure 4: An annotation example using 7 consecutive front-view images as the visual input, which results in incorrect annotations.

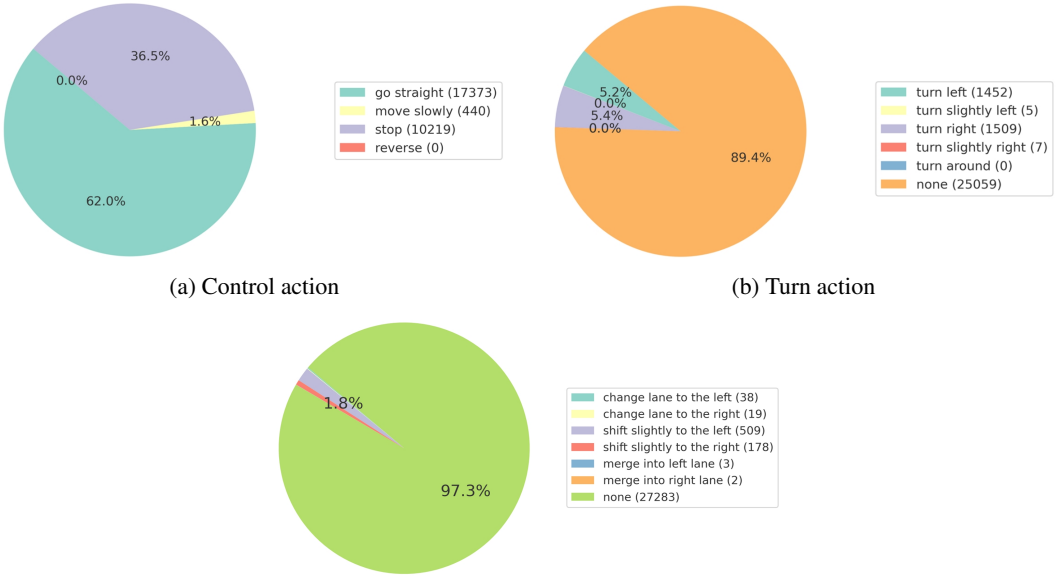
ment loss. This underscores the importance of normalizing with different temperatures to balance the smoothness and sharpness of teacher-student features.

Model Design. We investigate alternative design options in our method. First, we use an MLP layer instead of the MHCA block in our structured action classification head. Second, we study different language models, such as T5 [71], MPNet [72] in addition to CLIP for encoding text annotations from Q_1. From Tab. 5, we observe that MLP achieves slightly worse L2 performance and the same collision rate. Additionally, we observe that both T5 and MPNet outperform the UniAD baseline, but are slightly worse than CLIP.

Hyperparameter Study. Balancing the losses of different tasks is a critical challenge in multi-task learning. We study the hyperparameters λ_1 and λ_2 in the context of UniAD. The results, shown in Tab. 6, indicate that all three variants outperform UniAD. Among these variants, the performance is the worst when $\lambda_1 = 0.1$ and $\lambda_2 = 1$, as the annotations for Q_1 contain more valuable information compared to the annotations for Q_2.

Table 7: Statistics of freeform text annotation $\mathcal{A}_1$ from Q_1.

Word Length	Annotation $\mathcal{A}_1$		
	$\mathcal{A}_c$	$\mathcal{A}_f$	$\mathcal{A}_r$
Max	29	41	93
Min	6	7	13
Mean	11.30	15.34	35.85



(a) Control action

(b) Turn action

(c) Lane action

Figure 5: Distributions of control, turn, and lane actions.

D VLM Annotation

In this section, we provide a comprehensive analysis of our VLM annotations, including visual input format comparisons, annotation statistics, as well as the annotation quality, including representative successful and failure examples.

