RAPDrive: Shared-Latent Hybrid Decoding for Reasoning and Planning in Autonomous Driving

Yuheng Liu[1] **Le Hui**[2,3†] **Ziyue Zhu**[1] **Yigong Zhang**[4] **Jin Xie**[4,†] **Jian Yang**[1,4,†]
[1]PCA Lab, VCIP, College of Computer Science, Nankai University, Tianjin, China
[2]Electronics and Information, Northwestern Polytechnical University, Xi'an, China
[3]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[4]School of Intelligence Science and Technology, Nanjing University, Suzhou, China
liuyuhengnk@mail.nankai.edu.cn, csjyang@nankai.edu.cn

Abstract

Driving vision-language-action (VLA) models seek to integrate reasoning with trajectory planning, but language and motion exhibit different generation structures: reasoning text is typically generated sequentially, whereas trajectory planning requires modeling dependencies across multiple future time steps. We present RAPDrive, a shared-latent hybrid-decoding framework that jointly processes visual context, ego state, motion history, reasoning tokens, and future planning slots within a single transformer. By combining autoregressive text generation with iterative masked motion-token refinement and continuous trajectory decoding, RAPDrive enables reasoning and planning to interact through shared latent representations while retaining modality-specific generation mechanisms. Training combines supervised reasoning-and-planning objectives, with GRPO-based post-training providing additional improvements. Experiments on NAVSIM and Bench2Drive show that RAPDrive outperforms existing base driving VLA baselines under supervised training. Ablation studies further validate the effectiveness of the proposed components.

1 Introduction

End-to-end autonomous driving has made steady progress through unified perception, prediction, and planning systems [Chen et al., 2024, Hu et al., 2023, Jiang et al., 2026, Chitta et al., 2023]. Benchmarks such as NAVSIM and Bench2Drive make the remaining planning difficulty visible through planner-centric evaluation under different simulation assumptions and metric suites [Dauner et al., 2024, Jia et al., 2024]. In parallel, recent driving vision-language-action (VLA) models extend this line by coupling visual scene understanding, natural-language reasoning, and trajectory generation within a single policy. Broader embodied VLA and multimodal reasoning work has likewise emphasized large-scale policy pretraining, open-source VLA models, and chain-of-thought prompting as useful ingredients for grounded action generation [Brohan et al., 2022, Kim et al., 2024, Wei et al., 2022, Zhao et al., 2025].

This direction raises a structural question for driving VLA: how should language reasoning and trajectory planning interact within a unified model? Reasoning text is naturally sequential, making autoregressive decoding a natural fit for language generation. Future motion, however, is a geometric object whose quality depends on consistency over the full planning horizon. Existing driving VLA systems reflect different answers to this question: AutoVLA discretizes physical actions and generates reasoning and action tokens autoregressively [Zhou et al., 2025], whereas ReCogDrive conditions a diffusion planner on cognitive VLM representations [Li et al., 2025d]. We introduce RAPDrive, a

†Corresponding authors.

