DriveMind: Mind-Evolving Belief Tracking for Closed-Loop Autonomous Driving

Yan Shi
University of Washington
yanshi@uw.edu

Kehua Chen*
University of Washington
zeonchen@uw.edu

Jiarui Qi
University of Washington
jiaruiqi@uw.edu

Yihong Tang
McGill University
yihong.tang@mail.mcgill.ca

Haicheng Liao
University of Michigan
haichenl@umich.edu

Yinhai Wang*
University of Washington
yinhai@uw.edu

Abstract

Recent multimodal large language models (MLLMs) can perceive a traffic scene, reason about it, and propose a driving action within a single model. However, existing MLLM-based drivers reason from scratch at each frame and commit to a single explanation, discarding earlier hypotheses about the scene. When the scene is ambiguous, the agent locks in on the wrong interpretation and acts on it before evidence accumulates. In this paper, we propose DriveMind, an MLLM-based driving framework that keeps several hypotheses about the scene alive at the same time and updates their weights as new frames arrive. Inspired by particle filtering, DriveMind propagates each hypothesis through a VLM, scores it against the current observation, and resamples to maintain diversity. On Bench2Drive, adding belief tracking to the same driving pipeline improves the Driving Score (DS) from 68.50 to 81.37 and the Success Rate (SR) from 41.36% to 54.09%. The largest gains occur on ambiguous scenarios such as unsignalized intersections (+20.0 DS) and oncoming-traffic interactions (+20.5 DS). Beyond the overall metrics, DriveMind also improves fine-grained interactive driving abilities such as merging, emergency braking, and give-way. The code is available at: https://github.com/shiyannacl/drivemind.

1 Introduction

MLLMs have recently been adopted for autonomous driving, where a single model takes raw sensor input and outputs both a textual rationale and a driving decision [Cui et al., 2024]. This makes them a natural fit for tasks where the model must explain its choices, such as following natural-language navigation commands or justifying a lane change.

However, driving differs from one-shot question-answering tasks. The agent observes the scene only partially, and the scene keeps changing as it moves. Traffic is also full of ambiguity. For example, a slowing vehicle may be yielding, preparing to turn, or reacting to an unseen obstacle; a pedestrian near the curb may intend to cross or remain waiting. In these cases, multiple explanations can remain plausible at the same time. Several existing MLLM-based driving systems generate a single rationale or planned trajectory at each inference step [Xu et al., 2024, Mao et al., 2023, Renz et al., 2025]. This single-chain reasoning is fluent but brittle: the model picks one interpretation too early, then has to act on it whether or not later frames support it. Figure 1 contrasts the two paradigms: a single chain of thought per frame, versus a set of hypotheses that is carried forward and reweighted over time.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

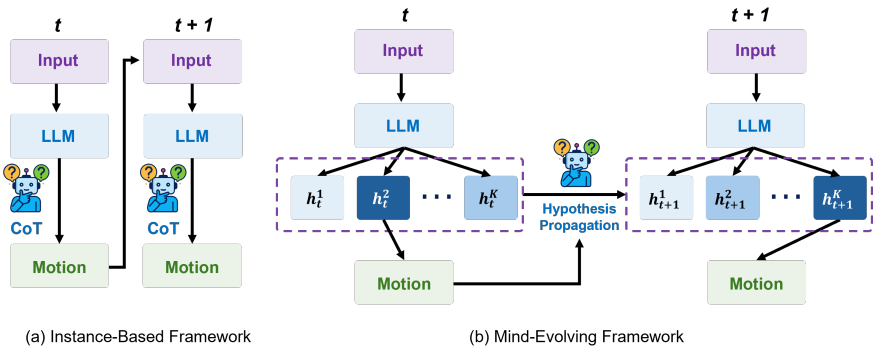

Figure 1: **From single-chain reasoning to mind-evolving belief.** (a) Instance-Based Framework: one rationale per timestep. ("CoT" stands for "Chain of Thought") (b) Mind-Evolving Framework: multiple hypotheses $\{h_t^k\}$ propagated over time; darker color indicates higher hypothesis weight.

This premature commitment is especially problematic in closed-loop control. Once the model forms a wrong interpretation, the resulting action may change subsequent observations in ways that reinforce the initial mistake. Existing uncertainty estimation methods for large language models (LLMs) mainly focus on output variability across multiple generations [Ye et al., 2024, Zhang et al., 2024a]. These methods are useful, but they measure uncertainty over multiple answers to the same question, not how uncertainty changes as new frames arrive.

To address this limitation, we propose *DriveMind*, a closed-loop driving framework that maintains a time-evolving belief state over multiple reasoning hypotheses. DriveMind represents the agent's reasoning state as a weighted set of natural-language hypotheses. Each hypothesis describes a possible interpretation of the scene, and its weight represents its relative support under the accumulated observations. As new frames arrive, DriveMind updates the hypotheses themselves and reweights them based on the new evidence.

For real-time closed-loop control, DriveMind adopts an asynchronous fast-slow Vision-Language-Action (VLA) architecture. The slow subsystem uses an MLLM to perform multi-hypothesis reasoning and belief tracking. The fast subsystem runs at control frequency and uses a diffusion-based controller to generate waypoint trajectories conditioned on the latest belief. On Bench2Drive, adding belief tracking improves DS from 68.50 to 81.37 and SR from 41.36% to 54.09%. The largest gains occur in scenarios requiring intent disambiguation, such as unsignalized intersections (+20.0 DS). On NAVSIM, belief tracking improves PDMS from 78.2 to 81.0. Ablations support the effectiveness of belief tracking, while runtime analysis demonstrates the efficiency of asynchronous inference. Our main contributions are summarized as follows:

- We introduce *DriveMind*, a closed-loop driving framework that maintains a weighted set of natural-language hypotheses and updates them as new observations arrive.
- We design an asynchronous fast-slow VLA architecture where a slow MLLM reasoner performs belief tracking and a fast diffusion controller generates real-time trajectories.
- We show on Bench2Drive and NavSim that explicit belief tracking improves driving performance and robustness, with the largest gains on long-tail scenarios requiring intent disambiguation.

2 Related Work

2.1 Autonomous Driving with LLM

LLMs and MLLMs have been applied to autonomous driving across a range of tasks, from scene understanding and decision-making to motion planning. Existing systems such as LLM-Driver [Chen et al., 2024], DriveGPT4 [Xu et al., 2024], DriveLM [Sima et al., 2024], GPT-Driver [Mao et al., 2023], and LanguageMPC [Sha et al., 2025] use language models for scene understanding, decision making, or motion planning. Other works incorporate commonsense knowledge or linguistic feedback to

improve interpretability and adaptability [Fu et al., 2024, Wen et al., 2024, Zhang et al., 2024b]. More recent closed-loop frameworks, including LMDrive [Shao et al., 2024], SimLingo [Renz et al., 2025], and ReCogDrive [Li et al., 2025], further connect vision-language reasoning with autonomous driving control. These approaches cover a range of reasoning and memory mechanisms; for example, DiLu includes a memory component and reflection to accumulate driving experience [Wen et al., 2024]. DriveMind focuses on carrying forward and reweighting multiple competing scene interpretations as new observations arrive.

2.2 Uncertainty in Reasoning and Prediction

Uncertainty estimation for LLMs has been studied from several perspectives. Some methods use step-level uncertainty signals or chain-of-thought information to improve reasoning or response-wise uncertainty estimates [Yin et al., 2024, Zhang and Zhang, 2025]. Other methods assess reasoning quality and output uncertainty using graph edit distance, coherence scoring, or NLI-based decomposition [Da et al., 2025, Golovneva et al., 2023, Lin et al., 2024, Zhang et al., 2024a]. These methods provide useful estimates of output uncertainty, but they mainly focus on static responses or final predictions rather than how uncertainty evolves over time during closed-loop interaction.

Classical planners address this temporal dimension through explicit belief maintenance over predefined state spaces [Cunningham et al., 2015, Zhang et al., 2020, Ding et al., 2022], but predefined state and behavior representations can limit coverage of rare or open-ended scenarios. DriveMind borrows the idea of temporal belief propagation and uses a VLM to generate hypotheses, allowing hypotheses to express scene interpretations in natural language.

