sensors

Article

Bidirectional Planning for Autonomous Driving Framework with Large Language Model

Zhikun Ma [1,*], Qicong Sun [2] and Takafumi Matsumaru [1]

[1] Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan; matsumaru@waseda.jp
[2] Singapore General Hospital, Singapore 169608, Singapore; sun.qicong@singhealth.com.sg
* Correspondence: zhikun@akane.waseda.jp

Abstract: Autonomous navigation systems often struggle in dynamic, complex environments due to challenges in safety, intent prediction, and strategic planning. Traditional methods are limited by rigid architectures and inadequate safety mechanisms, reducing adaptability to unpredictable scenarios. We propose SafeMod, a novel framework enhancing safety in autonomous driving by improving decision-making and scenario management. SafeMod features a bidirectional planning structure with two components: forward planning and backward planning. Forward planning predicts surrounding agents' behavior using text-based environment descriptions and reasoning via large language models, generating action predictions. These are embedded into a transformer-based planner that integrates text and image data to produce feasible driving trajectories. Backward planning refines these trajectories using policy and value functions learned through Actor–Critic-based reinforcement learning, selecting optimal actions based on probability distributions. Experiments on CARLA and nuScenes benchmarks demonstrate that SafeMod outperforms recent planning systems in both real-world and simulation testing, significantly improving safety and decision-making. This underscores SafeMod's potential to effectively integrate safety considerations and decision-making in autonomous driving.

Keywords: autonomous driving; multi-modal language model; decision-making

Citation: Ma, Z.; Sun, Q.; Matsumaru, T. Bidirectional Planning for Autonomous Driving Framework with Large Language Model. *Sensors* **2024**, *24*, 6723. https://doi.org/10.3390/s24206723

Academic Editor: Felipe Jiménez

Received: 3 October 2024
Revised: 14 October 2024
Accepted: 16 October 2024
Published: 19 October 2024

1. Introduction

Autonomous driving systems are often criticized for their lack of transparency, leading to them being perceived as a "black box". This opacity raises significant concerns about safety and reliability, as both users and regulators find it difficult to understand or explain the rationale behind a vehicle's decisions. To address these challenges, recent research efforts have concentrated on enhancing the interpretability and trustworthiness of these systems by analyzing how various factors influence their decision-making. For instance, Li et al. [1] employed experiments with controlled variables and alternative scenarios to identify factors that impact model decisions. Similarly, frameworks like LaMPilot [2] leverage large language models (LLMs) to interpret user intent and translate these inputs into comprehensible driving actions, demonstrating their applicability across different driving conditions. Furthermore, regulatory guidelines increasingly emphasize the need for autonomous vehicles to provide understandable explanations for their decisions, which is crucial for maintaining safety and fostering public trust in these technologies [3].

The introduction of autonomous driving technology marks a transformative step in the evolution of transportation, with the potential to significantly improve efficiency, safety, and accessibility. A driving force behind these advancements is Embedded Artificial Intelligence (EAI), enabling autonomous vehicles to operate effectively in complex environments by adhering to societal norms and accepted behaviors [4,5]. Central to autonomous driving is trajectory planning, which involves the vehicle predicting the real-time behaviors of road users, such as other vehicles, pedestrians, and cyclists, and adapting its movements accordingly. Accurate perception and prediction of the surrounding environment are vital to ensure safe and effective trajectory planning.

Despite notable technological progress, autonomous driving continues to face significant hurdles. A major issue is "occlusions", where obstacles block the vehicle's sensors, impairing accurate environmental perception [6]. Additionally, current systems often struggle in complex scenarios. Various approaches have been proposed to address these challenges: Li et al. [7] applied probabilistic modeling to anticipate pedestrian behavior, while Zhang et al. [5] used dynamic game theory to tackle trajectory planning problems arising from occluded views. However, both methods encountered difficulties in managing the unpredictability and dynamic nature of real-world driving.

Moreover, balancing high navigation performance with stringent safety requirements adds another level of complexity. Many systems either overlook or inadequately incorporate critical safety constraints essential for autonomous vehicles to operate securely in uncertain environments [8]. The lack of robust safety integration, combined with insufficient forward- and backward-looking planning mechanisms, limits their efficacy in real-world applications where both safety and efficiency are imperative.

To address these limitations, we propose SafeMod, a novel framework for modular, safety-enhanced navigation powered by LLMs. SafeMod introduces a flexible, modular architecture with a core principle of bidirectional planning, which involves two key components: forward planning and backward planning. Forward planning and backward planning are two key modules defined by the sequence of the planning process. Forward planning focuses on generating future navigation strategies by analyzing the agent's objectives and environmental context. Backward planning, on the other hand, retrospectively reviews these strategies for potential risks, ensuring that safety is maintained throughout the decision-making process.

We evaluate SafeMod on challenging benchmarks in nuScenes [9] and CARLA [10], where it demonstrates superior safety and task success rates compared to existing state-of-the-art methods. SafeMod's ability to analyze intent, adapt to dynamic environments, and ensure safety at every stage of planning showcases its potential for advancing autonomous navigation.

Our key contributions are:

- Proposing SafeMod, a modular framework for autonomous navigation powered by LLMs, integrating intent inference within the forward planner and including a backward planner for adaptive decision-making.
- Introducing a novel bidirectional planning approach that combines forward strategy generation (via the forward planner and intent analyzer) and backward safety evaluation (via the backward planner), ensuring optimal performance and strict safety compliance in dynamic environments.
- Demonstrating through extensive experiments on nuScenes and CARLA that SafeMod improves navigation success rates and safety performance, outperforming current state-of-the-art methods.

These results show that SafeMod consistently outperforms traditional navigation systems in both success rates and safety compliance. The bidirectional planning mechanism—combining the forward planner with its intent analyzer and the backward planner—allows SafeMod to adapt to complex scenarios while maintaining stringent safety standards.

2. Related Work

2.1. LLMs for Autonomous Driving Decision-Making

Embedded Artificial Intelligence (EAI) is becoming increasingly critical in the development and functioning of autonomous vehicles (AVs). A key focus of recent research has been improving AVs' ability to interact with their environments in human-like ways. For instance, Zhou et al. [11] explored models that mimic human driving behavior, while Sadigh et al. [12] examined how integrating human inputs can better align AV actions with natural driving tendencies. Additionally, Sun et al. [13] highlighted the importance of visual cues in pedestrian interactions for effective decision-making in urban settings. Within this evolving field, two distinct approaches have emerged. The first involves using large language models (LLMs) to comprehend driving scenes through question-answering tasks. The second approach focuses on LLM-driven scene understanding for planning purposes. For example, DriveGPT4 [14] leverages historical video and textual data to predict both answers and control signals. Similarly, LanguageMPC [15] translates perception results and map data into language descriptions to guide planning actions. These advances demonstrate the growing integration of AI techniques in autonomous driving, aiming to create vehicles that are more intuitive and context-aware. These work explores the current landscape of EAI in autonomous vehicles, highlighting key research directions and emerging approaches that aim to enhance the human-like interaction capabilities of AVs.

2.2. Knowledge-Driven Approaches for Autonomous Driving

While data-driven approaches, such as multimodal data fusion [16–18], and the use of imitation learning and reinforcement learning for autonomous driving models [19–22], have achieved remarkable success in both academia and industry, they also face significant challenges. These techniques have enabled autonomous driving technology to gradually integrate into daily life. However, their reliance on the distribution of training data limits their generalization capability. As a result, they often struggle with adaptability and the long-tail phenomenon when applied in less common or varied scenarios [23]. In contrast, human drivers possess strong common sense and adaptability, enabling them to handle unexpected situations with ease. This highlights the need to shift towards knowledge-driven approaches, which emphasize empirical reasoning and inductive learning from the environment [24,25]. Unlike methods based solely on predefined rules or domain-specific data, knowledge-driven techniques acquire general knowledge and evolve over time [26,27]. By incorporating human-like reasoning, these strategies improve performance, interpretability, and safety, particularly in complicated traffic conditions. With the emergence of foundation models, powerful tools like large language models (LLMs) and vision-language models (VLMs) have become prominent in reasoning and decision-making for autonomous driving [28,29]. By training on diverse datasets, these explorations of the transition from data-driven to knowledge-driven approaches in autonomous driving, emphasize the potential of foundation models to enhance reasoning and decision-making capabilities in complex traffic scenarios.

3. Problem Formulation

The goal of autonomous driving is to develop a navigation policy that ensures safety, efficiency, and adaptability in dynamic environments. The navigation task can be framed as a Markov decision process (ADNP), which is defined as a tuple $P = \langle S, A, T, C, \Pi \rangle$.

The state space S encompasses all possible configurations of the environment, including the agent's position (p_{agent}), velocity (v_{agent}), the positions of obstacles (p_{obst}), and road conditions (r_{road}). These elements form a state $s \in S$.

The action space A consists of the agent's possible control actions, such as steering, acceleration, and braking, denoted at each time step by $a_t = (a_{\text{steer}}, a_{\text{accel}}, a_{\text{brake}})$.

The transition function T defines how the environment evolves as a result of the agent's actions. Given a state s_t and an action a_t, the next state s_{t+1} is determined by the function $T(s_t, a_t)$, which models the dynamics of the environment, possibly incorporating differential equations to describe the behavior of the vehicle.

The cost function C assigns a value to each action a taken in state s, representing the associated cost. The total cost over a trajectory $\tau = \{s_0, a_0, \ldots, s_T\}$ is the sum of the costs incurred at each time step, denoted as $J(\tau) = \sum_{t=0}^{T} C(s_t, a_t)$. This function penalizes unsafe actions, such as collisions or speeding. The cost for a specific state–action pair $C(s_t, a_t)$ includes terms for collisions and speed violations.

Finally, in an Autonomous Driving Navigation Problem (ADNP), the primary objective is to determine a policy that minimizes the accumulated cost while ensuring safety and task success. Π is the set of all possible navigation policies $\pi : S \to A$, mapping states to actions. The goal is to find an optimal policy $\pi^* \in \Pi$ that minimizes the total cost:

$$\pi^* = \arg\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{T} C(s_t, \pi(s_t))\right]$$

By solving this optimization problem, the system selects actions that effectively balance safety and performance throughout the navigation task.