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

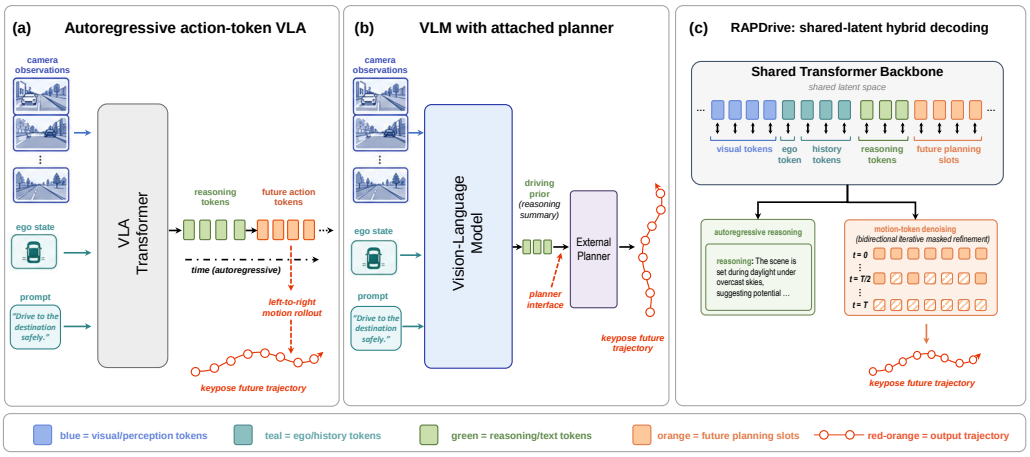

Figure 1: Comparison of reasoning-and-planning paradigms for driving VLA. (a) Autoregressive action-token VLA generates reasoning and action tokens sequentially. (b) VLM-based planning uses semantic representations to condition an external planner. (c) RAPDrive integrates reasoning tokens and future planning slots within a shared transformer, employing autoregressive text generation and iterative masked motion-token refinement.

shared-latent hybrid-decoding framework that places future planning slots directly inside the shared transformer computation while using a motion-specific generation rule for trajectory refinement.

Figure 1 contrasts these reasoning-and-planning paradigms.

RAPDrive forms a single transformer sequence from visual tokens, ego state, recent motion history, reasoning tokens, and future planning slots. A hybrid attention mask preserves causal decoding for reasoning text while enabling bidirectional interaction among future planning slots and the full multimodal context. Future motion is refined through iterative masked token denoising in a dedicated endpoint-contour motion-token space and then realized by a continuous local-delta decoder [Chang et al., 2022]. Language and motion retain distinct token spaces while interacting through shared hidden states. Training combines supervised reasoning-and-planning objectives, and GRPO is evaluated as an optional post-training extension [Shao et al., 2024b].

In the supervised NAVSIM setting, RAPDrive reaches 86.9 PDMS with camera-only inputs and improves over the listed base driving VLA baselines. GRPO-based post-training increases RAPDrive to 90.3 PDMS. On Bench2Drive, RAPDrive remains competitive with the collected driving VLA baselines, and controlled ablations support the main design choices. Our contributions are:

- We propose RAPDrive, a shared-latent hybrid-decoding framework for driving VLA that couples reasoning and trajectory planning within a single transformer sequence while using autoregressive text generation and iterative masked motion refinement.
- We introduce a motion-specific planning interface based on endpoint-contour motion tokens and continuous local-delta realization, enabling horizon-level trajectory refinement without collapsing motion generation into a purely autoregressive language suffix.
- We show that RAPDrive provides a stronger supervised/base driving VLA model than the listed prior non-world-model baselines, remains compatible with GRPO-based post-training, and is supported by controlled ablations on the main architectural choices.

2 Related Work

2.1 Traditional End-to-End Driving

Traditional end-to-end driving systems optimize perception, prediction, and planning directly for driving performance, typically without treating language reasoning as an explicit output. Representative examples include TransFuser, UniAD, VAD/VADv2, and other planners based on parallelized

decoding, distillation, latent or world-model-based planning, and continuous diffusion [Chitta et al., 2023, Hu et al., 2023, Jiang et al., 2023, 2026, Weng et al., 2024, Li et al., 2024, 2025a, Liao et al., 2025]. These methods remain strong planning baselines because their architectures and supervision are specialized for trajectory quality rather than joint language-action generation. They therefore provide important benchmark context for RAPDrive and help distinguish task-specialized end-to-end planning systems from driving VLA models.

2.2 Driving VLA and Generative Planning

Driving VLA methods connect visual scene understanding, language reasoning, and action generation. Early examples study language-conditioned closed-loop driving and driving-specific VLMs [Shao et al., 2024a, Tian et al., 2024, Hwang et al., 2024, Jiang et al., 2024]. This ecosystem is also supported by driving-language resources such as Talk2Car, NuScenes-QA, SUTD-TrafficQA, LingoQA, and DriveLM, together with specialist driving VLMs such as Dolphins, GPT-Driver, and OmniDrive [Deruyttere et al., 2019, Qian et al., 2024, Xu et al., 2021, Marcu et al., 2024, Sima et al., 2024, Ma et al., 2024, Mao et al., 2023, Wang et al., 2024b]. Generalist multimodal backbones such as Qwen2-VL are also increasingly used as reference points in this space [Wang et al., 2024a]. The closest base driving VLA prior to RAPDrive is AutoVLA, which discretizes trajectories into action tokens and generates reasoning and action autoregressively within one VLA stream [Zhou et al., 2025]. ReCogDrive is another close prior because it combines a cognitive VLM with a diffusion-based planner and reinforced post-training [Li et al., 2025d]. RAPDrive instead keeps planning slots inside the shared transformer computation and refines future motion with masked token denoising rather than a left-to-right action suffix or an attached planner. Related variants also explore adaptive reasoning, self-reflection, or vision-language-instructed action generation [Luo et al., 2025, Yuan et al., 2026, Fu et al., 2025].