D.1 Visual Input

While we use a front-view image (as shown in Fig. 3a) as the visual input to the VLM, we also experiment with other alternatives described in Sec. 3, including six images covering the full 360-degree surroundings of the ego vehicle (Fig. 3b) and a sequence of consecutive front-view images (Fig. 4). Compared to the first alternative that uses the full 360-degree surrounding as input, our approach yields similar annotations from the VLM while significantly reducing the computational cost by processing a much smaller input image. The second alternative, which utilizes consecutive front-view images, often results in incorrect annotations, such as misidentifying the current action status and failing to detect a left-turn action. This is due to the challenge VLMs face in understanding temporal dynamics, especially from ego-centric visual signals. Furthermore, using consecutive images increases the annotation time by approximately 80% compared to our approach.

D.2 Annotation Statistics

We annotate the training set of the nuScenes dataset, which consists of 700 scenes and 28,130 frames. Following the methodology described in Sec. 3, we set $T = 6$ to project the ego vehicle's future trajectory onto the front-view image. Consistent with UniAD [2], we exclude samples lacking sufficient input data, resulting in a total of 28,032 annotated samples.

Table 8: Summary of questionnaire results from 5 participants evaluating both freeform annotations and structured action labels. Std represents the standard deviation among the participants.

Participant	Average Score (1-5)			Accuracy (%)		
	$\mathcal{A}_c$	$\mathcal{A}_f$	$\mathcal{A}_r$	$\mathcal{A}_{\text{control}}$	$\mathcal{A}_{\text{turn}}$	$\mathcal{A}_{\text{lane}}$
1	4.58	4.66	4.26	0.88	0.96	0.98
2	4.34	4.26	4.34	0.86	0.92	0.94
3	4.86	4.66	4.54	0.98	0.84	0.96
4	4.12	4.34	4.40	0.80	0.84	0.94
5	4.50	4.62	4.56	0.98	0.96	0.98
Average	4.48	4.51	4.42	0.90	0.90	0.96
Std	0.28	0.19	0.13	0.08	0.06	0.02

For freeform reasoning annotations using Q_1, we calculate the word length of responses for each sub-question (Q_{1-1}, Q_{1-2}, and Q_{1-3}). The statistics are presented in Tab. 7, in which the average response length of $\mathcal{A}_r$ is the longest, as this sub-question focuses on detailed reasoning information.

For structured action annotations using Q_2, we analyze the distribution of three types of actions. The results are shown in Fig. 5a, Fig. 5b and Fig. 5c. Approximately 62% of frames are labeled as "go straight", 89.4% as "no turn action", and 97.3% as "no lane action". Notably, no frames are labeled with "reverse" or "turn around", and only a very small number of frames are labeled as "merge into left lane" or "merge into right lane". These statistics suggest that the nuScenes dataset has limited diversity in driving actions.

An interesting observation is that the VLM occasionally outputs actions not included in our predefined action list. For example, it generates actions such as "turn slightly left", "turn slightly right", "shift slightly to the left", and "shift slightly to the right". In our work, we merge these outputs into our predefined one-hot categories: "turn slightly left" is merged with "turn left", "turn slightly right" with "turn right", "shift slightly to the left" with "change lane to the left", and "shift slightly to the right" with "change lane to the right". This highlights the advantage of using structured annotations, as they help mitigate hallucinations by constraining VLM outputs to predefined categories.

D.3 Annotation Quality

To validate the annotation quality from the VLM, we make a questionnaire with a random sample of 50 cases for evaluation. For each case, participants are provided with the front-view image of the ego vehicle, projected with its future trajectory, along with the corresponding VLM annotations of Q_1 and Q_2.

Participants are then asked to score each response. For freeform reasoning annotations, we set a scoring criterion on a 1 to 5 scale as follows:

- *5 Points*: Highly Consistent.
  - The text description perfectly matches the image.
  - Key elements of the image (e.g., vehicle state, action, reasoning) are accurately described.
  - The text is clear, concise, and complete, with no unnecessary details or contradictions.
- *4 Points*: Mostly Consistent.
  - The text description mostly aligns with the image, with minor inaccuracies or omissions.
  - Key elements are described but may lack some secondary details.
  - Alternatively, the text may contain minor redundancies or slightly unrelated details that don't impact the overall match.
- *3 Points*: Partially Consistent.
  - The text description partially matches the image but has notable inaccuracies or missing details.

