IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Mimir: Hierarchical Goal-Driven Diffusion With Uncertainty Propagation for End-to-End Autonomous Driving

Zebin Xing, *Graduate Student Member, IEEE*, Yupeng Zheng, *Student Member, IEEE*, Qichao Zhang, *Member, IEEE*, Zhixing Ding, Pengxuan Yang, *Graduate Student Member, IEEE*, Songen Gu, Zhongpu Xia, and Dongbin Zhao, *Fellow, IEEE*

***Abstract*—End-to-end autonomous driving has emerged as a pivotal direction in the field of autonomous systems. Recent works have demonstrated impressive performance by incorporating high-level guidance signals to steer low-level trajectory planners. However, their potential is often constrained by inaccurate high-level guidance and the computational overhead of complex guidance modules. To address these limitations, we propose Mimir, a novel hierarchical dual-system framework capable of generating robust trajectories relying on goal points with uncertainty estimation: (1) Unlike previous approaches that deterministically model, we estimate goal point uncertainty with a Laplace distribution to enhance robustness; (2) To overcome the slow inference speed of the guidance system, we introduce a multi-rate guidance mechanism that predicts extended goal points in advance. Validated on challenging Navhard and Navtest benchmarks, Mimir surpasses previous state-of-the-art methods with a 20% improvement in the driving score EPDMS, while achieving 1.6× improvement in high-level module inference speed without compromising accuracy. The code and models will be released soon to promote reproducibility and further development.**

***Index Terms*—Learning from demonstration, imitation learning, autonomous vehicle navigation.**

Received 10 July 2025; accepted 24 November 2025. Date of publication 8 December 2025; date of current version 2 January 2026. This article was recommended for publication by Associate Editor H. Ravichandar and Editor A. Faust upon evaluation of the reviewers' comments. This work was supported in part by Beijing Natural Science Foundation-Xiaomi Innovation Joint under Grant Fund L253007, in part by Beijing Natural Science Foundation under Grant 4242052, and in part by the National Natural Science Foundation of China (NSFC) under Grant 62173325. *(Corresponding author: Qichao Zhang.)*

Zebin Xing, Yupeng Zheng, Qichao Zhang, Pengxuan Yang, Zhongpu Xia, and Dongbin Zhao are with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institue of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: zhangqichao2014@ia.ac.cn, xingzebin2024@ia.ac.cn).

Zhixing Ding is with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institue of Automation, Chinese Academy of Sciences, Beijing 100190, China, and with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the School of Artificial Intelligence, China University of Geosciences (Beijing), Beijing 100083, China.

Songen Gu is with the School of Data Science, Fudan University, Shanghai 200433, China.

The code is available at https://github.com/ZebinX/Mimir-Uncertainty-Driving .

Digital Object Identifier 10.1109/LRA.2025.3641129

I. Introduction

END-TO-END autonomous driving, which directly maps sensor observations (e.g., RGB images and ego status) to driving actions, has emerged as a pivotal research frontier due to its potential for scalable deployment.

Pioneering methods [3], [4], [5], [30], [34] regress a single-mode by constructing various scene representations and multiple auxiliary supervision tasks with raw sensors input, yet overlook the inherent variance and uncertainty of driving actions. Recently, some approaches [6], [7] model multi-modal action distributions with diffusion. DiffusionDrive [6] extracts anchors from trajectory vocabulary and accelerates denoising through anchor-guided diffusion, while SOTA method, GoalFlow [7], employs goal points as conditional constraints trajectory generation. Although GoalFlow ingeniously bridges perception and multi-modal planning via goal-dirven design, its real-world applicability faces two critical limitations: (1) Error propagation: goal point prediction errors propagate downstream to the planning module as shown in Fig. 1. The goal point specifies an exact short-term future position, strongly guiding the planner. In deterministic modeling, large prediction errors may mislead the decoder into generating unsafe trajectories. (2) Real-time bottleneck: cascaded goal construction and trajectory generation impede real-time performance. Hierarchical frameworks [7], [9] often adopt heavy high-level models to extract effective guidance, which challenges deployment on real vehicles due to high computational cost.

To address these challenges, we propose Mimir, a novel hierarchical dual-system framework comprising: (1) A slow guidance system that generates goal points with Laplacian-modeled uncertainty from RGB observations, ego status, and a trajectory vocabulary. It further extrapolates extended goal points and injects them into the fast planner. (2) A fast planning system first aligns perception features and the goal point guidance in two different coordinate systems via a Guidance Injection module. Then, a diffusion-based planner generates trajectories conditioned on the sampled perception features, goal points, and associated uncertainty.

Extensive evaluations on large-scale challenging Navhard and Navtest benchmarks demonstrate that Mimir achieves SOTA performance with 20% driving score improvement. Besides, leveraging the multi-rate architecture, Mimir increases 1.6× inference speed in the high-level module.

In summary, our contributions are threefold:

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

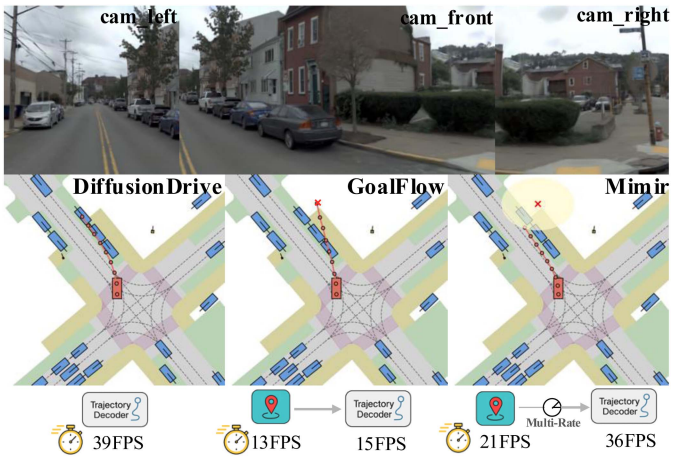

Fig. 1. Compared to existing methods, Mimir assesses the uncertainty of the predicted goal points from the high-level model, enabling safer and more reliable trajectory planning.

