LeapVAD: A Leap in Autonomous Driving via Cognitive Perception and Dual-Process Thinking

Yukai Ma, *Graduate Student Member, IEEE*, Tiantian Wei, Naiting Zhong, Jianbiao Mei, Tao Hu, Licheng Wen, Xuemeng Yang, Botian Shi, and Yong Liu

***Abstract*—While autonomous driving technology has made remarkable strides, data-driven approaches still struggle with complex scenarios due to their limited reasoning capabilities. Meanwhile, knowledge-driven autonomous driving systems have evolved considerably with the popularization of visual language models. In this article, we propose LeapVAD, a novel method based on cognitive perception and dual-process thinking. Our approach implements a human-attentional mechanism to identify and focus on critical traffic elements that influence driving decisions. By characterizing these objects through comprehensive attributes—including appearance, motion patterns, and associated risks—LeapVAD achieves more effective environmental representation and streamlines the decision-making process. Furthermore, LeapVAD incorporates an innovative dual-process decision-making module mimicking the human-driving learning process. The system consists of an analytic process (System-II) that accumulates driving experience through logical reasoning and a heuristic process (System-I) that refines this knowledge via fine-tuning and few-shot learning. LeapVAD also includes reflective mechanisms and a growing memory bank, enabling it to learn from past mistakes and continuously improve its performance in a closed-loop environment. To enhance efficiency, we develop a scene encoder network that generates compact scene representations for rapid retrieval of relevant driving experiences. Extensive evaluations conducted on two leading autonomous driving simulators, CARLA and DriveArena, demonstrate that LeapVAD achieves superior performance compared with camera-only approaches despite limited training data. Comprehensive ablation studies further emphasize its effectiveness in continuous learning and domain adaptation. Project page: https://pjlab-adg.github.io/LeapVAD/**

***Index Terms*—Autonomous driving, dual-process, knowledge-driven.**

Received 10 January 2025; revised 26 June 2025 and 5 October 2025; accepted 21 October 2025. This work was supported by NSFC under Grant 62088101, for Autonomous Intelligent Unmanned Systems. *(Corresponding authors: Botian Shi; Yong Liu.)*

Yukai Ma, Jianbiao Mei, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: yukaima@zju.edu.cn; jianbiaomei@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Tiantian Wei is with the TUM School of Engineering and Design, Technical University of Munich, 80333 Munich, Germany (e-mail: tiantian.wei@tum.de).

Naiting Zhong is with Tongji University, Shanghai 200092, China (e-mail: 2233590@tongji.edu.cn).

Tao Hu is with the Science Island Branch of the Graduate School, University of Science and Technology of China, Hefei 230026, China (e-mail: ht_simon@mail.ustc.edu.cn).

Licheng Wen, Xuemeng Yang, and Botian Shi are with Shanghai Artificial Intelligence Laboratory, Shanghai 200232, China (e-mail: wenlicheng@pjlab.org.cn; yangxuemeng@pjlab.org.cn; shibotian@pjlab.org.cn).

Digital Object Identifier 10.1109/TNNLS.2025.3626711

I. Introduction

SINCE the early 21st century, humans have been exploring the use of computer algorithms to replace human drivers. Recent data-driven approaches [1], [2] have achieved impressive results but remain highly dependent on training data distribution. This often yields shallow semantic understanding and misinterpretations in complex or unfamiliar scenarios. Lacking inferential ability, they merely generalize observed patterns, limiting performance to annotated data. For example, faced with anomalies like a plane on the road, they cannot make proper decisions. Hence, systems capable of reasoning beyond training data and imitating human cognition are urgently needed. Knowledge-based methods [3], [4] leverage large language and vision-language models as driving agents, representing a step toward advanced autonomy. Current evaluation methods such as open-loop testing do not capture how agents interact with the environment [5]. As a result, responsiveness and adaptability remain insufficiently assessed, underscoring the need for more comprehensive evaluation.

Human learning to drive involves continuous interaction in closed-loop environments, where decisions are made based on surroundings and feedback is received. According to dual-process theory [6], [7], human intelligence operates on two systems: 1) System-I (i.e., heuristic process, which is characterized by being fast, automatic, and empirical) and 2) System-II (i.e., analytic process, which is characterized by being slow, rational, and logical). This dual-process thinking is evident as novice drivers transition to experienced ones. Initially, they depend on common sense, but with training, they develop skills through trial and error and rational analysis (analytic process), leading to muscle memory that allows for quick, instinctive reactions in familiar situations (heuristic process). Even after obtaining a driver's license, individuals continue to learn from experiences and accidents to improve their driving skills.

Recent work, such as DriveVLM [4], has explored integrating Visual Language Models (VLMs) and data-driven planning pipelines to emulate this dual-process paradigm for autonomous driving. Contrary to the mentioned techniques, our method draws from attention mechanisms and observational learning, and decision-making in human driving. It utilizes a memory bank for experience storage and replay in a closed-loop scenario, enabling continuous learning via memory and reflection mechanisms. The dual-process analogy not only offers an intuitive interpretation of LeapVAD but also provides a principled foundation for integrating imme-

diate decision-making with long-term reflective learning in autonomous driving systems.

The initial work LeapAD [8] introduced a dual-process driving system inspired by the human attention mechanism. This system aimed to achieve knowledge-driven autonomous driving with limited training data, emulating the human ability to learn from experience and refine skills over time. While the previous method demonstrated promising performance, it had several limitations, such as supporting only single-frame image inputs and lacking precise motion prediction for traffic participants. To overcome these challenges, we propose an enhanced version named LeapVAD. Specifically, the algorithm has been extended to support multiview and multiframe inputs. Furthermore, we enhance the scene encoder to extract scene tokens that are more closely related to driving actions, thus improving the accuracy of the overall system.

Building on the dual-process continuous learning framework, this article presents the following new contributions.

1) *Temporal Scene Understanding:* We extend our approach to handle multiframe inputs, which provide a more comprehensive understanding of key traffic objects by offering attributes such as velocity and motion trends across multiple frames.
2) *Efficient Retrieval Mechanism:* We propose a scene encoder that effectively captures scene tokens pertinent to anticipated driving actions, facilitating the acquisition of few-shot samples. Compared with text embedding, our method achieves higher precision in encoding scene tokens.
3) *Knowledge Transferability in Domain Adaptation:* LeapVAD utilizes analytic process and reflection mechanisms to compile a memory bank of transferable experiences. Our results show that the knowledge stored in this memory bank can be applied not only to various towns in simulation within the same domain but also to high-fidality environments, such as high-fidelity environments.
4) *Extended Validation:* We evaluate our approach under more diverse setups and scenarios. The driving score (DS) on the CARLA [9] Town05 short and long benchmarks increases by 5.3% and 42.6%, over our previous results [8]. Moreover, we achieved the best performance on DriveArena [10] by utilizing the memory bank selected in CARLA.

Algorithm 1 LeapVAD

Input:
System Prompt P_s
Reflection Prompt P_r
Traffic Rules P_t
Images $\mathbf{I}$ from ego car
Current ego state $\mathbf{A}$
Navigation information N from LimSim [26]

Output:
Decisions S for LimSim [26]

ANALYTIC PROCESS (SECTION III-E)
for $i <$ length(sequence) **do**
  $D_i \leftarrow \text{VLM}(P_s, \mathbf{I}_i)$ // Section III-C
  $R_i, S_i \leftarrow \text{GPT}(P_t, D_i, N_i, \mathbf{A}_i)$ // reasoned decisions
  $\mathbf{t}_i \leftarrow \mathbf{E}(\mathbf{I}_i, \mathbf{A}_i)$ // Section III-D
  $\mathbf{B} \leftarrow \mathbf{B} \cup \{\mathbf{t}_i, D_i, R_i, S_i\}$ // update memory bank
  send S_i // send meta actions to low-level controller
end for
$\mathbf{B} \leftarrow \text{revise}(\mathbf{B})$ // check and update
HEURISTIC PROCESS (SECTION III-F)
for $j <$ length(sequence) **do**
  $D_j \leftarrow \text{VLM}(P_s, \mathbf{I}_j)$
  $\mathbf{t}_j \leftarrow \mathbf{E}(\mathbf{I}_j, \mathbf{A}_j)$
  $\{\mathbf{t}_i, D_i, R_i, S_i\}_{i=0}^{k-1} \leftarrow \text{TopK}(\boldsymbol{B}, k)$ // top-k samples
  $R_j, S_j \leftarrow \text{LLM}(P_t, \{D_i, R_i, S_i\}_{i=0}^{k-1}, D_j, N_j, \mathbf{A}_j)$
  $\mathbf{Q} \leftarrow \mathbf{Q} \cup \{\mathbf{t}_j, D_j, R_j, S_j\}$ // historical queues
  if length($\mathbf{Q}$) $> m$ **then**
    dequeue($\mathbf{Q}$) // remove oldest element
  end if
  send S_j
end for
REFLECTION MECHANISM (SECTION V-E4)
if detect_accident() = True **then**
  $O \leftarrow$ get_accident_info()
  $\mathbf{t}_n, D_n, R_n, S_n \leftarrow \text{GPT}(P_r, O, \mathbf{Q})$ // Section V-E4
  $\mathbf{B} \leftarrow \mathbf{B} \cup \{\mathbf{t}_n, D_n, R_n, S_n\}$ // update memory bank
end if

II. Related Works

A. Evolution of Vision Language Models

Built on large language models (LLMs) such as LLaMA [11] and Vicuna [12], a wide range of VLMs [13], [14], [15], [16], [17], [18] has emerged, extending their capabilities to multimodal understanding. These models have introduced innovative pretraining and fine-tuning techniques to bridge the gap between vision and language and raise extraordinary emergent abilities, typically including instruction following [19], in-context learning [13], and chain-of-thought [18] on multimodal data. Typically, VLMs use Q-former-like [17], [14] or multi-layer perceptron (MLP)-like [18], [19] connectors for modal alignment, trained in three stages: pretraining, instruction-tuning, and optional alignment tuning. For instance, LLaVA [16] and MiniGPT-4 [15] employ instruction tuning for interactive visual agents, while Qwen-VL [17] uses multistage training for multilingual and fine-grained comprehension. InternVL [18] introduces progressive alignment for multimodal understanding, significantly advancing vision-language integration.

B. Leveraging Foundation Models for Autonomous Driving

VLMs have been successfully adapted via supervised fine-tuning (SFT) for downstream vision tasks [20]. In autonomous driving, studies leverage foundation models for their world knowledge and reasoning [3], [21], [22]. For instance, DriveLM [23] uses VLMs for interpretable planning via graph-based reasoning, while DriveMLM [24] generates decisions from human instructions in simulations. ELM [25] specializes in embodied driving understanding, and Retrieval-Augmented Generation (RAG)-Driver [3] enhances prediction with retrieval-augmented generation. Recent work like DriveVLM-Dual [4] integrates VLMs into planning pipelines for real-world deployment.