A related line of work augments driving VLA with future prediction or world-model supervision. DriveVLA-W0, DynVLA, and Vega are representative examples, alongside broader driving world-model methods such as VISTA [Li et al., 2025b, Shang et al., 2026, Zuo et al., 2026, Gao et al., 2024]. Related generative formulations include GenAD and controllable traffic diffusion, alongside transformer diffusion backbones such as DiT [Zheng et al., 2024, Zhong et al., 2023, Peebles and Xie, 2023].

Beyond autoregressive action-token generation, generative approaches provide alternative mechanisms for trajectory prediction. DiffusionDrive performs trajectory refinement through truncated continuous diffusion [Liao et al., 2025]. In discrete generative modeling, MaskGIT introduces iterative masked-token prediction with confidence-based token selection [Chang et al., 2022]. Inspired by this refinement paradigm, RAPDrive operates on discrete endpoint-contour motion tokens and decodes the refined representations into continuous trajectories, while integrating future planning slots within a shared VLA transformer.

2.3 Reasoning Supervision and Policy Post-Training

Reasoning supervision has been explored in vision-language-action models to improve the connection between intermediate reasoning and action generation [Zhao et al., 2025, Zhou et al., 2025]. Beyond supervised learning, reinforcement learning provides an additional means of improving policy performance through task-level rewards. Group Relative Policy Optimization (GRPO) was introduced in DeepSeekMath [Shao et al., 2024b] and has subsequently been adopted in driving VLA methods such as AutoVLA and AutoDrive-R2 [Zhou et al., 2025, Yuan et al., 2026]. ReCogDrive and DynVLA further explore reinforcement-based refinement for driving policies [Li et al., 2025d, Shang et al., 2026].

3 Method

RAPDrive is a vision-language-action framework for end-to-end driving that couples scene understanding, natural-language reasoning, and trajectory planning inside a shared transformer computation. Its central design is hybrid decoding: reasoning text is generated autoregressively, while future motion is refined through iterative masked denoising over a dedicated endpoint-contour motion-token space