Bidirectional Planning Problem. To enhance safety and adaptability, we propose a bidirectional planning framework consisting of two phases:

- **Forward Planning:** Given the current state $s_t \in S$ and the inferred intent I, the forward planner generates a sequence of actions:

  $$\{a_t, a_{t+1}, \ldots, a_{t+k}\} \in A$$

  that optimally navigates towards the goal state s_g. This planning is based on minimizing the expected cost over future states:

  $$\mathbb{E}\left[\sum_{i=t}^{t+k} C(s_i, a_i)\right]$$

- **Backward Planning:** After forward planning, the backward planner evaluates the safety and feasibility of the generated trajectory $\tau = \{s_t, s_{t+1}, \ldots, s_{t+k}\}$. The backward planning phase involves minimizing a risk function $R(\tau)$ to ensure the trajectory is safe:

  $$\min_{\tau} R(\tau) = \sum_{i=t}^{t+k} (\mathbb{1}_{\text{unsafe}}(s_i) + \mathbb{1}_{\text{collision}}(s_i))$$

  The plan is adjusted iteratively to minimize risks, ensuring the trajectory remains within the safe operating bounds:

  $$\hat{\tau} = \arg\min_{\tau} R(\tau)$$

Optimal Policy. The final objective is to output a policy π^* that balances both forward and backward planning considerations. The optimal policy must minimize the combined cost and risk function over the entire trajectory:

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} C(s_t, \pi(s_t)) + R(\tau)\right]$$

This policy not only ensures efficient navigation but also adapts to unforeseen hazards, dynamically adjusting actions based on intent inference and safety evaluations.

4. Methodology

The SafeMod framework (Figure 1) for autonomous navigation uses a modular approach, combining forward and backward planning. During forward planning, the BEV-planning module processes bird's-eye-view (BEV) features using Motion, Map, and Planning Transformers to generate optimal trajectories. A video sense module interprets sensory data from multiple views. Employing a forward-thinking mechanism, it continuously updates the agent's internal state, refining decisions based on real-time environmental changes. Following forward planning, the system moves to backward planning. Backward planning involves a latent space system composed of a representation, transition, and reward model to enhance safety. These models predict and refine trajectories by integrating state, action, and reward data, ensuring optimal and safe actions. The process uses a Q-function to evaluate and select the best candidate states, balancing performance and safety more effectively than prior methods that lack safety constraints.

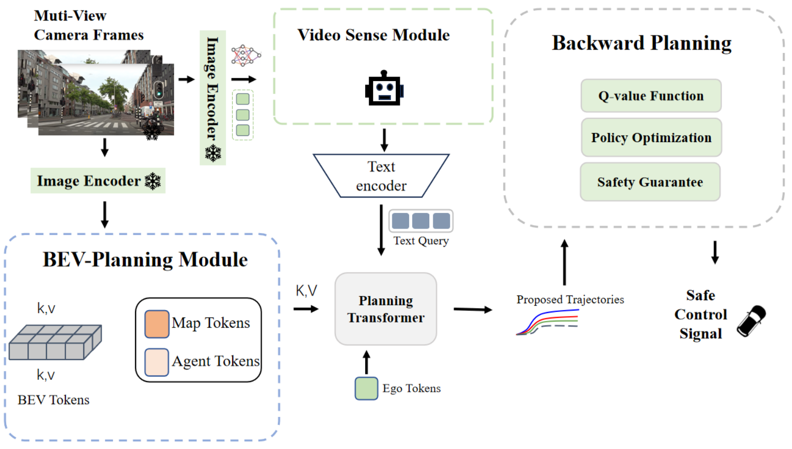


Figure 1. Overall framework of SafeMod. SafeMod takes multi-view image sequences as input, transforms them into BEV embedding and sense description, outputs them and samples one action to control the vehicle.

4.1. Forward Planning

The forward planning mechanism in the updated framework introduces two primary modules: the BEV-planning module and the VLM-based video sense module. Each module plays a crucial role in generating efficient trajectories based on the vehicle's current state and environment.

The BEV-planning module is responsible for creating a bird's-eye view (BEV) representation of the vehicle's surroundings. This spatial layout includes the road structure, nearby obstacles, and dynamic entities, which serve as the foundational elements for trajectory computation. By leveraging the BEV, the system can accurately plan paths that account for the spatial distribution of elements around the vehicle.

The video sense module, based on vision-language models (VLMs), adds a layer of semantic understanding to the planning process. It processes visual inputs from the vehicle's cameras to interpret scene dynamics, object interactions, and contextual information that are not captured in the BEV alone. This semantic enrichment allows the planning mechanism to make more informed decisions by incorporating temporal and context-aware features from the visual data.

The BEV-planning and VLM-based video sense modules engage in collaborative work to enhance trajectory generation. The BEV module provides a spatial blueprint for path planning, while the video sense module refines these trajectories by incorporating semantic and temporal context, leading to more accurate and reliable navigation.

4.1.1. BEV-Planning Module

As shown in Figure 2, the BEV-planning module consists of multiple transformers that work together to process bird's-eye-view (BEV) features to generate strategy in planning transformer. The structure includes:

BEV Feature: The BEV feature extraction provides a comprehensive understanding of the surrounding environment from a BEV perspective. In our approach, we use 2 s of historical information to plan a 3-s future trajectory. For encoding image features, we adopt ResNet50 [30] as the default backbone network.

To enhance the perception range, vectorized mapping and motion prediction are performed for a longitudinal range of 60 m and a lateral range of 30 m. The default settings for BEV queries, map queries, and agent queries are set at 200 × 200, 100 × 20, and 300, respectively. Each map vector query contains 100 vectors, with each vector consisting of

20 map points. The feature dimension and hidden size are configured to 256. To optimize efficiency, the number of encoder and decoder layers for the motion and map modules is reduced from 6 to 3. Additionally, the input image size is reduced from 1280 $\times$ 720 to 640 $\times$ 360 to lower computational load without sacrificing performance.

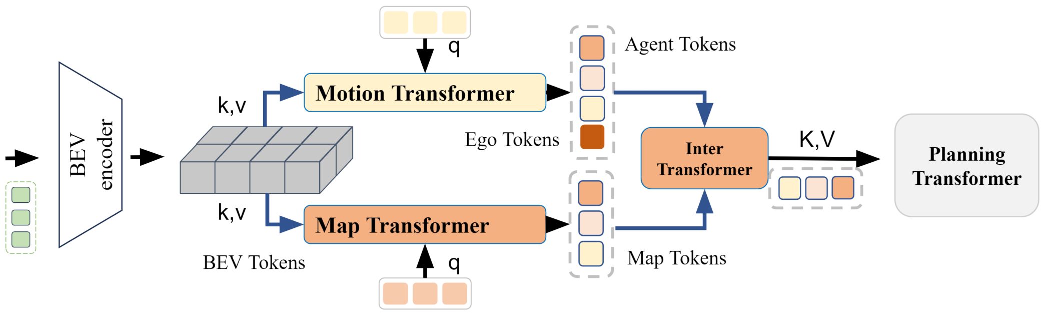


Figure 2. Framework detail of BEV-planning module. The BEV-planning module processes bird's-eye-view (BEV) features through multiple transformers to generate strategic vehicle control actions, incorporating BEV feature extraction, motion and map predictions, inter-query interactions, and final trajectory planning based on high-level driving commands.

Motion Transformer: The Motion Transformer utilizes agent queries, Q_a, for extracting agent-specific information from the common BEV feature representation via deformable attention [31]. Subsequently, an MLP-based decoder network processes these queries to infer agent characteristics, including position, category probability, and heading.

To enhance agent features for more accurate motion prediction, the module facilitates agent-agent and agent-map interactions via an attention mechanism [32,33]. This enables the system to consider agents' relative positions and their environmental relationships. Following these interactions, the module predicts future trajectories for each agent, represented as multi-modal motion vectors $\hat{V}_a \in \mathbb{R}^{N_a \times N_k \times T_f \times 2}$. Here, N_a is the number of predicted agents, N_k the number of modalities, and T_f the number of future timestamps. Each modality corresponds to a specific driving intention and outputs a probability score indicating its likelihood.

Map Transformer: The Map Transformer integrates map information to enhance navigation accuracy by accounting for road geometry and traffic rules. It employs a set of map queries, Q_m [34], to extract relevant map information from the shared BEV feature map. The system then predicts map vectors, $\hat{V}_m \in \mathbb{R}^{N_m \times N_p \times 2}$, assigning a class score to each. Here, N_m is the number of predicted map vectors, and N_p denotes the number of points in each vector.

Inter Transformer: The BEV-planning module uses a randomly initialized ego query Q_{ego} to learn implicit scene features essential for planning. Initially, the ego query interacts with agent queries Q_a via a Transformer decoder [35], utilizing position embeddings encoded by an MLP (PE$_1$) from the positions of the ego vehicle p_{ego} and agents p_a. This interaction captures relative spatial relationships:

$$Q'_{\text{ego}} = \text{TransformerDecoder}(Q_{\text{ego}}, Q_a, Q_a, \text{PE}_1(p_{\text{ego}}), \text{PE}_1(p_a)).$$

Subsequently, the updated ego query Q'_{ego} interacts with map queries Q_m to incorporate static scene information. Positions of the ego vehicle and map elements p_m are encoded using another MLP (PE$_2$), enriching Q''_{ego} with both dynamic and static environmental details for comprehensive planning.

Planning Transformer: The Planning Transformer synthesizes outputs from previous modules to generate the final trajectory for planning. Operating without HD maps, it relies on a high-level driving command c to guide navigation, using commands like *turn left*, *turn right*, and *go straight* as in common practices [36,37].

The planning head inputs the concatenated ego features $f_{\text{ego}} = [Q'_{\text{ego}}, Q''_{\text{ego}}, s_{\text{ego}}]$ and the driving command c to produce the planning trajectory $\hat{V}_{\text{ego}} \in \mathbb{R}^{T_f \times 2}$ using a simple MLP-based decoder.