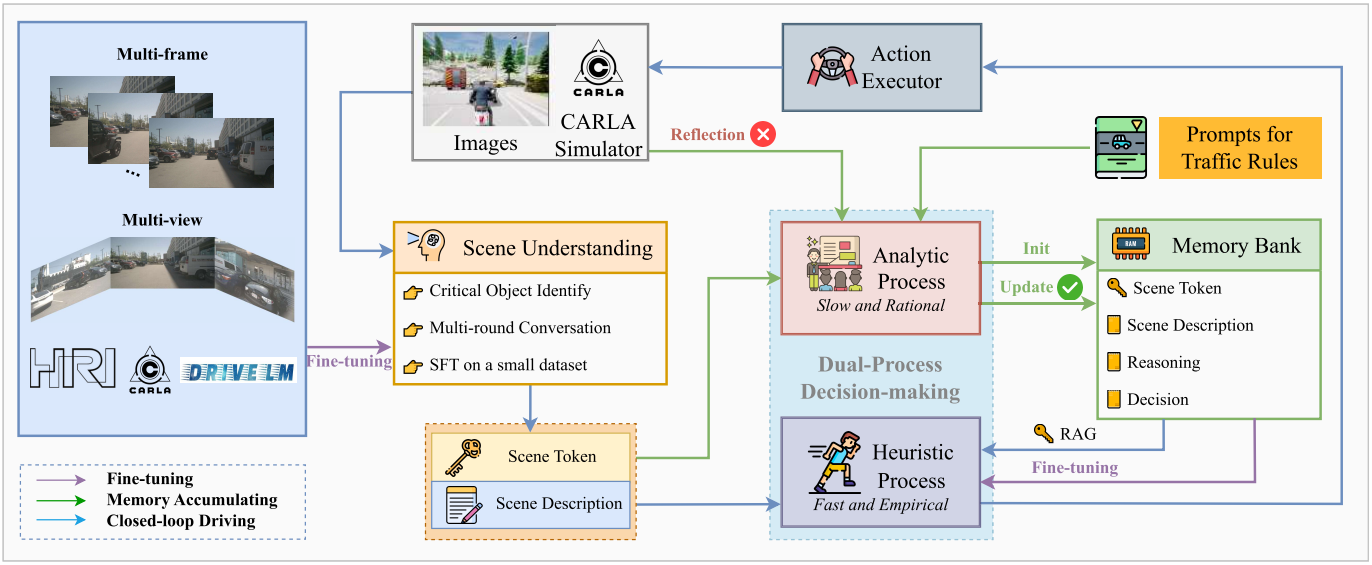


Fig. 1. Architecture of LeapVAD consists of two primary modules: scene understanding and dual-process decision-making. The scene understanding module analyzes multiview or multiframe images, identifying critical objects and generating a scene token. This token serves as a characteristic representation of the current scene. The dual-process decision-making module then uses this scene description and the guidance of traffic rules to make reasoning and decisions. These decisions are converted into control signals to navigate the ego car in the simulator. Specifically, analytic process accumulates an initial memory bank used to train heuristic process and updates it, especially when heuristic process encounters accidents. Heuristic process leverages scene tokens to efficiently retrieve the most relevant historical scenarios from this memory bank, enabling rapid and informed driving decisions. For visual clarity in Fig. 1, we color-code the three key operational phases: fine-tuning (purple), memory accumulating (green), and close-loop driving (blue).

In contrast to the approaches mentioned above, we draw inspiration from human driving behaviors. Through the dual-process system, the memory bank, and the reflection mechanism, we implement the storage, updating, and transfer of experiences, enabling continuous exploration, learning, and improvement in closed-loop scenarios.

C. Transitioning From Data-Driven to Knowledge-Driven Autonomous Driving

Data-driven methods [27], [28] have achieved success in autonomous driving but struggle with distribution shifts and long-tail scenarios [29], [30]. In contrast, knowledge-driven approaches that use reasoning and continuous learning, like humans leveraging common sense, are crucial [31], [32]. Foundation models (LLMs/VLMs) excel at reasoning and knowledge application for driving tasks [21], [33], drawing increasing research interest [25], [34].

Recent work has proposed a more integrated paradigm: vision-language-action model (VLA) [35] models that fuse camera streams, natural language instructions, and low-level actuation into a single policy. Compared with these approaches, this article focuses on the continual learning of the VLM component.

III. Methodology

A. Overview

Our proposed LeapVAD framework consists of four main components: the VLM for scene understanding (see Section III-C), scene encoder for extracting scene token (see Section III-D), a dual-process decision-making module consisting of the analytic process (see Section III-E) and the heuristic process (see Section III-F), which operates in conjunction with a controller detailed in Appendix B.

B. System Pipeline

Fig. 1 and Algorithm 1 illustrate LeapVAD, our knowledge-driven system that employs a fine-tuned VLM to analyze multiframe images and describe key objects in the closed-loop simulator. The scene encoder creates scene tokens based on the current image and vehicle state. These object descriptions and scene tokens are passed to the dual-process decision-making module (comprising analytic process and heuristic process), which performs scenario reasoning to generate driving decisions and corresponding reasoning. Initially, analytic process's outputs are used to construct the memory bank and fine-tune heuristic process. During testing, heuristic process makes driving decisions, sending high-level actions to the action executor to generate control signals, while analytic process is reserved for generating reflections when accidents occur through the automatic reflection mechanism for self-updating and continuous improvement.

C. Scene Understanding With VLM

Human drivers focus on key objects around the vehicle to avoid information overload, improve reaction times, and reduce cognitive load. This strategy enhances concentration and lowers accident risks. Inspired by this, the scene understanding module in LeapVAD selectively identifies critical objects, simplifying the environment's description and simplifying the decision-making process.

Off-the-shelf foundation VLMs often lack domain-specific knowledge for driving. To address this, we perform SFT and prompt the VLMs to generate detailed descriptions of objects that may impact driving decisions. These descriptions include semantic, spatial, and motion attributes and behavioral reasoning. By integrating these elements, the system achieves a more

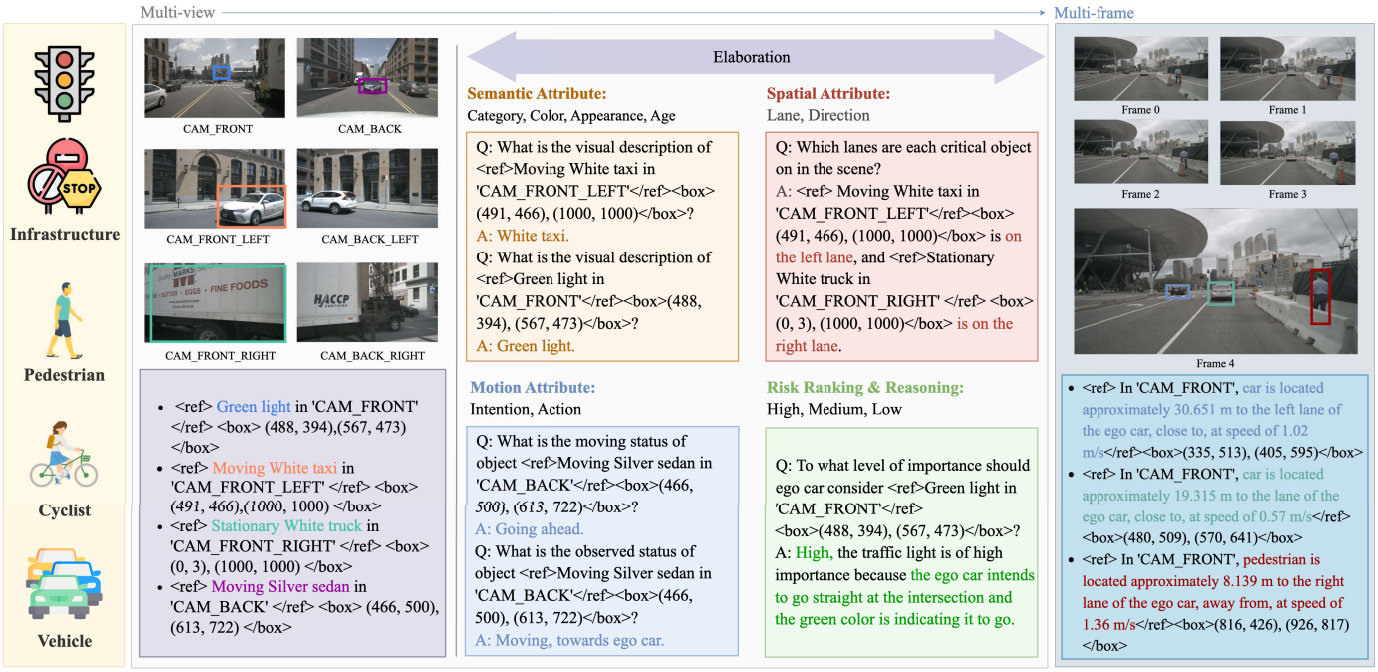


Fig. 2. We create a dataset for instruction learning in VLM derived from DriveLM [23], Rank2Tell [36], and CARLA [9]. This dataset can be categorized into two types: multiview and multiframe. The multiview annotations include a summary and elaboration, while the multiframe annotations solely consist of a summary. Compared with multiview annotations, the multiframe annotations provide additional information such as exact velocity and motion trends.

comprehensive understanding of the environment, improving safety and adaptability in complex driving scenarios.

To enhance the scene understanding capability of the VLM in autonomous driving, we develop a summary-elaboration data structure for SFT, as shown in Fig. 2. The data comprises two parts: multiview and multiframe. Notably, the summary is used only in the closed loop, while elaboration is reserved for training. For example, the descriptions generated by the VLM are represented as $\boldsymbol{D} = \{D_{s,i}, D_{l,i}, D_{m,i}, D_{r,i}\}_{i=0}^{N-1}$, where N denotes the number of critical objects. In multiview construction, each object includes the following.

1) *Semantic Attribute D_s:* Describes the object's category, typically important traffic participants (e.g., vehicles and cyclists), and infrastructure (e.g., traffic lights and stop signs).
2) *Spatial Attribute D_l:* Indicates its bounding box, lane position, and distance from the ego car, essential for safety and collision avoidance.
3) *Motion Attribute D_m:* Refers to the object's motion direction.
4) *Behavioral Reasoning D_r:* Explains the object's significance and influence on the ego car's driving decisions. For instance, a stop sign on the right is crucial when the ego car is going straight, as it indicates the need to stop at the intersection.

For multiframe data, dynamic attributes of objects within the scene are included in the summary, as illustrated in the right part of Fig. 2. Specifically, the VLM describes the location, motion trends, distance, and velocity of critical objects by analyzing the video data. In addition, it provides the bounding boxes of these objects in the final frame. Combined with the ego speed and the navigation info from LimSim, the final scene descriptions are fed into the decision-making module. Refer to Appendix E for the example of the format of scene descriptions.

D. Scene Token

To facilitate retrieving similar scenes in the memory bank for few-shot prompting, we propose a more efficient and precise method for extracting scene tokens. Differing from the previous work LeapAD [8], our LeapVAD uses scene tokens instead of text embeddings to avoid the ambiguity, synonyms, and paraphrasing inherent in language-based scene descriptions, leading to more stable and accurate performance. In this article, LeapVAD generates meta-actions for steering and speed regulation, mirroring the primary control mechanisms employed by human drivers. Our key insight is that scenes requiring similar human control responses (regarding steering and braking) can be considered analogous at the control level. Building upon this observation and inspired by ACO [37], we develop a comparative learning approach that operates in two distinct spaces: the action (ACT) space for steering and the acceleration (ACC) space for braking. This dual-space comparative learning framework enables us to derive scene tokens that capture the underlying control similarities.