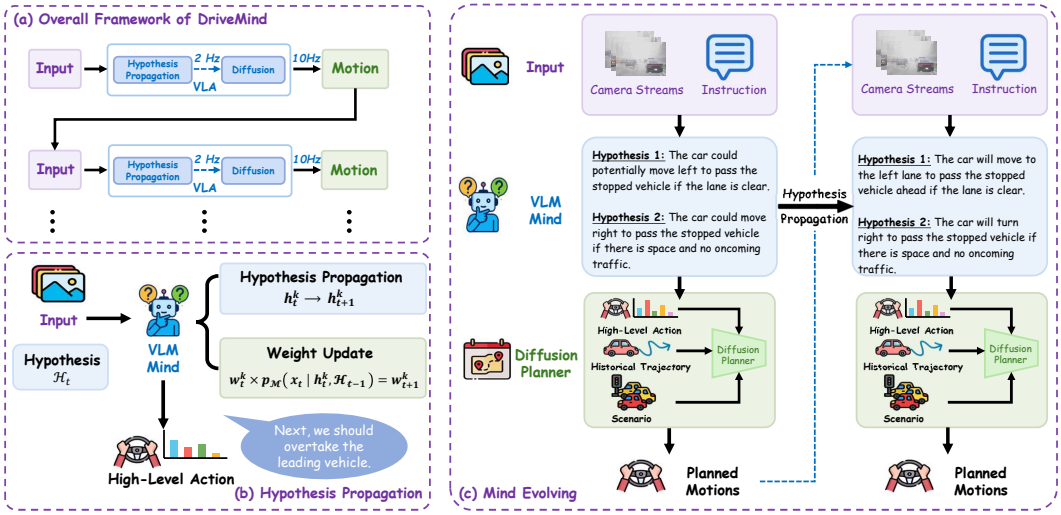

Figure 2: ***DriveMind* overview.** (a) An asynchronous fast–slow architecture: a slow vision-language model (VLM) maintains a multi-hypothesis belief state, while a fast diffusion planner generates trajectories. (b) Belief tracking through hypothesis propagation and reweighting. (c) The belief state evolves as new observations arrive.

3 Problem Formulation

We formulate autonomous driving as a sequential decision-making process in an interactive environment $\mathcal{E}$. At each time step t, the agent receives an observation $\boldsymbol{x}_t$, executes an action a_t, and the environment transitions to $\boldsymbol{x}_{t+1} = \mathcal{E}(\boldsymbol{x}_t, a_t)$. Let

$$\mathcal{H}_t = (\boldsymbol{x}_{1:t}, a_{1:t-1}) \tag{1}$$

denote the history available at time t.

Let $h_t \in \mathcal{R}$ denote a latent reasoning hypothesis, where $\mathcal{R}$ is the open-ended space of natural-language interpretations of the scene. A VLM-based driving policy can then be written as a marginalization

over hypotheses:

$$p(a_t \mid \mathcal{H}_t) = \sum_{h_t \in \mathcal{R}} p(a_t \mid h_t, \mathcal{H}_t) p(h_t \mid \mathcal{H}_t). \tag{2}$$

The key challenge is that the reasoning space $\mathcal{R}$ is open-ended and cannot be enumerated explicitly. A single-hypothesis driving policy can be viewed as approximating Eq. (2) by selecting one hypothesis and conditioning action generation on it. However, this approximation forces an early commitment: if the chosen hypothesis turns out to be wrong, the agent has already executed an action based on it, and the resulting observations may make recovery difficult.

DriveMind instead maintains a finite approximation to the posterior over reasoning hypotheses. At timestep t, we represent the belief state as a weighted set of K natural-language hypotheses:

$$\mathcal{B}_t = \{(h_t^k, w_t^k)\}_{k=1}^K, \qquad \sum_{k=1}^K w_t^k = 1, \tag{3}$$

where h_t^k is the k-th natural-language hypothesis and w_t^k is its normalized belief weight. This weighted set gives a discrete approximation to the posterior:

$$p(h_t \mid \mathcal{H}_t) \approx \sum_{k=1}^K w_t^k \delta(h_t = h_t^k), \tag{4}$$

where $\delta(\cdot)$ denotes a point mass on a hypothesis.

Substituting this approximation into Eq. (2) yields:

$$p(a_t \mid \mathcal{H}_t) \approx \sum_{k=1}^K w_t^k p(a_t \mid h_t^k, \mathcal{H}_t). \tag{5}$$

The belief state $\mathcal{B}_t$ is propagated through the closed-loop process and also serves as the conditioning signal for trajectory generation.

4 Methodology

As illustrated in Figure 2, DriveMind has two subsystems running asynchronously. A slow VLM reasoner updates the multi-hypothesis belief state $\mathcal{B}_t$ at a low frequency (around 2 Hz). A fast diffusion controller runs at control frequency (around 10 Hz) and generates waypoint trajectories conditioned on the most recently available belief $\mathcal{B}_t$; between two slow updates, the controller reuses the last belief without waiting. We describe the slow subsystem in Section 4.1 and the fast subsystem in Section 4.2.

4.1 High-Level Reasoner: Multi-Hypothesis Belief-State Propagation

Given the previous belief state $\mathcal{B}_{t-1} = \{(h_{t-1}^k, w_{t-1}^k)\}_{k=1}^K$, the high-level reasoner updates both the hypotheses and their weights using the new observation $\boldsymbol{x}_t$. We use a particle-filter-style [Gordon et al., 1993, Doucet et al., 2001] update with three steps: predict, correct, and resample, where each hypothesis h_t^k plays the role of a particle and its weight w_t^k reflects its current plausibility.

Hypothesis Propagation (Predict). For each previous hypothesis h_{t-1}^k, the VLM samples an updated hypothesis conditioned on the previous hypothesis, the previous action, and the current observation:

$$h_t^k \sim p_{\mathcal{M}}(\cdot \mid h_{t-1}^k, a_{t-1}, \boldsymbol{x}_t, \mathcal{H}_{t-1}). \tag{6}$$

This equation describes a prompted VLM update inspired by particle filtering, without specifying an explicit probabilistic state-transition model.

Evidence-Consistency Reweighting (Correct). After hypothesis propagation, we then ask the VLM to score how well each updated hypothesis is supported by the current observation:

$$s_t^k = f_{\mathcal{M}}(h_t^k, \boldsymbol{x}_t, \mathcal{H}_{t-1}), \tag{7}$$

where $f_{\mathcal{M}}(\cdot)$ is a VLM-based scoring function. In our implementation, we prompt the VLM with the hypothesis text, the current observation, and a fixed scoring rubric; the VLM outputs an integer score in [1,10] which we rescale to [0,1]. For example, a hypothesis that a pedestrian is likely to cross should receive a higher score when the pedestrian moves toward the lane and a lower score when the pedestrian remains stationary. We provide the prompt template in Appendix D.

We update each hypothesis weight by combining its previous weight with an exponential factor based on its consistency score:

$$\tilde{w}_t^k = w_{t-1}^k \exp(s_t^k/\beta), \tag{8}$$

$$w_t^k = \frac{\tilde{w}_t^k}{\sum_{j=1}^K \tilde{w}_t^j}, \tag{9}$$

Here, $\beta > 0$ controls the sensitivity to score differences, and the normalized weights represent relative support among hypotheses rather than calibrated probabilities.

Diversity Maintenance (Resample). A common issue in particle filters is degeneracy, where one hypothesis captures all the weight. To maintain diversity, we monitor the effective sample size (ESS) [Kim et al., 2025]:

$$ESS_t = \frac{1}{\sum_{k=1}^K (w_t^k)^2}. \tag{10}$$

When $ESS_t < \alpha K$, where α is a predefined threshold, we resample K hypotheses from the current belief state according to their weights. Because naive resampling can produce near-duplicate hypotheses, we add a diversity-aware regeneration step. We embed each hypothesis with the VLM's text encoder and compute pairwise cosine similarity. For pairs above a threshold ρ (we use $\rho = 0.9$), we keep the higher-weighted hypothesis and prompt the VLM to regenerate the other so that it remains consistent with $\boldsymbol{x}_t$ but covers a different plausible interpretation. See Appendix A.5.

Maneuver Marginal. In addition to maintaining the belief state, the VLM also predicts a high-level maneuver distribution for each hypothesis, $p_{\mathcal{M}}(u_t \mid h_t^k, \mathcal{H}_t)$, where u_t is a discrete action label defined in Appendix D. Marginalizing over the belief gives an interpretable maneuver distribution:

$$p(u_t \mid \mathcal{H}_t) = \sum_{k=1}^K w_t^k \, p_{\mathcal{M}}(u_t \mid h_t^k, \mathcal{H}_t), \tag{11}$$

which we use for analysis and visualization (Figure 4). The continuous trajectory itself is generated by the diffusion controller in Section 4.2, conditioned on the belief context c_t rather than on u_t.

4.2 Low-Level Controller: Belief-Conditioned Trajectory Generation

We use a diffusion-based controller $\mathcal{D}$ to map the latest belief into a short-horizon trajectory. Rather than conditioning on the single top-ranked hypothesis, we feed the controller a weighted aggregation of all hypotheses, so that uncertainty in the belief naturally translates into smoother conditioning when no single hypothesis dominates.

Belief Embedding Aggregation. To condition the controller on $\mathcal{B}_t = \{(h_t^k, w_t^k)\}_{k=1}^K$, we encode each hypothesis with the VLM's text encoder and take the mean-pooled last-layer hidden state, denoted $E_{\mathcal{M}}(h_t^k)$. The belief context is the weight-aggregated embedding:

$$c_t = \sum_{k=1}^K w_t^k E_{\mathcal{M}}(h_t^k). \tag{12}$$

This representation combines information from all hypotheses according to their belief weights.