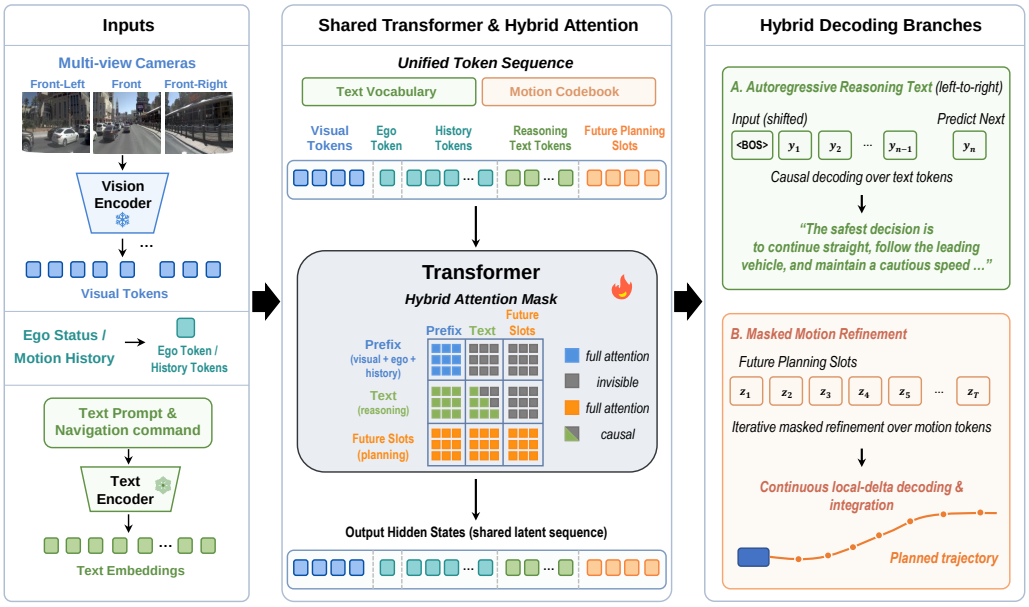

Figure 2: RAPDrive shared-latent hybrid-decoding architecture. Visual, ego-state, motion-history, and text embeddings are processed together with future planning slots under a hybrid attention mask. Reasoning text is generated autoregressively, while motion tokens are refined through iterative masked denoising in a dedicated endpoint-contour codebook space. Refined planning-slot states are decoded into continuous local motion deltas and integrated into the final trajectory.

and then realized as a continuous trajectory. Language and motion keep distinct token spaces, but their hidden states interact through the shared transformer sequence. Figure 2 summarizes the architecture.

3.1 Problem formulation

At each decision step, RAPDrive receives camera observations, ego-vehicle status, and recent ego-motion history, and predicts both reasoning text and a future ego trajectory. In the experimental setting used in this paper, the visual input consists of front, front-left, and front-right camera images over two historical frames. Let I denote the camera observations, $\boldsymbol{s}$ the ego-status vector, and H the recent ego-motion history. The ego-status vector contains rear-axle velocity, rear-axle acceleration, and a high-level navigation command:

$$\boldsymbol{s} = (v_x, v_y, a_x, a_y, \boldsymbol{\gamma}^\top)^\top, \qquad \boldsymbol{\gamma} \in \{0,1\}^4, \tag{1}$$

where v_x, v_y denote ego velocity, a_x, a_y denote ego acceleration, and $\boldsymbol{\gamma}$ is a one-hot command vector whose entries sum to one, giving $\boldsymbol{s} \in \mathbb{R}^8$. The history H contains recent ego poses represented in the current ego frame.

The text output is a semi-structured reasoning sequence

$$Y = (y_1, \ldots, y_L), \tag{2}$$

trained with standard causal language modeling. The motion output is a sparse future ego trajectory

$$P = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_T), \qquad \boldsymbol{p}_t = (x_t, y_t, \theta_t)^\top, \tag{3}$$

where each keypose is a planar ego-frame pose, $T = 8$ keyposes cover a $4\,\mathrm{s}$ horizon at $0.5\,\mathrm{s}$ intervals, and $\boldsymbol{p}_0 = (0,0,0)^\top$ is the current ego pose. RAPDrive models reasoning and planning as coupled outputs:

$$(Y, P) = f_\theta(I, \boldsymbol{s}, H). \tag{4}$$

During supervised training, future planning slots attend to teacher-forced reasoning tokens. At inference time, RAPDrive first generates reasoning text autoregressively and then conditions masked motion refinement on the generated reasoning context.

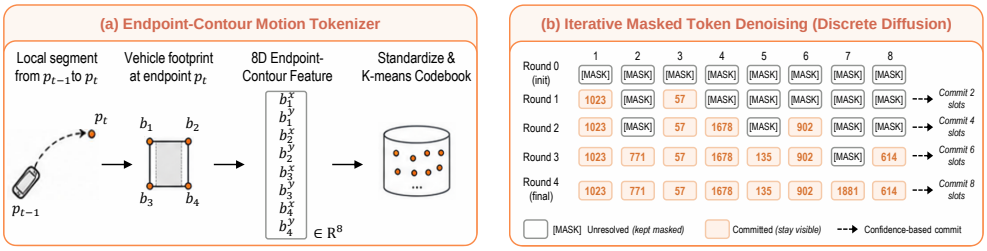

Figure 3: RAPDrive motion-token interface. (a) Endpoint-contour motion tokenizer: each local segment is represented by endpoint vehicle-footprint corners and assigned to a K-means motion codeword. (b) Iterative masked motion-token refinement: future planning slots are initialized as masked categorical tokens and denoised with confidence-based commitment.