1) Scene Encoder: As illustrated in Fig. 3, given a batch of image $\boldsymbol{I} \in \mathbb{R}^{B\times H\times W\times 3}$ and the ego state $\boldsymbol{A} \in \mathbb{R}^{B\times 9}$, we process the input through the scene encoder $\boldsymbol{E}$ to derive the scene token $\boldsymbol{t} \in \mathbb{R}^{B\times 256}$, which encompasses $\boldsymbol{g}_{\text{act}} \in \mathbb{R}^{B\times 128}$ and $\boldsymbol{g}_{\text{acc}} \in \mathbb{R}^{B\times 128}$. Following MoCo [38], we stabilize the feature distribution and reduce the gap between current and cached scene tokens during training. To achieve this, we introduce a momentum-updated scene encoder $\boldsymbol{E}_m$, which produces a feature vector $\boldsymbol{t}' \in \mathbb{R}^{B\times 256}$. The updated rule of the encoder's parameters is

$$\theta_{\boldsymbol{E}_m} \leftarrow \alpha\theta_{\boldsymbol{E}_m} + (1-\alpha)\theta_{\boldsymbol{E}} \tag{1}$$

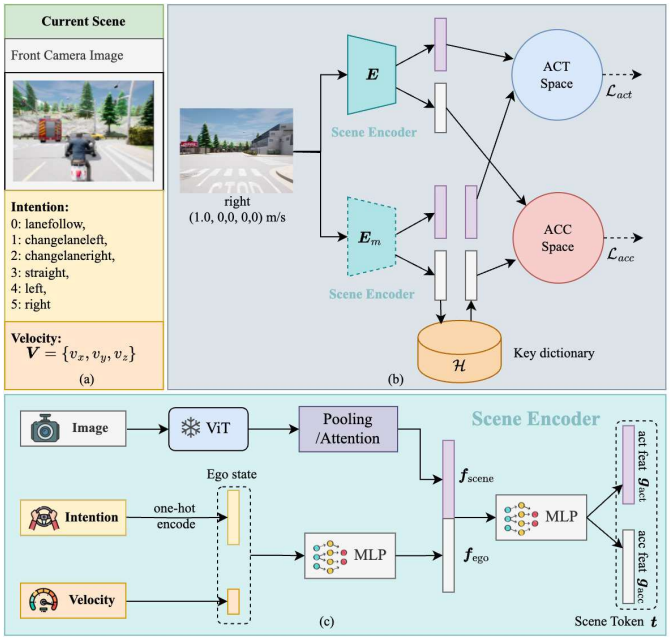


Fig. 3. Training pipeline of our scene encoder is outlined as follows. (a) Details about the input data. (b) How we form both ACT and ACC for the input images and update the model using contrastive loss in these two spaces. (c) Architecture of the scene encoder.

where α is the momentum coefficient and $\theta_{\boldsymbol{E}_m}, \theta_{\boldsymbol{E}}$ are the parameters of $\boldsymbol{E}_m$ and $\boldsymbol{E}$. Notably, $\boldsymbol{E}_m$ is used exclusively during training, while $\boldsymbol{E}$ is employed for retrieving few-shot samples in closed-loop testing scenarios.

Specifically, the intent is derived through one-hot encoding. The features are concatenated with ego velocity $\boldsymbol{V}$ via an MLP to generate the ego state features $\boldsymbol{f}_{\text{ego}} \in \mathbb{R}^{B\times 256}$. The image is processed to extract ViT features $\boldsymbol{f}_{\text{img}} \in \mathbb{R}^{B\times N\times C}$, which are then compressed into the scene features $\boldsymbol{f}_{\text{scene}} \in \mathbb{R}^{B\times 256}$ using either max-pooling or attention mechanisms. The final scene token $\boldsymbol{t}$ is produced by concatenating $\boldsymbol{f}_{\text{ego}}$ and $\boldsymbol{f}_{\text{scene}}$ after applying the MLP.

2) Key Dictionary: Following MoCo [38], we use a key dictionary $\mathcal{H}$ to store the historical encoded scene token $\boldsymbol{t}'$ to enable a larger contrastive batch size. In this study, we utilize the scene encoder $\boldsymbol{E}_m$ to generate the scene token $\boldsymbol{t}'$. This process establishes positive pairs with the current training features. It ensures that these tokens are stored in the key dictionary, $\mathcal{H}$, to form positive and negative sample pairs in subsequent training iterations. When the size of $\mathcal{H}$ exceeds its predetermined capacity, the samples within the dictionary are systematically replaced.

3) Traning Loss: During training, a batch of images and their corresponding ego states are sampled. Following data augments, these images are encoded by both $\boldsymbol{E}$ and $\boldsymbol{E}_m$ to obtain the query features $\boldsymbol{g}_{\text{sp}} \in \mathbb{R}^{B\times 128}$ and key features $\boldsymbol{g}'_{\text{sp}} \in \mathbb{R}^{B\times 128}$. Simultaneously, N key features are sampled from both $\mathcal{H}$ and $\boldsymbol{g}'_{\text{sp}}$ to form the key set $\boldsymbol{K}$. The losses $\mathcal{L}_{\text{act}}$ and $\mathcal{L}_{\text{acc}}$ are then computed as

$$\mathcal{L}_{\text{sp}} = -\log \frac{\sum_{z^+\in P_{\text{sp}}(\boldsymbol{g}_{\text{sp}})} \exp(\boldsymbol{g}_{\text{sp}} \cdot z^+/\tau)}{\sum_{z^-\in N_{\text{sp}}(\boldsymbol{g}_{\text{sp}})} \exp(\boldsymbol{g}_{\text{sp}} \cdot z^-/\tau)} \tag{2}$$

where $P_{\text{sp}}(\boldsymbol{g}_{\text{sp}})$ and $N_{\text{sp}}(\boldsymbol{g}_{\text{sp}})$ are the positive and negative keys of query $\boldsymbol{g}_{\text{sp}}$ in "sp" space, respectively. They are represented as

$$\begin{aligned} P_{\text{sp}}(\boldsymbol{g}_{\text{sp}}) &= \{z \,|\, \|\hat{a}'_{\text{sp}} - \hat{a}_{\text{sp}}\| < \sigma_{\text{sp}}, (z, \hat{a}'_{\text{sp}}) \in \boldsymbol{K}\} \\ N_{\text{sp}}(\boldsymbol{g}_{\text{sp}}) &\equiv \boldsymbol{K} \setminus P_{\text{sp}}(\boldsymbol{g}_{\text{sp}}) \\ &\text{sp} \in \{\text{act}, \text{acc}\} \end{aligned} \tag{3}$$

where $\hat{a}_{\text{sp}}$ denotes the label of the query feature $\boldsymbol{g}_{\text{sp}}$, and $\hat{a}_{\text{sp}}$ represents the label of the key feature z, indicating the steering and braking values in the ACT and ACC spaces, respectively. The set $N_{\text{sp}}(\boldsymbol{g}_{\text{sp}})$ is obtained by removing the positive pair $P_{\text{sp}}(\boldsymbol{g}_{\text{sp}})$ from the candidate set $\boldsymbol{K}$. In addition, σ_{sp} is the distance threshold used in the "sp" space.

The overall contrastive loss of the scene token is

$$\mathcal{L} = \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{acc}}\mathcal{L}_{\text{acc}} \tag{4}$$

where λ_{act} and λ_{acc} are hyperparameters used to adjust the weights of $\mathcal{L}_{\text{act}}$ and $\mathcal{L}_{\text{acc}}$.

E. Analytic Process

Leveraging scene descriptions offered by the VLM, we develop the analytic process, a framework designed to emulate the logical reasoning processes characteristic of human drivers. The analytic process utilizes logical reasoning to navigate complex scenarios, employing structured analysis and rational decision-making to ensure safety in driving tasks.

Through extensive pretraining on diverse datasets, LLMs inherently accumulate a vast repository of world knowledge, equipping them to address complex problems with nuanced reasoning and understanding [22]. This capability meets the demand of the analytic process, which relies on thorough analysis and contextual awareness to make informed decisions in driving scenarios. Our analytic process leverages the world knowledge embedded in LLMs to interpret scene descriptions and produce high-quality driving decisions. Empirical results indicate that incorporating specific traffic rules, as outlined in Appendix D, further enhances the system's safety and reliability in real-world driving scenarios. Moreover, we combine the VLM with the analytic process to conduct closed-loop experiments, enabling the collection of high-quality decision-making data and outcomes. These results, stored as "experience" in a memory bank, are incrementally accumulated and can be effectively transferred to the heuristic process. This transfer empowers the heuristic process to leverage prior experience for rapid response in analogous scenarios, as elaborated in Section III-F.

1) Reflection Mechanism: The analytic process is employed to facilitate reflection on traffic accidents, as demonstrated at the bottom of Algorithm 1. Precisely, in a closed-loop driving scenario integrating the VLM and heuristic process, the occurrence of any accident O activates a reflective mechanism. The analytic process subsequently analyzes the scene description $\boldsymbol{D}$, reasoning $\boldsymbol{R}$, and decision $\boldsymbol{S}$ from the historical frames $\boldsymbol{Q}$ preceding the incident. This analysis identifies causal factors, detects errors, and proposes corrective reasoning R_n alongside decision-making strategies S_n. A detailed example of the reflection process is provided in Appendix E4.

The insights derived from this reflection procedure are subsequently integrated into the system's memory bank,

enabling LeapVAD to learn continuously from past failures and make progressively better-informed decisions in future driving scenarios. Notably, the accumulated experience in the memory bank exhibits strong transferability and generalization, allowing it to be directly applied to other lightweight models and easily adapted to diverse scenarios, as discussed in Section IV-D6.

F. Heuristic Process

1) LLM: Although the analytic process excels at providing precise driving reasoning and decisions through its detailed analysis and thorough evaluation, it is inherently slow processing often results in duplicated and redundant efforts, limiting its practicality in real-world driving scenarios. Drawing inspiration from human driving behavior, where drivers develop muscle memory through repeated practice that enables efficient reactions with minimal cognitive load, we introduce a heuristic process within LeapVAD incorporating a lightweight language model.

To enable effective knowledge transfer, we apply SFT using the data accumulated in the memory bank, as described in Section III-E. This process distills knowledge from the analytic process into the lightweight model, allowing the heuristic process to adapt its behavior to diverse scenarios while operating significantly faster (approximately five times faster in our experiments). Our previous results [8] reveal that the lightweight model without SFT fails to produce reliable driving decisions.

2) Few-Shot Prompting: We utilize few-shot prompting [8], [22] to enhance the heuristic process's generalization for unseen scenes and reduce hallucinations, leading to more robust decisions. This approach allows the heuristic process to effectively draw on insights from its memory bank, improving the accuracy of future driving decisions.

Following Section III-D, we can obtain the scene token $\boldsymbol{t}_{\mathrm{q}}$ for the current scene. Subsequently, the cosine similarity between $\boldsymbol{t}_{\mathrm{q}}$ and the scene tokens $\{\mathbf{t}_i\}_{i=0}^{M-1}$ in the memory bank with the size of M is calculated by

$$\mathrm{cosine}(\boldsymbol{t}_{\mathrm{q}}, \mathbf{t}_{\mathrm{i}}) = \frac{\boldsymbol{t}_{\mathrm{q}} \cdot \mathbf{t}_{\mathrm{i}}}{||\boldsymbol{t}_{\mathrm{q}}||||\mathbf{t}_{\mathrm{i}}||}. \tag{5}$$

We select the top-k samples with the highest similarity scores as queried scenes. The scene descriptions $\{D_i\}_{i=0}^{k-1}$, reasoning $\{R_i\}_{i=0}^{k-1}$, and decisions $\{S_i\}_{i=0}^{k-1}$ of these samples, along with the current scene description D_c, are input into the heuristic process for final reasoning R_c and decision S_c.