Diffusion-Based Control Policy. We model motion generation as conditional denoising [Ho et al., 2020, Chi et al., 2025]. The controller $\mathcal{D}$ outputs a short-horizon waypoint sequence $a_t = [p_{t,1}, \ldots, p_{t,H}]$ with $H = 8$, where each waypoint $p_{t,i} \in \mathbb{R}^2$ is the (x, y) offset in the ego frame. At inference time, we initialize the trajectory from Gaussian noise $a_t^T \sim \mathcal{N}(0, I)$ and iteratively apply the reverse denoising process.

Given a noisy motion a_t^τ at diffusion step τ, the noise predictor ϵ_θ is conditioned on the belief context c_t and visual evidence $\Phi(\mathcal{H}_t)$:

$$\hat{\epsilon} = \epsilon_\theta(a_t^\tau, \tau, c_t, \Phi(\mathcal{H}_t)). \tag{13}$$

The predicted noise $\hat{\epsilon}$ is used to update a_t^τ toward a cleaner sample $a_t^{\tau-1}$. After the reverse steps, we obtain the denoised trajectory $a_t = a_t^0$, which is executed in a receding-horizon manner.

DriveMind is trained in two stages. In the first stage, the high-level reasoner is trained with maneuver supervision and score prediction supervision:

$$\mathcal{L}_{\text{high}} = \mathcal{L}_{\text{man}} + \lambda_{\text{score}} \mathcal{L}_{\text{score}}. \tag{14}$$

Maneuver Prediction Loss. Given the belief state $\mathcal{B}_t$, we marginalize over hypotheses to obtain the maneuver distribution and apply negative log-likelihood against the ground-truth maneuver u_t^*:

$$\mathcal{L}_{\text{man}} = -\log \left(\sum_{k=1}^{K} w_t^k \, p_{\mathcal{M}}(u_t^* \mid h_t^k, \mathcal{H}_t) \right). \tag{15}$$

Score Prediction Loss. For each natural-language hypothesis h_t^k, we supervise the VLM with an annotated consistency score $r_t^{k*} \in \{1, \ldots, 10\}$. The scores are assigned by human annotators and Claude. The model is trained to generate the score as a textual answer using cross-entropy:

$$\mathcal{L}_{\text{score}} = -\frac{1}{K} \sum_{k=1}^{K} \log p_{\mathcal{M}} \left(r_t^{k*} \mid h_t^k, \boldsymbol{x}_t, \mathcal{H}_{t-1} \right). \tag{16}$$

At inference time, the generated scores are rescaled and converted into belief weights using Eqs. (8)–(9).

4.3 Closed-loop Inference and Uncertainty Tracking

Algorithm 1 summarizes the closed-loop inference procedure of DriveMind. We use the entropy of the hypothesis weights to summarize how belief is distributed across hypotheses:

$$U_t = -\sum_{k=1}^{K} w_t^k \log(w_t^k). \tag{17}$$

Figure 3 shows how these belief statistics evolve in representative scenarios; additional qualitative examples are in Appendix E.

5 Experiments

5.1 Benchmarks

We evaluate DriveMind on two autonomous driving benchmarks: Bench2Drive and NAVSIM (Appendix C). Bench2Drive [Jia et al., 2024] is a CARLA-based benchmark with 220 short routes, each approximately 150 m long, spanning Town01–Town15 under varying weather and traffic. We follow the official Bench2Drive evaluation protocol and report standard closed-loop metrics, including Driving Score (DS), Success Rate (SR), efficiency, and comfort. We also report the Bench2Drive multi-ability breakdown, which evaluates performance on merging, overtaking, emergency braking, give-way, and traffic-sign handling, along with the mean score across abilities.

DriveMind w/o Belief. This baseline uses the same VLM and diffusion controller as DriveMind but conditions trajectory generation on a single scene interpretation generated from the current observation. It does not propagate hypotheses across reasoning steps or perform belief reweighting and resampling.

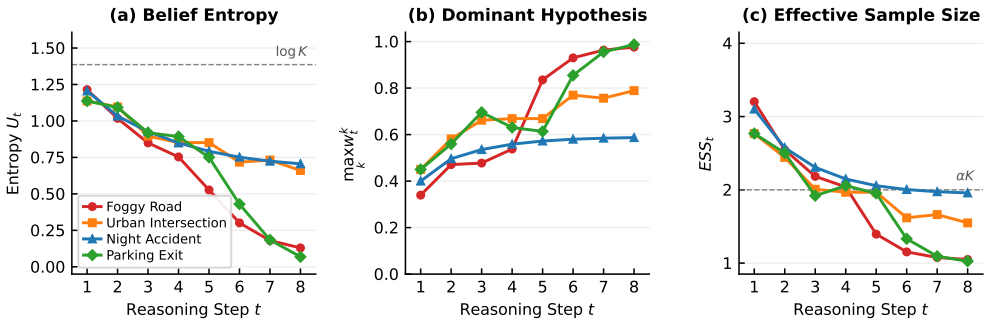

Figure 3: **Belief dynamics across four scenarios.** Foggy Road (red), Urban Intersection (orange), Night Accident (blue), Parking Exit (green). **(a)** Entropy U_t starts near $\log K$ and decays as the scene becomes less ambiguous. **(b)** The dominant weight $\max_k w_t^k$ climbs toward certainty in most scenarios; Night Accident plateaus at $\sim$0.6, reflecting persistent ambiguity. **(c)** ESS_t drops below the resampling threshold αK once belief concentrates, except in Night Accident where diversity is maintained throughout. Per-step visualizations appear in Appendix E.

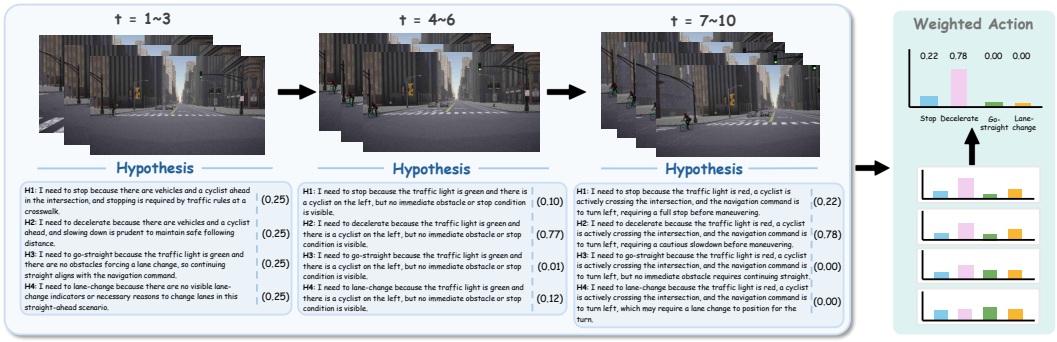

Figure 4: **Belief evolution and weighted action selection.** The ego vehicle approaches an intersection with a left-turn command and a cyclist near the crosswalk. DriveMind starts with four equally weighted hypotheses and concentrates belief on cautious behavior as the cyclist's relevance becomes clear: deceleration receives the highest final weight (0.78), with stopping retained as a fallback (0.22).

5.2 Results

Driving Performance on Bench2Drive. Table 1 reports the closed-loop driving metrics and multi-ability evaluation results on Bench2Drive. DriveMind achieves **81.37** DS and **54.09%** SR. Compared with DriveMind w/o belief, belief tracking improves DS from **68.50** to **81.37** (+12.87) and SR from **41.36%** to **54.09%** (+12.73 percentage points). Relative to DriveMind w/o belief, the largest multi-ability gains are in give-way (30.00% to 70.00%, +40.00 percentage points) and overtaking (57.78% to 91.11%, +33.33 percentage points), both of which involve interactions with other road users. DriveMind also achieves the highest multi-ability mean score among the methods in Table 1, at **58.17%**. At the same time, DriveMind does not dominate every individual ability, suggesting that belief tracking is most beneficial in scenarios requiring intent disambiguation rather than rule-following alone.

Figure 4 shows a representative intersection case where the ego vehicle must decide how to react to a cyclist while following a left-turn command. DriveMind initially keeps multiple hypotheses with equal weights and gradually shifts belief weight toward cautious deceleration as later observations confirm that the cyclist is crossing the intersection.

Scenario Analysis. Table 2 compares DriveMind with the baseline across nine Bench2Drive scenario categories. The largest DS gains occur in *Opposite Direction* (+20.5) and *Unsignalized Intersection* (+20.0), where driving decisions depend on uncertain interactions with other agents. *Pedestrian Crossing* (+16.9), *Obstacle/Hazard* (+14.7), and *Merging* (+13.6) also improve substan-

Table 1: Closed-loop evaluation and multi-ability test results on Bench2Drive. Best results are shown in bold.