3.2 Endpoint-contour motion token space

Figure 3 summarizes the endpoint-contour tokenizer and masked motion-token refinement interface. As shown in Figure 3(a), for each local segment, we describe the endpoint pose by transforming the vehicle footprint into the coordinate frame of the segment start. Let $\boldsymbol{q}_m \in \mathbb{R}^2$, $m = 1, \ldots, 4$, denote the four canonical footprint corners in the vehicle frame. Given local endpoint displacement $(\Delta x_t, \Delta y_t, \Delta\theta_t)$, the endpoint contour corners are

$$\boldsymbol{b}_{t,m} = \mathrm{Rot}(\Delta\theta_t)\boldsymbol{q}_m + \begin{bmatrix} \Delta x_t \\ \Delta y_t \end{bmatrix}, \qquad m = 1, \ldots, 4. \tag{5}$$

We vectorize these corners into an endpoint-contour feature

$$\boldsymbol{\phi}_t = \mathrm{vec}(\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,4}) \in \mathbb{R}^8, \tag{6}$$

standardize them with training-set mean $\boldsymbol{\mu}_\phi$ and standard deviation $\boldsymbol{\sigma}_\phi$,

$$\tilde{\boldsymbol{\phi}}_t = (\boldsymbol{\phi}_t - \boldsymbol{\mu}_\phi) \oslash (\boldsymbol{\sigma}_\phi + \epsilon\mathbf{1}), \tag{7}$$

where $\oslash$ denotes elementwise division and $\mathbf{1} \in \mathbb{R}^8$ broadcasts the scalar ϵ. We fit a K-means codebook

$$\mathcal{C} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K\}, \qquad K = 2048. \tag{8}$$

Each geometric codeword $\boldsymbol{e}_k \in \mathbb{R}^8$ defines a supervision target. Each ground-truth segment is assigned to its nearest codeword:

$$z_t^\star = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\| \tilde{\boldsymbol{\phi}}_t - \boldsymbol{e}_k \right\|_2^2. \tag{9}$$

Thus a future trajectory is represented as a discrete motion-token sequence

$$z^\star = (z_1^\star, \ldots, z_T^\star). \tag{10}$$

The codebook serves as a motion-specific planning interface and supervision target, while language and motion remain represented by separate token spaces. Additional reconstruction details are provided in Appendix A.1.

3.3 Shared transformer latent sequence and hybrid attention

RAPDrive embeds perception, ego state, motion history, reasoning tokens, and future planning slots into a single transformer sequence with hidden width D. Visual observations are encoded by the vision encoder and projected into this width. Ego-status and history inputs are projected by lightweight MLP modules. Reasoning tokens use the language-model embedding table. Future motion is represented by $T = 8$ planning slots whose content embeddings come from either a mask embedding or a committed motion-token embedding.

At refinement round n, future slot t is represented as

$$\boldsymbol{x}_{\text{plan},t}^{(n)} = \boldsymbol{e}_{\text{pos}}(t) + \begin{cases} \boldsymbol{e}_{\text{mask}}, & \text{if } z_t^{(n)} = [\text{MASK}], \\ \boldsymbol{e}_{\text{tok}}(z_t^{(n)}), & \text{otherwise.} \end{cases} \tag{11}$$

The embeddings are D-dimensional; the learned lookup $\boldsymbol{e}_{\text{tok}}$ maps token ids to hidden states, separately from the geometric codewords. The joint sequence is

$$\mathbf{X}^{(n)} = [\, \mathbf{X}_{\text{vis}} \mid \boldsymbol{x}_{\text{ego}} \mid \mathbf{X}_{\text{hist}} \mid \mathbf{X}_{\text{text}} \mid \mathbf{X}_{\text{plan}}^{(n)} \,], \tag{12}$$

where $\mathbf{X}_{\text{vis}}$ denotes visual embeddings, $\boldsymbol{x}_{\text{ego}}$ the ego embedding, $\mathbf{X}_{\text{hist}}$ history embeddings, $\mathbf{X}_{\text{text}} = \text{Embed}_{\text{text}}(Y)$ reasoning embeddings, and $\mathbf{X}_{\text{plan}}^{(n)}$ planning-slot embeddings. Given the hybrid attention mask $\mathbf{A}$ over this sequence, the shared transformer computes the hidden-state matrix

$$\mathbf{H}_\theta^{(n)} = \text{Transformer}_\theta(\mathbf{X}^{(n)}, \mathbf{A}). \tag{13}$$