IV. Experiments

A. Data Preparation

1) Data for VLM: We constructed an instruction-following dataset for the SFT of VLM by integrating data collected from Rank2Tell [36], DriveLM [23], and CARLA [9]. We standardized the referring format for key objects as follows: `<ref>In camera view, properties</ref><box>coordinates</box>`. Each dataset was processed with the original labels to obtain unique question-answer pairs. Overall, the dialog is constructed summary-elaborated, as shown in Fig. 2. The elaboration includes four aspects of key object properties: semantic attributes, spatial attributes, motion attributes, and importance. Apart from multiview dialogs, we added summaries for multiframe images, introducing properties such as distance, speed, and motion trends of key objects. We constructed dialogs using five frames, with Rank2Tell operating at 10 Hz and the others at 2 Hz. We collected 5K of multiview summary data and 2K of multiframe summary data from CARLA Towns 01–04, 06, 07, and 10 to train the VLM.

2) Data for Heuristic Process: We accumulate experience in a closed-loop setting and store it in a memory bank by integrating analytic process and VLM, which serves as few-shot examples for subsequent SFT and heuristic process prompting. Our approach also incorporates a dynamic updating mechanism to address issues encountered by heuristic process, as detailed in Section III-E. It is worth noting that these samples are collected in a closed-loop environment without human intervention. As with LeapAD [8], we used a memory bank size of 18.1K in Table I and used it as the default in this article. Notably, LeapVAD achieved superior performance while utilizing less data. In this context, "L" and "C" represent light detection and ranging (LiDAR) and camera modalities. Key abbreviations: "L" = LiDAR, "C" = Camera; "DD" = Data-driven, "KD" = Knowledge-driven; "OE" = OpenAI Embedding, "ST" = Our Scene Token. The method marked with "†" utilizes InternVL2-8B [18] as the VLM, and the method indicated by "*" employs analytic process. Specifically, the memory bank consists of 18k experiences collected from CARLA Towns 01–04, 06, 07, and 10 using analytic process at 2 Hz over three rounds, along with 0.1k reflection experiences from Town05. We apply K-means clustering to organize these experiences within the memory bank.

3) Data for Scene Encoder: We utilized data collected from CARLA and the nuScenes [39] (train set) to train the scene encoder. Specifically, we collected 90K driving frames from the CARLA simulator, of which 70% were designated for training. Each frame includes an image captured by the front-view camera and data on intent, velocity, steering, and braking. Steering values are normalized within the $[-1, 1]$ range, whereas braking values range from $[0, 1]$. We use steering and braking as labels for the ACT and ACC spaces, respectively, and set the distance threshold σ_{sp} ($\mathrm{sp} \in \{\mathrm{act}, \mathrm{acc}\}$) to 0.04.

B. Implementation Details

According to Section IV-D1, we utilize Qwen-VL-7B [17] and InternVL2-8B [18] as the VLM for scene understanding, employ GPT-4o as the slow system (analytic process) for reasoning and logic, and use Qwen1.5-1.8B [40] as the fast system (heuristic process) for automatic and rapid thinking in LeapVAD. The scene encoder $\boldsymbol{E}$ implements "pooling + state" to extract the scene token, where the input includes the ego state, and max pooling is used to extract $\boldsymbol{f}_{\mathrm{scene}}$, as demonstrated in Table II.

To fully exploit the capabilities of our VLM in closed-loop driving experiments, we conducted SFT with instruction-based data as discussed in Section IV-A1. For LLaVA-1.5-7B [48], we employed the AdamW optimizer [49] with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate followed a cosine decay schedule, initialized at 2×10^{-4}. Training

TABLE I

COMPARISON OF OUR LEAPVAD WITH COMPETITIVE METHODS ON TOWN05 SHORT BENCHMARK.

Row	Method	Modality	Type	Annotations	Retrieval	Reflection	DS ↑	RC ↑
01	InterFuser [41]	L+C	DD	3M	-	-	**94.95±1.91**	**95.19±2.57**
02	TransFuser [42]	L+C	DD	228K	-	-	54.52±4.29	78.41±3.75
03	VAD [28]	C	DD	228K	-	-	64.30	87.30
04	NEAT [43]	C	DD	130K	-	-	58.70±4.11	77.32±4.91
05	Roach [44]	C	DD	-	-	-	65.26±3.63	88.24±5.16
06	WOR [45]	C	DD	1M	-	-	64.79±5.53	87.47±4.68
07	LBC [46]	C	DD	157K	-	-	30.97±4.17	55.01±5.14
08	CILRS [47]	C	DD	720K	-	-	7.47±2.51	13.40±1.09
09	LeapAD* [8]	C	KD	11K	-	-	81.31±2.37	94.22±3.18
10	LeapAD [8]	C	KD	11K	OE	×	75.73±1.36	92.10±1.44
11	LeapAD [8]	C	KD	11K	OE	✓	83.11±0.28	94.98±0.54
12	**LeapVAD***	C	KD	41K	-	×	86.55±3.12	97.19±0.42
13	**LeapVAD**	C	KD	41K	OE	×	82.58±3.04	96.68±1.54
14	**LeapVAD**	C	KD	41K	ST	×	83.78±2.01	99.42±0.03
15	**LeapVAD**	C	KD	41K	ST	✓	**88.19±2.98**	**99.53±0.17**
16	**LeapVAD†**	C	KD	41K	ST	✓	80.26±2.46	98.80±0.67

TABLE II

PRECISION@1 FOR VARIOUS SCENE TOKENS ON NUSCENES [39]

Method	Ego state	Precision@1	
		steer(%)↑	brake(%)↑
LeapAD [8]		60.39	76.62
Scene Encoder(pooling)		82.69	83.55
Scene Encoder(pooling)	✓	**83.20**	**87.52**
Scene Encoder(attention)	✓	82.22	85.81

TABLE III

COMPARISON OF OUR LEAPVAD WITH COMPETITIVE METHODS IN TOWN05 LONG BENCHMARK

Method	Modality	Type	Annotations	DS (%) ↑	RC (%) ↑	IS (%) ↑
DriveMLM [24]	L+C	DD	2M	**76.1**	**98.1**	**78.0**
ThinkTwice [50]	L+C	DD	2M	70.9	95.5	75.0
InterFuser [41]	L+C	DD	3M	68.3	95.0	72.0
TransFuser [42]	L+C	DD	228K	31.0	47.5	77.0
VAD [28]	C	DD	228K	30.3	75.2	-
TCP [51]	C	DD	420K	57.2	80.4	73.0
NEAT [43]	C	DD	130K	37.7	62.1	61.0
Roach [44]	C	DD	-	43.6	80.4	54.4
WOR [45]	C	DD	1M	44.8	82.4	54.0
LBC [46]	C	DD	157K	7.1	32.1	22.1
CILRS [47]	C	DD	720K	3.7	7.2	51.4
LeapAD [8]	C	KD	11K	51.7	**100**	51.7
LeapVAD	C	KD	41K	**73.7**	95.7	**78.0**

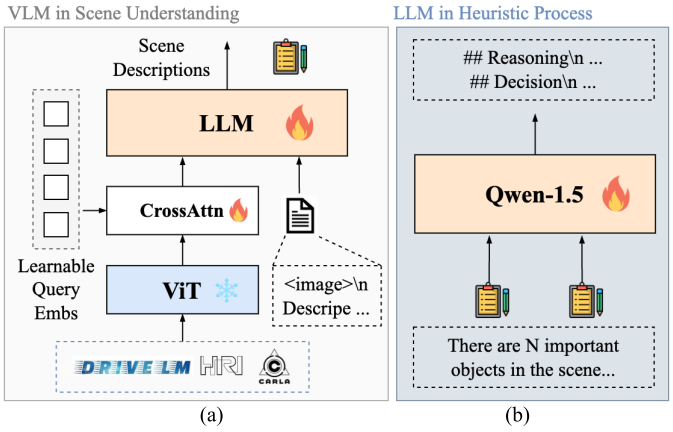


Fig. 4. Illustration depicts the fine-tuning process. (a) Fine-tuning of the VLM using 4.1K instruction-following data points for scene understanding. (b) Utilization of the collected samples in the memory bank to fine-tune Qwen-1.5, employed in the heuristic process model.

was conducted with a batch size of 8 × 8 NVIDIA A100 GPUs, completing 5 epochs in approximately 10 h. For Qwen-VL-7B [17], we utilized the AdamW optimizer [49] with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We combined this with a cosine decay of the learning rate, initially set to 1×10^{-5}. The batch size was set to 8, and the model was trained for 5 epochs on 8 A100 GPUs, taking approximately 69 h. For InternVL2-8B [18], we employed the same optimizer with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The initial learning rate was set to 4×10^{-5} and decayed using a cosine schedule. Similarly, the batch size was 8, and the model was trained for 5 epochs on 8 A100 GPUs, requiring around 7 h. The input image resolution was set at 448 × 448 pixels.

For heuristic process, we conducted SFT on Qwen1.5-1.8B for 5 epochs using samples stored in the memory bank, taking about 6 h. The training hyperparameters are consistent with VLM's training procedure. Fig. 4 shows the detailed fine-tuning process. The dual-process decision module outputs meta-actions (e.g., "ac," "dc," "*LCL*," "*LCR*," "IDLE," and "STOP") at a frequency of 2 Hz, which are further refined into control signals, as detailed in Appendix B.