4.1.2. Video Sense Module

As shown in Figure 3, the video sense module is responsible for interpreting sensory data and updating the internal state of the agent in real-time. This module includes:

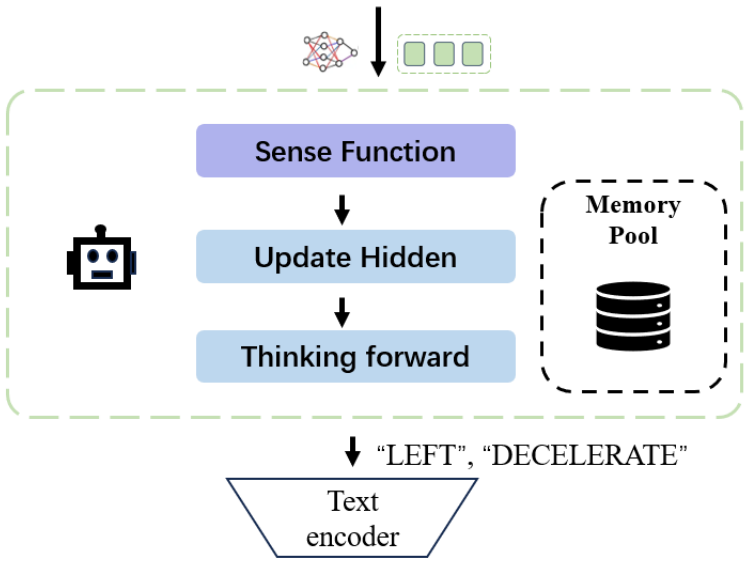


Figure 3. Overall framework of video sense module. The video sense module processes video inputs to extract structured sensory information and generate multi-turn dialogues for enhanced scene understanding, utilizing a pre-trained video encoder to obtain video embeddings, a cross-modality projector for alignment with language embeddings, and a perception module to continuously update the agent's hidden states based on the extracted data, ultimately predicting context-specific actions for navigation.

Video data input: To initiate the process, we mathematically formalize the extraction of data from the video input as follows:

$$D(t) = \text{extract\_data}(V(t)) \tag{1}$$

Here, $V(t)$ represents the raw video input at discrete time t, capturing the dynamic driving environment. The function `extract_data` translates visual cues from the video into a structured format for further processing.

For the extraction, we utilized the pre-trained LanguageBind video encoder [38], based on a ViT-B/32 vision transformer [39], as the frozen visual backbone f_v. Given an input video frame sequence $V_i = \{v_i^1, v_i^2, \ldots, v_i^k\} \in \mathbb{R}^{3\times k\times 224\times 224}$, the video is split into temporal sequences, with each patch sharing the same spatial location across different frames.

These patches are transformed through a linear projection, and a vision transformer is employed to generate video embeddings $z_{vo} \in \mathbb{R}^{2048\times 1024}$. The video encoder is pre-trained using video-language contrastive learning techniques, specifically CLIP4clip [40], without further fine-tuning.

Next, we leverage a two-layer MLP cross-modality projector to project and align the encoded video embedding z_{vo} with the language token embeddings $z_v \in \mathbb{R}^{2048\times 4096}$. This alignment step is crucial for connecting the visual and language features in a unified space. The projector f_p is defined as:

$$f_p(z_{vo}) = \text{GELU}(W_2 \cdot \text{GELU}(W_1 \cdot z_{vo})) \tag{2}$$

Here, GELU [41] is used as the activation function to introduce non-linearity, and the projector is trained in a two-stage process to refine the alignment between modalities.

Sense Function: Building on Video-LLaVA to generate multi-turn dialogues based on the extracted caption information and image content. These dialogues address various tasks, including object counting, color recognition, relative positioning, and OCR-based text extraction. The process is formalized as follows:

$$D_{\text{dialogue}} = \text{GenerateDialogue}(C, I), \tag{3}$$

where C represents the caption information extracted from the image I, and D_{dialogue} denotes the generated multi-turn dialogue. By engaging in iterative dialogue tasks, this method enhances the model's ability to recognize long-tail objects and improves its understanding of complex visual scenes.

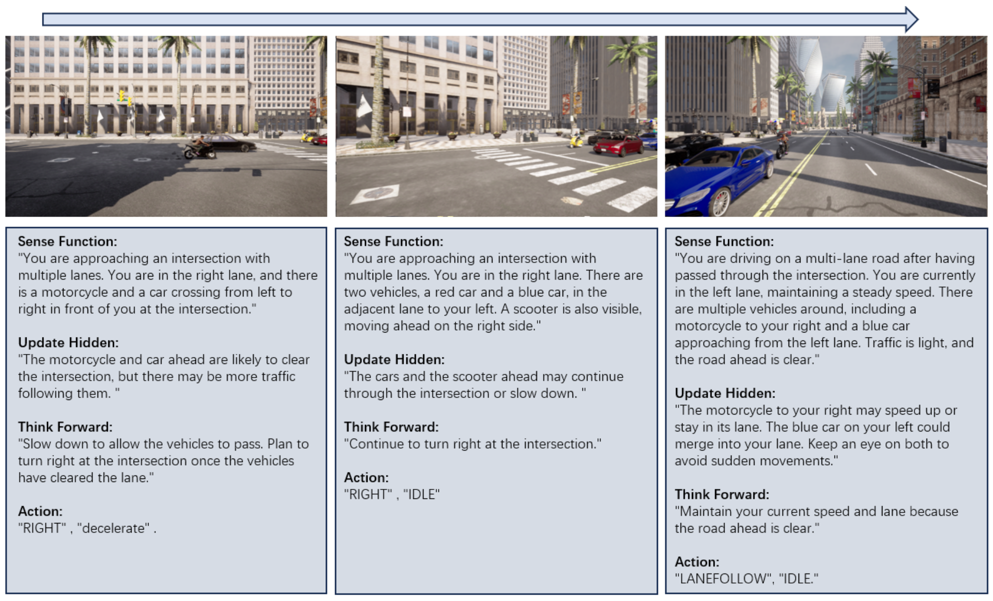
Update Hidden: After the video data extraction step, we proceed to process the multi-view video inputs V_i for navigation. The extracted data $D(t)$ from each video stream at time t is fed into a perception module, which generates the necessary sensory information $P(t)$. This sensory information plays a crucial role in the agent's decision-making and state updating process. Mathematically, the process is represented as follows:

$$P(t) = f_p(D(t)) \tag{4}$$

Here, f_p represents the perception module, responsible for transforming the extracted video data $D(t)$ into sensory information $P(t)$, such as object detection, scene understanding, or environmental awareness. This sensory information is then used to update the internal hidden states of the agent. Specifically, the hidden state h_t at time t is computed by using the previous hidden state h_{t-1} and the new sensory information $P(t)$, as follows:

$$h_t = \text{UpdateState}(h_{t-1}, P(t)) \tag{5}$$

This continuous update process allows the agent to dynamically adjust its internal state based on the evolving environment captured by the multi-view video inputs. By maintaining a consistent flow of sensory information through the perception module, the agent can make informed decisions and interact with the environment in real time. Figures 4 and 5 display, from left to right, the complete reasoning processes of the video sense module in lane changing and at intersections in CARLA, respectively.



Sense Function:
"You are approaching an intersection with multiple lanes. You are in the right lane, and there is a motorcycle and a car crossing from left to right in front of you at the intersection."

Update Hidden:
"The motorcycle and car ahead are likely to clear the intersection, but there may be more traffic following them. "

Think Forward:
"Slow down to allow the vehicles to pass. Plan to turn right at the intersection once the vehicles have cleared the lane."

Action:
"RIGHT" , "decelerate" .

Sense Function:
"You are approaching an intersection with multiple lanes. You are in the right lane. There are two vehicles, a red car and a blue car, in the adjacent lane to your left. A scooter is also visible, moving ahead on the right side."

Update Hidden:
"The cars and the scooter ahead may continue through the intersection or slow down. "

Think Forward:
"Continue to turn right at the intersection."

Action:
"RIGHT" , "IDLE"

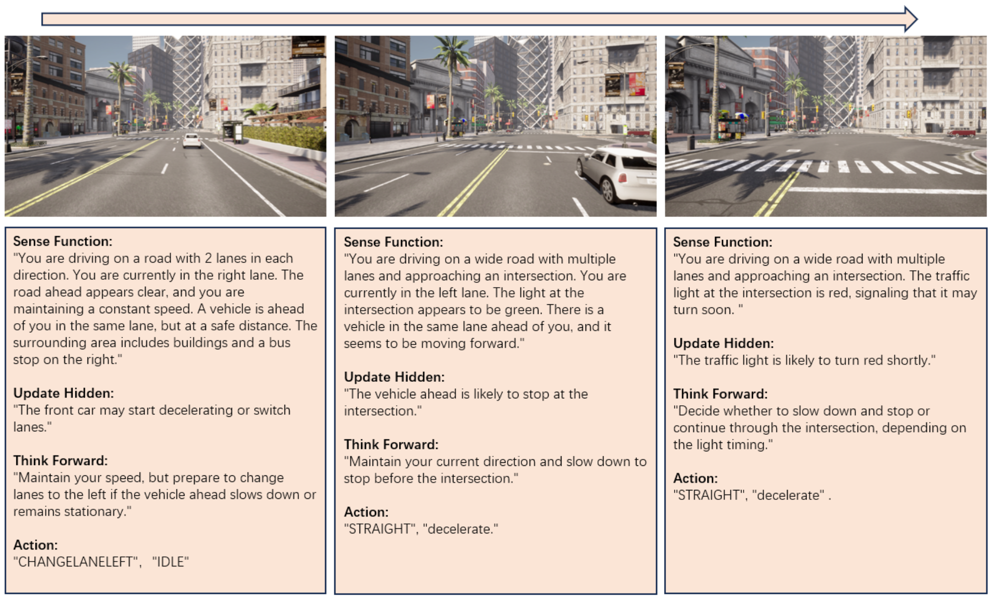
Sense Function:
"You are driving on a multi-lane road after having passed through the intersection. You are currently in the left lane, maintaining a steady speed. There are multiple vehicles around, including a motorcycle to your right and a blue car approaching from the left lane. Traffic is light, and the road ahead is clear."

