Asynchronous Large Language Model Enhanced Planner for Autonomous Driving

Yuan Chen[1,2⋆], Zi-han Ding[1⋆], Ziqin Wang[1⋆], Yan Wang[2†], Lijun Zhang[1], and Si Liu[1†]

[1] Beihang University
[2] AIR, Tsinghua University
{chenyuan1, wzqin, ljzhang, liusi}@buaa.edu.cn
zihanding819@gmail.com
wangyan@air.tsinghua.edu.cn

Abstract. Despite real-time planners exhibiting remarkable performance in autonomous driving, the growing exploration of Large Language Models (LLMs) has opened avenues for enhancing the interpretability and controllability of motion planning. Nevertheless, LLM-based planners continue to encounter significant challenges, including elevated resource consumption and extended inference times, which pose substantial obstacles to practical deployment. In light of these challenges, we introduce *AsyncDriver*, a new asynchronous LLM-enhanced closed-loop framework designed to leverage scene-associated instruction features produced by LLM to guide real-time planners in making precise and controllable trajectory predictions. On one hand, our method highlights the prowess of LLMs in comprehending and reasoning with vectorized scene data and a series of routing instructions, demonstrating its effective assistance to real-time planners. On the other hand, the proposed framework decouples the inference processes of the LLM and real-time planners. By capitalizing on the asynchronous nature of their inference frequencies, our approach have successfully reduced the computational cost introduced by LLM, while maintaining comparable performance. Experiments show that our approach achieves superior closed-loop evaluation performance on nuPlan's challenging scenarios. The code and dataset are available at https://github.com/memberRE/AsyncDriver.

Keywords: Autonomous Driving · Large Language Models · Motion Planning

1 Introduction

Motion planning plays a pivotal role in autonomous driving, garnering significant interest due to its direct impact on vehicle navigation and safety. One particularly noteworthy evaluation approach is the employment of closed-loop simulation, which involves the dynamic development of driving scenarios that

⋆ Equal contribution. † Corresponding author.

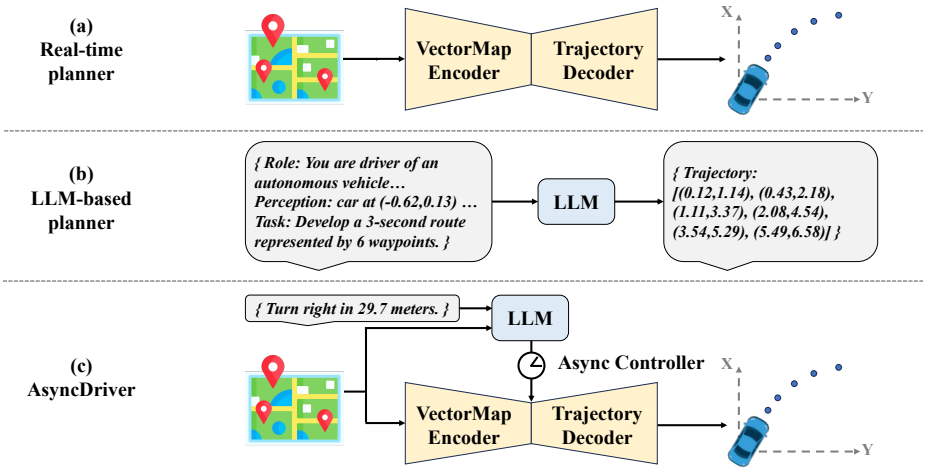


Fig. 1: Comparative Overview of Learning-based Autonomous Driving Planning Frameworks. (a) Real-time planner: Offers quick inference but has limited controllability. (b) LLM-based planner: Produces linguistic descriptions and controls, offering high interactivity and interpretability at the expense of inference speed. (c) AsyncDriver: While leveraging the reasoning capabilities of LLM, a balance between performance and inference speed is achieved through asynchronous control.

adapt to the planner's predicted trajectories, thus necessitating that the model exhibits stronger predictive accuracy and bias correction capabilities.

As illustrated in Fig.1(a), current learning-based real-time motion planning frameworks [13, 18, 20, 24, 32, 33] typically utilize vectorized map information as input and employ a decoder to predict trajectories. As purely data-driven methods, they are particularly vulnerable to long-tail phenomena, where their performance can significantly degrade in rare or unseen scenarios [6]. Moreover, while some rule-based strategies exist, their manual crafting of rules are found to be inadequate for capturing the entirety of potential complex scenarios, resulting in driving strategies that tend towards extremes either excessive caution or aggression. Furthermore, both learning-based and rule-based planning frameworks suffer from low controllability, which raises concerns regarding the safety and reliability of such systems in dynamic environments.

Recently, the considerable potential of Large Language Models (LLMs), including GPT-4 [1] and Llama2 [38], has been extensively explored within the realm of autonomous driving. Their extensive pre-training on large-scale datasets has established a robust foundation for comprehending traffic rules and scenarios. Consequently, LLM-based planners have demonstrated superior performance in scene analysis, reasoning, and human interaction, heralding new prospects for enhancing the interpretability and controllability of motion planning [9, 46, 48]. Nonetheless, as shown in Fig.1(b), these models frequently encounter several of these specific challenges: 1) The scene information is described through language, which could be constrained by the permissible input token length, making

it challenging to encapsulate complex scene details comprehensively and accurately [28, 29, 34, 44]. 2) Prediction via linguistic outputs entails either directly issuing high-level commands that are then translated into control signals, potentially leading to inaccuracies, or outputting trajectory points as floating-point numbers through language, a task at which LLMs are not adept [22, 29, 45]. 3) Prevalent frameworks primarily utilize LLMs as the core decision-making entity. While this strategy offers advantages in performance, the inherently large number of parameters in LLMs results in a noticeable decrease in inference speed relative to real-time planners, presenting substantial obstacles to their real-world implementation.