For the scene token, we trained the scene encoder on our custom training set, consisting of 63K CARLA data samples and the nuScenes dataset, for 12 epochs, taking approximately 1 h. We utilized a batch size of 128 and the SGD optimizer with hyperparameters set as follows: $\lambda_{act} = \lambda_{acc} = 1$, a learning rate of 0.03, a weight decay of 10^{-4}, and an $\mathcal{H}$ size of 4096. For image augmentation, we employed DriveArena [10] to generate two batches with aspect ratios of 336 × 600 and

TABLE IV

COMPARISON OF OUR LEAPVAD WITH END-TO-END AUTONOMOUS DRIVING ALGORITHMS IN DRIVEARENA [10]

Method	DA sing_route_1			DA sing_route_2			DA boston_route_1			DA boston_route_2			Avg.		
	PDMS↑	RC↑	ADS↑	PDMS↑	RC↑	ADS↑	PDMS↑	RC↑	ADS↑	PDMS↑	RC↑	ADS↑	PDMS↑	RC↑	ADS↑
VAD	53.15	4.67	2.48	51.47	4.00	2.06	58.30	6.04	3.52	50.54	3.66	1.85	53.36±3.46	4.59±1.05	2.48±0.74
UniAD	76.15	16.84	12.82	72.15	16.9	8.75	49.52	9.1	4.50	68.88	12.1	8.35	66.68±11.82	13.74±3.82	8.60±3.40
LeapVAD + InternVL2 (Mem.)	78.92	**98.52**	77.75	90.67	**56.79**	51.49	72.39	14.36	10.40	87.92	44.00	38.69	82.48±8.39	**53.42±34.93**	44.58±27.99
LeapVAD + QwenVL	81.03	98.51	79.82	94.23	51.44	48.47	82.53	18.51	15.28	84.48	12.08	10.21	85.57±5.95	45.14±39.54	38.45±32.38
LeapVAD + QwenVL (Mem.)	**82.33**	98.44	**81.05**	**95.70**	54.42	**52.08**	**85.41**	**18.66**	**15.94**	**93.19**	**35.42**	**33.01**	**89.16±6.31**	51.74±34.39	**45.52±27.90**

TABLE V

VALIDATION STUDY ON SCENE UNDERSTANDING

Dataset	Method	Precision↑	Recall↑	F1 Score↑	ROUGE↑	GPT Score↑
Rank2Tell [37]	llava-1.5-7b [48]	28.49	25.22	26.75	69.26	65.20
	qwen-vl2-7b [17]	46.70	37.37	41.52	70.23	66.59
	internvl2-8b [18]	45.67	36.94	40.84	62.71	63.60
CarlaSim	llava-1.5-7b [48]	34.95	32.00	33.41	78.04	58.09
	qwen-vl-7b [17]	51.41	47.14	49.18	83.24	63.01
	internvl2-8b [18]	53.55	44.80	48.79	73.77	58.13

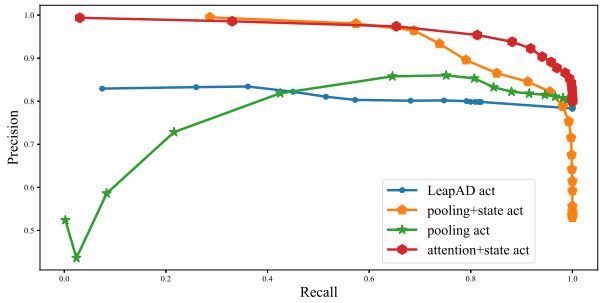
448 × 800, respectively, while ensuring consistency with the nuScene dataset in terms of layouts and boxes. In Fig. 3, "pooling" refers to the application of max pooling to generate the scene feature $\boldsymbol{f}_{\text{scene}}$ from ViT features, whereas "attention" denotes the use of a learnable weight matrix to create the scene feature $\boldsymbol{f}_{\text{scene}}$ from ViT features. The training procedure for this knowledge-driven model is detailed in the Appendix.

C. Evaluation in Closed-Loop Driving

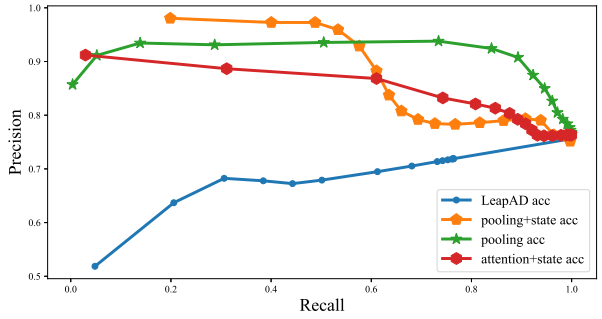
1) CARLA Leaderbord: We conduct closed-loop experiments in CARLA, a popular open-source simulator, to evaluate the performance of our LeapVAD. To assess our method's effectiveness, we perform a detailed evaluation in a closed-loop driving environment using the Town05 benchmark. We utilize several metrics: DS, route completion (RC), and infraction score (IS). RC represents the percentage of the route completed by the agent, while IS reflects the penalties from accidents. The final metric, DS, is calculated by combining RC and IS, allowing us to evaluate the driving performance on a given route.

Specifically, we provide five different configurations to thoroughly evaluate our method, as shown in the highlighted rows in Table I. From top to bottom, these configurations are: 1) closed-loop experiments using the analytic process directly; 2) LeapVAD with OpenAI embedding [52] as the scene token and a memory bank without reflection; 3) using our scene encoder to extract the scene token and a memory bank without reflection; 4) employing our scene encoder for scene token extraction and a memory bank with reflection; and 5), which utilizes InternVL2-8B [18] as the VLM, while the rest remains consistent with *iv)*. For additional VLM ablation studies, refer to Section IV-D1.

As shown in Table I, our LeapVAD outperforms all camera-only methods, including the previous LeapAD approach. Notably, our technique outperforms even TransFuser [42], which utilizes LIDAR sensor input. While our method shows slightly lower performance compared with InterFuser [41], another multisensor approach, it is noteworthy that InterFuser



(a)



(b)

Fig. 5. Precision–recall curves on nuScenes [39] dataset. (a) ACT space. (b) ACC space.

requires a substantially larger dataset with approximately 73 times more camera and LiDAR annotations. The comparison between entries in rows 09 and 12, as well as rows 10 and 13, shows the performance improvement of our proposed temporal module over LeapAD [8], fully confirming the superiority of employing multiframe data for analysis. Replacing the previous OpenAI embedding with our designed scene encoder (as seen in rows 13 and 14) yields notable improvements in DS and RC, providing empirical support for the scene encoder's effectiveness. We further substantiate the advancements of the scene encoder in Section III-D of the ablation experiment. Finally, the comparison between rows 15 and 12 highlights the efficacy of our dual-process decision module, with LeapVAD achieving superior DS and RC scores.

In addition, we present the evaluation results on the more challenging Town05 Long benchmark in Table III. The results indicate that LeapVAD significantly improves over the previous LeapAD and outperforms all other vision-only methods. Notably, LeapVAD's performance is comparable to that of DriveMLM, despite the latter being trained with LiDAR and 48 times more data. It is also worth noting that LeapVAD adopts a more cautious driving policy. As a result, compared

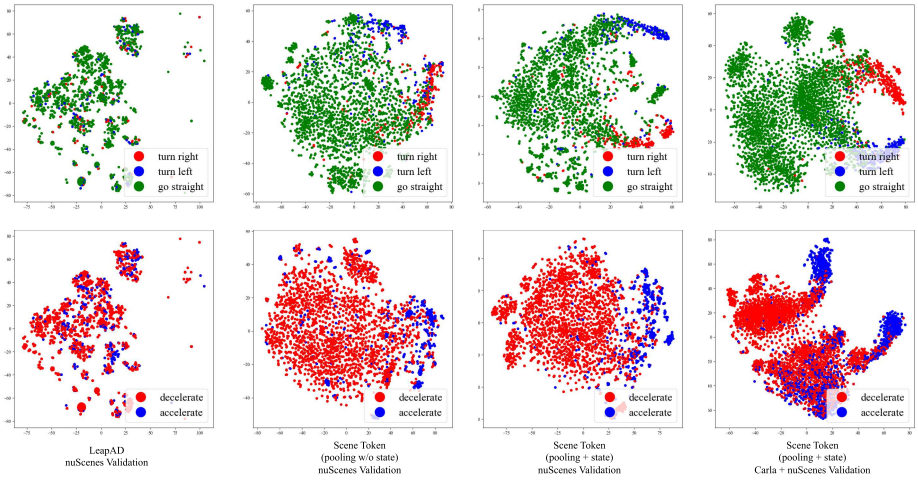


Fig. 6. t-SNE [54] visualization of extracted features. The first row illustrates the relationship between $\boldsymbol{g}_{\text{act}}$ and the steering, whereas the second row depicts the correlation between $\boldsymbol{g}_{\text{acc}}$ and the brake. The category of the scene token is indicated at the bottom of each respective column.

with LeapAD, it occasionally exceeds the allowed time limit, leading to a lower RC score.

2) DriveArena: We conducted closed-loop experiments in DriveArena [10], a high-fidelity closed-loop simulation platform, to evaluate LeapVAD's driving performance in a simulation environment resembling real-world conditions. These experiments were performed on four predefined routes, with two selected in Boston and two in Singapore. The evaluation utilized three key metrics: predictive driver model score (PDMS), RC, and Arena DS (ADS). PDMS, initially introduced by NAVSIM [53], evaluates the quality of the predicted trajectory at each time step. ADS, as defined in DriveArena [10], combines PDMS with RC, where RC follows the same definition as the CARLA Leaderboard.

The closed-loop performance of LeapVAD and end-to-end autonomous driving algorithms is summarized in Table IV. Notably, LeapVAD generates meta-actions, unlike other algorithms that directly output trajectories. We evaluated LeapVAD under three distinct configurations in DriveArena [10]: 1) LeapVAD + InternVL2 (Mem.), where InternVL2 serves as the VLM for scene understanding, and the system utilizes a memory bank; 2) LeapVAD + QwenVL, where QwenVL is used as the VLM for scene understanding, and the system operates without a memory bank; and 3) LeapVAD + QwenVL (Mem.), where QwenVL is employed as the VLM, and the system incorporates a memory bank.

As demonstrated in Table IV, our knowledge-driven approach exhibits robust performance in high-fidelity simulation environments. We evaluated the LeapVAD with InternVL2 and QwenVL for scene understanding. To demonstrate the effectiveness of the memory bank, we also tested the LeapVAD with and without the memory bank module in DriveArena [10]. The calculated PDMS, RC, and ADS are presented in percentages. In particular, the memory bank accumulated from the CARLA simulator works effectively when transferred to another simulator. This shows the strong generalization capability of our approach, enabling adaptation to diverse environmental conditions without excessive reliance on training data.

D. Ablation Study

We conduct extensive ablation studies about the number of few shots, size of the memory bank, reflection mechanism, and accumulated experience in a closed-loop driving setup to demonstrate the generalization and continuous learning capabilities of our LeapVAD.

1) Ablation of VLM: To select the most suitable VLM model for this task, we assessed the performance of various VLMs using the Rank2Tell (real environment) and CARLA (simulation environment) test sets. Specifically, we compared the performance of three models: LLaVA-1.5-7B [48], Qwen-VL-7B [17], and InternVL2-8B [18]. We utilized the grounded score, encompassing accuracy, recall, and $F1$ value, to assess the semantic alignment capabilities of the models. In addition, we employed the chat score, which comprises the language score ROUGE and the GPT (GPT-4-turbo) score, to evaluate their question-answer reasoning abilities. Table V presents the experimental results, indicating that Qwen-VL-7B and InternVL2-8B outperform the other models.

We further assessed the performance of Qwen-VL-7B and InternVL2-8B in a closed-loop setting. We conducted closed-loop tests using these two models on the CARLA Town05 short benchmark, and the experimental results are presented in the penultimate two rows of Table I. We assess the grounding and conversational capabilities of prominent VLMs using the validation set in Rank2Tell dataset [36] and the CarlaSim dataset introduced in Section IV-A1. The experiment demonstrated the superior performance of QwenVL; thus, it was selected for subsequent experiments in this article.