- We propose a hierarchical dual-system framework for autonomous driving, Mimir, which achieves real-time performance by predicting extended goal points through an multi-rate guidance mechanism.
- We explicitly model the uncertainty of the goal point using a Laplace distribution, and incorporate this uncertainty into the trajectory planner to enable more reliable and robust trajectory generation.
- Our method achieves SOTA performance on the large-scale, challenging Navhard and Navtest benchmarks, with up to a 20% improvement in the EPDMS driving score and a 1.6× speedup in high-level module inference.

II. Related Work

A. End-to-End Autonomous Driving

Earlier autonomous driving approaches [31], [32], [33] followed a modular pipeline. End-to-end autonomous driving directly maps raw sensory inputs like camera images and LiDAR scans to driving actions via neural networks. Some approaches rely on direct trajectory regression for prediction. Methods like Transfuser [3] advance this by effectively fusing multimodal sensor data and utilizing a GRU to output the predicted trajectory. UniAD [4] focuses on designing specialized transformer modules to extract critical information such as maps and visual features, ultimately also regressing the trajectory using a transformer. Pushing efficiency further, VAD [5] improves significantly by replacing dense BEV features with vector representations. CIL++ [28] investigates multi-view fusion for trajectory prediction. Subsequently, SparseDrive [8] builds upon this by employing sparse representations for both surrounding agents and map elements and leveraging pre-defined goal points to elicit multi-modal trajectory predictions.

Leveraging the strong modeling capabilities of diffusion [21] frameworks, more recent end-to-end methods increasingly favor using a diffusion planner to model the multimodal distribution of possible trajectories. HE-Drive [22], for instance, models trajectory distributions with a diffusion planner, aligning the predictions more closely with human driving preferences. DiffusionDrive [6] employs a truncated diffusion process, starting from pre-sampled anchor trajectories to generate the future path. Similarly, GoalFlow [7] initially predicts future goal points and uses this guidance to steer the goal-conditioned trajectory generation performed by its flow matching [23] model. Although these methods attempt to model the multimodality of trajectories using the diffusion framework, they still overlook the aleatoric uncertainty of the condition. This limitation restricts the performance when the condition is inaccurate.

B. Uncertainty Estimation

Standard neural network models often lack inherent uncertainty estimation capabilities. To tackle this, certain approaches like [12] explicitly model uncertainty using dedicated network components, for instance, by caching the k nearest neighbors of a predicted label and employing a separate classifier to predict uncertainty. Another method [13] leverages the discrepancy between outputs from multiple diffusion processes to quantify the uncertainty associated with a predicted reward function. Alternative strategies [14] utilize model ensembles, aggregating results from multiple models to form the final prediction and estimate uncertainty. Further techniques apply Bayesian methods [16] or Test-Time Augmentation [17] to evaluate model confidence.

Several methods specifically attempt to model uncertainty for perception or planning tasks in autonomous systems. [18] has explored Bayesian uncertainty estimation for quantitatively assessing uncertainty and evaluating its predictive quality. [19], [29] estimates map uncertainty by modeling the distribution of points in vectorized maps. Furthermore, [20] proposes a method using model ensembles to assess epistemic uncertainty in driving plans, enabling the detection of distribution shifts and facilitating recovery. However, this ensemble-based approach requires training multiple models, significantly increasing the cost of model deployment.

III. Problem Definition

End-to-end autonomous driving typically requires modeling perceived environmental information and observable ego-motion states without reliance on high-definition maps to generate future trajectories. Given history sensory inputs $\{I_{-n}^{\text{sensor}}, \ldots, I_0^{\text{sensor}}\}$ from time $-n$ to time 0 (current time), where $I^{\text{sensor}} = \{I^{\text{cam}}, I^{\text{lidar}}\}$, and ego-motion states $\{I_{-n}^{\text{motion}}, \ldots, I_0^{\text{motion}}\}$ where $I^{\text{motion}} = \{I^{\text{command}}, I^{\text{vel}}, I^{\text{acc}}\}$, the neural network needs to output a future trajectory sequence $\{O_1^{\text{traj}}, \ldots, O_T^{\text{traj}}\}$, where each trajectory point satisfies $O^{\text{traj}} = \{x, y, heading\}$. Here, I^{command} denotes the high-level driving intention (turn left, turn right, go straight, or unknown) encoded as a one-hot vector.

This task can be formally defined as finding the maximum likelihood for the ego vehicle's future trajectory:

$$\arg\max_{\theta} P_\theta(O_{1:T}^{\text{traj}} \mid I_{-n:0}^{\text{sensor}}, I_{-n:0}^{\text{motion}}) \tag{1}$$

Several approaches [7], [9], [24] employ a hierarchical framework for trajectory prediction. They first predict the high-level guidance G like meta-action or goal point, then utilizes this to direct the ego vehicle's specific motion. These methodologies can be formally defined as:

$$\arg\max_{\theta} P_\theta^{\text{hight}}(G \mid I_{-n:0}^{\text{sensor}}, I_{-n:0}^{\text{motion}}) \tag{2}$$