In this work, we introduce AsyncDriver, a novel asynchronous LLM-enhanced framework for closed-loop motion planning. As depicted in Fig.1(c), this method aligns the modalities of vectorized scene information and series of routing instructions, fully leveraging the considerable capabilities of LLM for interpreting instructions and understanding complex scenarios. The Scene-Associated Instruction Feature Extraction Module extracts high-level instruction features, which are then integrated into the real-time planner through the proposed Adaptive Injection Block, significantly boosting prediction accuracy and ensuring finer trajectory control. Moreover, our approach preserves the architecture of the real-time planner, allowing for the decoupling of inference frequency between LLM and the real-time planner. By controlling the asynchronous intervals of inference, it significantly enhances computational efficiency and alleviates the additional computational cost introduced by LLM. Furthermore, the wide applicability of our proposed Adaptive Injection Block ensures that our framework can be seamlessly extended to any transformer-based real-time planner, underscoring its versatility and potential for broader application.

To summarize, our paper makes the following contributions:

- We propose AsyncDriver, a novel asynchronous LLM-enhanced framework, in which the inference frequency of LLM is controllable and can be decoupled from that of the real-time planner. While maintaining high performance, it significantly reduces the computational cost.
- We introduce the Adaptive Injection Block, which is model-agnostic and can easily integrate scene-associated instruction features into any transformer-based real-time planner, enhancing its ability in comprehending and following series of language-based routing instructions.
- Compared with existing methods, our approach demonstrates superior closed-loop evaluation performance in nuPlan's challenging scenarios.

2 Related Work

2.1 Motion Planning For Autonomous Driving

The classic modular pipeline for autonomous driving includes perception, prediction, and planning. In this framework, the planning stage predict a future

trajectory based on the perception outputs, then executed by the control system. This architecture, widely adopted in industry frameworks like Apollo [3], contrasts with end-to-end approaches [16, 17] by enabling focused research on individual tasks through well-defined data interfaces between modules.

Autonomous driving planners can be mainly categorized into rule-based and learning-based types. Rule-based planners [2, 10, 11, 21, 23, 39, 40] rely on predefined rules for determining the vehicle's trajectory, such as maintaining a safe following distance and obeying traffic signals. For instance, IDM [39] ensures a safe distance from the leading vehicle by calculating an appropriate speed based on braking and stopping distances. PDM [10] builds on IDM by selecting the highest-scoring IDM proposal as the final trajectory, achieving state-of-the-art performance in the nuPlan Challenge 2023 [4]. However, rule-based planners often struggle with complex driving scenarios beyond their predefined rules.

Learning-based planners [13, 18, 20, 24, 32, 33] aim to replicate human expert driving trajectories using imitation learning or offline reinforcement learning from large-scale datasets. However, they face limitations due to the scope of the datasets and model complexity, leaving substantial room for improvement in areas like routing information comprehension and environmental awareness.

2.2 LLM For Autonomous Driving

The rapid advancement of Large Language Models has been noteworthy. These models, including GPT-4 [1] and Llama2 [38], have been trained on extensive textual datasets and exhibit exceptional generalization and reasoning abilities. A growing body of research has explored the application of LLMs' decision-making capacities to the domain of autonomous driving planning [5, 7, 8, 12, 14, 19, 25–29, 31, 34–37, 41–45, 47].

Some efforts [12, 28, 29, 44] have explored integrating scene information, including the ego vehicle's status and information about obstacles, pedestrians, and other vehicles into LLMs using linguistic modalities for decision-making and explanations. These approaches face limitations due to finite context length, making it challenging to encode precise information for effective decision-making and reasoning. To overcome these constraints, multi-modal strategies such as DrivingWithLLM [7], DriveGPT4 [45], and RAGDriver [47] have been developed. These methods align vectorized or image/video modalities with linguistic instructions for a more comprehensive interpretation of driving scenarios. However, using language to express control signals has its limitations. DrivingWithLLM outputs high-level commands in linguistic expressions, improving QA interaction but reducing the fidelity of translating complex reasoning into precise vehicle control. DriveGPT4 expresses waypoints through language, showing strong open-loop performance but lacking closed-loop simulation evaluation.

Furthermore, some efforts [34,35] focus on closed-loop evaluation by connecting low-level controllers or regressors behind large language models for precise vehicle control. LMDrive [35] uses continuous image frames and navigation instructions for closed-loop driving but requires complete LLM inference at each planning step. LanguageMPC [34] employs LLMs to obtain Model Predictive

Control parameters, achieving control without training. However, these methods necessitate serial language decoding or full LLM inference at each planning step, challenging real-time responsiveness and limiting practical deployment.

In our approach, we shift from using LLMs for direct language output to enhancing real-time learning-based planners. This strategy improves environmental comprehension and allows LLMs and real-time planners to operate independently at different inference rates. This decoupling reduces LLM inference latency, facilitating real-world deployment.

3 Data Generation

The nuPlan [4] dataset is the first large-scale benchmark for autonomous driving planning, comprising $1,200$ hours of real-world human driving data from Boston, Pittsburgh, Las Vegas, and Singapore. To support various training stages, we developed pre-training and fine-tuning datasets from the nuPlan *Train* and *Val* sets, focusing on 14 official [30] challenging scenario types.

3.1 Pre-training Data Generation

To enhance LLM's understanding of instructions in autonomous driving, we created a dataset of language-based QAs, aligning with LLM's native modality to better grasp instruction semantics, which includes *Planning-QA* and *Reasoning1K*, with sample datasets provided in the supplementary material.

Planning-QA is created using a rule-based approach for scalability. It is designed to enhance the LLM's understanding of the relationships among waypoints, high-level instructions, and control. In this context, waypoints are arrays of points, high-level instructions are composed of velocity commands (*stop, accelerate, decelerate, maintain speed*) and routing commands (*turn left, turn right, go straight*), and control involves velocity and acceleration values. Planning-QA includes six types of questions, each focusing on the conversion between waypoints, high-level instructions, and control.

Reasoning1K includes $1,000$ pieces of data generated by GPT-4, beyond merely providing answers, it further supplements the reasoning and explanation based on the scene description and is used for mixed training with Planning-QA.