(a) Closed-loop metrics ↑

	Method	Efficiency	Comfort	SR (%)	DS
w/ dist.	TCP [Wu et al., 2022]	54.26	47.80	15.00	40.70
	TCP-ctrl [Wu et al., 2022]	55.97	**51.51**	7.27	30.47
	TCP-traj [Wu et al., 2022]	76.54	18.08	30.00	59.90
	ThinkTwice [Jia et al., 2023b]	69.33	16.22	31.23	62.44
	DriveAdapter [Jia et al., 2023a]	70.22	16.01	33.08	64.22
w/o dist.	AD-MLP [Zhai et al., 2023]	48.45	22.63	0.00	18.05
	UniAD-T. [Hu et al., 2023]	123.92	47.04	13.18	40.73
	UniAD-B. [Hu et al., 2023]	129.21	43.58	16.36	45.81
	VAD [Jiang et al., 2023]	**157.94**	46.01	15.00	42.35
	DriveTransformer-L. [Jia et al., 2025]	100.64	20.78	35.01	63.46
	ReCogDrive [Li et al., 2025]	138.18	17.45	45.45	71.36
	DriveMind w/o belief	134.75 ± 1.86	18.01 ± 0.08	41.36 ± 1.85	68.50 ± 1.39
	DriveMind (Ours)	114.83 ± 0.33	18.98 ± 0.12	$\mathbf{54.09 \pm 1.10}$	$\mathbf{81.37 \pm 1.41}$

(b) Multi-ability test (%) ↑

	Method	Merging	Overtaking	Emerg. Brake	Give-way	Traffic Sign	Mean
w/ dist.	TCP	16.18	20.00	20.00	10.00	6.99	14.63
	TCP-ctrl	10.29	4.44	10.00	10.00	6.45	8.23
	TCP-traj	8.89	24.29	51.67	40.00	46.28	34.22
	ThinkTwice	27.38	18.42	35.82	50.00	54.23	37.17
	DriveAdapter	28.82	26.38	48.76	50.00	56.43	42.08
w/o dist.	AD-MLP	0.00	0.00	0.00	0.00	4.35	0.87
	UniAD-T.	8.89	9.33	20.00	20.00	15.43	14.73
	UniAD-B.	14.10	17.78	21.67	10.00	14.21	15.55
	VAD	8.11	24.44	18.64	20.00	19.15	18.07
	DriveTransformer-L.	17.57	35.00	48.36	40.00	52.10	38.60
	ReCogDrive	29.73	20.00	**69.09**	20.00	**71.34**	42.03
	DriveMind w/o belief	31.25 ± 0.70	57.78 ± 2.77	31.67 ± 1.33	30.00 ± 0.00	34.21 ± 1.47	36.98 ± 1.21
	DriveMind (Ours)	$\mathbf{41.25 \pm 1.10}$	$\mathbf{91.11 \pm 1.70}$	51.67 ± 1.20	$\mathbf{70.00 \pm 0.00}$	36.84 ± 2.01	$\mathbf{58.17 \pm 1.20}$

Table 2: Results across nine Bench2Drive scenario categories. ΔDS is the gain over DriveMind w/o belief.

Scenario Category	DS ↑		SR (%) ↑		ΔDS
	DriveMind w/o belief	DriveMind	DriveMind w/o belief	DriveMind	
Unsignalized Intersection	54.4	74.4	16.7	30.0	**+20.0**
Opposite Direction	42.7	63.2	13.3	13.3	**+20.5**
Pedestrian Crossing	53.3	70.2	20.0	30.0	+16.9
Obstacle / Hazard	84.0	98.7	73.3	96.7	+14.7
Merging	72.0	85.6	50.0	70.0	+13.6
Cut-in / Lane Change	81.6	94.0	60.0	85.0	+12.4
Actor Flow / Highway	73.6	80.8	40.0	55.0	+7.2
Signalized Intersection	58.8	64.6	12.0	4.0	+5.8
Emergency / Accident	88.0	91.7	76.0	84.0	+3.7
Overall	68.5	81.4	41.4	54.1	**+12.9**

tially, consistent with benefits in interpreting interactions and potential hazards. Gains are smaller in *Emergency/Accident*. SR remains unchanged in *Opposite Direction* (13.3%) and decreases from 12.0% to 4.0% in *Signalized Intersection*.

Long-Tail Scenario Analysis. Table 3 examines six challenging scenarios involving pedestrian crossings, merging, and roadside hazards. DriveMind improves DS and SR in all six. The largest gain occurs in *Parking–Crossing Pedestrian*, where DS rises from 53.2 to 100.0 and SR from 20% to 100%. *Merger into Slow Traffic (V2)* and *Hazard at Side Lane (Two Ways)* gain 35.2 and 33.4 DS, respectively. Maintaining competing interpretations may help resolve these ambiguous interactions.

Table 3: Results on six selected challenging Bench2Drive scenarios. Baseline denotes DriveMind w/o belief; SR is reported in percent.

Scenario	DS ↑		SR (%) ↑		ΔDS
	Baseline	DriveMind	Baseline	DriveMind	
Parking–Crossing Pedestrian	53.2	100.0	20	100	**+46.8**
Merger into Slow Traffic (V2)	44.0	79.2	0	60	**+35.2**
Hazard at Side Lane (Two Ways)	66.6	100.0	40	100	**+33.4**
Non-Signalized Junction Right Turn	49.7	81.6	20	40	+31.9
T-Junction	59.2	86.0	0	60	+26.8
Interurban Advanced Actor Flow	68.0	92.0	20	80	+24.0

Table 4: Ablation of multi-hypothesis belief tracking on Bench2Drive.

Method	DS ↑	SR (%) ↑
DriveMind w/o belief	68.50	41.36
DriveMind (K=2)	70.00	48.00
DriveMind (K=4)	**81.37**	**54.09**

Table 5: Effect of weight source on maneuver prediction accuracy.

Mode	Maneuver Acc. ↑	Entropy
Random	0.644	1.090
Uniform ($1/K$)	0.779	1.386
VLM score (ours)	**0.930**	**0.711**

5.3 Ablation Studies

We conduct ablation studies to evaluate four key design choices in DriveMind: multi-hypothesis belief tracking, the hypothesis count, scoring mode and inference latency.

Effect of Multi-Hypothesis Belief Tracking. Table 4 compares the single-interpretation baseline with multi-hypothesis belief tracking. With K=2, DS increases from 68.50 to 70.00 and SR from 41.36% to 48.00%; with K=4, they reach 81.37 and 54.09%, respectively. The K=2 setting initializes hypotheses for going straight and stopping, while K=4 additionally includes deceleration and lane change. Thus, this comparison varies both hypothesis count and initial maneuver coverage.

Effect of Hypothesis Count. We further analyze how the number of maintained hypotheses K affects behavioral diversity during belief tracking. Figure 5 reports three statistics as K varies. Let u_t^k be the maneuver label of hypothesis h_t^k, and define its aggregated action weight as $\pi_t(u) = \sum_{k=1}^{K} w_t^k \mathbf{1}[u_t^k = u]$. We measure (i) **action disagreement** as $1 - \sum_{u \in \mathcal{U}} \pi_t(u)^2$; (ii) **active unique predictions** as the number of maneuver categories with $\pi_t(u) \geq 0.05$; and (iii) **effective hypotheses** using ESS_t. As K increases, action disagreement and the number of unique predictions grow rapidly until $K \approx 4$, after which they plateau. Similarly, the effective number of hypotheses grows sublinearly with K, indicating that the weights are not uniformly distributed across hypotheses. The plateau in unique maneuver predictions should be interpreted in light of the fixed four-category vocabulary: K=4 already covers all maneuver categories at initialization. This saturation is therefore consistent with the chosen maneuver partition, rather than evidence of unrestricted semantic coverage. We use K=4 in all main experiments.

Effect of Scoring Mode. We evaluate maneuver prediction accuracy using the most probable maneuver under the belief-weighted marginal of Eq. (11):

$$\hat{u}_t = \arg\max_{u \in \mathcal{U}} \sum_{k=1}^{K} w_t^k p_{\mathcal{M}}(u \mid h_t^k, \mathcal{H}_t), \qquad \text{Acc.} = \frac{1}{|\mathcal{T}_{\text{eval}}|} \sum_{t \in \mathcal{T}_{\text{eval}}} \mathbf{1}[\hat{u}_t = u_t^*], \tag{18}$$

where $\mathcal{T}_{\text{eval}}$ is the set of evaluated frames and u_t^* is the reference maneuver derived from the expert autopilot's recorded state and controls, using the four-way vocabulary in Appendix D.

We compare VLM-based weighting with uniform weights ($w_t^k = 1/K$) and random weights, keeping the hypothesis set and reference labels fixed. As shown in Table 5, VLM-based weighting achieves 93.0% maneuver prediction accuracy, compared with 77.9% for uniform weighting and 64.4% for random weighting. These results support the usefulness of consistency-based reweighting. The lower

Table 6: Runtime analysis of DriveMind on a single NVIDIA RTX A6000 GPU. Left: isolated component latency. Right: system-level latency under shared-GPU execution. P95 denotes the 95th-percentile latency. s denotes the number of fast control steps between two slow reasoner updates.