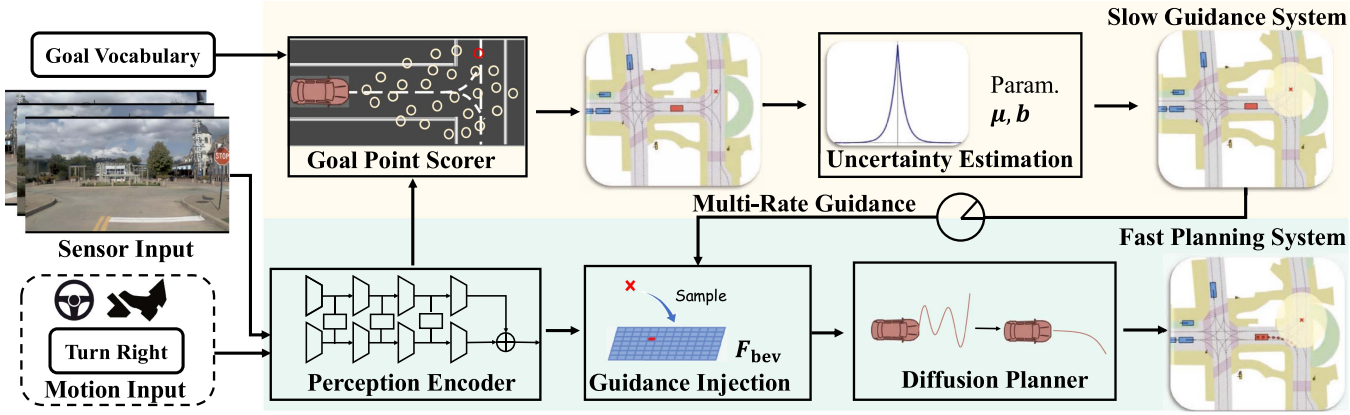

Fig. 2. **Overview of the Mimir architecture.** The high-level model (orange) selects the optimal goal point from a predefined vocabulary $\mathbb{V}$ and estimates its uncertainty using a Laplace distribution to generate guidance $G = \{\boldsymbol{\mu}, \boldsymbol{b}\}$. The multi-rate Guidance further predicts the extended goal point, allowing the system to operate at different rates. The low-level planner (green) fuses the guidance information with perception features through a Guidance Injection module, then generates trajectories via the diffusion planner. Mimir decouples guidance generation from trajectory planning for optimized efficiency.

$$\arg\max_{\theta} P_{\theta}^{\text{low}}(O_{1:T}^{\text{traj}} \mid I_{-n:0}^{\text{sensor}}, I_{-n:0}^{\text{motion}}, G) \tag{3}$$

Where $P_{\theta}^{\text{high}}(\cdot)$ represents the high-level model that captures guidance distributions. In contrast, $P_{\theta}^{\text{low}}(\cdot)$ denotes the low-level model that generates trajectory distribution.

IV. Methodology

As illustrated in Fig. 2, we propose Mimir, a robust and efficient fast-slow system that enhances trajectory generation quality through high-level uncertainty modeling. We begin by introducing the Perception Encoder and Goal Point Scorer to select the raw goal point initially. Next, we introduce Uncertainty Estimation of the goal point using the Laplace distribution. Following this, we present an extended goal point prediction strategy to construct our Multi-Rate Guidance. Finally, we describe how Guidance Injection is applied to effectively incorporate guidance into the Diffusion Planner.

A. Perception Encoder and Goal Point Scorer

Given the sensor inputs and ego-motion, Mimir first extracts the perception features using the Perception Encoder and then selects a raw goal point from the goal point vocabulary $\mathbb{V}$ using a Goal Point Scorer.

Perception Encoder. Typical sensor inputs include image $I \in \mathbb{R}^{3\times H_{\text{img}}\times W_{\text{img}}}$ and LiDAR $L \in \mathbb{R}^{K\times 3}$. Following Transfuser [3], we extract features from camera and LiDAR inputs using two separate resnet50 backbones. To enable a deep fusion between image I and LiDAR L, a transformer is utilized at each block of the resnet, progressively integrating the backbone features into BEV feature F_{bev}. During training, the BEV features F_{bev} are supervised via two heads: a map head for HD map annotations, and an agent head for bounding boxes of surrounding agents. Notably, due to the synthetic image in navsimv2 [2] lacking corresponding LiDAR, a learnable feature substitutes the LiDAR input in the deployment of navsimv2.

Goal Point Scorer. We follow GoalFlow [7] to model the goal point as high-level guidance, representing the ego vehicle's location at the 4-second horizon. Initially, we cluster the endpoints of trajectories to construct a vocabulary $\mathbb{V} = \{g_i\}_{i=1}^{8192}$ containing 8,192 candidate goal points. For each goal point $g_i \in \mathbb{V}$, we predict two scores: δ^{dac}, which measures Drivable Area Compliance (DAC), and δ^{dis}, which indicates the proximity between g_i and the endpoint g_{end} of the ground-truth trajectory τ_{gt} using softmax. Finally, the candidate goal point with the highest combined score is then selected as the raw goal point g_{raw}.

B. Uncertainty Estimation

In previous work, hierarchical frameworks [7], [9] often employed a powerful model to plan deterministic high-level commands like meta-actions, goal points, and language instructions to control the low-level planners. However, in autonomous driving, driving behaviors are often not deterministic. Goal points are often used as guiding information in motion planning tasks [26], [27]. Approaches like GoalFlow [7] and GoalGan [24] sample the goal points from a fixed candidate set, which limits the model's ability to generalize across out-of-distribution scenarios. To address these issues, we propose using a lightweight model to capture the uncertainty in goal point guidance and extend the fixed set sampling into continuous space modeling.

The raw goal point g_{raw}, defined by x and y coordinates, is a deterministic quantity. To model its inherent uncertainty, we transform it into a distribution over the 2D coordinate space. Common approaches [10], [11] represent this uncertainty using parametric distributions such as Laplace or Gaussian centered at the predicted coordinate. Specifically, we use Laplace modeling to represent goal point distributions based on the given raw goal points. For each scenario, the x, y coordinate of the goal point are designed to satisfy the Laplace distribution, whose probability density satisfies:

$$f(v_i|\mu_i, b_i) = \frac{1}{2b_i}\exp\left(-\frac{|v_i - \mu_i|}{b_i}\right) \tag{4}$$

where v_i denotes the coordinate value (i.e., v_1 for the x-axis and v_2 for the y-axis), μ_i is the location parameter, and b_i is the scale parameter controlling the spread of the distribution.

As for the module structure, we first embed g_{raw} into a goal point query, which interacts with the BEV feature map F_{bev} through spatial cross-attention. This step allows spatial semantic

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

context to be injected into the goal representation. Subsequently, agent-level cross-attention is performed between the goal query and F_{agent}, which derives from the agent head in Perception Encoder, enabling the modeling of dynamic interactions between the goal and surrounding agents. The output is processed by a feed-forward network for non-linear transformation. Two MLP heads then predict the refined goal point $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{R}^2$ and its associated uncertainty $\boldsymbol{b} = (b_1, b_2) \in \mathbb{R}^2_+$, modeled as the scale parameter of a Laplace distribution.