2) Different Type of Scene Token: To promote standardization, we assessed the performance of varying scene token representations on the nuScenes dataset [39]. We utilize the nuScenes training set for model training, while the validation set serves as queries for retrieving data from the training set during evaluation. Table II presents the precision@1 for each method, with our scene token approach demonstrating significantly superior performance compared with LeapAD. For various scene tokens on nuScenes [39]. For each scene token in the validation set, the most similar token from the training set is selected as the top-1 match. A pair of scene

TABLE VI
ABLATION ON THE SIZE OF $\mathcal{H}$

Size	Precision@1		Size	Precision@1	
	steer(%)↑	brake(%)↑		steer(%)↑	brake(%)↑
256	80.33	78.69	2048	85.25	90.16
512	77.05	81.97	4096	83.20	87.52
1024	83.61	88.52	8192	83.61	80.33

TABLE VII
ABLATION ON THE NUMBER OF FEW SHOTS

Shots	DS (%) ↑	RC (%) ↑	IS (%) ↑
0	71.43	97.57	73.31
1	74.57	95.91	77.52
2	77.63	95.20	81.76
3	**82.40**	**99.53**	**82.80**

TABLE VIII
AFFECT OF THE MEMORY SIZE FOR FEW-SHOT LEARNING

Mem Sizes	DS (%) ↑	RC (%) ↑	IS (%) ↑
0	71.43	97.57	73.31
90	74.39	97.72	76.50
900	78.67	**99.64**	78.86
9000	**82.40**	99.53	**82.80**

TABLE IX
COMPUTATIONAL EFFICIENCY

Methods	Params	FPS
UniAD (single)	131.9M	1.8
LeapAD + QwenVL	11.1B	0.14
LeapVAD + QwenVL	11.1B	0.10

tokens is deemed a positive match for steering if the offset of the steering angle is within 0.04. The pair indicates a positive match for braking if the braking signals are concurrent. For evaluation, we retrieved the scene from the memory bank with the highest cosine similarity and utilized its associated driving steering and braking labels. The results revealed that the "pooling + state" setting yielded the best model performance, and thus, we adopted this setting as the default for the closed-loop experiments within this article.

To assess the performance of varying scene token representations more comprehensively, we present the precision–recall curves for different methods in Fig. 5. The legend labels the method, respective settings, and classification space. It is evident from the figure that LeapAD [8] exhibits the poorest performance among the compared methods. In contrast, the "pooling + state" and "attention+state" methods demonstrate a more balanced performance, with "pooling+state" exhibiting superior results in the ACC space, which is particularly crucial for closed-loop experiments.

To investigate how the size of the key dictionary $\mathcal{H}$ and the staleness of cached scene tokens affect learning, we conduct an ablation study on different sizes of $\mathcal{H}$, where the scene encoder employs pooling with ego-state inputs. As shown in Table VI, increasing the size of $\mathcal{H}$ appropriately improves the performance of the scene encoder. However, when the size becomes too large, the presence of overly outdated entries in the key dictionary leads to degraded performance.

To better illustrate the enhancements attributed to the scene token, we employ t-distributed stochastic neighbor embedding (t-SNE) [54] to visualize its impact on clustering different scene token representations. In the first column, Fig. 6 reveals that the scene token utilized by LeapAD fails to form distinct clusters associated with possible driving actions. In contrast, the second column highlights our proposed scene token method's successful differentiation of steering and acceleration/deceleration patterns, even without ego state input. The inclusion of ego-state information further enhances the clustering effect. Notably, the fourth column demonstrates the effectiveness of our proposed scene token extraction technique on mixed nuScenes and CARLA data, yielding favorable clustering results.

Furthermore, rows 13 and 14 of Table I show the results of closed-loop experiments conducted by LeapVAD using OpenAI embedding [52] and our scene token for sample retrieval, respectively. It is evident that utilizing our scene token yields superior outcomes.

3) Ablation on the Number of Few Shots: We conducted an ablation experiment on few-shot prompting in our previous version [8]. To determine if the same outcome would occur after updating the representation of the scene token, we replicated the experiment. The tests were conducted on the Town05 Short benchmark, comprising 9K samples randomly collected from the memory bank introduced in Section IV-A2. As illustrated in Table VII, the results demonstrate that our new scene token effectively retrieves relevant samples and guides decision-making. Due to our VLM updates, LeapVAD outperforms other camera-only methods, even in a zero-shot setting. A notable performance enhancement is observed when advancing from zero-shot to three-shot scenarios, underscoring the advantages of leveraging experience stored in memory and the efficacy of the few-shot strategy.

4) Impact of Memory Sizes: Memory capacity plays a pivotal role in knowledge-driven systems by aggregating critical experiences. Through comprehensive ablation studies, we demonstrate that larger memory capacities lead to enhanced system performance. We conduct closed-loop evaluations using three different memory sizes (9000, 900, and 90 experiences), with samples randomly drawn from the memory bank described in Section IV-A2. As illustrated in Table VIII, the quantitative results reveal a consistent performance improvement with increasing memory size. This underscores the continuous learning ability of LeapVAD and establishes a direct correlation between model performance and experience accumulation.

5) Effectiveness of the Reflection Mechanism: In our prior work [8], we sampled nine sequences with scores below 50 on the Town05 short benchmark for reflection experiments. Specifically, we used a three-shot and 900-experience memory bank setting and found that the scores of most sequences improved with additional reflection rounds, effectively demonstrating the validity of our designed reflection mechanism. We conducted additional ablation experiments to further substantiate this mechanism's robustness. As evidenced by experiments 14 and 15 in Table I, with all other parameters held constant, we observed a DS improvement of 4.41 and an RC enhancement of 0.11 after four reflection rounds.

6) Generalization of Accumulated Knowledge: Our previous research demonstrated the robust transferability of memory

TABLE X
EVALUATION OF DISTANCE AND VELOCITY ESTIMATION

Method	Distance			Velocity		
	MAE (m)↓	RMSE (m)↓	RE (%)↓	MAE (m/s)↓	RMSE (m/s)↓	RE (%)↓
Depth-Anything-V2	1.69	2.43	9.95	1.06	1.97	14.82
Qwen-VL-7B	2.94	4.31	14.70	1.80	3.27	20.31
InternVL2-8B	2.63	3.93	14.08	1.65	3.19	17.03

banks across different towns in CARLA. In this section, we further investigate experience transfer from simulation environments to high-fidelity driving scenarios. As shown in the penultimate row of Table IV, we first conducted closed-loop evaluations of LeapVAD in DriveArena without utilizing any memory bank. Subsequently, we incorporated 18.1K memories accumulated from CARLA, with results presented in the last row of Table IV. The substantial improvement of 3.59 points in average PDMS demonstrates that experiences acquired in simulation environments can effectively transfer not only between different maps but also to real-world driving scenarios. This cross-domain performance enhancement, which surpasses the capabilities of existing methods, validates the superiority and robust transferability of our approach.

V. CONCLUSION

In this article, we propose a human-like, knowledge-driven autonomous driving framework using a VLM to identify critical objects in the environment. Our system mimics human perception and employs a dual-process decision-making module to model driver learning. A reflective mechanism and a memory bank support continuous self-improvement. We also introduce an efficient scene token to represent driving-relevant scenes and retrieve similar samples for decision guidance. Our method achieves strong closed-loop performance with much less data, reaching near state-of-the-art results using only 1/73 of the training data. It improves DSs on the Town05 short and long benchmarks by 5.3% and 42.6% over LeapAD [8], and achieves the best results on DriveArena [10]. Extensive ablation studies further confirm the effectiveness, continuous learning ability, and knowledge transfer of our framework.

However, as shown in Table IX, the current LLM-based model adds high computational cost, making it unsuitable for direct use on real vehicles. To address this, we consider two directions for future work: 1) speeding up the heuristic process via model compression or hardware acceleration and 2) using LeapVAD as a long-horizon decision module combined with a high-frequency end-to-end controller for real-time inference.

APPENDIX

A. Training Details

1) Training Paradigm: Fig. 1 shows our knowledge-driven architecture with three learnable modules: scene understanding, dual-process decision-making, and scene token. The scene understanding module encodes common knowledge and is fine-tuned for better instruction following and output format. The dual-process decision-making module takes scene descriptions as input and generates reasoning steps and meta-actions, trained with experience from the analytic process. The scene token module processes the current image and ego state to produce a RAG key for accurate memory retrieval. Each module is trained independently on datasets described in Section IV. Our current prototype already shows commonsense reasoning in handling corner cases. To further improve, we add a plugin memory bank that stores past experiences, enabling few-shot learning for the heuristic process.

2) Loss Function: The Scene Understanding module employs cross-entropy loss [17], [48] for fine-tuning both Qwen-VL and LLaVA-1.5. For InternVL-2, we adopt a multicomponent loss function comprising image–text contrastive (ITC) loss, image–text matching (ITM) loss, and image-grounded text generation (ITG) loss, following the BLIP-2 framework [14]. The heuristic process utilizes cross-entropy loss to optimize Qwen-1.5. And the training objective for the scene token module is formally defined in Section III-D3.

B. Low-Level Control

Our dual-process decision-making module outputs meta-actions (e.g., "AC," "DC," "*LCL*," "*LCR*," "IDLE," and "STOP"), which serve as inputs for trajectory generation. A proportional-integral-derivative controller (PID) controller then tracks these trajectories to compute the final control signals, such as steering, throttle, and brake.

1) Planned Waypoints: To improve route tracking, we addressed the large gaps (up to tens of meters) between default CARLA waypoints. Using HD maps, we converted sparse waypoints into dense path points at 1-m intervals, forming an accurate reference path. When needed, we also added adjacent lane information to support maneuvers such as overtaking and obstacle avoidance. This refinement allows temporary deviation from navigation waypoints and greatly enhances the flexibility and execution of the control system.

Combining detailed lane data, our controller uses a planner provided by LimSim [26] to generate trajectories for the next 5 s, ensuring that the vehicle travels on the desired path. The actions "AC" and "DC" determine the target state by calculating the target acceleration from the current acceleration, establishing the speed and position 5 s ahead. The actions "*LCL*" and "*LCR*" use a spatiotemporal sampling strategy to sample target positions within the target lane and the speed range to determine the target state. These target states are inputted into the trajectory optimizer in Frenét coordinate [55], generating quintic polynomial trajectories. As a result, the trajectory is optimized and selected based on cost factors such as smoothness, speed matching, acceleration, jerk, and obstacle avoidance, resulting in the optimal trajectory.

Our approach can utilize alternative methods rather than relying on high-definition maps. Techniques like those in DriveCot [56] and TransFuser [42] leverage distinct neural networks to predict future paths from camera images and sparse navigation data, seamlessly aligning with controller design while preserving our core methodology.

2) PID Controller: The controller selects trajectory points with target speeds and waypoints from the LimSim planner based on the vehicle's current speed: at low speeds, it chooses farther points, while at high speeds, it selects closer ones for stable tracking.