Update Hidden:
"The motorcycle to your right may speed up or stay in its lane. The blue car on your left could merge into your lane. Keep an eye on both to avoid sudden movements."

Think Forward:
"Maintain your current speed and lane because the road ahead is clear."

Action:
"LANEFOLLOW", "IDLE."

Figure 4. Intersection traffic negotiation. The module identifies the vehicles at the intersection, anticipates their behavior, and selects a safe right-turn maneuver while avoiding collisions. It analyzes the scooter and cars' positions, predicting their paths and adjusting speed or direction to maintain safe distances. After passing the intersection, the framework continues monitoring for lane changes or sudden movements, ensuring smooth traffic flow and preventing conflicts.



Sense Function:
"You are driving on a road with 2 lanes in each direction. You are currently in the right lane. The road ahead appears clear, and you are maintaining a constant speed. A vehicle is ahead of you in the same lane, but at a safe distance. The surrounding area includes buildings and a bus stop on the right."

Update Hidden:
"The front car may start decelerating or switch lanes."

Think Forward:
"Maintain your speed, but prepare to change lanes to the left if the vehicle ahead slows down or remains stationary."

Action:
"CHANGELANELEFT", "IDLE"

Sense Function:
"You are driving on a wide road with multiple lanes and approaching an intersection. You are currently in the left lane. The light at the intersection appears to be green. There is a vehicle in the same lane ahead of you, and it seems to be moving forward."

Update Hidden:
"The vehicle ahead is likely to stop at the intersection."

Think Forward:
"Maintain your current direction and slow down to stop before the intersection."

Action:
"STRAIGHT", "decelerate."

Sense Function:
"You are driving on a wide road with multiple lanes and approaching an intersection. The traffic light at the intersection is red, signaling that it may turn soon. "

Update Hidden:
"The traffic light is likely to turn red shortly."

Think Forward:
"Decide whether to slow down and stop or continue through the intersection, depending on the light timing."

Action:
"STRAIGHT", "decelerate" .

Figure 5. Lane changing. The module identifies the vehicle ahead while driving on a multi-lane road, anticipates its stationary behavior, and selects a safe lane change maneuver to the left. It analyzes the traffic light at the intersection and the presence of vehicles, predicting possible stops and adjusting speed accordingly to maintain safe movement.

Thinking Forward: In this phase, the system generates potential trajectories based on current observations and predicted future states. Specifically, we utilize the multi-turn dialogue output D_{dialogue} as the descriptive prompt input, while the hidden state h_t serves as the input for the intentions prediction module. These inputs guide the model in predicting a set of discrete available actions, which can vary depending on the context. The data output at this stage is shown in Figure 6.

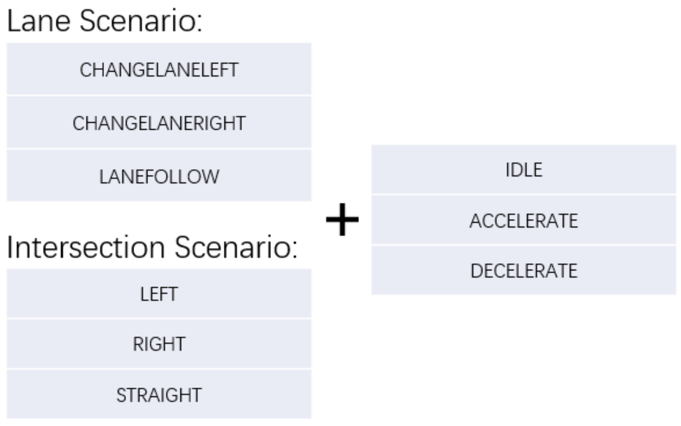


Figure 6. Example of action output.

In a normal driving scenario, possible actions include:

- `CHANGELANELEFT`: Move one lane to the left.
- `CHANGELANERIGHT`: Move one lane to the right.
- `LANEFOLLOW`: Continue in the current lane.

At an intersection, the actions are:

- `LEFT`: Turn left at the intersection.
- `RIGHT`: Turn right at the intersection.
- `STRAIGHT`: Keep straight at the intersection.

In both cases, additional decisions such as `IDLE`, `ACCELERATE`, or `DECELERATE` are also possible. For example, the final action output may look like `LANEFOLLOW, IDLE` or `STRAIGHT, DECELERATE`. The process can be described as follows:

$$A_{\text{atomic}} = \text{PredictAction}(h_t, D_{\text{dialogue}}), \tag{6}$$

where A_{atomic} represents the predicted atomic action with two elements, and PredictAction combines both the hidden state and the dialogue-based description to produce a reasoning step followed by a single atomic action. This atomic action is then passed to the text encoder, which encodes the action as a tokenized query:

$$Q_{\text{action}} = \text{TextEncoder}(A_{\text{atomic}}), \tag{7}$$

Finally, the tokenized query Q_{action} is aligned within the planning transformer to ensure proper integration into the trajectory planning process. This completes the process of generating reasoning-based atomic actions that are used for planning and control.

Memory Pool: To enhance decision-making based on past experiences, we implemented a two-stage search strategy to retrieve the most similar past driving scenario that matches the current query scenario. This allows the agent to learn from historical data and apply relevant knowledge to new situations.

In the first phase, for each prior scenario i, a vectorized key $k_i \in \mathbb{R}^{1\times(n_e+n_g+n_h)}$generated. This key consolidates the ego-states $e_i \in \mathbb{R}^{1\times n_e}$, mission goals $g_i \in \mathbb{R}^{1\times n_g}$, and the trajectories from historical information $h_i \in \mathbb{R}^{1\times n_h}$, make them into a single vector, and then prior scenarios forms a key tensor$K \in \mathbb{R}^{N\times(n_e+n_g+n_h)}$:

$$K = \{[e_i, g_i, h_i] \mid i = \{1, 2, \dots, N\}\}. \tag{8}$$

Similarly, the current query scenario is vectorized as $Q = [e, g, h] \in \mathbb{R}^{1\times(n_e+n_g+n_h)}$, representing the ego-states, mission goals, and historical trajectories of the current scenario.

Next, we compute the similarity scores $S \in \mathbb{R}^N$ between the query scenario Q and each of the past scenarios represented by K:

$$S = Q\Lambda K^\top, \tag{9}$$

where $\Lambda = \text{diag}(\lambda_e, \lambda_g, \lambda_h) \in \mathbb{R}^{(n_e+n_g+n_h)\times(n_e+n_g+n_h)}$ is a diagonal matrix that assigns weights to the different components (ego-states, goals, and trajectories), reflecting their relative importance in the search process.

After calculating the similarity scores, the top-K scenarios with the highest scores are selected as candidates for the second-stage search, which refines the selection process to identify the most relevant past experience.

In the second stage, we employed an existing algorithm similar to those used in [42,43]. This algorithm utilizes vector databases, where both the input query and each memory record are encoded into embeddings. A K-nearest neighbors (K-NN) search is then performed in the embedding space to retrieve the top-K most similar records.

However, due to the diverse and dynamic nature of driving scenarios, embedding-based searches may struggle to generalize across varied contexts, leading to suboptimal retrieval of relevant experiences. To address this limitation, we integrate human knowledge and past driving experiences through a cognitive memory system. This system enhances retrieval by considering contextual nuances that pure embedding methods might overlook, thereby improving the system's ability to generalize across diverse driving environments.

4.2. Backward Planning

Building on the planning transformer's output of $P(z \mid s)$, we developed a latent space system composed of previous models for backward planning to ensure safety. This system integrates state annotations and is expected to improve the accuracy and efficiency of the planning process.

- **Representation Model:** This model encodes the latent state based on past states, actions, and observations, expressed as:

$$p(s_t \mid s_{t-1}, a_{t-1}, o_t), \tag{10}$$

  where s_t represents the state at time t, a_{t-1} is the previous action, and o_t is the observation at time t.
- **Transition Model:** The transition model predicts future states through a Gaussian distribution, ensuring consistency between the predicted and actual dynamics:

$$q(s_t \mid s_{t-1}, a_{t-1}) \sim \mathcal{N}(\mu, \sigma^2), \tag{11}$$

  with KL divergence applied to measure the discrepancy:

$$\mathcal{D}_{KL}(q(s_t \mid s_{t-1}, a_{t-1}) \parallel p(s_t)). \tag{12}$$

- **Reward Model:** This model calculates the expected rewards for each state, optimizing agent actions by:

$$q(r_t \mid s_t). \tag{13}$$

In our latent state space, p represents the true state distribution from environmental interaction, while q reflects the predicted state from the imagination model. The agent generates future trajectories and iteratively selects optimal behaviors while avoiding unsafe paths.

As illustrated in Figure 7, candidate trajectories are evaluated using a Q-function, $\arg\max Q(s,z)$, where z is the trajectory and s the state. The agent chooses the trajectory that maximizes rewards while maintaining safety. Unlike Dreamer [44], our method incorporates safety constraints, balancing reward and risk effectively.

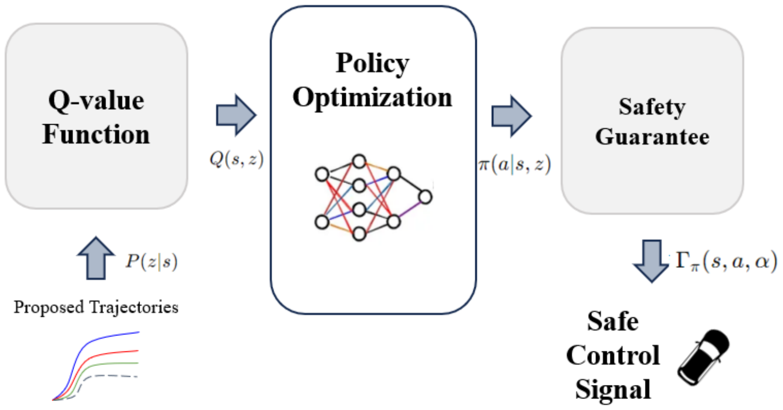


Figure 7. Policy generation in backward planning using Q-function optimization.