- Important aspects of the image (e.g., vehicle speed, road conditions) may be under- or misrepresented.
- There could be some conflicting or vague statements.

- *2 Points*: Mostly Inconsistent.
  - The text description is largely inconsistent with the image but contains a small amount of relevant information.
  - The description fails to capture critical details of the image or includes noticeable inaccuracies.
  - Logical errors or contradictions in the text are present.
- *1 Point*: Completely Inconsistent.
  - The text description does not match the image at all.
  - The text is entirely irrelevant or contradicts the image in significant ways.
  - Misleading information that severely detracts from interpretability.

For structured action annotations, we ask the participants to score True or False for each action.

We evaluated the results from 5 participants, as summarized in Tab. 8. The scores validate the overall annotation quality. Specifically, the annotation $\mathcal{A}_f$, which predicts future actions, received the highest score, while the annotation $\mathcal{A}_r$, which describes reasoning, received the lowest. Additionally, for action annotations, the accuracy for all three action types is 90% or higher, with the lane action achieving the highest accuracy at 96%.

D.4 Succesful Annotation Examples

We present three examples to demonstrate the quality of VLM annotations, as shown in Fig. 6a, Fig. 6b, and Fig. 6c.

In Fig. 6a, the VLM accurately identifies the red traffic light and suggests a stop action at the intersection. It also predicts a reasonable future action and clearly explains the rationale behind its decisions.

In Fig. 6b, a white van is observed ahead of the ego vehicle but in the opposite lane. The VLM correctly assesses that the van will not affect the ego vehicle's movement and outputs appropriate driving actions.

In Fig. 6c, the ego vehicle is stopped at an intersection on a rainy day. Despite low visibility, the VLM successfully identifies the red traffic light and predicts the future movement based on the traffic light's status.

D.5 Imperfect Annotation Examples

We also present three annotation failure cases, illustrated in Fig. 7a, Fig. 7b, and Fig. 7c.

In Fig. 7a, the VLM accurately recognizes that the traffic light is green and predicts a future right-turn action from its reasoning annotation. However, it incorrectly outputs a left-turn action from the action annotation. Since we query Q_1 and Q_2 separately, the response to Q_1 does not influence Q_2. One potential solution is to introduce additional prompts to establish a progressive questioning process toward more accurate action annotations.

In Fig. 7b, the VLM outputs "stop" or "move slowly" as the ego vehicle's current status. While these outputs are plausible, they are inconsistent with the ground truth, as the projected future trajectory indicates that the ego vehicle is currently turning right at the intersection. On the other hand, the action annotation successfully predicts the correct future action.

In Fig. 7c, the VLM mistakenly identifies the red pedestrian light as a traffic light and provides incorrect responses.

Overall, despite making occasional mistakes, the VLM demonstrates the ability to generate meaningful annotations that uncover the underlying rationale behind driving decisions, bringing benefits to our proposed method as validated by our experiments. By querying two independent annotation questions, our method is robust to VLM mistakes that often appear in only one of the two responses, as seen in Fig. 7a and Fig. 7b. We defer to obtaining more accurate VLM responses as future work to further advance the performance of E2E planning models.

E Qualitative Examples

We present a total of eight qualitative examples: Fig. 8, Fig. 9, Fig. 10, Fig. 11, Fig. 12, Fig. 13, Fig. 14, and Fig. 15 to compare the planning results of our proposed method against UniAD.

In Fig. 8, Fig. 10, Fig. 11, Fig. 12 and Fig. 13, the planning trajectories generated by UniAD are winding, lack smoothness, and fail to stay within the center of the lane. In contrast, our method produces trajectories that are significantly smoother and stay within the lane boundaries.

Similarly, in Fig. 9, Fig. 10, Fig. 11, Fig. 14, and Fig. 15, the commands generated by UniAD are incorrect, as the ego vehicle is moving forward. However, our action head successfully predicts the correct actions in these scenarios.

These qualitative examples highlight the capability of VLM-AD to produce smoother and more accurate planning trajectories in challenging driving scenarios, while also providing enhanced interpretability.