3.2 Fine-tuning Data Generation

To further achieve multimodal understanding and alignment, we constructed a fine-tuning dataset based on $10,000$ scenarios, capturing one frame every 8 seconds, resulted in a training set of $180,000$ frames and a validation set of $20,000$ frames, each incorporating both vectorized map data and linguistic prompts. Importantly, the scenario type distribution in both training and validation datasets matches the distribution of the entire nuPlan trainval dataset.

For the extracted vectorized scene information, similar to [18], ego information and 20 surrounding agent information over 20 historical frames, in addition to global map data centered around the ego are involved.

LLM's prompt is comprised of two parts: system prompt and series of routing instructions. Concerning routing instructions, a rule-based approach is employed to transform the pathway into a series of instructions enhanced with distance information. Regarding training dataset preparation, the ego vehicle's ground truth trajectory over the ensuing 8 seconds is harnessed as the pathway for routing instruction generation. During simulation, based on the observation of the current scene, a pathway that adheres to a specified maximum length is found through a hand-crafted method as a reference path for instruction generation.

4 Methodology

As illustrated in Fig. 2, we introduce the asynchronous LLM-enhanced closed-loop framework, AsyncDriver, which mainly includes two components: 1) The Scene-Associated Instruction Feature Extraction Module; 2) The Adaptive Injection Block. Additionally, due to our framework's design, the inference frequency between the LLM and the real-time planner could be decoupled and regulating by asynchronous interval, which markedly enhancing inference efficiency.

Within this section, we present the Scene-Associated Instruction Feature Extraction Module (Section 4.1), detail the design of Adaptive Injection Block (Section 4.2), discuss the concept of Asynchronous Inference (Section 4.3) and outline the training details employed (Section 4.4).

4.1 Scene-Associated Instruction Feature Extraction Module

Multi-modal Input In each planning iteration, the vectorized scene information is acquired from the simulation environment. Analogous to the approach employed by GameFormer [18], historical trajectory and state information of both the ego and other agents are extracted, alongside global map data. Vectorized scene information for the real-time planner is provided in the same manner. All vector data are relative to the position of the ego. Subsequently, through the processing by the Vector Map Encoder and Map Adapter, we derive map embeddings. These map embeddings, along with language embeddings, are then fed into the Llama2-13B backbone to obtain the final hidden features $h = \{h_0, h_1, ..., h_{-1}\} \in \mathbb{R}^{N_h \times D_{llm}}$.

Alignment Assistance Module To grasp the essence of routing instructions while maintaining a fine-grained comprehension of vectorized scene information for enhanced extraction of scene-associated high-level instruction features, we employ the Alignment Assistance Module to facilitate the alignment of multi-modal input. Concretely, we have pinpointed five critical scene elements essential to the autonomous driving process for multi-task prediction, which is implemented by five separate 2-layer MLP prediction heads. About the current states

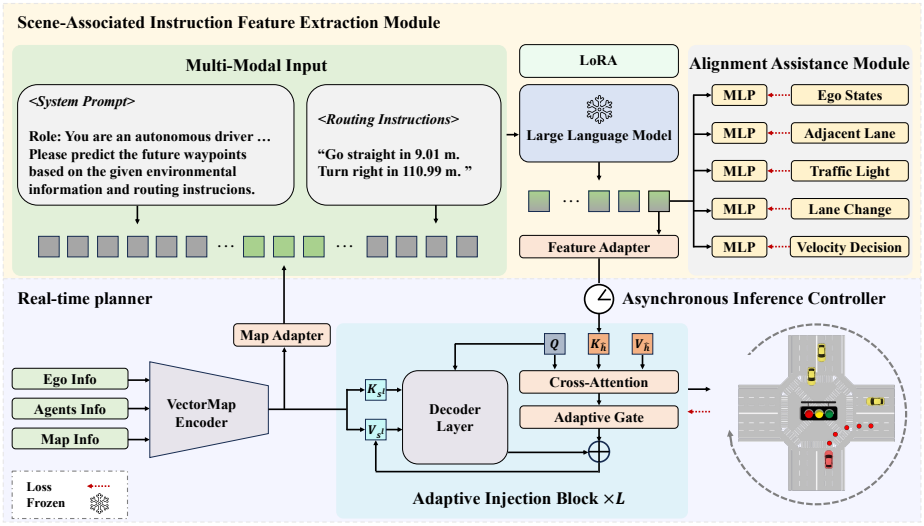


Fig. 2: Overview of our proposed AsyncDriver framework. Scene information, together with routing instructions, is encoded through the Scene-Associated Instruction Feature Extraction Module. Subsequently, the Adaptive Injection Block asynchronously enhances the features of the real-time planner, facilitating closed-loop control for autonomous vehicles. The Alignment Assistance Module is exclusively employed for multi-modality alignment during training.

of the ego vehicle, we perform regression to estimate the vehicle's velocity and acceleration along the X and Y axes. For map information, we undertake classification tasks to identify the existence of adjacent lanes on both the left and right sides and to assess the status of traffic lights relevant to the current lane. Additionally, with a view towards future navigation strategies, we classify the requirement for lane changes in upcoming trajectories and identify future velocity decision, which includes options *acceleration*, *deceleration*, and *maintaining current speed*. It is worth noting that the Alignment Assistance Module is only used to assist multi-modal alignment in the training phase and does not participate in the inference stage.

4.2 Adaptive Injection Block

We adopt the decoder structure of [18] as our basic decoder layer, facilitating the adaptive integration of scene-associated instruction features by evolving the conventional transformer-based decoder layer into an Adaptive Injection Block.