Component-level latency			*System-level latency*	
Component	Mean (ms)	P95 (ms)	Setting	Mean (ms)
High Level Hypothesis Evolving (K=2)	74.0	77.9	Synchronous baseline	177.0
High Level Hypothesis Evolving (K=4)	120.0	125.4	Async refresh (s=2)	94.7
High Level Hypothesis Evolving (K=6)	165.5	167.6	Async refresh (s=4)	65.5
Low-level control	45.0	45.0	Async refresh (s=6)	57.1

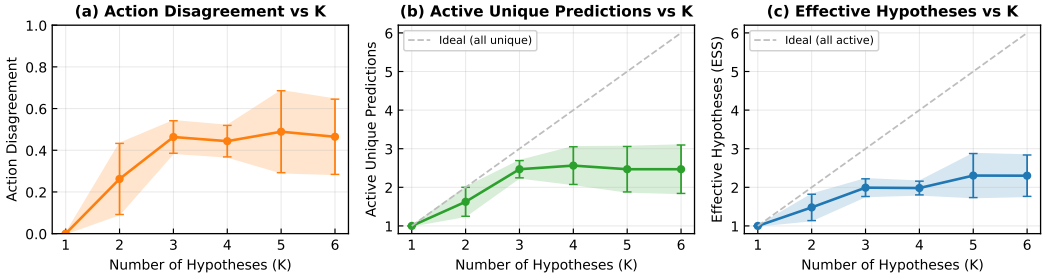

Figure 5: **Effect of hypothesis count K on behavioral diversity.** (a) Weighted action disagreement. (b) Maneuver categories with total weight at least 0.05. (c) ESS after reweighting; dashed lines mark K. Curves show means across four scenarios, each averaged over eight reasoning steps; shaded bands and error bars show one standard deviation across scenario means.

entropy (0.711 vs. 1.386 for uniform weighting) indicates more concentrated weights, rather than calibrated confidence.

Inference Latency. Table 6 reports both isolated component latency and system-level latency on a single NVIDIA RTX A6000 GPU. The high-level belief update remains within the slow-reasoner budget even as the hypothesis count increases: it takes 74.0 ms for K=2, 120.0 ms for the default K=4, and 165.5 ms for K=6. The low-level controller takes 45.0 ms, below the 100 ms budget of a 10 Hz control loop. We also profile shared-GPU execution. A synchronous baseline takes 177.0 ms per step, while asynchronous belief refresh reduces the average control-step latency to 94.7 ms at s=2, 65.5 ms at s=4, and 57.1 ms at s=6. These results show that the reasoning module can be updated at low frequency without dominating runtime, and that asynchronous refresh substantially reduces control-side latency compared with blocking reasoning updates.

6 Conclusion and Limitations

We presented *DriveMind*, a closed-loop MLLM-based driving framework that maintains a time-evolving belief state over multiple natural-language hypotheses. By propagating and reweighting hypotheses as new observations arrive, DriveMind reduces premature commitment under ambiguity. Its asynchronous fast–slow architecture decouples slow VLM-based belief tracking from fast diffusion-based trajectory generation. Experiments on Bench2Drive confirm improved performance and robustness in ambiguous scenarios. Ablations support the effectiveness of multi-hypothesis belief tracking, while runtime analysis demonstrates the efficiency of asynchronous inference.

DriveMind uses a fixed number of hypotheses and a fixed update frequency. We plan to adjust both based on the traffic scene and include hypotheses about other road users' intentions. We also tried adding DriveMind's belief tracking to SimLingo and other driving models, but the results were unstable. This may be related to their backbones or earlier fine-tuning. Future work will test different backbones and fine-tuning settings to make the improvements more consistent.

Acknowledgments

This work was supported in part by the Pacific Northwest Transportation Consortium (PacTrans), funded by the U.S. Department of Transportation (USDOT). We thank Dr. Ryan Feng Lin for his insightful discussions and valuable feedback.

The authors declare no competing interests.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL `https://arxiv.org/abs/2511.21631`.

Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. Driving with LLMs: Fusing object-level vector modality for explainable autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14093–14100. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10611018.

Yuntao Chen, Yuqi Wang, and Zhaoxiang Zhang. DrivingGPT: Unifying driving world modeling and planning with multi-modal autoregressive transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26890–26900, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. TransFuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12878–12895, 2023. doi: 10.1109/tpami.2022.3200245.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, Tianren Gao, Erlong Li, Kun Tang, Zhipeng Cao, Tong Zhou, Ao Liu, Xinrui Yan, Shuqi Mei, Jianguo Cao, Ziran Wang, and Chao Zheng. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops*, pages 958–979, 2024.

Alexander G. Cunningham, Enric Galceran, Ryan M. Eustice, and Edwin Olson. MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1670–1677. IEEE, 2015. doi: 10.1109/icra.2015.7139412.

Longchao Da, Xiaoou Liu, Jiaxin Dai, Lu Cheng, Yaqing Wang, and Hua Wei. Understanding the uncertainty of LLM explanations: A perspective based on reasoning topology. In *Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=p4wZfBFgyI`.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems*, volume 37, pages 28706–28719, 2024. doi: 10.52202/079017-0902.

Wenchao Ding, Lu Zhang, Jing Chen, and Shaojie Shen. EPSILON: An efficient planning system for automated vehicles in highly interactive environments. *IEEE Transactions on Robotics*, 38(2):1118–1138, 2022. doi: 10.1109/tro.2021.3104254.

Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001. doi: 10.1007/978-1-4757-3437-9.

Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. Drive like a human: Rethinking autonomous driving with large language models. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pages 910–919. IEEE, 2024. doi: 10.1109/wacvw60836.2024.00102.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=xYlJRpzZtsY`.

Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993. doi: 10.1049/ip-f-2.1993.0015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. DriveAdapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7953–7963, 2023a.

Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21983–21994, 2023b. doi: 10.1109/cvpr52729.2023.02105.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2Drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *Advances in Neural Information Processing Systems*, volume 37, pages 819–844, 2024.

Xiaosong Jia, Junqi You, Zhiyuan Zhang, and Junchi Yan. DriveTransformer: Unified transformer for scalable end-to-end autonomous driving. In *International Conference on Learning Representations*, pages 67227–67243, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a7afc9957f1190223763b6ea93218f98-Abstract-Conference.html.

Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. VAD: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8350, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Jiang_VAD_Vectorized_Scene_Representation_for_Efficient_Autonomous_Driving_ICCV_2023_paper.html.

Bo Jiang, Shaoyu Chen, Hao Gao, Bencheng Liao, Qian Zhang, Wenyu Liu, and Xinggang Wang. VADv2: End-to-end vectorized autonomous driving via probabilistic planning. In *International Conference on Learning Representations*, pages 68886–68900, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/70925fe2336ae05917c817d5e4660ab5-Abstract-Conference.html.

Hyunwoo Kim, Melanie Sclar, Tan Zhi-Xuan, Lance Ying, Sydney Levine, Yang Liu, Joshua B Tenenbaum, and Yejin Choi. Hypothesis-driven theory-of-mind reasoning for large language models. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=yGQqTuSJPK.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. ReCogDrive: A reinforced cognitive framework for end-to-end autonomous driving. *arXiv preprint arXiv:2506.08052*, 2025.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=DWkJCSxKU5.

Jiageng Mao, Yuxi Qian, Junjie Ye, Hang Zhao, and Yue Wang. GPT-Driver: Learning to drive with GPT. *arXiv preprint arXiv:2310.01415*, 2023.

Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. SimLingo: Vision-only closed-loop autonomous driving with language-action alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11993–12003, 2025. doi: 10.1109/cvpr52734.2025.01120.

Hao Sha, Yao Mu, Yuxuan Jiang, Li Chen, Chenfeng Xu, Ping Luo, Shengbo Eben Li, Masayoshi Tomizuka, Wei Zhan, and Mingyu Ding. LanguageMPC: Large language models as decision makers for autonomous driving, 2025. URL https://arxiv.org/abs/2310.03026v3. Version 3.

Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L. Waslander, Yu Liu, and Hongsheng Li. LMDrive: Closed-loop end-to-end driving with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15120–15130, 2024.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. DriveLM: Driving with graph visual question answering. In *European Conference on Computer Vision*, pages 256–274. Springer, 2024. doi: 10.1007/978-3-031-72943-0_15.

Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. DiLu: A knowledge-driven approach to autonomous driving with large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OqTMUPuLuC.

Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *Advances in Neural Information Processing Systems*, volume 35, pages 6119–6132, 2022. doi: 10.52202/068431-0443.

Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K Wong, Zhenguo Li, and Hengshuang Zhao. DriveGPT4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 9(10):8186–8193, 2024. doi: 10.1109/lra.2024.3440097.

Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek F Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. Benchmarking LLMs via uncertainty quantification. In *Advances in Neural Information Processing Systems*, volume 37, pages 15356–15385, 2024. doi: 10.52202/079017-0491.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Zhiyuan Zeng, Xiaonan Li, Junqi Dai, Qinyuan Cheng, Xuanjing Huang, and Xipeng Qiu. Reasoning in flux: Enhancing large language models reasoning through uncertainty-aware adaptive guidance. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2401–2416, 2024. doi: 10.18653/v1/2024.acl-long.131. URL https://aclanthology.org/2024.acl-long.131/.

Jiang-Tian Zhai, Ze Feng, Jinhao Du, Yongqiang Mao, Jiang-Jiang Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuScenes. *arXiv preprint arXiv:2305.10430*, 2023.