4.2.1. Q-Value Function

The Q-value function evaluates the expected cumulative reward for a trajectory:

$$Q(s,z) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, z_t)\right], \tag{14}$$

where γ is the discount factor and r is the reward at time step t. The agent selects actions based on distributional reinforcement learning, focusing on both value and cost expectations [45].

4.2.2. Policy Optimization

To ensure real-world safety (e.g., in autonomous driving), we introduce a Lagrangian method with constraints to balance reward maximization and safety:

$$\max_{\pi} \mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t)\right], \quad \text{s.t.} \quad \mathbb{E}\left[\sum_t \gamma^t c(s_t, a_t)\right] \leq d, \tag{15}$$

where $c(s_t, a_t)$ represents the cost associated with unsafe actions. A control barrier function adjusts the policy's risk sensitivity:

$$h(s_{t+m}) \leq (1-\alpha)h(s_t), \tag{16}$$

where α controls the degree of conservativeness.

4.2.3. Safety Guarantee

We use control barrier functions to enforce safety constraints dynamically, optimizing the policy until the risk metric Γ_π satisfies:

$$\Gamma_\pi(s, a, \alpha) = Q^c_\pi(s, a) + \alpha^{-1}\phi\left(\Phi^{-1}(\alpha)\right)\sqrt{V^c_\pi(s, a)}, \tag{17}$$

where Q^c_π represents the cost critic, and V^c_π the variance. The agent balances safety and reward by adjusting its policy accordingly.

4.2.4. Policy Optimization

Safety constraints are introduced using a Lagrangian method, ensuring that the policy maximizes rewards while adhering to safety requirements:

$$\max_{\pi} \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}}\left[\sum_t \gamma^t r(s_t, a_t)\right], \quad \text{s.t.} \; \mathbb{E}\left[\sum_t \gamma^t c(s_t, a_t)\right] \leq d. \tag{18}$$

We further employ a control barrier function to dynamically adjust risk values in decision-making:

$$h(s_{t+m}) \leq (1-\alpha)h(s_t), \tag{19}$$

where α controls the agent's risk sensitivity.

This approach, combined with distributional RL, enables safe exploration by evaluating the trade-off between reward and risk.

5. Algorithm Overview

Algorithm 1 outlines the SafeMod framework. The agent begins by analyzing the environment using the LLM to infer intent. In the forward planning phase, the system generates an initial trajectory based on a series of perception and prediction modules. This trajectory is then passed to the backward planning phase, where it is evaluated for safety and feasibility using a set of optimization and safety guarantee functions.

Algorithm 1: SafeMod Framework with Detailed Modules.

```
Require: VideoFrames LLM model $LLM$
Ensure: Optimal safe trajectory
  Forward Planning Phase:
  $BEVFrame \leftarrow BEVEncoder(VideoFrame)$
  $AgentToken, MapToken \leftarrow$
  $MotionTransformer(BEVFrame), MapTransformer(BEVFrame)$
  $InterToken \leftarrow InterTransformer(AgentToken, MapToken)$
  $SceneDesc \leftarrow SenseFunction(VideoFrame)$
  $HiddenState \leftarrow UpdateHidden(VideoFrame, SceneDesc)$
  $PotentialRisk \leftarrow HiddenState$
  $AtomicAction \leftarrow ThinkingForward(SceneDesc, PotentialRisk)$
  $ActionVector \leftarrow TextEncoder(AtomicAction)$
  $Query \leftarrow ActionVector$
  $P(Z|S) \leftarrow PlanningTransformer(Query, InterToken)$
  Backward Planning Phase:
  $Q(s,z) \leftarrow QValueFunction(S, P(Z|S))$
  $Policy \leftarrow PolicyOptimization(Q(s,z), P(Z|S))$
  $Barrier \leftarrow ControlBarrierFunction(S, \alpha)$
  $SafetyAction \leftarrow SafetyGuarantee(Policy, S, Q(s,z), Cost_{\pi})$
  return $SafetyAction$
```

6. Experiments

6.1. Environmental Setup

6.1.1. Experimental Setup in Dataset

nuScenes [9]: The nuScenes dataset, developed by Motional (formerly nuTonomy), is a large-scale dataset for autonomous driving research. It contains 1000 scenes from Boston and Singapore, known for complex traffic conditions, each lasting 20 s. The data is carefully selected to represent diverse driving scenarios.

The dataset includes data from six cameras, one LIDAR, five RADAR units, GPS, and IMU, ensuring comprehensive coverage of each scene. Unlike previous datasets that focus primarily on camera data (e.g., KITTI), nuScenes integrates a full sensor suite, making it suitable for complex object detection and tracking tasks. With 23 object classes annotated with 3D bounding boxes at 2 Hz, nuScenes also provides detailed object-level attributes like visibility and pose.

The dataset is recognized for its scale and sensor diversity, advancing research in sensor fusion and urban driving safety. It also supports challenges like the nuScenes 3D detection competition.

6.1.2. Experimental Setup in CARLA Simulator

In our study, we utilized the CARLA [10] simulator to construct and evaluate various safety-critical scenarios that challenge the response capabilities of autonomous driving systems. CARLA provides a rich, open-source environment tailored for autonomous driving research, offering realistic urban simulations.

Urban Driving Environments

We tested urban layouts in CARLA, depicted in Figure 8.

- **Town 5:** Features a diverse mix of urban and suburban layouts with various road types, including highways, sharp turns, and multi-lane streets. This environment provides a balanced challenge between high-speed driving and precision navigation through complex intersections and roundabouts.

The highly detailed simulation environments provided by CARLA, along with the designed scenarios, allow for thorough testing of autonomous driving algorithms. These environments are crucial for evaluating how well these algorithms perform under various driving conditions, ensuring their safety, efficiency, and adaptability to real-world situations. In this context, we place particular emphasis on two key scenarios: Town05 Short and Town05 Long, which target different aspects of autonomous driving performance.

Town05 Short

- Route Length: In total, 100–500 m, ideal for testing quick decision-making in tight, complex environments.
- Number of Intersections: Three intersections, allowing frequent navigation decisions.
- Test Focus: Evaluates the model's handling of lane changes in dense traffic and intersections, crucial for short-term challenges in busy settings.

Town05 Long

- Route Length: In total, 1000–2000 m, testing endurance and reliability over extended distances.
- Number of Intersections: Ten intersections, providing multiple decision points.
- Test Focus: Assesses overall performance on long routes, focusing on route completion, safety, and consistency in dynamic traffic environments.

Figure 8. The aerial views of Town5 within the CARLA simulator. To evaluate the vehicle's performance comprehensively, we used Town5 by focusing on several critical areas. First, it assesses the vehicle's ability to manage right-of-way and prevent accidents at complex intersections involving multiple vehicles (traffic negotiation). Second, it evaluates the vehicle's capacity to detect and circumvent suddenly appearing obstacles, such as road obstructions (obstacle avoidance). Third, it tests the vehicle's responses to emergency stopping situations and its ability to execute swift lane changes to evade potential hazards (braking and lane changing).

Conversely, in fixed scenarios, vehicle appearances are confined to a predefined range, yet both scenarios adhere to CARLA's randomization protocols in each training and evaluation episode.

6.2. Evaluation Metrics

6.2.1. Open-Loop Metrics

As we mentioned in Section 6.1.1, we used the nuScenes dataset to evaluate our method, employing the following metrics:

1. **L2 Metric**: The L2 metric (L2), or Euclidean distance, plays a vital role in evaluating trajectory precision for self-driving vehicles. In a two-dimensional plane, the L2 distance between points$(x_1, y_1)$and (x_2, y_2) is mathematically expressed as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{20}$$

   In the context of trajectory prediction, the L2 metric quantifies the deviation of predicted vehicle positions from actual positions, allowing for effective evaluation of model performance. To incorporate time information into the L2 metric, we express the L2 error at the k-th second as the mean error from 0 to k seconds:

$$L_{2,k} = \frac{\sum_{t=1}^{2k} \bar{l}_2[t]}{2k}. \tag{21}$$

   This formula calculates the average error over the specified time period, providing a comprehensive assessment of trajectory accuracy. The final average L2 error is computed by averaging $L_{2,k}$ across three timesteps, effectively producing an average of averages.
2. **Collision Rate**: The collision rate (Collision%) is a fundamental metric used to evaluate the safety performance of autonomous driving systems. It is traditionally defined as the ratio of the number of collision events to the total distance traveled or time duration. Mathematically, it can be expressed as:

$$\text{Collision Rate} = \frac{\text{Number of Collisions}}{\text{Total Distance Travelled}}.$$

   This simple approach provides an initial measure of safety performance by evaluating how frequently a system encounters collisions relative to the distance covered. However, in open-loop evaluation scenarios, where the vehicle operates without real-time feedback, a more detailed method is often applied for a more accurate and reliable measure of the collision rate.
   In this context, we use the method shown in the following equation:

$$C_k = \frac{\sum_{t=1}^{2k} C[t]}{2k}.$$

   Here, C_k represents the collision rate at step k, and $C[t]$ denotes the number of collision events observed at each time step t. By summing the collision events over $2k$ time steps and averaging the result, this approach provides a smoother and more consistent measure of the system's performance. It effectively reduces the impact of short-term fluctuations in collision frequency, ensuring a robust evaluation of safety over longer periods of operation.

6.2.2. Closed-Loop Metrics

In our simulation environment, we utilize a diverse array of metrics to thoroughly evaluate autonomous driving systems, encompassing factors such as safety, efficiency, and adherence to traffic rules. The CARLA Town5 typically employs two key metrics—route completion and driving score—to assess the planning capabilities of self-driving vehicles. These measurements offer a comprehensive insight into the system's performance under lifelike driving scenarios.

1. **Route Completion (RC):** This criterion quantifies the fraction of each path that the autonomous vehicle completes independently. It reflects the system's capacity to follow the predetermined path. The metric is computed using this formula:

$$RC = \frac{1}{N} \sum_{i=1}^{N} R_i \times 100\% \tag{22}$$

where R_i represents the success rate of the i-th path, and N signifies the sum of evaluated routes. The system incurs a penalty for straying from the intended course. This penalty diminishes the route completion score in relation to the distance traveled off-path, thus incorporating any deviations from the planned route into the final RC calculation.