$\boldsymbol{A_c}$: The ego vehicle is currently stopped or moving very slowly at the intersection.
$\boldsymbol{A_f}$: The ego vehicle will remain stopped until the traffic light turns green and the pedestrian has safely crossed the street.
$\boldsymbol{A_r}$: The red traffic light and the pedestrian crossing in front of the vehicle indicate that it is necessary for the ego vehicle to stop to ensure safety and compliance with traffic rules.
$\boldsymbol{A_{control}}$: stop
$\boldsymbol{A_{turn}}$: none
$\boldsymbol{A_{lane}}$: none

(a) A successful annotation example where the red traffic light is correctly identified, and the annotations are complete and informative.

$\boldsymbol{A_c}$: The ego vehicle is likely moving forward at a moderate speed.
$\boldsymbol{A_f}$: The ego vehicle will continue to move forward along the road.
$\boldsymbol{A_r}$: The road ahead is clear, and there are no visible obstacles or traffic signals that would necessitate stopping. The white van is not in the direct path of the ego vehicle, allowing it to proceed safely.
$\boldsymbol{A_{control}}$: go straight
$\boldsymbol{A_{turn}}$: none
$\boldsymbol{A_{lane}}$: none

(b) A successful annotation example where the white van is correctly identified as irrelevant to the ego vehicle's driving.

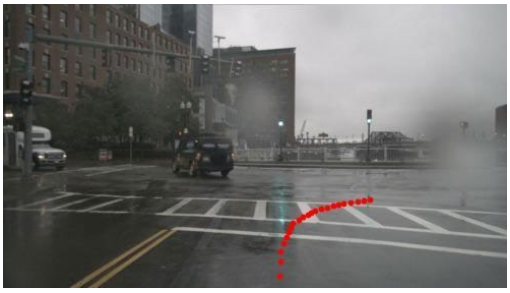
$\boldsymbol{A_c}$: The ego vehicle is currently stopped or moving very slowly.
$\boldsymbol{A_f}$: The ego vehicle will likely remain stopped until the traffic light turns green, at which point it may proceed cautiously.
$\boldsymbol{A_r}$: The red traffic light indicates that the vehicle should stop to comply with traffic regulations. The wet road conditions also suggest that the vehicle should proceed with caution once the light changes.
$\boldsymbol{A_{control}}$: stop
$\boldsymbol{A_{turn}}$: none
$\boldsymbol{A_{lane}}$: none

(c) A successful annotation example where the rainy condition and red traffic light are correctly identified, resulting in high-quality annotations.

Figure 6: Qualitative examples of successful VLM-based annotations.

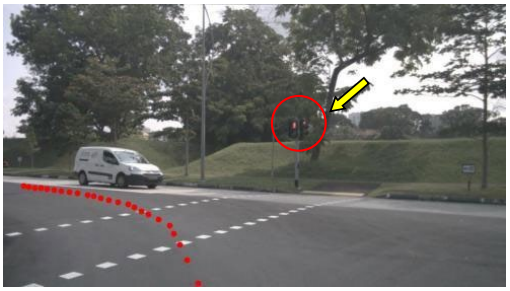
$\boldsymbol{A_c}$:The ego vehicle is currently stopped or moving very slowly at an intersection.
$\boldsymbol{A_f}$: The ego vehicle is likely to proceed forward and make a slight right turn.
$\boldsymbol{A_r}$: The traffic light is green, indicating it is safe to proceed. The road ahead is clear, with no visible obstacles or traffic, allowing the vehicle to move forward and follow the road's curve to the right.
$\boldsymbol{A_{control}}$: go straight
$\boldsymbol{A_{turn}}$: turn left
$\boldsymbol{A_{lane}}$: none

(a) An imperfect annotation example where reasoning annotations are accurate but the turn action is incorrectly annotated as "turn left".