Boxuan Zhang and Ruqi Zhang. CoT-UQ: Improving response-wise uncertainty quantification in LLMs with chain-of-thought. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26114–26133. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1339. URL https://aclanthology.org/2025.findings-acl.1339/.

Caiqi Zhang, Fangyu Liu, Marco Basaldella, and Nigel Collier. LUQ: Long-text uncertainty quantification for LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5244–5262. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.emnlp-main.299. URL https://aclanthology.org/2024.emnlp-main.299/.

Jimuyang Zhang, Zanming Huang, Arijit Ray, and Eshed Ohn-Bar. Feedback-guided autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15000–15011, 2024b.

Lu Zhang, Wenchao Ding, Jing Chen, and Shaojie Shen. Efficient uncertainty-aware decision-making for automated driving using guided branching. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3291–3297. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9197302.

A Implementation Details

A.1 Model Architecture

DriveMind consists of a high-level VLM reasoner and a low-level diffusion controller. For the high-level reasoner, we use Qwen3VL-2B-Instruct [Bai et al., 2025] as the backbone.

The VLM generates natural-language hypotheses, predicts maneuver distributions, and produces belief scores for hypothesis reweighting.

For the low-level controller, we use a DDPM-based diffusion policy [Ho et al., 2020] with a Transformer-based backbone. The controller is conditioned on the belief context c_t and visual feature history $\Phi(\mathcal{H}_t)$ to generate short-horizon waypoint trajectories.

A.2 Training the Low-Level Diffusion Controller

We train the diffusion controller $\mathcal{D}$ to generate a continuous future waypoint trajectory a_t conditioned on the visual feature history $\Phi(\mathcal{H}_t)$ and the belief context c_t.

Decoupled Training Strategy. After training the high-level reasoner, we freeze $\mathcal{M}$ and pre-compute belief contexts for controller learning.

Phase I: Belief Caching (Offline). Using the frozen $\mathcal{M}$, we run belief tracking over the training sequences and compute c_t at each timestep. We cache $(c_t, \Phi(\mathcal{H}_t), a_t^0)$, where a_t^0 denotes the expert waypoint trajectory.

Phase II: Controller Training. We train $\mathcal{D}$ by minimizing the conditional diffusion denoising objective:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\tau,\epsilon,a_t^0}\left[\|\epsilon - \epsilon_\theta(a_t^\tau, \tau, c_t, \Phi(\mathcal{H}_t))\|_2^2\right]. \tag{19}$$

where a_t^τ is obtained by adding Gaussian noise to the expert trajectory a_t^0 at diffusion step τ.

During inference, the two subsystems are coupled online: $\mathcal{M}$ updates the belief state $\mathcal{B}_t$ and predicts the maneuver distribution, while $\mathcal{D}$ consumes the latest belief context c_t to synthesize a trajectory in real time.

A.3 Training Details

DriveMind is trained in two stages. In the first stage, we fine-tune the VLM using a mixture of VQA and action-prediction samples with a 1:1 sampling ratio. The VLM is optimized with maneuver (STOP, DECELERATE, GO_STRAIGHT, LANE_CHANGE) prediction and score prediction losses. For score prediction, we apply cross-entropy to the annotated score answers for individual natural-language hypotheses.

In the second stage, we freeze the VLM and pre-compute belief contexts over the training sequences. The diffusion controller is then trained to denoise expert waypoint trajectories conditioned on the cached belief context and visual features. All experiments are conducted on 2 NVIDIA A6000 GPUs. The training hyperparameters are summarized in Table 7.

Table 7: Training hyperparameters for DriveMind.

Parameter	Value	Parameter	Value
VLM Learning Rate	2×10^{-5}	Diffusion Learning Rate	1×10^{-4}
VLM Batch Size	8	Diffusion Batch Size	256
Weight Decay (VLM)	0.0001	DDPM Steps	20
Max Sequence Length	2048	Weight Decay (Diffusion)	0.0005

A.4 Closed-loop Inference Algorithm

Algorithm 1 Closed-loop Belief Tracking and Control (Inference)

Require: Reasoning model $\mathcal{M}$, diffusion controller $\mathcal{D}$, observations $\boldsymbol{x}_{1:T}$, hypothesis count K, ESS threshold α, similarity threshold ρ, temperature β.

Initialize trace $\tau = \emptyset$, hypotheses $\{h_0^k\}_{k=1}^K$, weights $w_0^k = 1/K$, and belief state $\mathcal{B}_0 = \{(h_0^k, w_0^k)\}_{k=1}^K$.
for $t = 1, \ldots, T$ **do**
  Construct history $\mathcal{H}_{t-1} = (\boldsymbol{x}_{1:t-1}, a_{1:t-2})$.
  ▷ — *Predict: Hypothesis Propagation (Slow)* —
  for $k = 1, \ldots, K$ **do**
    Sample $h_t^k \sim p_{\mathcal{M}}(\cdot \mid h_{t-1}^k, a_{t-1}, \boldsymbol{x}_t, \mathcal{H}_{t-1})$. ▷ Eq. (6)
  end for
  ▷ — *Correct: Evidence-Consistency Reweighting (Slow)* —
  for $k = 1, \ldots, K$ **do**
    Compute score $s_t^k = f_{\mathcal{M}}(h_t^k, \boldsymbol{x}_t, \mathcal{H}_{t-1})$. ▷ Eq. (7)
    Compute unnormalized weight $\tilde{w}_t^k = w_{t-1}^k \exp(s_t^k/\beta)$. ▷ Eq. (8)
  end for
  Normalize weights $w_t^k = \tilde{w}_t^k / \sum_{j=1}^K \tilde{w}_t^j$. ▷ Eq. (9)
  ▷ — *Resample: Diversity Maintenance (Slow)* —
  Compute ESS_t via Eq. (10).
  if $ESS_t < \alpha K$ **then**
    Resample K hypotheses from $\{(h_t^k, w_t^k)\}$ according to $\{w_t^k\}$.
    Compute pairwise cosine similarity $\mathrm{sim}(j, k)$ using normalized VLM text embeddings.
    for each pair (j, k) with $\mathrm{sim}(j, k) > \rho$ **do**
      Keep the higher-weighted hypothesis and regenerate the other so that its similarity to the kept hypothesis is at most ρ.
    end for
    Reset all weights to $w_t^k = 1/K$ and update $\mathcal{B}_t = \{(h_t^k, w_t^k)\}_{k=1}^K$.
  else
    Set $\mathcal{B}_t = \{(h_t^k, w_t^k)\}_{k=1}^K$.
  end if
  ▷ — *Control: Belief-Conditioned Trajectory Generation (Fast)* —
  Construct current history $\mathcal{H}_t = (\boldsymbol{x}_{1:t}, a_{1:t-1})$.
  Compute belief context $c_t = \sum_{k=1}^K w_t^k E_{\mathcal{M}}(h_t^k)$. ▷ Eq. (12)
  Sample trajectory $a_t \sim p_{\mathcal{D}}(a \mid c_t, \Phi(\mathcal{H}_t))$ via reverse diffusion.
  Execute a_t in a receding-horizon manner.
  Compute uncertainty $U_t = -\sum_{k=1}^K w_t^k \log w_t^k$.
  Add $(\mathcal{B}_t, c_t, a_t, U_t, ESS_t)$ to trace τ.
end for
return τ.

A.5 Diversity Maintenance Visualization

Figure 6 illustrates the diversity-maintenance step used after evidence-consistency reweighting. When the belief distribution becomes highly concentrated, naive resampling may repeatedly select the same high-weight hypothesis, reducing the diversity of the belief state. DriveMind addresses this issue by regenerating near-duplicated hypotheses into semantically distinct alternatives while keeping them consistent with the current observation.

B Formal Definitions of Evaluation Metrics

To provide a rigorous evaluation of *DriveMind*, we employ two sets of metrics corresponding to the Bench2Drive and NAVSIM benchmarks. These metrics evaluate the agent's performance from the perspectives of safety, efficiency, comfort, and similarity to human driving behavior.

B.1 Bench2Drive (CARLA-based) Metrics

The closed-loop performance in the CARLA simulator is primarily characterized by the Driving Score (DS) and Success Rate (SR).

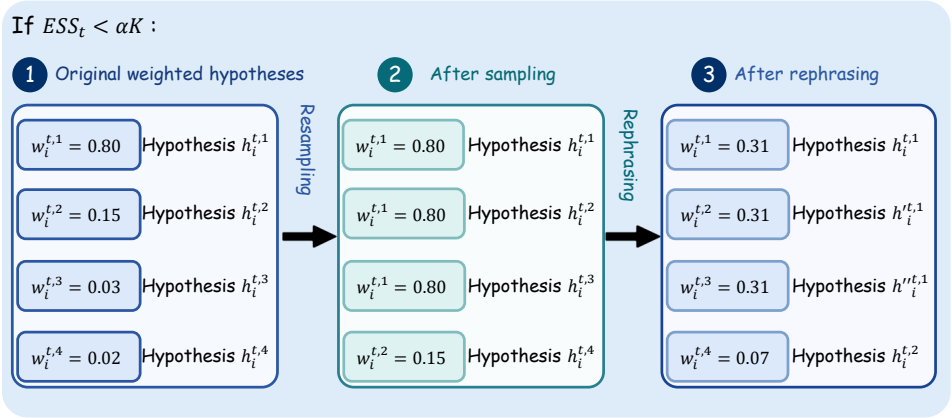

Figure 6: **Diversity maintenance through resampling and regeneration.** When the effective sample size ESS_t falls below the threshold αK, DriveMind resamples high-weight hypotheses and regenerates near-duplicated hypotheses to maintain semantic diversity while preserving consistency with the current observation.