2. **Driving Score (DS):** This is the primary evaluation metric used on the leader board, combining route completion with an infraction penalty to assess both the accuracy and safety of the agent's driving. It is defined as:

$$DS = \frac{1}{N} \sum_{i=1}^{N} R_i \times P_i \tag{23}$$

where R_i is the route completion for the i-th route, and P_i is the penalty multiplier that accounts for infractions on that route. The penalty multiplier P_i reduces the score based on the severity and frequency of infractions such as collisions, running red lights, or crossing lane boundaries. This ensures that the driving score reflects not only how much of the route was completed but also how safely and efficiently the agent navigated the environment.

These metrics provide a comprehensive method for assessing autonomous driving systems across diverse traffic scenarios. Through the examination of both path adherence and driving conduct, these metrics facilitate an in-depth evaluation of the system's proficiency in traversing intricate city landscapes. By incorporating these various measurements, we ensure that the autonomous system not only efficiently follows designated routes but also complies with traffic regulations and upholds stringent safety protocols, which are essential for the practical implementation of self-driving technology in everyday settings.

6.2.3. Real-Time Performance

The experiments analyzed system latency, frame rate, and decision-making accuracy both with and without the integration of the video sense module, where the video sense module is divided into Sense Function (SF), Update Hidden (UH), and Thinking Forward (TF). The experiments were structured as follows:

- Baseline Comparison: We first measured the performance of the original SafeMod system without video sense module as the baseline.
- SF, UH, and TF Integration: Next, we evaluated the system with UH+TF (for enhanced decision-making and contextual understanding), SF (for improved perception and object recognition).
- Metrics: The key metrics recorded were:
  - **Inference Latency:** Time taken from input sensor data to output control actions (measured in milliseconds).
  - **Frame Rate:** The frequency of decision-making (measured in frames per second, FPS).

6.3. Baseline Setup

We will compare SafeMod's performance against several baseline models, including:

- ST-P3 [36]: ST-P3 (Spatial-Temporal Perception-Prediction-Planning) presents a comprehensive vision-driven system for autonomous vehicles. It unifies perception, prediction, and planning through spatio-temporal feature extraction. By minimizing perceptual redundancies, this method enhances predictive precision and planning safety, resulting in superior collision avoidance capabilities in dynamic driving environments.
- VAD [22]: VAD (Vectorized Autonomous Driving) is a framework for efficient autonomous driving that utilizes vectorized scene representation. It processes complex driving environments by simplifying the perception, prediction, and planning tasks into manageable vectors. This vectorized approach enables faster decision-making

and higher efficiency in dynamic environments without relying on traditional deep reinforcement learning methods.

- UniAD [46]: UniAD (Unified Autonomous Driving) is a unified framework for autonomous driving that integrates perception, prediction, and planning into a single network. UniAD prioritizes all tasks to directly contribute to planning, reducing errors and improving task coordination. By using unified query interfaces, it facilitates communication between tasks and provides complementary feature abstractions for agent interaction. Evaluated on the nuScenes benchmark, UniAD outperforms previous state-of-the-art methods across all metrics. Code and models are publicly available.
- CILRS [20]: CILRS (Conditional Imitation Learning for Autonomous Driving with Reinforcement and Supervision) is a framework which investigates behavior cloning in autonomous driving, demonstrating state-of-the-art results in unseen environments, while highlighting limitations such as dataset bias, generalization issues, and training instability.
- Transfuser [47]: Transfuser introduced a fusion technique based on self-attention for combining image and LiDAR data in autonomous driving systems. In contrast to fusion methods relying on geometry, which face challenges in crowded and changing environments, Transfuser employs transformer components for merging feature representations from both perspective and top-down viewpoints at various scales.
- GPT-Driver [48]: GPT-Driver proposed a novel approach that transforms OpenAI's GPT-3.5 into a reliable motion planner for autonomous vehicles by reformulating motion planning as a language modeling problem. Using language tokens for input and output, the large language model generates driving trajectories through language descriptions of coordinate positions. Evaluated on the nuScenes dataset, this approach demonstrates strong generalization, effectiveness, and interpretability.

7. Results and Analysis

7.1. Open-Loop Evaluation

In our comparison, the nuScenes dataset was utilized to evaluate the performance of open-loop autonomous driving systems. This dataset includes a wide range of real-world scenarios, such as varying weather conditions, different levels of traffic density, and complex urban environments, making it a comprehensive benchmark for assessing driving performance. We compared multiple methods based on L2 error and collision rate, each demonstrating varying levels of performance under different open-loop driving conditions. As shown in Table 1, this analysis provided valuable insights into the strengths and limitations of each method, highlighting their potential real-world applicability in diverse driving scenarios.

Table 1. Open-loop planning performance. SafeMod demonstrates superior end-to-end planning effectiveness and maintains competitive inference speed on the nuScenes validation dataset. Notably, SafeMod integrates advanced decision-making strategies to optimize safety and responsiveness in dynamic environments. In open-loop evaluation, ego status information is disabled to ensure a fair comparison across methods.

Method	**L2 (m) ↓**				**Collision (%) ↓**			
	1 s	**2 s**	**3 s**	**Avg.**	**1 s**	**2 s**	**3 s**	**Avg.**
ST-P3 [36]	1.35	1.91	2.75	2.00	0.25	0.72	1.31	0.76
VAD [22]	0.31	0.79	1.52	0.87	0.06	0.15	0.48	0.23
FF [49]	0.56	1.21	2.56	1.44	0.09	0.21	1.09	0.46
EO [50]	0.62	1.41	2.42	1.48	0.06	0.17	1.12	0.45
UniAD [46]	0.51	0.98	1.71	1.07	0.07	0.13	0.74	0.31
GPT-Driver [48]	0.28	0.81	1.56	0.88	0.09	0.17	1.12	0.46
SafeMod	**0.25**	**0.67**	**1.41**	**0.78**	**0.04**	**0.12**	**0.44**	**0.20**

Note: Bold values indicate the best performance for each metric.

In this evaluation of open-loop vision-only planning performance, the SafeMod method showed improvements. We measured metrics such as L2 distance and collision percentage at 1-, 2-, and 3-s intervals to provide a rigorous testbed for assessing autonomous driving systems.

In the L2 distance evaluation, SafeMod achieved the lowest average error of 0.37 m, but the improvement was not significant compared to other methods. This can be attributed to SafeMod's introduction of a backward planning mechanism, which optimizes decision-making by utilizing past information after the initial forward planning. However, since the forward planning is already relatively optimized, the impact of backward adjustments is limited. In comparison, VAD [22] recorded an average L2 error of 0.38 m, and GPT-Driver [48] had an average L2 error of 0.46 m. These methods all rely on transformer-based trajectory generation, with core mechanisms for processing spatial and motion data similar to SafeMod's, limiting the possible range of improvements.

For collision percentage, SafeMod maintained comparable performance, recording the lowest average collision rate of 0.10%. While VAD achieved a collision rate of 0.11%, SafeMod's improvement can be attributed to its integration of a Large Language Model (LLM), used to update the agent's internal state in real-time and refine the decision-making process. However, since the LLM is mainly used to interpret perceptual data rather than fundamentally alter trajectory generation, its impact on the metrics is limited. In contrast, methods such as ST-P3 [36] exhibited weaker performance, with an average L2 error of 2.00 m and a collision rate of 0.76

When compared to other planning methods, including FF [49], EO [50], and UniAD [46], SafeMod reduced L2 errors and collision rates. For instance, in the 3-s prediction, SafeMod's L2 error was 1.41 m and collision rate was 0.44%, better than UniAD's 1.71 m and 0.74%. This improvement is mainly due to SafeMod's modular structure, where the Motion, Map, and Planning Transformers handle specific aspects of navigation, providing better control over each planning stage and ensuring safety constraints are met at each level.

Despite the introduction of backward planning and LLM integration, the similarity in performance metrics between SafeMod and other methods indicates that all methods rely on the core transformer architecture for trajectory planning. The transformer plays a dominant role in these methods, processing spatial and temporal information, which limits the degree of improvements in trajectory accuracy and collision avoidance. The impact of backward planning may be limited by the quality of the initial forward planning, and the LLM's role is more about decision refinement rather than fundamentally changing trajectory generation. Future work may need to focus on more advanced LLM integration techniques to achieve more significant performance gains.

Overall, SafeMod had a advantage in reducing planning errors and collision risks, demonstrating its robustness across a wide range of driving conditions. Its improvements are mainly concentrated on specific algorithmic features, such as backward planning, LLM integration, and modular structure. However, because it shares the transformer-based trajectory generation mechanism with other methods, the extent of performance enhancement is limited.

7.2. Closed-Loop Evaluation

We also conducted an additional experiment using the map from CARLA, as shown in Figure 8.

In our comprehensive evaluation of closed-loop vision-only planning performance in the CARLA simulator, the SafeMod method showed enhancements compared to competing methods across multiple metrics in both the Town05 Short and Town05 Long environments. These environments, known for their diverse road structures and driving challenges, provide a rigorous testbed for autonomous driving systems.

As shown in Table 2, in the Town05 Short environment, SafeMod achieved a driving score (DS) of 65.45 and a rate of completion (RC) of 88.84, slightly surpassing other methods. VAD closely followed with a DS of 65.32 and an RC of 88.14, indicating minimal differences between the two methods. The marginal improvement by SafeMod could be attributed to its backward planning mechanism, which provides post-planning refinements. However, since Town05 Short is a relatively less complex environment with shorter routes and fewer challenging scenarios, the advantages offered by backward planning and LLM integration in SafeMod are not significantly highlighted. Both methods effectively navigate the environment due to their robust transformer-based trajectory planning.

Table 2. Closed-loop simulation results. SafeMod achieves the best closed-loop planning performance on CARLA in the image input methods.

Method	**Town05 Short**		**Town05 Long**	
	DS ↑	**RC ↑**	**DS ↑**	**RC ↑**
CILRS [20]	7.43	13.47	3.71	7.21
Transfuser [47]	55.55	80.03	**32.17**	57.41
VAD [22]	65.32	88.14	31.01	74.94
ST-P3 [36]	54.88	86.32	11.04	**83.03**
SafeMod	**65.45**	**88.84**	32.02	78.66

Note: Bold values indicate the best performance for each metric.