$\boldsymbol{A_c}$: The ego vehicle is currently stopped or moving very slowly at the intersection.
$\boldsymbol{A_f}$: The ego vehicle is likely to proceed through the intersection, making a right turn.
$\boldsymbol{A_r}$: The vehicle is positioned at a stop line, indicating it is waiting for a safe opportunity to proceed. The intersection appears clear, and the vehicle is aligned to make a right turn, which is a common maneuver at intersections when clear.
$\boldsymbol{A_{control}}$: move slowly
$\boldsymbol{A_{turn}}$: turn right
$\boldsymbol{A_{lane}}$: none

(b) An imperfect annotation example where an overly cautious annotation is generated, despite the ego vehicle continuing without stopping or slowing down.



$\boldsymbol{A_c}$: The ego vehicle is likely stopped or moving very slowly at the intersection.
$\boldsymbol{A_f}$: The ego vehicle will remain stopped or proceed very slowly until the traffic light changes to green.
$\boldsymbol{A_r}$: The red traffic light indicates that the ego vehicle must stop to allow cross traffic to pass safely. The presence of the van crossing the intersection further supports the need for the ego vehicle to wait until it is safe to proceed.
$\boldsymbol{A_{control}}$: stop
$\boldsymbol{A_{turn}}$: none
$\boldsymbol{A_{lane}}$: none

(c) An imperfect annotation example where the pedestrian traffic light is mistakenly considered, leading to an incorrect stop annotation.

Figure 7: Qualitative examples of imperfect VLM-based annotations.

(a) UniAD.

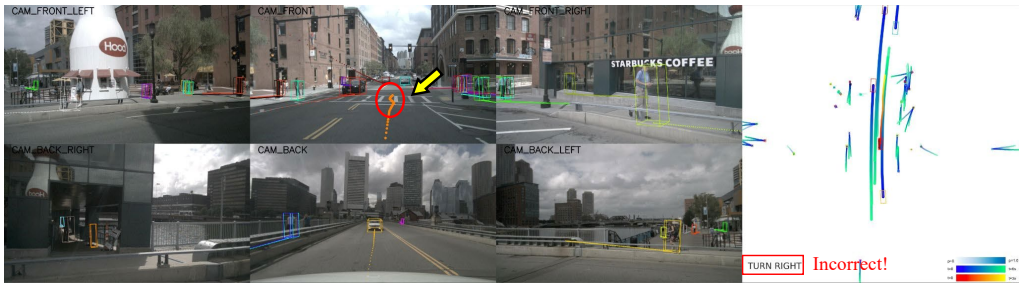
(b) VLM-AD (Ours).

Figure 8: Our method generates a smooth trajectory for an evening driving scenario, in contrast to the baseline method that predicts a winding trajectory.

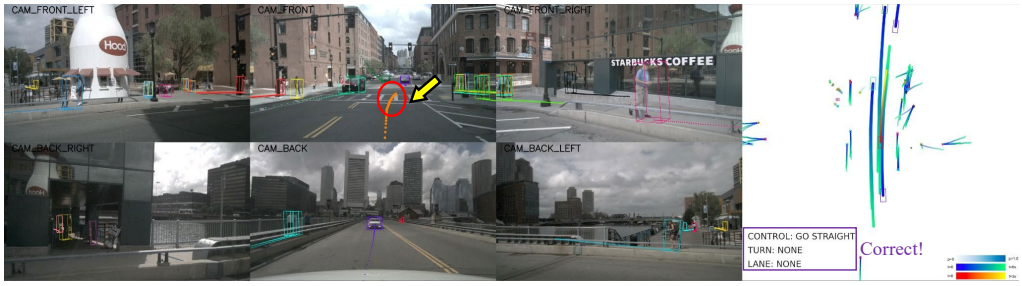
(a) UniAD.

(b) VLM-AD (Ours).

Figure 9: Our method accurately predicts the correct actions when driving on a curved road, while the command generated by UniAD is incorrect.



(a) UniAD.