Driving Score (DS): The DS is the primary metric in Bench2Drive, defined as the product of the Route Completion (RC) and the Infraction Penalty (IP):

$$DS = \frac{1}{N} \sum_{i=1}^{N} RC_i \times IP_i, \tag{20}$$

where RC_i is the percentage of the i-th route completed by the agent. IP_i is a cumulative penalty factor that starts at 1.0 and is reduced whenever an infraction occurs (e.g., collisions with pedestrians, vehicles, or static objects, and red-light violations).

Success Rate (SR): SR is the ratio of routes completed without any infractions:

$$SR = \frac{\text{Number of Successful Routes}}{\text{Total Number of Routes}}. \tag{21}$$

Infraction Penalty (IP): Each type of infraction has a specific weight ω_{inf}. If multiple infractions occur, their penalties are multiplied: $IP = \prod \omega_{inf}$. Common penalty weights include:

- **Collision (Pedestrian/Vehicle):** 0.60
- **Collision (Static Layout):** 0.65
- **Red Light Violation:** 0.70

B.2 NAVSIM (PDM-based) Metrics

For NAVSIM [Dauner et al., 2024], we follow the official Predictive Driver Model Score (PDMS), which summarizes multiple aspects of closed-loop planning quality, including collision avoidance, drivable-area compliance, time-to-collision safety, comfort, and route progress. The score is computed as

$$\text{PDMS} = \text{NC} \cdot \text{DAC} \cdot \left(\frac{5\,\text{EP} + 5\,\text{TTC} + 2\,\text{C}}{12} \right). \tag{22}$$

Where NC denotes the no-collision score, DAC denotes drivable-area compliance, EP denotes ego progress, TTC denotes time-to-collision safety, and C denotes comfort.

C NAVSIM Results

We further evaluate DriveMind on NavSim [Dauner et al., 2024] to test whether the planner generalizes beyond CARLA. Table 8 reports the NavSim sub-scores and the overall PDMS.

Table 8: Driving performance on Navsim

Method	NC↑	DAC↑	TTC↑	Comf.↑	EP↑	PDMS↑
Constant Velocity	68.0	57.8	50.0	100	19.4	20.6
Ego Status MLP	93.0	77.3	83.6	100	62.8	65.6
VADv2-V_{8192} [Jiang et al., 2026]	97.2	89.1	91.6	100	76.0	80.9
DrivingGPT [Chen et al., 2025]	**98.9**	90.7	**94.9**	95.6	**79.7**	82.4
UniAD [Hu et al., 2023]	97.8	91.9	92.9	100	78.8	83.4
TransFuser [Chitta et al., 2023]	97.7	**92.8**	92.8	100	79.2	**84.0**
DriveMind w/o belief	94.3	88.9	86.7	100.0	75.0	78.2
DriveMind w belief	96.5	90.6	91.1	100.0	76.1	81.0

DriveMind reaches 81.0 PDMS, slightly below TransFuser (84.0) and UniAD (83.4). The sub-scores show that DriveMind remains competitive on safety-related metrics: it achieves 96.5 NC, 91.1 TTC, and 100.0 comfort. The remaining gap mainly comes from DAC and EP, where DriveMind is slightly weaker than the strongest single-step planners.

In the NAVSIM ablation, adding belief tracking increases DriveMind's PDMS from 78.2 to 81.0 (Table 8). NC increases from 94.3 to 96.5, TTC from 86.7 to 91.1, DAC from 88.9 to 90.6, and EP from 75.0 to 76.1. Comfort remains at 100.0 in both variants.

This observation is consistent with the different roles of the two benchmarks. NavSim is a non-reactive, pseudo closed-loop benchmark: the planner is evaluated against logged sensor data, and the environment does not change in response to the planned trajectory. As a result, the main source of performance is the quality of the current predicted trajectory, while the benefit of propagating and reweighting multiple hypotheses across future interactive frames is naturally limited. In contrast, Bench2Drive evaluates the agent in an interactive simulator, where early misinterpretations can affect future observations and require recovery. Under this setting, belief tracking provides much larger gains, especially in ambiguous and interaction-heavy scenarios.

We therefore interpret the NavSim result as complementary to the Bench2Drive result. On NavSim, belief tracking provides a small positive gain and does not harm single-step planning performance. On Bench2Drive, where closed-loop interaction makes premature commitment more costly, the same belief mechanism yields substantially larger improvements.

D Prompt and Maneuver Space

Hypothesis propagation prompt. For each particle k, the VLM receives the previous hypothesis h_{t-1}^k, the previous executed action a_{t-1}, and the current camera observation $\boldsymbol{x}_t$, and generates one updated belief h_t^k. The propagation prompt is:

```
You are the high-level reasoning module of an autonomous driving system.
Your task is to maintain one plausible belief about the current traffic scene
and the immediate driving intention of the ego vehicle.

Previous belief about the scene: {h_{t-1}^k}
Previous action taken by the ego vehicle: {a_{t-1}}
Current speed: {speed} m/s
History trajectory: {trajectory history}

Based on the current camera image, update this belief.
Focus on:
- the most relevant nearby agents or obstacles,
- whether they are yielding, crossing, merging, braking, or turning,
- what the ego vehicle should do next at a high level.
```

```
Output exactly one hypothesis in 1-2 sentences, then output one maneuver label
from {STOP, DECELERATE, GO_STRAIGHT, LANE_CHANGE} on a new line.
```

Evidence-consistency scoring prompt. After propagation, each updated hypothesis h_t^k is evaluated by a second VLM pass that checks whether it is consistent with the current observation and a reference action. The scoring prompt is:

```
You are verifying whether a driving belief is consistent with the current scene.

Belief: {h_t^k}

Given the current camera image, rate how consistent this belief is with what the
ego vehicle should do now. Answer with exactly one integer from 1 to 10.
```

We rescale the integer output to $[0, 1]$ and use it as the scalar evidence-consistency score s_t^k.

Maneuver space. We use a four-way maneuver vocabulary:

$$\mathcal{U} = \{\texttt{STOP}, \texttt{DECELERATE}, \texttt{GO_STRAIGHT}, \texttt{LANE_CHANGE}\}. \tag{23}$$

STOP triggers an emergency brake, DECELERATE covers cautious slowing for hazards or yielding, GO_STRAIGHT denotes unimpeded forward progress, and LANE_CHANGE captures lateral maneuvers including overtaking and branch selection at intersections.

Reference Maneuver Labels. For the accuracy metric in Eq. (18), reference labels are constructed from expert autopilot recordings using ordered rules. LANE_CHANGE is assigned first using lateral navigation commands or steering. Among the remaining frames, STOP is identified using low speed and braking; DECELERATE is assigned using braking while moving or an active `should_brake` flag. Remaining frames are labeled GO_STRAIGHT.

E Additional Qualitative Results and Failure Cases

We provide additional qualitative case studies to illustrate how DriveMind updates its hypotheses and belief weights over time. Each case visualizes the observed scene sequence, the maintained natural-language hypotheses, their belief weights, and the resulting weighted action distribution. Figures 7, 8, 9, and 10 present representative cases covering intersection uncertainty, foggy-weather pedestrian uncertainty, nighttime accident uncertainty, and stable parking exit behavior.

Failure Cases. Despite strong overall performance, DriveMind can still fail in highly challenging scenarios, including extremely dense multi-agent interactions, sudden appearance of fast-moving objects, and prolonged occlusions. These failures are mainly caused by delayed high-level belief updates or insufficient visual evidence. For example, when an occluded agent appears abruptly between two slow reasoning updates, the controller may temporarily rely on an outdated belief context. Future work will explore faster belief updates, adaptive hypothesis allocation, and stronger uncertainty calibration for safety-critical deployment.