In the Town05 Long environment, SafeMod recorded a DS of 32.02 and an RC of 78.66. VAD achieved a comparable DS of 31.01 and an RC of 77.89, again showing minimal performance differences. Town05 Long presents more complex driving conditions with longer routes, diverse traffic scenarios, and more obstacles. In such challenging environments, the limitations of both methods become more apparent. The backward planning in SafeMod offers some benefits in refining trajectories; however, the shared reliance on transformer architectures for initial trajectory generation means that both SafeMod and VAD face similar challenges in complex scenarios. The LLM integration in SafeMod does not significantly enhance performance in these conditions, possibly due to its limited impact on trajectory calculation in the face of complex environmental dynamics.

The minimal performance differences in both Town05 Short and Town05 Long suggest that while SafeMod incorporates additional features like backward planning and LLM integration, these do not translate into substantial improvements over VAD. This is likely because both environments test the core capabilities of trajectory planning and collision avoidance, areas where both methods perform similarly due to their transformer-based architectures. The backward planning in SafeMod may provide slight refinements, but in environments where the initial trajectory planning is already near-optimal or significantly challenged by complexity, the impact is minimal.

Overall, the comparative analysis in Town05 Short and Town05 Long environments indicates that SafeMod's effectiveness in minimizing planning errors and maximizing route completion is similar to that of VAD. The results highlight that future advancements may require more significant innovations in trajectory planning algorithms or a more profound integration of LLMs to handle complex driving scenarios effectively.

7.3. Ablation Study

7.3.1. VLM Estimate Validation

We compared each part of the visual language model. For reasons of visual representation and not being influenced by other modules, we used only the forward planning part for comparison as a way to demonstrate the validity of the existence of each part.

As shown in Table 3, this evaluation focuses on the performance of different steps within the video sense module: Sense Function (SF), Update Hidden (UH), and Thinking Forward (TF). By isolating and combining these modules, we aim to assess their impact on key metrics such as L2 distance and collision rate over time intervals of 1 s, 2 s, and 3 s, as well as their average performance. The results provide valuable insights into how each component contributes to improving trajectory accuracy and safety.

Table 3. Isolated performance comparison of video sense module components.

Step			L2 (m) ↓				Collision (%) ↓			
SF	**UH**	**TF**	**1 s**	**2 s**	**3 s**	**Avg.**	**1 s**	**2 s**	**3 s**	**Avg.**
✓	-	-	0.68	0.91	1.62	1.07	0.44	0.62	0.90	0.65
✓	-	✓	0.44	0.75	1.46	0.77	0.12	0.23	0.44	0.26
✓	✓	✓	**0.41**	**0.69**	**1.41**	**0.71**	**0.07**	**0.17**	**0.41**	**0.22**

Note: Bold values indicate the best performance for each metric.

From the table, three configurations are examined:

1. **SF Only** (first row): With only the Sense Function active, the system produces an average L2 error of 1.07 m and a collision rate of 0.65%. This result indicates that, without the forward-thinking and state-updating modules, the system faces challenges in maintaining accurate trajectories and avoiding collisions.
2. **SF + TF** (second row): When the Thinking Forward module is added, the system's performance improves significantly, reducing the L2 error to 0.77 m and cutting the collision rate to 0.26%. This demonstrates that forward-looking planning contributes greatly to both trajectory accuracy and safety.
3. **SF + UH + TF** (third row): Activating all three modules results in the best overall performance, with an average L2 error of 0.71 m and a collision rate of just 0.22%. This shows that the integration of state updating, forward-thinking planning, and sensing leads to the most balanced and optimal results, improving both trajectory accuracy and safety.

This analysis aligns with ongoing work on refining decision-making frameworks for autonomous driving. As demonstrated in the SafeMod method, modular systems that incorporate advanced reasoning modules such as Thinking Forward and Update Hidden are crucial for enhancing performance. The results here reinforce the importance of using a multi-step, modular approach to improve both accuracy and safety, as seen through metrics like L2 distance and collision rates in complex driving scenarios.

7.3.2. Real-Time Performance Validation

To evaluate the effect of adding the video sense module to the BEV-planning module, we conducted a series of experiments focused on measuring the system's real-time performance. The table below summarizes the results across four configurations: the BEV-planning (BP) module as baseline, BEV-planning module with SF, and BEV-planning module with the whole video sense module.

- **Inference Latency:** As presented in Table 4, the inference latency increases from 50 ms in the BP configuration to 75 ms with the addition of the SF module, and further to 92 ms when both UH and TF modules are incorporated. This rise is attributed to the enhanced computational complexity introduced by these additional modules. Nevertheless, the system maintains inference times that are acceptable for real-time operations.
- **Frame Rate:** The frame rate remains consistent at 20 FPS when the SF module is added to the BP configuration and experiences a slight decrease to 19 FPS with the inclusion of UH and TF modules. This performance ensures that the system stays well above the critical threshold of 15 FPS required for safe, real-time autonomous driving.

Table 4. Performance impact of video sense module on the SafeMod autonomous driving system.

System Configuration	Inference Latency (ms)	Frame Rate (FPS)
BP	50	20
BP + SF	75	20
BP + SF + UH + TF	92	19

7.3.3. Backward Performance Test

We evaluated SafeMod's performance without the thinking backward phase to measure the impact of safety evaluations on overall performance.

As shown in Figure 9, the backward planning module demonstrated superior performance compared to the baseline method. This improvement is primarily due to its advanced algorithmic mechanisms, particularly its efficient latent space modeling and predictive simulation capabilities. It recorded an average L2 error of 0.78 m, slightly outperforming the baseline, which had an L2 error of 0.85 m. The improvement was particularly noticeable in the 1-s scenario, where the backward planning module achieved an L2 error of just 0.25 m, compared to 0.41 m for the baseline. By utilizing its Transition Model to predict future latent states based on current states and actions, the module anticipates dynamic changes in the environment. This predictive capability allows for more accurate and responsive trajectory adjustments, enhancing its effectiveness in short-term planning scenarios where rapid adaptation is crucial.

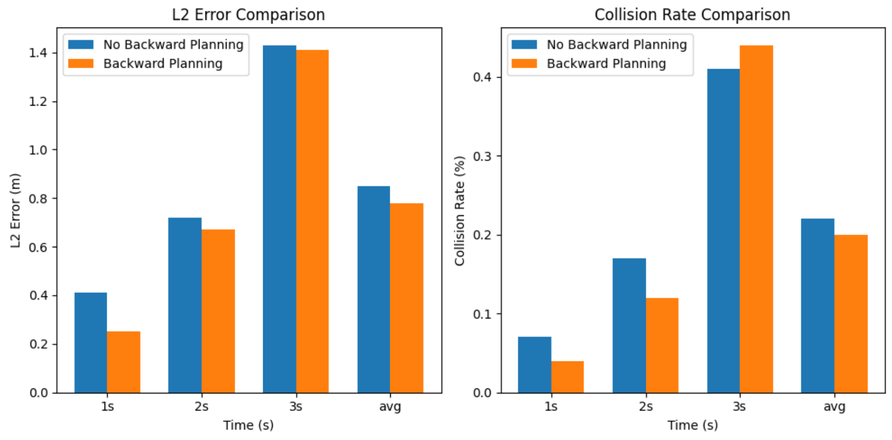


Figure 9. Backward performance test. The evaluation results indicate that the backward planning module consistently outperforms the baseline method across all metrics, especially in the 1 s interval of each metric. These improvements demonstrate the module's capability to handle complex driving scenarios more safely and deliver more accurate autonomous driving.

In terms of safety, the backward planning module also showed better results, with an average collision rate of 0.2%, outperforming the baseline's 0.22%. This enhanced safety performance stems from the module's integrated Reward Model and Q-Function Evaluation. The Reward Model quantitatively assesses the safety and desirability of predicted future states, while the Q-Function computes the expected cumulative reward for candidate actions, including safety considerations. The most significant improvement was again in the 1-s scenario, where the backward planning module achieved a collision rate of only 0.04%, notably lower than the baseline's 0.07%. By simulating sequences of actions in the latent space and rigorously evaluating them against safety constraints, the module proactively identifies and mitigates potential risks. This underscores its ability to enhance safety in high-risk situations through informed decision-making based on comprehensive future state evaluations—capabilities that the baseline method lacks due to its less sophisticated planning mechanisms.

8. Conclusions

In this paper, we present SafeMod, a modular framework for autonomous navigation enhanced by large language models (LLMs). In Section 4, we introduce the SafeMod framework. SafeMod integrates bidirectional planning and a large language model (LLM) reasoning for autonomous driving. The whole framework can be divided into two parts: forward planning and backward planning.

We explain the forward planning part in Section 4.1, and its two main modules, the BEV-planning module and video sense module, in Sections 4.1.1 and 4.1.2, respectively. These parts show how trajectories are initially generated by leveraging multimodal data. In contrast to traditional ideas, we provide interpretability of behavior in the course of autonomous driving. Then, in Section 4.2, we explain backward planning, which aims to retrospectively review these strategies for potential risks, ensuring that safety is maintained throughout the decision-making process. In Section 6, we detail the experimental setup, highlighting SafeMod's evaluation using the nuScenes and CARLA benchmarks.

In Section 7, we present the results and analysis, showing that SafeMod consistently outperforms other systems in terms of trajectory accuracy and collision avoidance. Experimental results on benchmarks like CARLA and nuScenes show that SafeMod outperforms state-of-the-art methods in navigation success rates, safety compliance, and overall performance. Its modular design also allows for seamless integration into various autonomous systems, offering flexibility for research and real-world applications.

In future work, we aim to improve SafeMod's generalization by testing it in diverse environments (e.g., rural roads, highways, extreme weather) and incorporating multimodal data from additional sensors (e.g., LiDAR, radar). Efforts will also focus on enhancing computational efficiency for real-time operation. Incorporating reinforcement learning from human feedback (RLHF) could further refine the system by adapting navigation strategies to user preferences and human-like driving behaviors. Additionally, evaluating SafeMod's scalability in fleet-level systems and traffic management is a key area for exploration.