We adopt the Negative Log-Likelihood of the Laplace distribution as our loss function to independently train the $\boldsymbol{\mu}$ and $\boldsymbol{b}$ parameters for the x and y coordinates. The final loss is formally defined as:

$$L_{\text{unc}} = \sum_{i=1}^{2} \log(2b_i) + \frac{|v_i - \mu_i|}{b_i} \tag{5}$$

C. Multi-Rate Guidance

In hierarchical frameworks, the high-level planner is typically implemented using a large model, which significantly slows down the overall inference process. For instance, GoalFlow operates at an inference rate of only 12 FPS, significantly lower than the lightweight trajectory decoder of DiffusionDrive (39 FPS), which limits its practical deployment. To address this issue, we propose three lightweight approaches (MLP-Based Predictor, Linear Kinematic Extrapolation, and DAC-Score Guided Kinematic Extrapolation) that allow the high-level module to avoid generating guidance at every frame, thereby enabling multi-rate guidance for the low-level planner. Specifically, at each frame, we predict an extended goal point $\boldsymbol{\mu}_{\text{extend}}$ for the next frame, while directly inheriting the current frame's uncertainty parameter $\boldsymbol{b}$.

MLP-Based Predictor. We first attempt to use a learning-based approach to invoke the high-level module once every two frames. For goal prediction head, we explore two designs: (1) a shared MLP head that jointly predicts goal points for the next two frames, and (2) two separate MLP heads, each predicting one frame with uncertainty guidance. The shared-head design shows unstable training, likely due to differing optimization targets across frames. Therefore, we adopt the separate-head approach. Details are presented in the experiments. *Linear Kinematic Extrapolation.* We also explore rule-based approaches. A naive interpolation method involves extrapolating the future trajectory using basic laws of motion. Assuming the vehicle undergoes uniform linear motion over a short horizon, we extrapolate the goal point based on the last two points of the current predicted trajectory. This can be formulated using the following physics-based equation:

$$\boldsymbol{\mu}_{\text{linear}} = \boldsymbol{\mu}_{\text{current}} + \mathbf{v_0}\Delta t + \frac{1}{2}\mathbf{a}(\Delta t)^2 \quad \text{where} \quad \Delta t = t - t_0 \tag{6}$$

where $\mathbf{r}$, $\mathbf{v}$, and $\mathbf{a}$ are all two-dimensional vectors in the plane. $\mathbf{v}_0$ and t_0 represent the velocity and timestamp of the last point in the current predicted trajectory.

DAC-Score Guided Kinematic Extrapolation. As shown in Fig. 3, while simple kinematic extrapolation is effective in most cases due to the linear nature of vehicle motion, it can lead to off-road predictions during turns. To address this, we refine the extrapolated goal point using the DAC score map to encourage drivable area compliance.

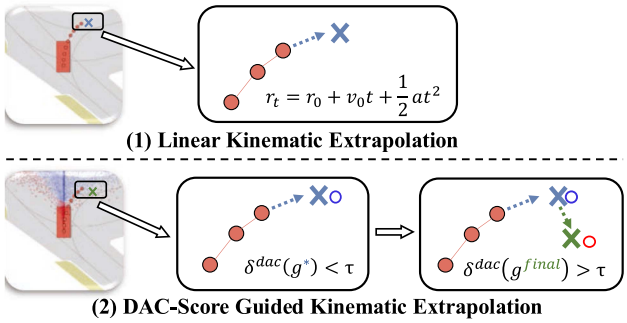

Fig. 3. **Extended Goal Point Extrapolation.** (1) Linear Kinematic Extrapolation directly performs linear extrapolation based on historical trajectory points; (2) DAC-Score Guided Kinematic Extrapolation refines the extended goal point prediction by incorporating DAC scores during extrapolation. In the DAC map visualization, red points indicate higher DAC scores approaching 1, while blue points represent lower scores near 0.

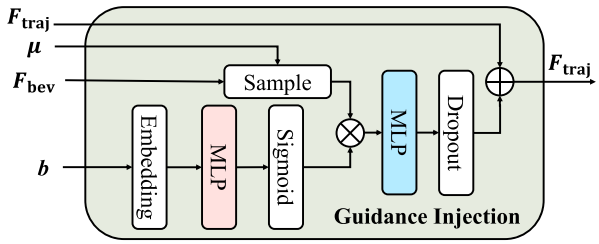

Fig. 4. Guidance Injection module.

In Section IV-B, each candidate point in the vocabulary $\mathbb{V}$ is associated with a DAC score δ^{dac}, where a score closer to 1 indicates higher likelihood of being in the drivable area, while a score closer to 0 suggests the point is likely off-road. Given a linearly kinematic extrapolated goal point $\boldsymbol{\mu}_{\text{linear}}$, we first locate its nearest neighbor $\boldsymbol{\mu}* \in \mathbb{V}$ and approximate its DAC score using $\delta^{\text{dac}}(\boldsymbol{\mu}^8)$. If this score exceeds a threshold τ, we set $\boldsymbol{\mu}_{\text{extend}} = \boldsymbol{\mu}_{\text{linear}}$. Otherwise, we select the most spatially similar candidate from $\mathbb{V}$ whose DAC score exceeds the threshold:

$$\boldsymbol{\mu}_{\text{extend}} = \begin{cases} \boldsymbol{\mu}_{\text{linear}}, & \text{if } \delta^{\text{dac}}(\boldsymbol{\mu}^8) \geq \tau \\ \arg \min\limits_{\substack{\boldsymbol{\mu}_j \in \mathbb{V} \\ \delta^{\text{dac}}(\boldsymbol{\mu}_j) \geq \tau}} \|\boldsymbol{\mu}_j - \boldsymbol{\mu}_{\text{linear}}\|_2, & \text{otherwise} \end{cases} \tag{7}$$

This selection strategy ensures that the extended goal point is not only consistent with physical extrapolation but also satisfies safety constraints by remaining in the drivable area.