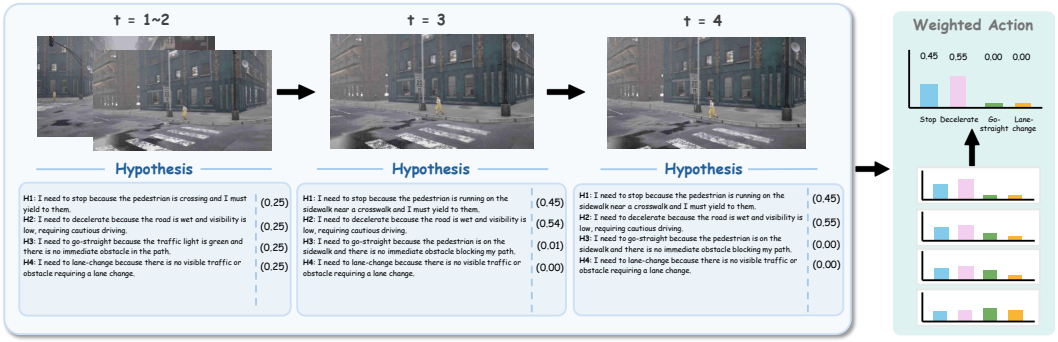

Figure 7: **Intersection uncertainty.** DriveMind initially maintains multiple plausible actions under ambiguity. As the pedestrian remains near the crosswalk and wet-road conditions persist, the belief concentrates on cautious behavior, assigning final weights of 0.55 to deceleration and 0.45 to stopping, while go-straight and lane-change actions are down-weighted to nearly zero.

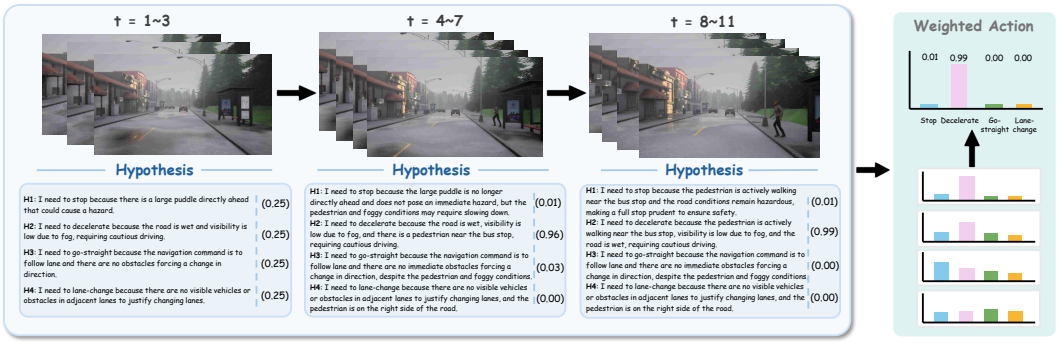

Figure 8: **Foggy weather with pedestrian uncertainty.** DriveMind handles a foggy and wet-road scenario where a pedestrian appears near the roadside. Early hypotheses include stopping for a potential hazard, decelerating due to low visibility, continuing straight, and lane changing. As the scene evolves, the belief concentrates on the deceleration hypothesis, reaching weight 0.99 at the final timestep. This illustrates how belief tracking supports cautious control under degraded visibility and pedestrian uncertainty.

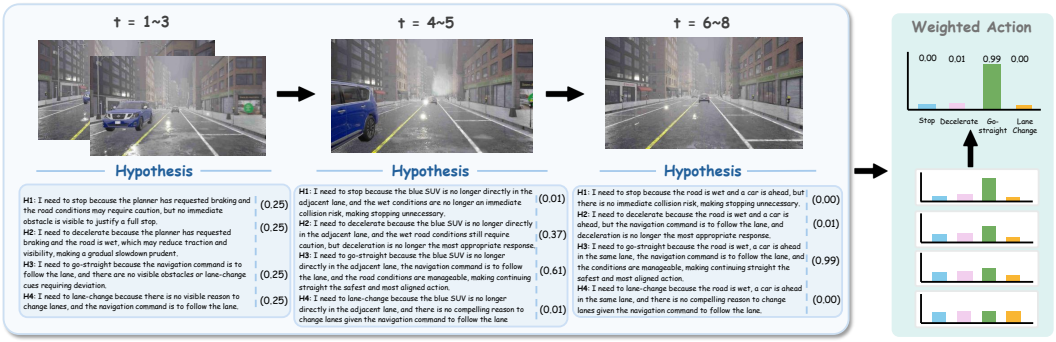

Figure 9: **Stable parking exit behavior in a low-ambiguity scenario.** DriveMind initially maintains multiple plausible hypotheses because of wet road conditions and a nearby vehicle. As the front vehicle moves farther ahead and the lane remains clear, DriveMind quickly concentrates its belief on the go-straight hypothesis. At the final timestep, the weighted action assigns 0.99 to go-straight, while stopping, deceleration, and lane-change are down-weighted to nearly zero. This case shows that DriveMind does not remain overly cautious when the visual evidence supports safe forward progress.

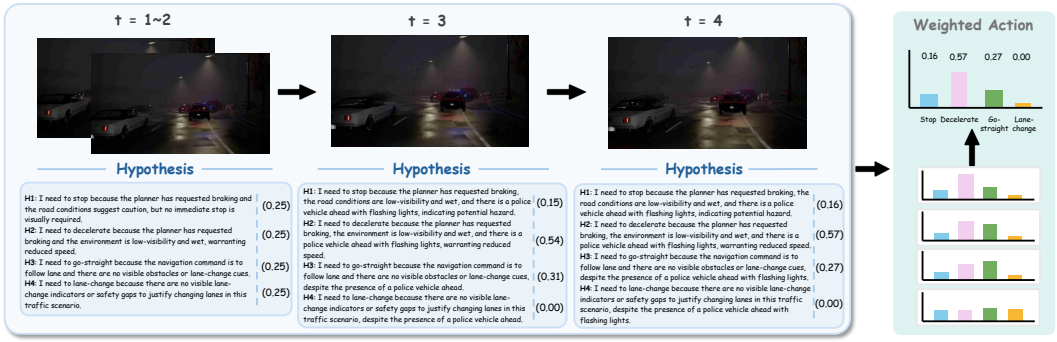

Figure 10: **Nighttime driving with accident uncertainty.** The ego vehicle drives in a dark, wet, and low-visibility scene with a police vehicle ahead. DriveMind initially maintains multiple plausible hypotheses with nearly uniform weights, including stopping, decelerating, going straight, and lane changing. As new observations confirm persistent poor visibility and the presence of the emergency vehicle, the belief shifts toward conservative behavior, with the final weighted action assigning the highest probability to deceleration while still retaining smaller probabilities for stopping and going straight.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution of DriveMind as a closed-loop driving framework that maintains and updates a multi-hypothesis belief state over time. The claims are supported by the proposed method, ablation studies, latency analysis, and experimental results on Bench2Drive and NAVSIM.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section. It discusses that the high-level reasoner runs at a lower frequency than the controller, which may reduce responsiveness to fast or abruptly changing events. It also notes that the learned belief weights may become miscalibrated under distribution shift, especially in rare scenarios not well covered by the training data. These limitations clarify the scope of the claims and identify conditions where the method may require further validation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results, theorems, lemmas, or proofs. The mathematical expressions are used to define the belief update procedure, uncertainty measures, diffusion objective, and training losses, rather than to state theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the proposed algorithm, model architecture, training objectives, two-stage training procedure, benchmark settings, evaluation metrics, and main hyperparameters. The appendix further provides the closed-loop inference algorithm and implementation details. These details provide a reasonable path for reproducing the main experimental results on the public Bench2Drive and NAVSIM benchmarks, independent of whether code is released with the submission.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper uses public benchmarks and cites the corresponding datasets. The code, configuration files, and reproduction scripts will be released in `https://github.com/shiyannacl/drivemind`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper describes the benchmark and evaluation protocol, training data setup, two-stage training procedure, model architecture, action discretization, diffusion controller training, and main hyperparameters. Additional implementation and metric details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports results using mean ± standard deviation in Table 1. The experiments were conducted over multiple runs and variability is quantified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments were run on a local GPU server with 2 NVIDIA A6000 GPUs. The high-level reasoner was fine-tuned on 2 GPUs, and the diffusion controller was trained on the same hardware. Inference latency was measured on a single NVIDIA A6000 GPU. We report the main training and evaluation settings used for the experiments in Table 7. Preliminary experiments used the same hardware configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is evaluated on autonomous driving benchmarks and does not involve human-subject experiments, private data collection, or deployment on public roads. The paper discusses limitations relevant to safety and robustness.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: DriveMind aims to improve the safety and interpretability of autonomous driving systems by explicitly tracking uncertainty over multiple hypotheses. This may help reduce premature decisions in ambiguous traffic scenes and provide more transparent evidence for why a vehicle chooses a particular action.

    At the same time, autonomous driving systems can cause serious harm if deployed without sufficient validation. In our method, delayed belief updates or miscalibrated hypothesis weights may lead to unsafe actions, especially under distribution shift or rare events not covered by training. Our experiments are conducted in simulation benchmarks, and the method is not intended for direct real-world deployment without extensive testing, safety monitoring, and compliance with relevant regulations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new high-risk pretrained model, image generator, scraped dataset, or other asset requiring special release safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original paper, the main benchmarks and models used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset, benchmark, or public software asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject experiments, crowdsourcing, or user studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper explicitly describes the use of multimodal large language models and vision-language models as core components of the proposed method. They are used for hypothesis generation, belief tracking, belief reweighting, and maneuver reasoning. In addition, LLM-based tools were used only for writing assistance, including grammar checking, typo correction, and wording refinement; all technical content, experimental design, and conclusions were developed and verified by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.