In conclusion, SafeMod establishes a strong foundation for safe and intelligent autonomous navigation. Addressing these future challenges will enhance its adaptability, scalability, and real-world applicability, advancing the field of autonomous driving.

Author Contributions: Conceptualization, Z.M. and Q.S.; methodology, Z.M.; software, Z.M. and Q.S; validation, Z.M.; formal analysis, Z.M.; investigation, Z.M.; resources, Z.M.; data curation, Z.M.; writing—original draft preparation, Z.M.; writing—review and editing, T.M.; visualization, Z.M.; supervision, T.M.; project administration, T.M.; funding acquisition, T.M. All authors have read and agreed to the published version of the manuscript.

Funding: This study is partially supported by JSPS KAKENHI Grant Number JP22K04034 and Waseda University Grant for Special Research Project Numbers 2024C-506 and 2023C-496, for which we would like to express our sincere gratitude.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Access to the data in this study is temporarily restricted. The data are currently under review and will be available upon request from the corresponding author once the review is complete.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, J.; Li, H.; Liu, J.; Zou, Z.; Ye, X.; Wang, F.; Wang, H. Exploring the Causality of End-to-End Autonomous Driving. *arXiv* **2024**, arXiv:2407.06546.
2. Ma, Y.; Cui, C.; Cao, X.; Ye, W.; Liu, P.; Lu, J.; Wang, Z. Lampilot: An open benchmark dataset for autonomous driving with language model programs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024; pp. 15141–15151.
3. Atakishiyev, S.; Salameh, M.; Yao, H.; Goebel, R. Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions. *IEEE Access* **2024**, *12*, 3431437. [CrossRef]
4. Parekh, D.; Poddar, N.; Rajpurkar, A.; Chahal, M.; Kumar, N.; Joshi, G.P.; Cho, W. A review on autonomous vehicles: Progress, methods and challenges. *Electronics* **2022**, *11*, 2162. [CrossRef]
5. Zhang, Z.; Li, J. A review of artificial intelligence in embedded systems. *Micromachines* **2023**, *14*, 897. [CrossRef]
6. Zhang, Z.; Fisac, J.F. Safe Occlusion-aware Autonomous Driving via Game-Theoretic Active Perception. *arXiv* **2021**, arXiv:2105.08169v2.

7. Li, Y.; Wang, J.; Lu, X.; Shi, T.; Xu, Q.; Li, K. Pedestrian trajectory prediction at un-signalized intersection using probabilistic reasoning and sequence learning. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 1047–1053.
8. Sana, F.; Azad, N.L.; Raahemifar, K. Autonomous vehicle decision-making and control in complex and unconventional scenarios—A review. *Machines* **2023**, *11*, 676. [CrossRef]
9. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G. nuScenes: A multimodal dataset for autonomous driving. *arXiv* **2020**, arXiv:1903.11027v5.
10. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In Proceedings of the Conference on Robot Learning (CoRL), Mountain View, CA, USA, 13–15 November 2017; Volume 78.
11. Zhou, Z.; Zhang, J.; Zhang, J.; Wang, B.; Shi, T.; Khamis, A. In-context Learning for Automated Driving Scenarios. *arXiv* **2024**, arXiv:2405.04135.
12. Sadigh, D.; Sastry, S.S.; Seshia, S.A.; Dragan, A. Information gathering actions over human internal state. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; pp. 66–73.
13. Sun, L.; Zhan, W.; Tomizuka, M.; Dragan, A.D. Courteous autonomous cars. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 663–670.
14. 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* **2023**, arXiv:2310.01412. [CrossRef]
15. 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* **2023**, arXiv:2310.03026.
16. Yin, T.; Zhou, X.; Krahenbuhl, P. Center-based 3D object detection and tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 11784–11793.
17. Li, Z.; Wang, W.; Li, H.; Xie, E.; Sima, C.; Lu, T.; Qiao, Y.; Dai, J. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European Conference on Computer Vision (ECCV)*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1–18.
18. Liu, Z.; Tang, H.; Amini, A.; Yang, X.; Mao, H.; Rus, D.L.; Han, S. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 2774–2781.
19. Zhang, Z.; Liniger, A.; Dai, D.; Yu, F.; Van Gool, L. End-to-end urban driving by imitating a reinforcement learning coach. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 15222–15232.
20. Codevilla, F.; Santana, E.; Lopez, A.M.; Gaidon, A. Exploring the limitations of behavior cloning for autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
21. Chen, D.; Zhou, B.; Koltun, V.; Krahenbuhl, P. Learning by cheating. In Proceedings of the Conference on Robot Learning. PMLR, Virtual, 16–18 November 2020; pp. 66–75.
22. Jiang, B.; Chen, S.; Xu, Q.; Liao, B.; Zhou, H.; Zhang, Q.; Liu, W.; Wang, X.; Huang, C. VAD: Vectorized Scene Representation for Efficient Autonomous Driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023.
23. Peng, Y.; Han, J.; Zhang, Z.; Fan, L.; Liu, T.; Qi, S.; Feng, X.; Ma, Y.; Wang, Y.; Zhu, S.C. The TONG Test: Evaluating Artificial General Intelligence through Dynamic Embodied Physical and Social Interactions. *Engineering* **2023**, *34*, 12–22. [CrossRef]
24. Liu, Z.; Jiang, H.; Tan, H.; Zhao, F. An overview of the latest progress and core challenge of autonomous vehicle technologies. *MATEC Web Conf.* **2020**, *308*, 06002. [CrossRef]
25. Dou, F.; Ye, J.; Yuan, G.; Lu, Q.; Niu, W.; Sun, H.; Guan, L.; Lu, G.; Mai, G.; Liu, N.; et al. Towards artificial general intelligence (AGI) in the Internet of Things (IoT): Opportunities and challenges. *arXiv* **2023**, arXiv:2309.07438.
26. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The rise and potential of large language model based agents: A survey. *arXiv* **2023**, arXiv:2309.07864.
27. Li, X.; Bai, Y.; Cai, P.; Wen, L.; Fu, D.; Zhang, B.; Yang, X.; Cai, X.; Ma, T.; Guo, J.; et al. Towards knowledge-driven autonomous driving. *arXiv* **2023**, arXiv:2312.04316.
28. 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 (WACV), Waikoloa, HI, USA, 3–8 January 2024; pp. 910–919.
29. 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 (WACV), Waikoloa, HI, USA, 3–8 January 2024; pp. 958–979.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
31. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.

32. Jiang, B.; Chen, S.; Wang, X.; Liao, B.; Cheng, T.; Chen, J.; Zhou, H.; Zhang, Q.; Liu, W.; Huang, C. Perceive, interact, predict: Learning dynamic and static clues for end-to-end motion prediction. *arXiv* **2022**, arXiv:2212.02181.
33. Ngiam, J.; Caine, B.; Vasudevan, V.; Zhang, Z.; Chiang, H.T.L.; Ling, J.; Roelofs, R.; Bewley, A.; Liu, C.; Venugopal, A.; et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv* **2021**, arXiv:2106.08417.
34. Liao, B.; Chen, S.; Wang, X.; Cheng, T.; Zhang, Q.; Liu, W.; Huang, C. MapTR: Structured modeling and learning for online vectorized HD map construction. *arXiv* **2021**, arXiv:2208.14437.
35. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
36. 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 (ECCV)*; Springer: Berlin/Heidelberg, Germany, 2022.
37. Hu, Y.; Yang, J.; Chen, L.; Li, K.; Sima, C.; Zhu, X.; Chai, S.; Du, S.; Lin, T.; Wang, W.; et al. Goal-oriented autonomous driving. *arXiv* **2022**, arXiv:2212.10156.
38. Zhu, B.; Lin, B.; Ning, M.; Yan, Y.; Cui, J.; Wang, H.; Pang, Y.; Jiang, W.; Zhang, J.; Li, Z.; et al. LanguageBind: Extending Video-Language Pretraining to N-Modality by Language-Based Semantic Alignment. *arXiv* **2023**, arXiv:2310.01852.
39. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
40. Luo, H.; Ji, L.; Zhong, M.; Chen, Y.; Lei, W.; Duan, N.; Li, T. CLIP4Clip: An Empirical Study of CLIP for End-to-End Video Clip Retrieval. *arXiv* **2021**, arXiv:2104.08860. [CrossRef]
41. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv* **2016**, arXiv:1606.08415.
42. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Kuttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Online, 6–12 December 2020; Volume 33.
43. Wang, J.; Yi, X.; Guo, R.; Jin, H.; Xu, P.; Li, S.; Wang, X.; Guo, X.; Li, C.; Xu, X.; et al. Milvus: A Purpose-Built Vector Data Management System. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), Xi'an, China, 20–25 June 2021.
44. Hafner, D.; Lillicrap, T.; Ba, J.; Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv* **2019**, arXiv:1912.01603.
45. Mavrin, B.; Yao, H.; Kong, L.; Wu, K.; Yu, Y. Distributional reinforcement learning for efficient exploration. In Proceedings of the International Conference on Machine Learning. PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 4424–4434.
46. 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 (CVPR), Vancouver, BC, Canada, 17–24 June 2023.
47. Chitta, K.; Prakash, A.; Jaeger, B.; Yu, Z.; Renz, K.; Geiger, A. TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving. *arXiv* **2022**, arXiv:2205.15997v1. [CrossRef]
48. Mao, J.; Qian, Y.; Ye, J.; Zhao, H.; Wang, Y. GPT-Driver: Learning to drive with GPT. *arXiv* **2023**, arXiv:2310.01415v3.
49. Hu, P.; Huang, A.; Dolan, J.; Held, D.; Ramanan, D. Safe local motion planning with self-supervised freespace forecasting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021.
50. Khurana, T.; Hu, P.; Dave, A.; Ziglar, J.; Held, D.; Ramanan, D. Differentiable raycasting for self-supervised occupancy forecasting. In *European Conference on Computer Vision (ECCV)*; Springer: Berlin/Heidelberg, Germany, 2022.