Specifically, the hidden feature of the last token h_{-1} is projected via the feature adapter and subsequently fed into Adaptive Injection Block.

$$\hat{h} = Linear(h_{-1}) \tag{1}$$

Within the Adaptive Decoder Block, the foundational decoder architecture of the real-time planner is elegantly extended to ensure that the query in each layer not only preserves the attention operation intrinsic to the original scene information but also engages in cross-attention with scene-associated instruction features, thereby incorporating instructional guidance into the prediction process. Afterwards, the updated instruction-enhanced query feature is modulated by the learnable adaptive gate, which is initialized by zero and reintegrated into the original attention output of the decoder layer. The adaptive injection process of the *l-th* decoder block can be formulated as follows:

$$s^{l+1} = g \cdot softmax(\frac{QK_{\hat{h}}{}^T}{\sqrt{C}})V_{\hat{h}} + softmax(\frac{QK_{s^l}{}^T}{\sqrt{C}})V_{s^l} \tag{2}$$

where g is the value of adaptive gate, Q denotes query in original decoder layer, K_i and V_i represent key and value respectively of feature i, and s^l notes the scene feature of the *l-th* decoder layer.

The proposed adaptive injection method not only maintains the original decoder layer's ability to process complete scene information within the real-time planner but also enhances the planner's understanding and compliance with a series of flexible linguistic instructions. This advancement allows for the production of more refined and controllable predictions. It is worth noting that due to the simple yet effective design of our Adaptive Injection Block, it can be seamlessly integrated into any transformer-based architecture, thereby affording our approach the flexibility to be adapted to other real-time planner frameworks.

4.3 Asynchronous Inference

Our design leverages LLM to guide the real-time planner, significantly enhancing its performance through series of flexibly combined linguistic instructions without compromising its structural integrity. This method facilitates controlled asynchronous inference, effectively decoupling the inference frequencies of the LLM and the real-time planner, thereby LLM is not required to process every frame. During asynchronous intervals, the previously derived high-level instruction features continue to guide the prediction process of the real-time planner, which significantly boosts the inference efficiency and reduces the computational cost introduced by LLM. Notably, our framework accommodates a series of flexibly combined routing instructions, capable of delivering long-term, high-level routing insights. Therefore, even amidst asynchronous intervals, prior high-level features could still offer effective guidance, reinforcing the performance robustness throughout LLM inference intervals.

Experimental results reveal that our architecture maintains near-robust performance when the asynchronous inference interval of LLM is extended. By controlling the LLM to perform inference every 3 frames can achieve a reduction in inference time of nearly 40%, with only a minimal accuracy loss of about 1%, which demonstrates the efficacy of our approach in striking an optimal balance between accuracy and inference speed. For a more comprehensive exploration of experimental results and their analysis, please refer to Section 5.2.

4.4 Training Details

During the pre-training stage, the entirety of Reasoning1K, augmented with $1,500$ samples randomly selected from Planning-QA, was utilized to train LoRA. This process enabled the LLM to evolve from a general-purpose large language model into one specifically optimized for autonomous driving. As a direct outcome of this focused adaptation, the LLM has become adept at understanding instructions more accurately within the context of motion planning.

During the fine-tuning stage, since the architectures of VectorMap Encoder and Decoder are preserved, we load weights of the real-time planner pre-trained on the same dataset to enhance training stability. The total loss of the fine-tuning stage is comprised of Alignment Assistance Loss and Planning Loss. The former is partitioned into five components: $l1$ loss for 1) ego velocity and acceleration prediction $\tilde{x}_{va} \in \mathbb{R}^4$, cross-entropy loss for 2) velocity decision prediction $\tilde{x}_{dec} \in \mathbb{R}^3$ and 3) traffic light state prediction $\tilde{x}_{traf} \in \mathbb{R}^4$, binary cross-entropy loss for 4) adjacent lane presence prediction $\tilde{x}_{adj} \in \mathbb{R}^2$ and 5) lane change prediction $\tilde{x}_{chg} \in \mathbb{R}$. The complete Alignment Assistance Loss can be expressed as follows, where $\tilde{x}$ and x represent prediction and ground truth respectively:

$$\begin{aligned} L_{align} = L_1(\tilde{x}_{va}, x_{va}) + CE(\tilde{x}_{dec}, x_{dec}) + CE(\tilde{x}_{traf}, x_{traf}) \\ + BCE(\tilde{x}_{adj}, x_{adj}) + BCE(\tilde{x}_{chg}, x_{chg}) \end{aligned} \tag{3}$$

Following [18], the planning loss comprises two parts: 1) Mode prediction loss. m different modes of trajectories of neighbor agents are represented by Gaussian mixture model (GMM). For each mode, at any given timestamp t, its characteristics are delineated by a mean μ^t and covariance σ^t forming a Gaussian distribution. The best mode m^* is identified through alignment with the ground truth and refined via the minimization of negative log-likelihood. 2) Ego Trajectory Prediction loss. Future trajectory points of ego vehicle are predicted and are refined by $l1$ loss. Consequently, the planning loss is as follows:

$$L_{plan} = \sum_t L_{NLL}(\tilde{\mu}^t_{m^*}, \tilde{\sigma}^t_{m^*}, \tilde{p}_{m^*}, \tilde{s}_t) + \sum_t L_1(\tilde{s}_t - s_t) \tag{4}$$

where $\tilde{\mu}^t_{m^*}$, $\tilde{\sigma}^t_{m^*}$, $\tilde{p}_{m^*}$, $\tilde{s}_t$ represents the predicted mean, covariance, probability, and position respectively, corresponding to the best mode m^* at timestamp t, and s_t indicates the ground truth position at timestamp t.

In summary, the complete loss of fine-tuning stage is formulated as:

$$L = L_{align} + L_{plan} \tag{5}$$

5 Experiments

5.1 Experimental Setup

Evaluation Settings In accordance with the nuPlan challenge 2023 [30] settings, we selected 14 official challenging scenario types for training and eval-

Table 1: Evaluation on nuPlan Closed-Loop Reactive Challenges on *Hard20* split. The best results are highlighted in **bold** , while the second-best results are underscored with an underline for clear distinction. *Score*: final score in average. *Drivable*: drivable area compliance. *Direct.*: driving direction compliance. *Comf.*: ego is comfortable. *Prog.*: ego progress along expert route. *Coll.*: no ego at-fault collisions. *Lim.*: speed limit compliance. *TTC*: time to collision within bound.