For motion control, we use two independent PID controllers [42], [56]: a longitudinal controller for throttle/brake and a lateral controller for steering. The longitudinal PID ($K_P = 1.95, K_I = 0.05$, and $K_D = 0.2$) tracks target speed with a

```
You are a large multi-modal model trained by OpenAI. Now you act as a mature driving assistant, who can give accurate and correct advice for human driver in complex urban driving scenarios. You'll receive some scene description from the view of onboard camera. You'll need to make driving inferences and decisions based on the information. At each decision frame, you receive information about the current scene and a collection of actions. You will perform reasoning based on the the description of front-view image. Eventually you will select the appropriate action output from the action set.
Make sure that all of your reasoning is output in the `## Reasoning` section, and in the `## Decision` section you should only output the name of the action, e.g. `AC`, `IDLE` etc.

## Available Actions
 - AC: increasing your speed.
 - DC: slow down your speed.
 - IDLE: remain in the current lane with current speed.
 - STOP: stop, your current speed shold be zero.
 - LCL: change lane to the left of the current lane.
 - LCR: change lane to the right of the current lane.

You must obey these important rules below:
- You should pay more attention to the object on the ego lane. Vehicles that are not on the ego lane that are moving towards the ego car bring less potential risk. If they are always moving on the left or right lane, you don't need to slow down. You need to notice if they change to the ego lane and come to a close, then you need to slow down or stop to keep your distance and avoid the potential collision.
- When your steering value is negative, you should pay attention to objects in the left lane; when your steering value is positive, you should pay attention to objects in the right lane.
- You should always keep your distance from the objects around you, especially the object on the ego lane. You should 'DC' or even 'STOP' when an object is very close on the ego lane
- If you see a red light or yellow light on the ego lane, you should 'DC' or 'STOP' immediately. You should stop just before the white line (perpendicular to the direction you are running) in front of you and do not cross it.
- If there is any object that is less than 25m away from the vehicle, you should slow down ('DC'), and if the distance is less than 15 meters, you should 'STOP' immediately to avoid collision. When you are turning, you only need to focus on objects within 5m distance, no need to slow down or stop. If the distances of all objects are large than 25m, there are no red traffic lights and stop signs in front of you, and you are at low speeds (e.g. lower than 1m/s), you should select "AC" insteand of "IDLE".
- If there is any stop sign that is less than 10m away, you should 'DC' and if it is less than 5m away from the vehicle, you should 'STOP' immediately.

Your answer should follow this format:
## Reasoning
reasoning based on the descrioption of the front-view.
## Decison
one of the actions in the action set.(SHOULD BE exactly same and no other words!)

You must obey the rules above. \n
```

Fig. 7. System prompt for analytic process.

```
You are a part of an automotive safety analysis team. Your task is to reevaluate reasoning and decisions based on current vehicle accident information and descriptions of historical frames. You should identify potential inference errors in historical frames and propose revised inferences and decisions based on this reevaluation. Note that you should select one historical frame that has the best chance of avoiding the current accident to suggest your modifications about the reasoning and decision.
Make sure that all of your reasoning is output in the `## Reasoning` section, and in the `## Decision` section you should only output the name of the action, e.g. `AC`, `IDLE` etc.

Available Actions and important rules are the same as the system prompt for Analytic Process.

Your answer should follow this format:
## Time
time of the selected historical frame \n
## Reasoning
reasoning about the decision based on the descrioption of the selected frame. Pay attention to modifying pronouns, such as using "this frame" or "this moment" instead of using "the historical frame" or "at time of".
## Decision
one of the actions in the action set.(SHOULD BE exactly same and no other words!)

You must obey the rules above.
```

Fig. 8. System prompt in the reflection mechanism.

```
Current scene: {}, Frame 0: <img>{}</img>\nFrame 1: <img>{}</img>\nFrame 2: <img>{}</img>\nFrame 3: <img>{}</img>\nFrame 4: <img>{}</img>\n. The time interval between frames is 0.5s. You are driving a car in an urban street and this image indicates the scene that you see. Please describe in detail the key objects in the scene that may affect your driving. The object information should contain the object's category, the object's location relative to the ego car or the ego lane, the object's motion state, and the object's bounding box, as well as the approximate distance from the ego car. Especially, if there are any traffic lights, the information should contain the traffic light's color and bounding box.
```

Fig. 9. VLM system prompt.

10-frame buffer, while the lateral PID ($K_P = 1.0, K_I = 0.05$, and $K_D = 0.0$) computes steering from the heading angle difference to the next waypoint, also with a 10-frame buffer.

C. Distance and Speed Prediction

To evaluate distance and velocity estimation, we use ground-truth annotations from 5000 Carla simulation frames. VLM-generated object descriptions are aligned with ground-truth via bounding box intersection over union (IoU) (≥ 0.6). For valid matches, we report mean absolute error (MAE), root mean squared error (RMSE), and relative error (RE) with respect to the ground-truth mean values. As shown in Table X, while Qwen-VL-7B and InternVL2-8B achieve similar performance, both underperform compared with the task-specific baseline (Depth-Anything-V2). Note that the velocity of Depth-Anything-V2 [57] is derived from multiframe distance differences, providing a reference for task-specific speed estimation and highlighting the potential of augmenting VLMs with dedicated estimation modules.

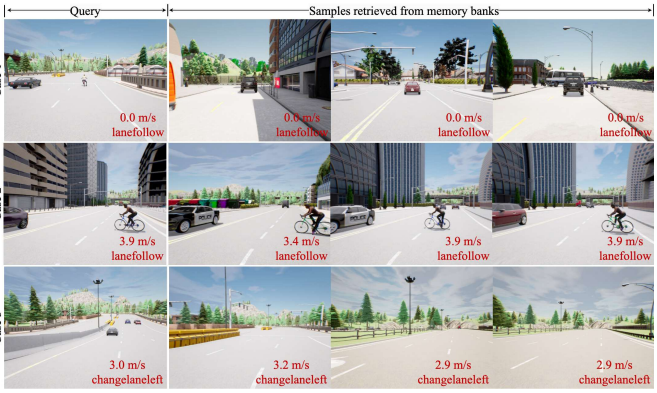


Fig. 10. Retrieved samples in the CARLA closed-loop experiment.

Fig. 11. Retrieved samples in the DriveArena closed-loop experiment.

D. Prompt Details

1) Prompt for Analytic Process: We detail the system prompt (see Fig. 7) used by analytic process to accumulate experience in a closed-loop environment. The prompt has four parts: task definition, meta-actions, traffic rules, and output format. Task definition specifies input–output requirements, while enumerated actions allow easy extraction from the agent's output. Traffic rules are included to assess driving compliance. The structured output requires both a decision and its reasoning, ensuring explainability and guiding heuristic process.

2) Prompt for Reflection: The reflection prompt (see Fig. 8) builds upon the system prompt while introducing error-detection criteria for historical frames. Unlike the analytic process prompt, this version modifies the task definition to instruct the agent to identify frames containing potential errors, analyze the associated scene descriptions and previous decisions, and output corrected reasoning and decisions.

3) Prompt for Scene Understanding: We describe the VLM prompts (see Fig. 9) for identifying critical objects in traffic scenes. These prompts combine image tokens with frame

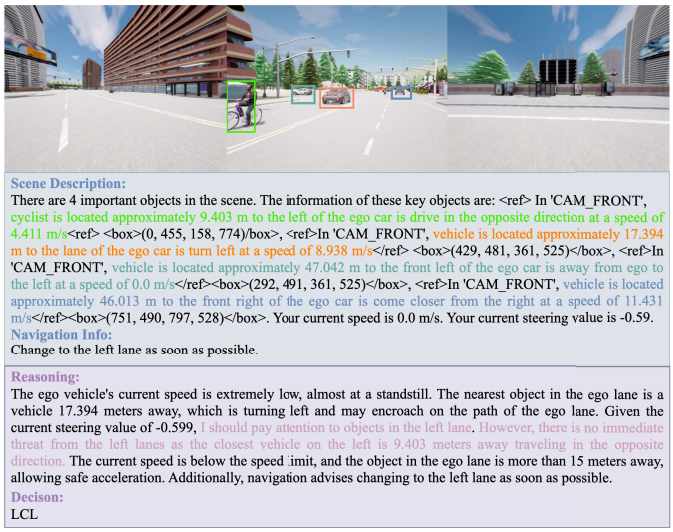


Fig. 12. Case of "*LCL*."

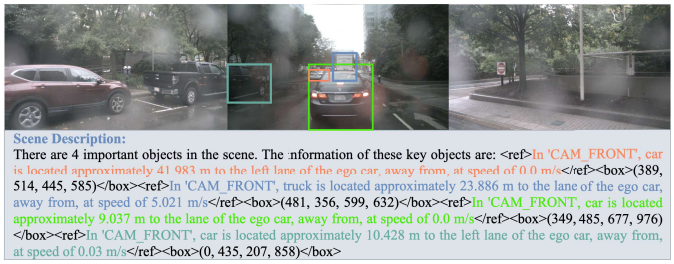


Fig. 13. Example case in rainy weather.

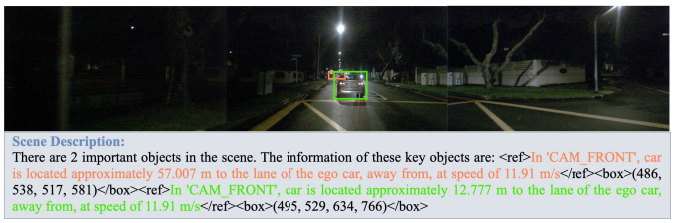


Fig. 14. Example case at night.

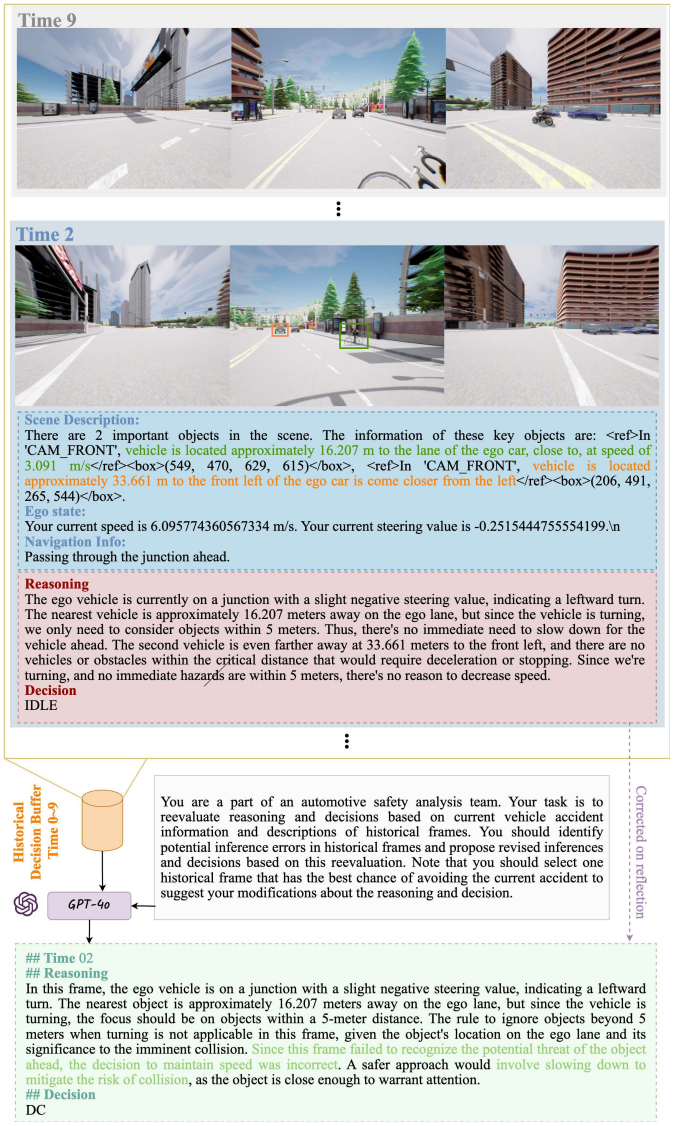


Fig. 15. Case study for reflection mechanism.

frequency for temporal context and instruct the VLM to capture each object's location, motion, bounding box, and distance from the ego vehicle. The bounding box is essential as it uniquely links visual and textual representations.