D. Guidance Injection Module and Diffusion Planner

In this section, the refined goal point $\boldsymbol{\mu}$ and its associated uncertainty $\boldsymbol{b}$ are injected into a Guidance Injection module, which enhances the trajectory representation. This enriched representation is then passed to the Diffusion Planner to generate the final trajectory. **Guidance Injection module.** As shown in Fig. 4, the refined goal point $\boldsymbol{\mu}$ is first projected from the ego coordinate frame into the BEV space using a projection operator Φ, allowing us to sample the corresponding spatial features from F_{bev}. To reflect the uncertainty in guidance, we compute a confidence weight from $\boldsymbol{b}$ using sinusoidal embedding followed

TABLE I
Navtest LEADERBOARD OF NAVSIMV1

Method	NC ↑	DAC ↑	TTC ↑	Comf. ↑	EP ↑	PDMS ↑
CV	68.0	57.8	50.0	100	19.4	20.6
MLP	93.0	77.3	83.6	100	62.8	65.6
LTF [3]	97.4	92.8	92.4	100	79.0	83.8
TransFuser [3]	97.7	92.8	92.8	100	79.2	84.0
UniAD [4]	97.8	91.9	92.9	100	78.8	83.4
PARA-Drive [25]	97.9	92.4	93.0	99.8	79.3	84.0
GoalFlow‡ [7]	98.2	96.4	93.8	99.4	82.6	87.9
Diff. [6]	98.2	96.2	**94.7**	100	82.2	88.1
Mimir	**98.2**	**97.5**	94.6	**100**	**83.6**	**89.3**

‡ denotes version of the trajectory decoder built upon a ResNet backbone with 256-dimensional feature encoding.

by an MLP and a sigmoid activation. The sampled BEV feature is then modulated by this confidence weight through element-wise multiplication to produce feature F_{guidance}:

$$F_{\text{guidance}} = \Phi(F_{\text{bev}}, \boldsymbol{\mu}) \times \text{Sigmoid}(\text{MLP}(\text{Emb}(\boldsymbol{b}))) \quad (8)$$

This guidance is injected into the trajectory query F_{traj} through residual fusion:

$$F_{\text{traj}} = F_{\text{traj}} + \text{Dropout}(\text{MLP}(F_{\text{guidance}})) \quad (9)$$

The trajectory query F_{traj} is initialized from a set of anchor trajectories, which encode prior knowledge of typical motion patterns and act as carriers for downstream information integration.

Diffusion Planner. The final trajectory $\{O_1^{\text{traj}}, \ldots, O_T^{\text{traj}}\}$ is predicted using a diffusion-based decoder adapted from DiffusionDrive [6]. This decoder takes the updated trajectory query F_{traj} as input and iteratively denoises it to generate the final trajectory.

V. EXPERIMENT

A. Dataset and Metrics

Our method is evaluated on two benchmarks: Navsimv1 [1] and Navsimv2 [2]. Navsimv1 uses Navtest as the test set, which includes 134 real-world scenarios totaling over 10,000 frames. All test scenes in Navtest are distributionally similar to the training data, and surrounding agents follow non-reactive replayed logs. In Navsimv1, PDMS is used as the metric, which is a weighted combination of sub-scores, including No at-fault Collisions (NC), where at-fault collisions refer to those that should have been avoided by proper planning, Drivable Area Compliance (DAC), Time-to-collision (TTC), Comfort (Comf.), and Ego Progress (EP).

In contrast, Navsimv2 employs a more challenging test set, Navhard, consisting of 77 scenarios and nearly 6,000 frames, including 450 real and 5,462 synthetic frames. Its evaluation protocol provides a pseudo closed-loop assessment through a two-stage process: In stage 1, it runs a 4-second open-loop simulation using real-world initial scenes without environmental feedback; in stage 2, it evaluates the system in synthesized follow-up scenes initialized from perturbed states, simulating future interactions. The main evaluation metric is EPDMS, which extends the v1 version by incorporating five additional sub-metrics Driving Direction Compliance (DDC), Traffic Light Compliance (TLC), Lane Keeping (LK), History Comfort (HC), and Extended Comfort (EC). These sub-metrics are integrated

TABLE II
Navhard LEADERBOARD OF NAVSIMV2

Metric	Stage	CV	MLP	LTF [3]	GoalFlow† [7]	Diff.† [6]	Mimir
NC ↑	S1	88.8	93.2	96.2	96.0	**96.8**	95.6
	S2	**83.2**	77.2	77.7	79.4	80.3	80.6
DAC ↑	S1	42.8	55.7	79.5	**92.6**	88.2	92.2
	S2	59.1	51.9	70.2	**78.2**	74.4	77.1
DDC ↑	S1	70.6	86.6	99.1	99.3	99.3	**99.7**
	S2	76.5	74.4	84.2	86.4	86.1	**89.3**
TLC ↑	S1	99.3	99.3	99.5	99.3	99.3	**99.5**
	S2	98.0	98.2	98.0	97.7	**98.4**	97.7
EP ↑	S1	77.5	81.2	84.1	84.0	**84.5**	84.0
	S2	71.3	77.1	85.1	86.5	**87.4**	86.4
TTC ↑	S1	87.3	92.2	95.1	**95.7**	94.6	94.6
	S2	81.1	75.0	75.6	76.0	76.9	**77.3**
LK ↑	S1	78.6	83.5	94.2	97.1	95.5	**98.0**
	S2	47.9	40.8	45.4	45.5	**50.4**	48.8
HC ↑	S1	97.1	97.5	97.5	97.5	97.5	**97.5**
	S2	97.1	**97.8**	95.7	94.4	95.5	94.5
EC ↑	S1	60.4	77.7	79.1	40.4	**79.1**	78.2
	S2	61.9	79.8	75.9	40.4	**69.4**	64.0
EPDMS ↑		10.9	12.7	23.1	28.7	28.9	**34.6**

† indicates the version trained using open-source code without LiDAR input.