RAPDrive uses a block attention mask that combines causal language generation with non-causal future-motion refinement, as visualized in Figure 2. Let $\mathbf{X}_{\text{ctx}} = [\mathbf{X}_{\text{vis}} \mid \boldsymbol{x}_{\text{ego}} \mid \mathbf{X}_{\text{hist}}]$ be the conditioning prefix. The prefix uses bidirectional attention, reasoning embeddings attend to the prefix and previous reasoning positions, and future planning slots attend to the prefix, the reasoning block, and all planning slots. Text-position hidden states are decoded by the language head, while future-slot hidden states are decoded by motion heads. The language embedding table and the endpoint-contour motion codebook remain separate; their interaction occurs through shared hidden states and attention inside the transformer.

3.4 Masked motion refinement

Future motion is generated by iterative masked-token denoising in the endpoint-contour motion-token space. Inference begins from an all-mask future sequence:

$$z^{(0)} = ([\text{MASK}], \ldots, [\text{MASK}]), \qquad \mathcal{U}^{(0)} = \{1, \ldots, T\}. \tag{14}$$

The default schedule uses $N = 4$ refinement rounds for $T = 8$ future slots. At refinement round $n = 0, \ldots, N-1$, the shared transformer predicts a categorical distribution for each unresolved slot $t \in \mathcal{U}^{(n)}$:

$$p_t^{(n)}(k) = \text{softmax}(\boldsymbol{\ell}_t^{(n)})_k, \qquad \boldsymbol{\ell}_t^{(n)} = \mathbf{W}_{\text{mot}} \boldsymbol{h}_{\text{plan},t}^{(n)}. \tag{15}$$

Here $\boldsymbol{h}_{\text{plan},t}^{(n)} \in \mathbb{R}^D$, $\mathbf{W}_{\text{mot}} \in \mathbb{R}^{K \times D}$, and $\boldsymbol{\ell}_t^{(n)} \in \mathbb{R}^K$. The provisional token and scalar confidence are

$$\hat{z}_t^{(n)} = \arg\max_k p_t^{(n)}(k), \qquad c_t^{(n)} = \max_k p_t^{(n)}(k). \tag{16}$$

Each round commits the two most confident unresolved slots:

$$C^{(n)} = \underset{t \in \mathcal{U}^{(n)}}{\text{Top2Indices}}\, c_t^{(n)}, \qquad |C^{(n)}| = 2, \tag{17}$$

Committed tokens remain visible in later rounds, while unresolved slots remain masked:

$$z_t^{(n+1)} = \begin{cases} \hat{z}_t^{(n)}, & t \in C^{(n)}, \\ z_t^{(n)}, & t \notin C^{(n)}, \end{cases} \qquad \mathcal{U}^{(n+1)} = \mathcal{U}^{(n)} \setminus C^{(n)}. \tag{18}$$

The reveal pattern is therefore $2/8$, $4/8$, $6/8$, and $8/8$, corresponding to the masked refinement process in Figure 3(b). Following MaskGIT-style masked-token refinement [Chang et al., 2022], we use "discrete diffusion" to denote this categorical masked-denoising process: the sequence starts from fully masked future slots and is refined by repeated prediction, confidence-based commitment, and conditioning on previously committed tokens. Additional training-time masked-state details are given in Appendix A.2.

3.5 Training objectives

RAPDrive uses a multi-term supervised objective that couples reasoning and planning supervision. The text branch is trained with causal language modeling on teacher-forced reasoning tokens, together with a text-state alignment loss and a small hybrid-path text cross-entropy term. The motion branch is trained by sampling partial masked-refinement states under the same four-round commit-two schedule used at inference and applying token denoising loss only on masked future slots. After the final refinement round, refined future-slot hidden states are decoded into ego-local motion deltas, sequentially integrated from $\boldsymbol{p}_0 = (0, 0, 0)^\top$, and supervised by local-delta and rollout-level keypose losses.