Method	Score	Drivable	Direct.	Comf.	Prog.	Coll.	Lim.	TTC
UrbanDriver [33]	35.35	75.53	97.12	**98.56**	**85.23**	55.21	81.62	47.84
GCPGP [13]	36.85	81.29	98.20	77.33	46.96	72.30	97.92	68.34
IDM [39]	53.07	84.94	98.02	83.15	64.79	74.01	96.38	60.57
GameFormer [18]	62.05	93.54	98.74	83.15	66.27	86.02	98.19	74.55
PDM-Hybird [10]	64.05	95.34	99.10	75.98	67.93	**87.81**	**99.57**	72.75
PDM-Closed [10]	64.18	95.69	99.10	77.06	68.20	**87.81**	**99.57**	73.47
AsyncDriver	65.00	94.62	98.75	83.15	67.13	85.13	98.15	73.48
AsyncDriver*	**67.48**	**96.77**	**99.10**	83.87	66.30	87.63	98.24	**76.70**

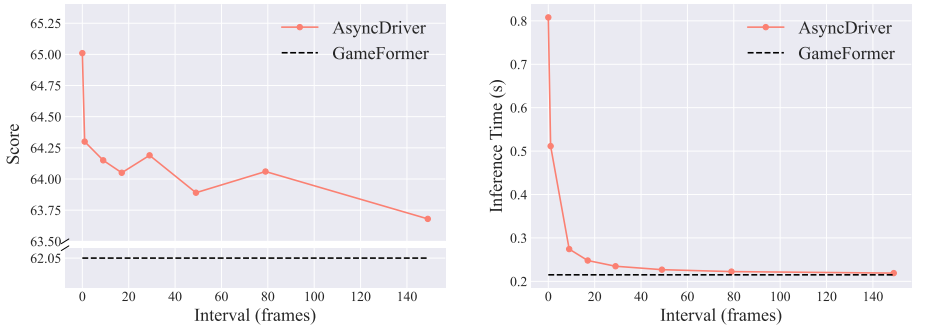
Table 2: Scores per scenario types in evaluation on nuPlan Closed-Loop Reactive Challenges on *Hard20* split. The best results are highlighted in **bold** , while the second-best results are underscored with an underline for clear distinction. Types 0-13 represent the 14 official scenario types of the nuPlan challenge 2023 [30], with specific details provided in the supplementary materials.

Methods	type0	type1	type2	type3	type4	type5	type6	type7	type8	type9	type10	type11	type12	type13
UrbanDriver [33]	69.39	15.78	44.59	7.42	13.7	27.14	0.00	19.44	20.8	23.61	68.33	93.16	33.78	56.39
GCPGP [13]	59.50	35.91	33.6	29.33	42.84	17.24	4.86	32.33	8.22	36.23	71.32	80.46	36.19	23.61
IDM [39]	70.69	44.84	91.54	54.08	50.22	41.71	4.76	53.97	37.53	60.33	83.97	93.03	45.77	12.44
GameFormer [18]	84.32	65.78	83.62	49.03	71.79	36.8	0.00	51.76	55.03	77.24	82.83	98.24	49.41	56.16
PDM-Hybird [10]	**87.68**	69.61	87.20	49.95	82.80	41.32	4.23	54.98	51.72	82.95	80.37	98.40	37.77	71.99
PDM-Closed [10]	87.67	70.93	87.20	49.95	82.80	41.25	4.22	53.53	51.57	**82.96**	80.37	**98.40**	39.95	**72.04**
AsyncDriver	83.53	67.24	83.14	**62.89**	83.28	49.26	5.94	62.53	**65.96**	64.86	**84.88**	96.62	49.60	54.35
AsyncDriver*	83.12	**74.04**	**91.82**	62.42	**85.26**	**53.51**	**6.01**	**65.26**	57.31	77.16	84.02	97.56	**62.18**	49.32

uation. While nuPlan [4] includes $757,844$ scenarios, most simple scenarios are insufficient for critical planner performance assessment, and the vast data volume extends evaluation times. We chose the *Hard20* dataset by randomly selecting 100 scenarios per type from the test set, then using the PDM [10] planner (nuPlan 2023 champion) to retain the 20 lowest-scoring scenarios per type, resulting in a test set of 279 scenarios.

Implementation Details Regarding implementation details, all experiments were conducted in the closed-loop reactive setting, where agents in scenarios could be equipped with an IDM [39] Planner, enabling them to react to ego vehicle's maneuvers. The simulation frequency is $10Hz$, at each iteration, the predicted trajectory has a time horizon of $8s$. We follow the closed-loop metrics proposed in nuPlan challenge, which is detailed in the supplementary material. For model settings, our AsyncDriver is built based on Llama2-13B [38], LoRA

[15] was configured with $R = 8$ and $\alpha = 32$. We use the AdamW optimizer and warm-up with decay scheduler with learning rate 0.0001.



(a) score variation with intervals

(b) average inference time variation with intervals

Fig. 3: Asynchronous Inference. Evaluation of expanding the inference interval to $[1, 9, 17, 29, 49, 79, 149]$ between the LLM and the real-time planner, executing asynchronous inference. Inference time measured on GPU Tesla A30.

5.2 Main Results

Hard20 Evaluation As illustrated in Tab. 1, our approach AsyncDriver achieved the highest performance on *Hard20* compared with existing planners, leading to an improvement of 4.6% over GameFormer [18] about 2.95 in score, even surpassing the current SOTA rule-based planners. For fair comparison, given the significant impact of trajectory refinement and alignment in closed-loop evaluations, we adapt the PDM [10] scorer to our AsyncDriver (denoted as AsyncDriver*), which lead to an improvement of 5.3% and 5.1% over PDM-Hybrid [10] and PDM-Closed [10] respectively, equivalent to approximately 3.43 and 3.30 in score, and an 8.7% increase (approximately 5.43 in score) over the learning-based planner GameFormer. From a different perspective, Tab. 2 illustrates the scores of each individual scenario type on the *hard20* split, as well as a comparison with existing planners. It is evident that our solution has delivered exceptional results in the majority of scenario types.