TABLE III
FPS OF SOTA METHODS ON Navhard

	Module	DiffusionDrive† [6]	GoalFlow† [7]	Mimir
FPS↑	high-level	/	13	**21**
	low-level	**39**	15	36

TABLE IV
ABLATION OF DIFFERENT COMPONENTS

Metric	Stage	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
NC ↑	S1	**96.8**	95.6	96.4	96.2
	S2	**80.3**	78.9	79.6	79.6
DAC ↑	S1	88.2	**92.0**	90.4	90.6
	S2	74.4	**78.6**	77.6	78.5
DDC ↑	S1	99.3	100	99.3	**100**
	S2	86.1	88.0	88.2	**88.8**
TLC ↑	S1	99.3	99.3	99.3	**99.3**
	S2	**98.4**	97.6	97.8	97.5
EP ↑	S1	84.5	**84.9**	84.4	84.2
	S2	**87.4**	86.4	86.3	86.0
TTC ↑	S1	94.6	95.1	95.1	**95.3**
	S2	76.9	75.9	76.7	**77.0**
LK ↑	S1	95.5	**98.0**	97.1	97.5
	S2	**50.4**	47.2	48.2	48.5
HC ↑	S1	97.5	97.7	97.7	**97.7**
	S2	**95.5**	94.6	93.8	94.2
EC ↑	S1	79.1	70.2	77.7	**78.2**
	S2	**69.4**	53.7	59.8	64.1
EPDMS ↑		28.9	31.1	31.1	**33.3**

TABLE V
EXTENDED GOAL POINT PREDICTION

Metric	Stage	Γ_0	Γ_1	Γ_2	Γ_3	Γ_4
Latency (ms)↓	/	3.09	3.61	**3.27**	3.62	4.13
NC ↑	S1	**96.2**	95.8	95.6	95.6	95.6
	S2	79.6	**81.4**	80.5	80.6	80.7
DAC ↑	S1	90.6	91.5	90.8	**92.2**	92.2
	S2	**78.5**	78.0	77.1	77.1	76.9
DDC ↑	S1	**100**	99.6	99.5	99.7	blue99.6
	S2	88.8	88.8	88.0	**89.3**	89.2
TLC ↑	S1	99.3	99.3	99.5	**99.5**	99.5
	S2	97.5	**97.8**	97.6	97.7	97.7
EP ↑	S1	84.2	84.1	**84.5**	84.0	84.0
	S2	86.0	86.3	**87.1**	86.4	86.2
TTC ↑	S1	**95.3**	95.3	94.2	94.6	94.4
	S2	77.0	77.6	77.0	**77.3**	77.3
LK ↑	S1	97.5	98.0	98.0	**98.0**	97.9
	S2	48.5	**49.0**	48.3	48.8	49.0
HC ↑	S1	**97.7**	97.7	97.7	97.5	97.5
	S2	94.2	94.4	94.4	**94.5**	**94.6**
EC ↑	S1	78.2	78.2	**79.1**	78.2	78.2
	S2	64.1	63.0	**64.3**	64.0	63.2
EPDMS ↑		33.3	33.5	34.0	**34.6**	34.4

through a combination of multiplicative factors and additive terms, as detailed in the navsim-v2 documentation[1].

B. Implementation Details.

For sensor inputs, we concatenate the left-front, front, and right-front camera views into a 256×1024 image and combine it with LiDAR data in Navtest. As the Navhard includes synthetic scenes without LiDAR, we replace the LiDAR input in these scenarios with a learnable latent feature. The goal point scorer uses the checkpoint of GoalFlow, while the low-level trajectory decoder employs a ResNet-50 backbone. In the DAC-Score Guided Kinematic Extrapolation module, we set the threshold $\tau = 0.5$. Defaultly, the slow guidance system performs reasoning once every two frames, while the fast planning system performs reasoning at every frame.

For training, the goal point module is supervised using the uncertainty-aware loss L_{unc} introduced in Section IV-B, and the trajectory decoder is trained with an L1 loss. All models are trained on 8 NVIDIA H20 GPUs using the navtrain with a batch size of 64. We use a hidden dimension of 256 and a learning rate of 6×10^{-4}. No external datasets, ensembled models, or test-time augmentations are used.

C. Main Result.

In Table I, we compare our method, Mimir, with several state-of-the-art approaches on the Navtest benchmark, which primarily consists of real-world driving scenarios. To ensure a fair comparison, we use trajectory decoders of the same capacity across all methods to ensure a fair comparison.: a Resnet-50 backbone with 256-dimensional feature representations. Mimir presents a strong performance, achieving the best scores in DAC and EP, which suggests that our method generalizes well to regular in-distribution scenes. Furthermore, it demonstrates the ability to improve driving efficiency (high EP score) without compromising safety.

[1]https://github.com/autonomousvision/navsim/blob/main/docs/metrics.md

TABLE VI
PDMS ON `Navtest` UNDER DIFFERENT EXTRAPOLATION HORIZONS

n	Method	0	3	5	10	20
PDMS ↑	**L-KE**	89.3	82.2	55.0	36.2	26.6
	DAC-KE	89.3	**86.1**	**65.1**	**45.7**	**43.2**

As shown in Table II, we report results on the more challenging benchmark Navhard, including a large portion of synthetically generated scenarios. Our method achieves nearly a 20% improvement over the best baseline in terms of overall EPDMS and ranks first in several sub-metrics. Compared to DiffusionDrive, Mimir achieves consistently higher scores across key safety and compliance metrics such as DAC, and DDC with gains of 3–4 points. We attribute this to the hierarchical structure of our model, where a strong high-level module provides more accurate guidance for downstream trajectory generation. Interestingly, for critical metrics like DDC, Mimir performs better on stage 2 synthetic scenes than on Stage 1 real scenes. This suggests that our model is robust to distribution shifts and better equipped to handle difficult situations. Furthermore, compared to the similar hierarchical framework GoalFlow, Mimir achieves a significantly higher score in EC, which measures the consistency between adjacent frames in the generated trajectory, indicating smoother and more stable driving behavior.

In Table III, we compare the FPS of different modules across methods. Compared to GoalFlow, our high-level module achieves a 1.6× speedup, owing to the design of our multi-rate guidance mechanism. Moreover, by adopting a truncated diffusion process inherited from DiffusionDrive, our low-level module achieves faster inference with fewer denoising steps compared to GoalFlow (2vs. 5), achieving higher FPS (32vs. 15).