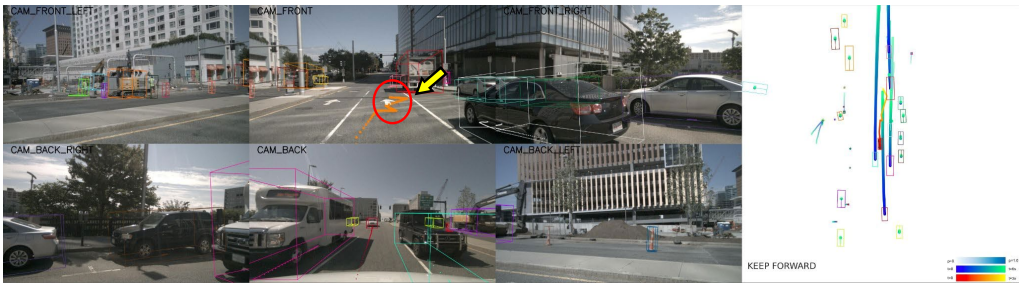
(b) VLM-AD (Ours).

Figure 10: Our method predicts accurate actions and a smooth trajectory in an urban driving scenario, in contrast to the baseline method that predicts a winding trajectory based on an incorrect command.

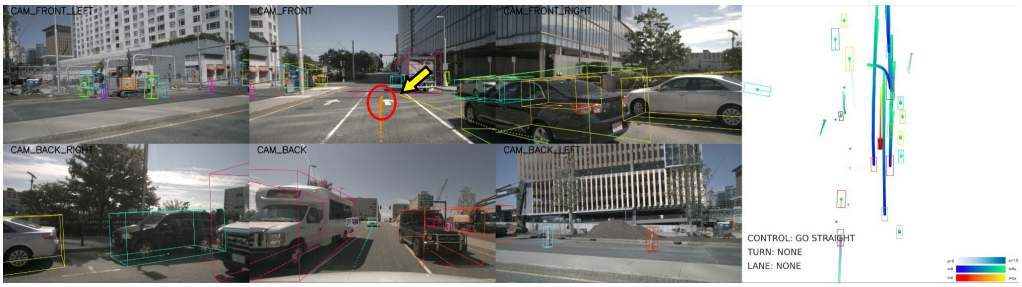
(a) UniAD.

(b) VLM-AD (Ours).

Figure 11: Our method predicts accurate actions and a smooth trajectory in rainy conditions, in contrast to the baseline method that predicts a winding trajectory based on an incorrect command.



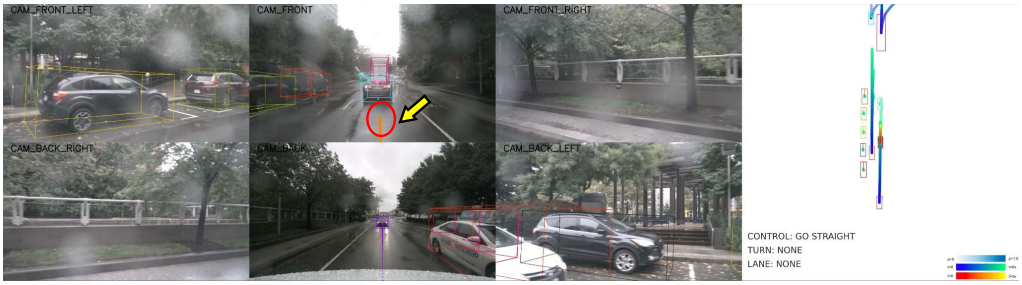
(a) UniAD.



(b) VLM-AD (Ours).

Figure 12: Our method predicts an accurate trajectory that stays within the lane boundaries, in contrast to the baseline method that predicts a zigzagging trajectory.

(a) UniAD.



(b) VLM-AD (Ours).

Figure 13: Our method predicts a smooth trajectory that stays within the lane boundaries, in contrast to the baseline method that predicts a swerving trajectory.

(a) UniAD.

(b) VLM-AD (Ours).

Figure 14: The baseline model predicts an accurate trajectory but is based on an incorrect command, while our method predicts both accurate actions and a precise future trajectory.

(a) UniAD.

(b) VLM-AD (Ours).

Figure 15: The baseline model predicts an accurate trajectory but is based on an incorrect command, while our method predicts both accurate actions and a precise future trajectory.