Quantitative results show that the high-level features extracted by our Scene-Associated Instruction Feature Extraction Module significantly enhance real-time planners' performance in closed-loop evaluation. Detailed metrics reveal that our approach improves drivable area performance by 3.23 points compared to GameFormer, demonstrating superior capability in identifying and navigating viable driving spaces due to advanced scene contextual understanding. Additionally, AsyncDriver* outperforms PDM in Time to Collision (TTC) metrics by 4.39%, approximately 3.23 points, indicating enhanced predictive accuracy,

which is crucial for ensuring safer driving by effectively forecasting and reducing the potential collision scenarios.

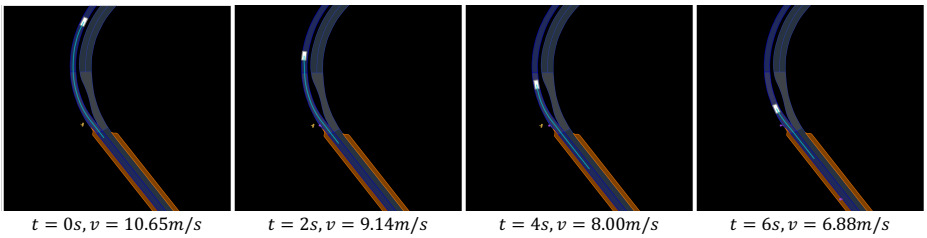
Asynchronous Inference We contend that, particularly for generalized high-level instructions, there exhibits a notable similarity within frames of short intervals. Consequently, given its role in extracting these high-level features, the LLM does not require involvement in the inference process for each frame, which could markedly enhance the inference speed. To explore this, experiments were designed to differentiate the inference frequency of the LLM and real-time planners, and during each LLM inference interval, the previous instruction features are employed to guide the predicted process of the real-time planner. As depicted in Fig. 3, the performance of our approach demonstrates remarkable robustness as the planning interval of the LLM increases, which suggests that LLM is capable of providing a long-term high-level instructions. We observed that even with an interval of 149 frames, meaning only one inference is made within a scenario, it still surpasses GameFormer by more than 1.0 point, while the inference time is nearly at the real-time level. As the inference interval increases, the required inference time drops dramatically, while accuracy remains almost stable. Therefore, by employing a strategy of dense training with asynchronous inference, our method achieves an optimal balance between accuracy and inference speed.

Instruction Following Fig. 4 shows the reactions of our method to different routing instructions, demonstrating its capability in instruction following. Fig. 4a illustrates the outcomes predicted when the scene employs conventional routing instructions. We note that the ego vehicle decelerates slightly, a maneuver that reflects the common sense of slowing down for curves. Nonetheless, given the open road conditions, the ego vehicle sustains a comparatively high velocity. In contrast, Fig. 4b depicts the scenario where *stop* serves as the routing instruction. Remarkably, even without the presence of external obstacles, the ego vehicle promptly executes a braking response to the command, reducing its speed from $10.65m/s$ to $1.06m/s$ within a mere 6 seconds. Consequently, it is evident that our AsyncDriver can function as a linguistic interaction interface, providing the ability of precisely interpreting and following human instructions to circumvent anomalous conditions.

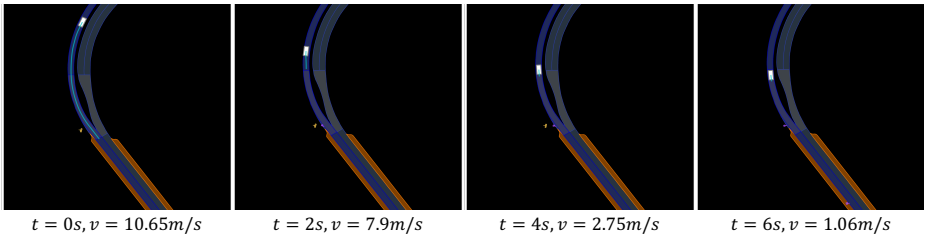
5.3 Ablation Study

In this section, we explore the necessity of LLM and investigate the effectiveness of individual components in AsyncDriver.

Necessity of LLM We conducted experiments by replacing the LLM with transformer blocks of different dimensions (256 and 5120), incorporating learnable routing instruction embeddings. Meanwhile, the Alignment Assistance Module and Adaptive Injection Block remained unchanged. The results, shown in



(a) receiving conventional routing instructions



(b) receiving *stop* instruction

Fig. 4: Visualization of AsyncDriver following human instruction. The light blue line represents ego's trajectory for the next 8 seconds. It contrasts the planning trajectories of the ego receiving conventional routing instructions against a forced *stop* instruction.

Tab. 3, indicate that despite a 20-fold increase in the number of transformer parameters, the performance only marginally improved from 63.27 to 63.59. In contrast, our AsyncDriver achieved a performance of 65.00, highlighting the significant impact of pretrained knowledge from LLM.

Ablation of Components Integrating a simple MLP as a prediction head after the LLM for planning significantly degraded the performance, indicating that simple trajectory regression cannot effectively align multi-modal information, thus failing to leverage the LLM's knowledge for scene reasoning. We replaced the MLP with a real-time planner and progressively added four structures: (i) Adaptive Injection Block, (ii) Alignment Assistance Module, (iii) LoRA, and (iv) pre-trained LoRA. As shown in Tab. 4, each module improved performance, with the Alignment Assistance Module and pre-trained LoRA weights contributing the most, yielding score increases of 0.94 and 0.97, respectively.

6 Conclusions

In this paper, we introduce AsyncDriver, a new asynchronous LLM-enhanced, closed-loop framework for autonomous driving. By aligning vectorized scene information with a series of routing instructions to form multi-modal features, we fully leverage LLM's capability for scene reasoning, extracting scene-associated instruction features as guidance. Through the proposed Adaptive Injection Block,

Table 3: Replacing LLM with non-pretrained cross attention. Utilize map information and learnable instruction embeddings to perform cross-attention, replacing the features generated by the LLM in the Scene-Associated Instruction Feature Extraction Module.