D. Ablation of Different Components.

In this section, we perform an ablation study to examine the impact of individual components in our model, as shown in Table IV. Compared to $\mathcal{M}_0$, $\mathcal{M}_1$ yields notable improvements in key metrics such as DAC and DDC, suggesting that the additional information effectively enhances spatial guidance in the low-level trajectory decoder. However, we also observe a considerable drop in the EC score compared to $\mathcal{M}_0$, which we attribute to a common limitation in hierarchical frameworks like GoalFlow, where the top and bottom modules are not optimized in a fully end-to-end manner.

Compared to $\mathcal{M}_1$, $\mathcal{M}_2$ achieves a similar EPDMS score but suffers from training instability. We believe this is due to the quadratic penalty in the log-likelihood of Gaussian distributions, which makes them more sensitive to outliers than the linear penalty in Laplace distributions.

In contrast, $\mathcal{M}_3$ shows clear improvements over $\mathcal{M}_1$, with a 2-point gain in overall EPDMS and significant improvements in TTC and EC. Notably, TTC estimates the likelihood of collision within a short time horizon. We hypothesize that uncertainty modeling in $\mathcal{M}_3$ helps the model capture potential interaction risks with nearby agents, thereby reducing collision probability.

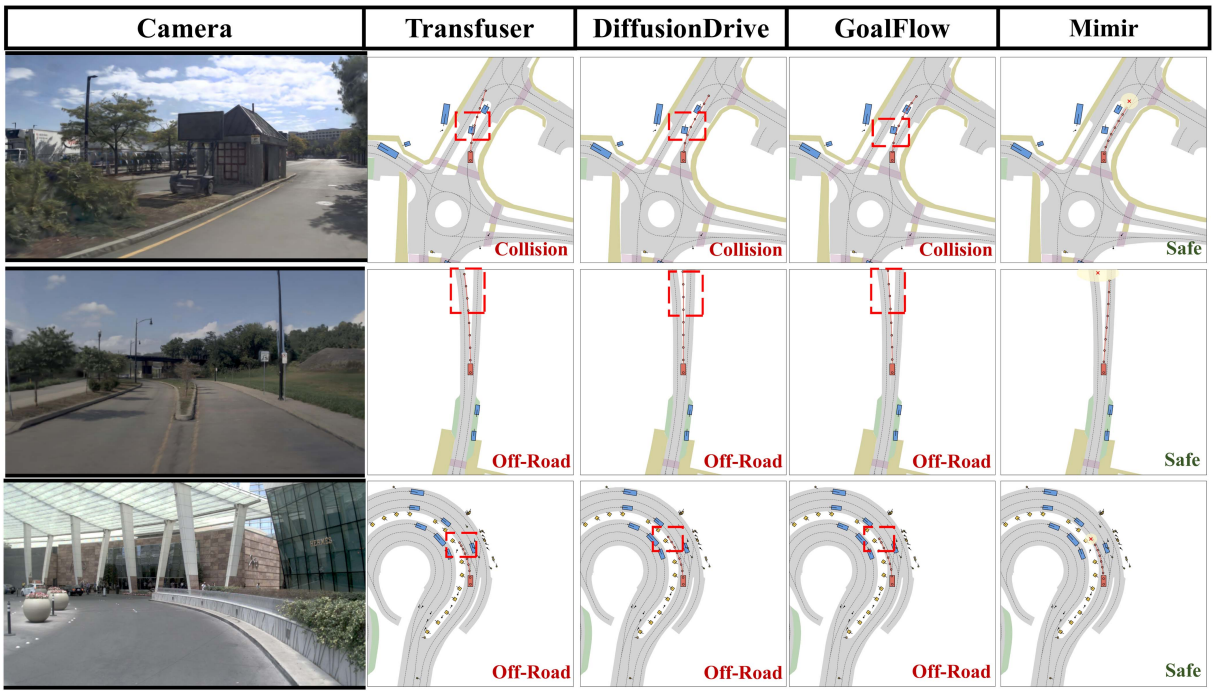

Fig. 5. **Visualization.** From top to bottom, we illustrate the capabilities of the four algorithms in Exit-Ramp Scenario, forked-intersection scenario, and roundabout scenario, respectively. The red cross represents the predicted goal point and the size of the yellow area around the goal point represents the level of uncertainty. Mimir leverages uncertainty estimation to mitigate the effects of inaccurate high-level guidance, enabling the generation of safer trajectories.

Additionally, the improved EC score compared to both $\mathcal{M}_1$ and $\mathcal{M}_2$ suggests that modeling uncertainty in goal point predictions provides temporal tolerance for slight fluctuations between adjacent frames.

E. Extended Goal Point Prediction.

A comparative study, as shown in Table V, focuses on single extended goal point prediction, due to the constraint of at most two adjacent frames in the Navhard benchmark. We analyze four approaches using a single NVIDIA H20 GPU. Our latency analysis, which measures the combined inference time of the uncertainty estimation and multi-rate guidance modules, reveals that Γ_1 introduces minimal computational overhead compared to other methods, while Γ_2 and Γ_3 exhibit similar latency levels. Notably, all variants maintain sub-10ms execution times, demonstrating their practical feasibility for real-time systems.

Despite Γ_1's lightweight design, it achieves comparable performance to Γ_0 in terms of overall EPDMS. We believe this is because the temporal gap between two consecutive frames is very small, resulting in highly similar condition features between them.

For Γ_2, this simple rule-based method yields strong performance surprisingly. We attribute this to the fact that, over short time horizons (e.g., 0.5 seconds), vehicle motion tends to follow linear kinematics.

Γ_3 achieves a higher EPDMS, with a particularly notable improvement of 1.4 points in the Stage 1 DAC score. This highlights the effectiveness of DAC-guided replanning. However, we observe that both Γ_2 and Γ_3 perform similarly on Stage 2 DAC. We hypothesize that this is due to prediction errors in the DAC score map within the synthetic data.

Γ_4 shows that the performance remains comparable to the Γ_3 . We speculate that this is because the prediction horizon is relatively short (within 0.5s), where the fluctuation range of uncertainty is small, leading to limited differences between the two strategies.