E. Visualization Cases

1) Case of Scene Token: In our closed-loop experiments, we examined cases of samples retrieved from the memory bank using scene tokens. Fig. 10 shows Town05 short benchmark examples, where retrieved samples share similar scenarios and traffic conditions, with ego states closely matching the query. Fig. 11 presents DriveArena cases, showing clear query–sample correlations: case 1 (traffic lights), case 2 (intersections), and case 3 (oncoming traffic). However, due to the larger data gap in DriveArena, retrieval quality is lower than in Fig. 10.

2) Close-Loop Case in CARLA: Fig. 12 illustrates a closed-loop case in CARLA, where the ego car navigates through a city with heavy traffic. The navigation system advises the car to switch to the left lane. As depicted in VLM, the majority of the vehicles are at a safe distance from the ego car. The closest cyclist is traveling in the opposite direction on the left, but heuristic process assesses that changing lanes presents no risk and makes the decision to "*LCL*."

3) Cases Under Different Weather Conditions: To test VLM reliability under different conditions, we visualize its outputs in rainy and night scenarios (see Figs. 13 and 14). The VLM accurately describes key objects in the rain, e.g., assigning 0 m/s to both a parked car and a braking car. At night, however, it sometimes hallucinates, misidentifying light sources as vehicles.

4) Cases of Reflection Mechanism: As described in Section III-E, analytic process reflects on accidents to improve the system. We set the decision buffer $\boldsymbol{Q}$ to $m = 10$ at 1 Hz. In the case shown in Fig. 15, when heuristic process collided with a cyclist at step 9, we fed the accident type O, buffer $\boldsymbol{Q}$, and reflection prompt (see Fig. 8) into analytic process. The output indicated the car should have chosen "DC" at step 2 instead of "IDLE" to avoid the collision.

ACKNOWLEDGMENT

This work was completed when Yukai Ma, Jianbiao Mei, Tiantian Wei, Naiting Zhong, and Tao Hu were research interns at Shanghai Artificial Intelligence Laboratory, Shanghai, China.

REFERENCES

[1] Z. Li et al., "BEVformer: Learning birds-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. Eur. Conf. Comput. Vis.* Switzerland: Springer, 2022, pp. 1–18.

[2] Z. Liu et al., "BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 2774–2781.

[3] J. Yuan et al., "RAG-driver: Generalisable driving explanations with retrieval-augmented in-context learning in multi-modal large language model," 2024, *arXiv:2402.10828*.

[4] X. Tian et al., "DriveVLM: The convergence of autonomous driving and large vision-language models," 2024, *arXiv:2402.12289*.

[5] Z. Li et al., "Is ego status all you need for open-loop end-to-end autonomous driving?," 2023, *arXiv:2312.03031*.

[6] J. S. B. T. Evans and K. E. Stanovich, "Dual-process theories of higher cognition: Advancing the debate," *Perspect. Psychol. Sci.*, vol. 8, no. 3, pp. 223–241, May 2013.

[7] P. C. Wason and J. S. B. T. Evans, "Dual processes in reasoning?," *Cognition*, vol. 3, no. 2, pp. 141–154, Jan. 1974.

[8] J. Mei et al., "Continuously learning, adapting, and improving: A dual-process approach to autonomous driving," 2024, *arXiv:2405.15324*.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. robot Learn.*, 2017, pp. 1–16.

[10] X. Yang et al., "DriveArena: A closed-loop generative simulation platform for autonomous driving," 2024, *arXiv:2408.00415*.

[11] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.

[12] W.-L. Chiang et al., "Vicuna: An open-source chatbot impressing GPT-4 with 90% ChatGPT quality," Large Model Syst. Org. (LMSYS Org), USA, Tech. Rep., Mar. 2023.

[13] J.-B. Alayrac et al., "Flamingo: A visual language model for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 23716–23736.

[14] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," 2023, *arXiv:2301.12597*.

[15] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "MiniGPT-4: Enhancing vision-language understanding with advanced large language models," 2023, *arXiv:2304.10592*.

[16] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 34892–34916.

[17] J. Bai et al., "Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond," 2023, *arXiv:2308.12966*.

[18] Z. Chen et al., "InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, May 2023, pp. 24185–24198.

[19] H. Zhang et al., "LLaVA-grounding: Grounded visual chat with large multimodal models," 2023, *arXiv:2312.02949*.

[20] J. Zhang, J. Huang, S. Jin, and S. Lu, "Vision-language models for vision tasks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 8, pp. 5625–5644, Aug. 2024.

[21] D. Fu et al., "Drive like a human: Rethinking autonomous driving with large language models," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. Workshops (WACVW)*, Jan. 2024, pp. 910–919.

[22] L. Wen et al., "DiLu: A knowledge-driven approach to autonomous driving with large language models," in *Proc. ICLR*, 2024, pp. 20165–20184.

[23] C. Sima et al., "Drivelm: Driving with graph visual question answering," 2023, *arXiv:2312.14150*.

[24] W. Wang et al., "DriveMLM: Aligning multi-modal large language models with behavioral planning states for autonomous driving," 2023, *arXiv:2312.09245*.

[25] Y. Zhou et al., "Embodied understanding of driving scenarios," 2024, *arXiv:2403.04593*.

[26] L. Wen et al., "LimSim: A long-term interactive multi-scenario traffic simulator," 2023, *arXiv:2307.06648*.

[27] Y. Hu et al., "Planning-oriented autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2023, pp. 17853–17862.

[28] B. Jiang et al., "VAD: Vectorized scene representation for efficient autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 8340–8350.

[29] Y. Peng et al., "The tong test: Evaluating artificial general intelligence through dynamic embodied physical and social interactions," *Engineering*, vol. 34, pp. 12–22, Mar. 2024.

[30] S. Gildert and G. Rose, "Building and testing a general intelligence embodied in a humanoid robot," 2023, *arXiv:2307.16770*.

[31] B. Zhang, J. Zhu, and H. Su, "Toward the third generation artificial intelligence," *Sci. China Inf. Sci.*, vol. 66, no. 2, Feb. 2023, Art. no. 121101.

[32] Z. Xi et al., "The rise and potential of large language model based agents: A survey," 2023, *arXiv:2309.07864*.

[33] C. Cui et al., "A survey on multimodal large language models for autonomous driving," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, May 2024, pp. 958–979.

[34] Wayve, "Lingo-1: Exploring natural language for autonomous driving," London, U.K., Tech. Rep., 2023.

[35] S. Jiang et al., "A survey on vision-language-action models for autonomous driving," 2025, *arXiv:2506.24044*.

[36] E. Sachdeva et al., "Rank2Tell: A multimodal driving dataset for joint importance ranking and reasoning," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2024, pp. 7498–7507.

[37] Q. Zhang, Z. Peng, and B. Zhou, "Learning to drive by watching YouTube videos: Action-conditioned contrastive policy pretraining," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2022, pp. 111–128.

[38] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9729–9738.

[39] H. Caesar et al., "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11621–11631.

[40] J. Bai et al., "Qwen technical report," 2023, *arXiv:2309.16609*.

[41] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," in *Proc. Conf. Robot Learn.*, 2023, pp. 726–737.

[42] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "TransFuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 12878–12895, Nov. 2023.

[43] K. Chitta, A. Prakash, and A. Geiger, "NEAT: Neural attention fields for end-to-end autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15793–15803.

[44] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "End-to-end urban driving by imitating a reinforcement learning coach," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2021, pp. 15222–15232.

[45] D. Chen, V. Koltun, and P. Krahenbuhl, "Learning to drive from a world on rails," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15590–15599.

[46] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Proc. Conf. Robot Learn.*, 2020, pp. 66–75.

[47] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, May 2019, pp. 9329–9338.

[48] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 26296–26306.

[49] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[50] X. Jia et al., "Think twice before driving: Towards scalable decoders for end-to-end autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 21983–21994.

[51] 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 *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 6119–6132.

[52] A. Neelakantan et al., "Text and code embeddings by contrastive pre-training," 2022, *arXiv:2201.10005*.

[53] D. Dauner et al., "NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking," 2024, *arXiv:2406.15349*.

[54] G. Hinton and L. Van Der Maaten, "Visualizing data using t-SNE journal of machine learning research," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Jul. 2008.

[55] L. Wen, P. Cai, D. Fu, S. Mao, and Y. Li, "Bringing diversity to autonomous vehicles: An interpretable multi-vehicle decision-making and planning framework," 2023, *arXiv:2302.06803*.
[56] T. Wang, E. Xie, R. Chu, Z. Li, and P. Luo, "DriveCoT: Integrating chain-of-thought reasoning with end-to-end driving," 2024, *arXiv:2403.16996*.
[57] L. Yang et al., "Depth anything V2," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 21875–21911.

Yukai Ma (Graduate Student Member, IEEE) received the B.Eng. degree in electrical engineering and its automation from Zhejiang University of Technology, Hangzhou, China, in 2021. He is currently pursuing the Ph.D. degree with the College of Control Science and Engineering, Zhejiang University, Hangzhou.

He is currently a Visiting Graduate Researcher at UCLA. His main research focuses on perception and planning in robotics and autonomous driving.

Tiantian Wei received the B.E. degree in mechanical engineering and the M.Sc. degree in mechatronics and robotics from the Technical University of Munich, Munich, Germany, in 2023 and 2025, respectively.

Her research interests include computer vision, deep learning, autonomous driving, and robot learning, with an emphasis on embodied artificial intelligence (AI) and cognitive perception-driven robotic systems.

Naiting Zhong received the B.S. and M.S. degrees from the College of Automotive Studies, Tongji University, Shanghai, China, in 2022 and 2025, respectively.

This work was accomplished during his internship at the Artificial Intelligence (AI) Lab. His research interests include autonomous driving validation technology and reinforcement learning agents.

Jianbiao Mei received the B.S. degree in control science and engineering from Zhejiang University, Zhejiang, China, in 2021, where he is currently pursuing the Ph.D. degree with the Laboratory of Advanced Perception on Robotics and Intelligent Learning, College of Control Science and Engineering.

His research interests include video segmentation, 3-D perception, and autonomous driving.

Tao Hu received the B.E. degree from Anhui University, Hefei, China, in 2023. He is currently pursuing the M.S. degree with the University of Science and Technology of China, Hefei, China.

His research interests include image restoration and video generation.

Licheng Wen received the B.S. and M.Sc. degrees in control science from Zhejiang University, Hangzhou, China, in 2019 and 2022, respectively.

He is currently a Researcher with Shanghai Artificial Intelligence (AI) Laboratory and collaborates with the Shanghai Institute of Innovation. His research interests include multimodal large language models, autonomous driving, and embodied artificial intelligence.

Xuemeng Yang received the M.S. degree in control science and engineering from the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China, in 2022.

She is now a Researcher at Shanghai Artificial Intelligence Laboratory, Shanghai, China. Her primary research interests encompass autonomous driving and artificial intelligence (AI) agents.

Botian Shi received the Ph.D. degree from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2021.

He is currently a Researcher at the ADLab, Shanghai Artificial Intelligence Laboratory, Shanghai, China. His research interests include autonomous driving systems, embodied artificial intelligence, as well as knowledge-driven autonomous driving.

Yong Liu received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively.

He is a Professor at the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University. His research interests include machine learning, robotics vision, multiple-sensor fusion, and intelligent systems.