Hidden Feature Dimensions	Instruction Embedding	LLM	Score
256	✓		63.27
5120	✓		63.59
5120		✓	**65.00**

Table 4: Component ablation study. MLP, RT-Planner, Ada, Align, LoRA, and LoRAPre represent i) direct regression of waypoints using MLP, ii) integration of a real-time planner, iii) Adaptive Injection Block, iv) Alignment Assistance Module, v) incorporation of LoRA, and vi) pre-trained LoRA with *Reasoning1K*, respectively.

MLP	RT-Planner	Ada	Align	LoRA	LoRAPre	Score
✓						33.91
	✓					62.01
	✓	✓				62.84
	✓	✓	✓			63.78
	✓	✓	✓	✓		64.03
	✓	✓	✓	✓	✓	**65.00**

we achieve the integration of series of routing information into any transformer-based real-time planner, enhancing its ability to understand and follow language instructions, and achieving outstanding closed-loop performance in nuPlan's challenging scenarios. Notably, owing to the structural design of our method, it supports asynchronous inference between the LLM and the real-time planner. Experiments show that our approach significantly increases inference speed with minimal loss in accuracy, reducing the computational costs introduced by LLM.

Broader Impacts And Limitations. Should the method prove successful, the proposed asynchronous inference scheme could significantly enhance the prospects for integrating LLMs into the practical application within the autonomous driving sector. Nevertheless, while this research has employed LLMs, it falls short of substantiating their generalization properties for the planning task. Future endeavors aim to rigorously assess the generalization and transfer potential of LLMs in in vectorized scenarios.

Acknowledgements

This research is supported in part by the National Science and Technology Major Project (No. 2022ZD0115502), the National Natural Science Foundation of China (No. 62122010, U23B2010), Zhejiang Provincial Natural Science Foundation of China (No. LDT23F02022F02), Beijing Natural Science Foundation (No. L231011), Beihang World TOP University Cooperation Program, and Lenovo Research.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., et al.: Odin: Team victortango's entry in the darpa urban challenge. Journal of field Robotics **25**(8), 467–492 (2008)
3. Baidu: Apollo auto. `https://github.com/ApolloAuto/apollo` (July 2019)
4. Caesar, H., Kabzan, J., Tan, K.S., Fong, W.K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., Omari, S.: nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. arXiv preprint arXiv:2106.11810 (2021)
5. Chen, L., Zhang, Y., Ren, S., Zhao, H., Cai, Z., Wang, Y., Wang, P., Liu, T., Chang, B.: Towards end-to-end embodied decision making via multi-modal large language model: Explorations with gpt4-vision and beyond. arXiv preprint arXiv:2310.02071 (2023)
6. Chen, L., Li, Y., Huang, C., Li, B., Xing, Y., Tian, D., Li, L., Hu, Z., Na, X., Li, Z., et al.: Milestones in autonomous driving and intelligent vehicles: Survey of surveys. IEEE Transactions on Intelligent Vehicles **8**(2), 1046–1056 (2022)
7. Chen, L., Sinavski, O., Hünermann, J., Karnsund, A., Willmott, A.J., Birch, D., Maund, D., Shotton, J.: Driving with llms: Fusing object-level vector modality for explainable autonomous driving. arXiv preprint arXiv:2310.01957 (2023)
8. Cui, C., Ma, Y., Cao, X., Ye, W., Wang, Z.: Receive, reason, and react: Drive as you say with large language models in autonomous vehicles. arXiv preprint arXiv:2310.08034 (2023)
9. Cui, C., Ma, Y., Cao, X., Ye, W., Zhou, Y., Liang, K., Chen, J., Lu, J., Yang, Z., Liao, K.D., et al.: A survey on multimodal large language models for autonomous driving. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 958–979 (2024)
10. Dauner, D., Hallgarten, M., Geiger, A., Chitta, K.: Parting with misconceptions about learning-based vehicle motion planning. arXiv preprint arXiv:2306.07962 (2023)
11. Fan, H., Zhu, F., Liu, C., Zhang, L., Zhuang, L., Li, D., Zhu, W., Hu, J., Li, H., Kong, Q.: Baidu apollo em motion planner. arXiv preprint arXiv:1807.08048 (2018)
12. Fu, D., Li, X., Wen, L., Dou, M., Cai, P., Shi, B., Qiao, Y.: Drive like a human: Rethinking autonomous driving with large language models. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 910–919 (2024)
13. Hallgarten, M., Stoll, M., Zell, A.: From prediction to planning with goal conditioned lane graph traversals. arXiv preprint arXiv:2302.07753 (2023)