The full supervised objective is

$$\mathcal{L}_{\text{sup}} = \mathcal{L}_{\text{text}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}} + \lambda_{\text{hyb}}\mathcal{L}_{\text{hyb}} + \lambda_{\text{token}}\mathcal{L}_{\text{token}} + \lambda_{\text{delta}}\mathcal{L}_{\text{delta}} + \lambda_{\text{key}}\mathcal{L}_{\text{keypose}}. \quad (19)$$

These terms supervise causal language generation, hybrid text states, categorical motion refinement, local motion accuracy, and rollout-level trajectory fidelity. Detailed formulas for each term are provided in Appendix A.3.

3.6 Optional GRPO-based post-training

RAPDrive can be further refined after supervised training with a planner-centric GRPO stage. For a scene context $x = (I, \boldsymbol{s}, H)$, the rollout policy samples a group of candidate reasoning-and-planning outputs. Each sample contains autoregressively generated reasoning text and a trajectory obtained from masked motion refinement followed by continuous decoding. In the reported NAVSIM post-training setting, the scalar reward is the official PDMS computed on the final trajectory. Group-normalized advantages are then used to update the joint text-and-motion policy with KL regularization to the supervised reference checkpoint. We report this post-training setting separately from supervised/base results. The full objective and notation are provided in Appendix A.4.

3.7 Inference

At inference time, RAPDrive first generates reasoning text autoregressively from the multimodal context. It then initializes all future planning slots as masked motion tokens and applies the four-round commit-two refinement schedule described in Section 3.4. The refined future-slot hidden states are passed to the continuous local-delta decoder, and the predicted local deltas are integrated from $\boldsymbol{p}_0 = (0, 0, 0)^\top$ to obtain the final 4 s trajectory.

4 Experiments

We evaluate RAPDrive on NAVSIM and Bench2Drive, separating supervised/base results from optional GRPO-based post-training so that architectural gains are not conflated with reward refinement.

4.1 Experimental Setup

Datasets and metrics. We report results on NAVSIM [Dauner et al., 2024] and Bench2Drive [Jia et al., 2024]. NAVSIM evaluates planning on nuPlan-derived scenarios [Caesar et al., 2021] and reports planner-centric component metrics. Bench2Drive evaluates closed-loop driving in CARLA [Dosovitskiy et al., 2017] and reports driving score, success rate, efficiency, and comfort. Because the two benchmarks use different simulators, metrics, and comparison sets, we discuss them separately. For NAVSIM, the primary score is PDMS:

$$\text{PDMS} = \text{NC} \cdot \text{DAC} \cdot \frac{5\text{TTC} + 2\text{C} + 5\text{EP}}{12}, \quad (20)$$

where NC is No at-fault Collision, DAC is Drivable Area Compliance, TTC is Time-to-Collision, C is Comfort, and EP is Ego Progress. The formula is computed using normalized metric values in $[0, 1]$, while the tables report NAVSIM metrics as percentages.

Implementation details. RAPDrive uses camera observations, ego status, and recent ego-motion history without lidar input. The visual input consists of front, front-left, and front-right camera images over two historical frames, resized to 448×448. The model predicts $T = 8$ future keyposes over a 4 s horizon.

We use InternVL3-2B [Zhu et al., 2025] as the multimodal backbone. The vision encoder is frozen, and we train the multimodal projector, the RAPDrive modules, the top four language-model decoder layers, the final normalization layer, and the language head. The endpoint-contour motion codebook is fit with K-means using $K = 2048$ tokens, with each token corresponding to one 0.5 s future segment.

Models are trained with AdamW using BF16 mixed precision on eight H20 GPUs. We use separate learning rates for pretrained language-model parameters and newly introduced modules; detailed hyperparameters are provided in Appendix A.5.

Reasoning supervision is generated offline with Qwen2.5-VL-72B using training-split data. The annotation inputs and output format are described in Appendix A.5.

4.2 Main Results

We first compare RAPDrive with prior methods on NAVSIM and Bench2Drive. For NAVSIM, we report supervised/base and post-training results separately.