In Tab. VI, we further assess both Linear Kinematic Extrapolation (L-KE) and DAC-Score Guided Kinematic Extrapolation (DAC-KE) on Navtest, which provides continuous history frames suitable for long-horizon analysis. In this setting, the goal points are extrapolated from the n-th historical frame, while $n = 0$ denotes using the goal point from the current frame. We find DAC-KE effectively alleviates the limitations of the L-KE and its practical extrapolation limit is around 3.

F. Quality Results

In Fig. 5, we present a visual comparison between Mimir and four other methods. Compared to the baselines, Mimir consistently generates safer trajectories. We observe that when the goal point is inaccurate, Mimir exhibits higher uncertainty, which serves as a safeguard against low-quality high-level guidance. On the other hand, when the goal point is relatively accurate, our trajectory planner produces higher-quality trajectories. We attribute this to our guidance injection module, which effectively integrates environmental context with high-level guidance for improved planning.

VI. Conclusions and Future Work

We present Mimir, a hierarchical dual-system framework for end-to-end autonomous driving with integrated uncertainty-aware guidance. Goal point uncertainty is modeled via a Laplace distribution to improve robustness. A DAC-guided kinematic

extrapolation enables fast extended goal prediction for multi-rate processing, while a Guidance Injection module effectively integrates high-level guidance into the trajectory planner. Experiments on Navhard and Navtest show SOTA performance, with ablation studies confirming the effectiveness of each component.

Looking forward, we plan to extend our work to VLA by exploring how uncertainty in language models can influence the performance of low-level planners. We will also continue to push the performance ceiling of the guidance extrapolation method on longer time horizons, as well as develop a goal-caching mechanism to enable asynchronous communication between hierarchical modules.

REFERENCES

[1] D. Dauner et al., "NAVSim: Data-driven non-reactive autonomous vehicle simulation and benchmarking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 28706–28719.

[2] W. Cao et al., "Pseudo-simulation for autonomous driving," in *Proc. Conf. Robot Learn.*, 2025, pp. 4709–4722.

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

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

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

[6] B. Liao et al., "Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2025, pp. 12037–12047.

[7] Z. Xing et al., "Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2025, pp. 1602–1611.

[8] W. Sun, X. Lin, Y. Shi, C. Zhang, H. Wu, and S. Zheng, "Sparsedrive: End-to-end autonomous driving via sparse scene representation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2025, pp. 8795–8801.

[9] B. Jiang et al., "Senna: Bridging large vision-language models and end-to-end autonomous driving," 2024, *arXiv:2410.22313*.

[10] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5580–5590.

[11] T. Yeo, O. F. Kar, and A. Zamir, "Robustness via cross-domain ensembles," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 12189–12199,.

[12] T. Ramalho and M. Miranda, "Density estimation in representation space to predict model uncertainty," in *Proc. Int. Workshop Eng. Dependable Secure Mach. Learn. Syst.*, 2020, pp. 84–96.

[13] G. Zhang, H.-a. Gao, Z. Jiang, H. Zhao, and Z. Zhedong, "Ctrl-U: Robust conditional image generation via uncertainty-aware reward modeling," in *Proc. Int. Conf. Learn. Representations*, 2025, pp.–1–20.

[14] M. Valdenegro-Toro, "Deep sub-ensembles for fast uncertainty estimation in image classification," in *Proc. Int. Adv. Conf. Neural Inf. Process Syst.*, Workshop, 2025, pp. 1–7.

[15] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proc. 6th Annu. Conf. Comput. Learn. Theory*, 1993, pp. 5–13.

[16] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 681–688.

[17] A. Lyzhov, Y. Molchanova, A. Ashukha, D. Molchanov, and D. Vetrov, "Greedy policy search: A simple baseline for learnable test-time augmentation," in *Proc. Conf. Uncertainty Artif. Intell.*, 2020, pp. 1308–1317.

[18] R. Michelmore, M. Wicker, L. Laurenti, L. Cardelli, Y. Gal, and M. Kwiatkowska, "Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 7344–7350.

[19] X. Gu, G. Song, I. Gilitschenski, M. Pavone, and B. Ivanovic, "Producing and leveraging online map uncertainty in trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 14521–14530.

[20] A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3145–3153.

[21] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 6840–6851.

[22] J. Wang et al., "HE-Drive: Human-like end-to-end driving with vision language models," 2024, *arXiv:2410.05051*.

[23] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *Proc. Int. Conf. Learn. Representations*, 2023, pp.–1–28.

[24] P. Dendorfer, A. Osep, and L. Leal-Taixé, "Goal-GAN: Multimodal trajectory prediction based on goal position estimation," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp.–1–17.

[25] X. Weng, B. Ivanovic, Y. Wang, Y. Wang, and M. Pavone, "Para-drive: Parallelized architecture for real-time autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 15449–15458.

[26] C. Wang, Y. Wang, M. Xu, and D. J. Crandall, "Stepwise goal-driven networks for trajectory prediction," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2716–2723, Apr. 2022.

[27] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "BiTraP: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1463–1470, Apr. 2021.

[28] Y. Xiao, F. Codevilla, D. Porres, and A. M. López, "Scaling vision-based end-to-end autonomous driving with multi-view attention learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 1586–1593.

[29] P. Yang et al., "UncAD: Towards safe end-to-end autonomous driving via online map uncertainty," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2025, pp. 1–8.

[30] Y. Gao, Q. Zhang, D.-W. Ding, and D. Zhao, "Dream to drive with predictive individual world model," *IEEE Trans. Intell. Veh.*, vol. 9, no 12, pp. 8824–8238, Dec. 2024.

[31] Y. Zheng et al., "PlanAgent: A multi-modal large language agent for closedloop vehicle motion planning," 2024, *arXiv:2406.01587*.

[32] D. Li et al., "Planning-inspired hierarchical trajectory prediction via lateral-longitudinal decomposition for autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 692–703, Jan. 2024.

[33] Q. Zhang et al., "TrajGen: Generating realistic and diverse trajectories with reactive and feasible agent behaviors for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24474–24487, Dec. 2022.

[34] Y. Zheng et al., "World4Drive: End-to-end autonomous driving via intention-aware physical latent world model," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2025, pp. 28632–28642.