14. Han, W., Guo, D., Xu, C.Z., Shen, J.: Dme-driver: Integrating human decision logic and 3d scene perception in autonomous driving. arXiv preprint arXiv:2401.03641 (2024)
15. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
16. Hu, S., Chen, L., Wu, P., Li, H., Yan, J., Tao, D.: St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In: European Conference on Computer Vision. pp. 533–549. Springer (2022)
17. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al.: Planning-oriented autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17853–17862 (2023)
18. Huang, Z., Liu, H., Lv, C.: Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. arXiv preprint arXiv:2303.05760 (2023)
19. Jin, Y., Shen, X., Peng, H., Liu, X., Qin, J., Li, J., Xie, J., Gao, P., Zhou, G., Gong, J.: Surrealdriver: Designing generative driver agent simulation framework in urban contexts based on large language model. arXiv preprint arXiv:2309.13193 (2023)
20. Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.M., Lam, V.D., Bewley, A., Shah, A.: Learning to drive in a day. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8248–8254. IEEE (2019)
21. Kesting, A., Treiber, M., Helbing, D.: General lane-changing model mobil for car-following models. Transportation Research Record: Journal of the Transportation Research Board p. 86–94 (Jan 2007). https://doi.org/10.3141/1999-10, http://dx.doi.org/10.3141/1999-10
22. Keysan, A., Look, A., Kosman, E., Gürsun, G., Wagner, J., Yu, Y., Rakitsch, B.: Can you text what is happening? integrating pre-trained language encoders into trajectory prediction models for autonomous driving. arXiv preprint arXiv:2309.05282 (2023)
23. Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., et al.: A perception-driven autonomous urban vehicle. Journal of Field Robotics **25**(10), 727–774 (2008)
24. Li, Z., Nie, F., Sun, Q., Da, F., Zhao, H.: Boosting offline reinforcement learning for autonomous driving with hierarchical latent skills. arXiv preprint arXiv:2309.13614 (2023)
25. Liu, J., Hang, P., Qi, X., Wang, J., Sun, J.: Mtd-gpt: A multi-task decision-making gpt model for autonomous driving at unsignalized intersections. In: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). pp. 5154–5161. IEEE (2023)
26. Ma, Y., Cao, Y., Sun, J., Pavone, M., Xiao, C.: Dolphins: Multimodal language model for driving. arXiv preprint arXiv:2312.00438 (2023)
27. Ma, Y., Cui, C., Cao, X., Ye, W., Liu, P., Lu, J., Abdelraouf, A., Gupta, R., Han, K., Bera, A., et al.: Lampilot: An open benchmark dataset for autonomous driving with language model programs. arXiv preprint arXiv:2312.04372 (2023)
28. Mao, J., Qian, Y., Zhao, H., Wang, Y.: Gpt-driver: Learning to drive with gpt. arXiv preprint arXiv:2310.01415 (2023)
29. Mao, J., Ye, J., Qian, Y., Pavone, M., Wang, Y.: A language agent for autonomous driving. arXiv preprint arXiv:2311.10813 (2023)

30. Motional: nuplan challange. https://github.com/motional/nuplan-devkit (2023)
31. Nie, M., Peng, R., Wang, C., Cai, X., Han, J., Xu, H., Zhang, L.: Reason2drive: Towards interpretable and chain-based reasoning for autonomous driving. arXiv preprint arXiv:2312.03661 (2023)
32. Renz, K., Chitta, K., Mercea, O.B., Koepke, A., Akata, Z., Geiger, A.: Plant: Explainable planning transformers via object-level representations. arXiv preprint arXiv:2210.14222 (2022)
33. Scheel, O., Bergamini, L., Wolczyk, M., Osiński, B., Ondruska, P.: Urban driver: Learning to drive from real-world demonstrations using policy gradients. In: Conference on Robot Learning. pp. 718–728. PMLR (2022)
34. Sha, H., Mu, Y., Jiang, Y., Chen, L., Xu, C., Luo, P., Li, S.E., Tomizuka, M., Zhan, W., Ding, M.: Languagempc: Large language models as decision makers for autonomous driving. arXiv preprint arXiv:2310.03026 (2023)
35. Shao, H., Hu, Y., Wang, L., Waslander, S.L., Liu, Y., Li, H.: Lmdrive: Closed-loop end-to-end driving with large language models. arXiv preprint arXiv:2312.07488 (2023)
36. Sharan, S., Pittaluga, F., Chandraker, M., et al.: Llm-assist: Enhancing closed-loop planning with language-based reasoning. arXiv preprint arXiv:2401.00125 (2023)
37. Sima, C., Renz, K., Chitta, K., Chen, L., Zhang, H., Xie, C., Luo, P., Geiger, A., Li, H.: Drivelm: Driving with graph visual question answering. arXiv preprint arXiv:2312.14150 (2023)
38. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
39. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical Review E p. 1805–1824 (Jul 2002). https://doi.org/10.1103/physreve.62.1805, http://dx.doi.org/10.1103/physreve.62.1805
40. Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., et al.: Autonomous driving in urban environments: Boss and the urban challenge. Journal of field Robotics **25**(8), 425–466 (2008)
41. Wang, S., Zhu, Y., Li, Z., Wang, Y., Li, L., He, Z.: Chatgpt as your vehicle co-pilot: An initial attempt. IEEE Transactions on Intelligent Vehicles (2023)
42. Wang, W., Xie, J., Hu, C., Zou, H., Fan, J., Tong, W., Wen, Y., Wu, S., Deng, H., Li, Z., et al.: Drivemlm: Aligning multi-modal large language models with behavioral planning states for autonomous driving. arXiv preprint arXiv:2312.09245 (2023)
43. Wang, Y., Jiao, R., Lang, C., Zhan, S.S., Huang, C., Wang, Z., Yang, Z., Zhu, Q.: Empowering autonomous driving with large language models: A safety perspective. arXiv preprint arXiv:2312.00812 (2023)
44. Wen, L., Fu, D., Li, X., Cai, X., Ma, T., Cai, P., Dou, M., Shi, B., He, L., Qiao, Y.: Dilu: A knowledge-driven approach to autonomous driving with large language models. arXiv preprint arXiv:2309.16292 (2023)
45. Xu, Z., Zhang, Y., Xie, E., Zhao, Z., Guo, Y., Wong, K.K., Li, Z., Zhao, H.: Drivegpt4: Interpretable end-to-end autonomous driving via large language model. arXiv preprint arXiv:2310.01412 (2023)
46. Yang, Z., Jia, X., Li, H., Yan, J.: A survey of large language models for autonomous driving. arXiv preprint arXiv:2311.01043 (2023)

47. Yuan, J., Sun, S., Omeiza, D., Zhao, B., Newman, P., Kunze, L., Gadd, M.: Rag-driver: Generalisable driving explanations with retrieval-augmented in-context learning in multi-modal large language model. arXiv preprint arXiv:2402.10828 (2024)
48. Zhou, X., Liu, M., Zagar, B.L., Yurtsever, E., Knoll, A.C.: Vision language models in autonomous driving and intelligent transportation systems. arXiv preprint arXiv:2310.14414 (2023)