NAVSIM supervised/base results. Table 1 reports the NAVSIM supervised/base comparison. RAPDrive reaches 86.9 PDMS using camera-only inputs, improving over AutoVLA, ReCogDrive, and AdaThinkDrive and matching DRAMA among the reported camera-only models. Several specialized end-to-end and world-model-augmented methods remain higher in the broader benchmark, while the direct VLA-to-VLA comparison favors RAPDrive.

Table 1: NAVSIM supervised/base comparison. E2E denotes traditional end-to-end driving, WM-VLA denotes world-model-augmented driving VLA, and VLA denotes base driving VLA. C denotes camera input and L denotes lidar input. Bold numbers mark the best score within each family.

Method	Type	Sensors	NC↑	DAC↑	TTC↑	Comf.↑	EP↑	PDMS↑
Ego Status MLP [Dauner et al., 2024]	E2E	–	93.1	78.3	84.0	**100.0**	63.2	66.4
VADv2 [Jiang et al., 2026]	E2E	C	97.2	89.1	91.6	**100.0**	76.0	80.9
UniAD [Hu et al., 2023]	E2E	C	97.8	91.9	92.9	**100.0**	78.8	83.4
TransFuser [Chitta et al., 2023]	E2E	C+L	97.7	92.8	92.8	**100.0**	79.2	84.0
PARA-Drive [Weng et al., 2024]	E2E	C	97.9	92.4	93.0	99.8	79.3	84.0
LAW [Li et al., 2025a]	E2E	C	96.4	95.4	88.7	99.9	81.7	84.6
Hydra-MDP [Li et al., 2024]	E2E	C+L	98.3	96.0	94.6	**100.0**	78.7	86.5
DiffusionDrive [Liao et al., 2025]	E2E	C+L	98.2	96.2	94.7	**100.0**	82.2	88.1
WoTE [Li et al., 2025c]	E2E	C+L	98.5	96.8	94.9	99.9	81.9	88.3
DRAMA [Yuan et al., 2024]	E2E	C	98.2	95.2	94.2	**100.0**	81.3	86.9
Centaur [Sima et al., 2025]	E2E	C+L	**99.2**	**98.7**	**97.2**	**100.0**	**86.0**	**92.1**
DriveVLA-W0 [Li et al., 2025b]	WM-VLA	C	98.7	**96.2**	95.5	**100.0**	**82.2**	**88.4**
Vega [Zuo et al., 2026]	WM-VLA	C	**98.9**	95.3	**96.1**	**100.0**	81.6	87.9
DynVLA [Shang et al., 2026]	WM-VLA	C	98.8	94.4	95.8	**100.0**	79.9	86.6
AutoVLA [Zhou et al., 2025]	VLA	C	96.9	92.4	88.1	99.9	75.8	80.5
ReCogDrive [Li et al., 2025d]	VLA	C	98.1	**94.7**	94.2	**100.0**	80.9	86.5
AdaThinkDrive [Luo et al., 2025]	VLA	C	**98.5**	94.4	**94.9**	**100.0**	79.9	86.2
RAPDrive	VLA	C	98.3	**94.7**	94.5	**100.0**	**81.0**	**86.9**

NAVSIM post-training results. Table 2 reports results after each method's post-training or policy-refinement stage. GRPO improves RAPDrive from 86.9 to 90.3 PDMS, placing it above AutoVLA and AutoDrive-R2 and matching AdaThinkDrive in this comparison, while DynVLA and ReCogDrive remain higher.

Table 2: NAVSIM results after each method's reported post-training or policy-refinement stage. RAPDrive uses optional GRPO-based refinement. Bold numbers denote the best value in each column.

Method	NC↑	DAC↑	TTC↑	Comf.↑	EP↑	PDMS↑
AutoVLA [Zhou et al., 2025]	98.4	95.6	**98.0**	99.9	81.9	89.1
AdaThinkDrive [Luo et al., 2025]	98.4	97.8	95.2	**100.0**	84.4	90.3
AutoDrive-R2 [Yuan et al., 2026]	98.5	95.9	95.4	**100.0**	82.7	89.1
DynVLA [Shang et al., 2026]	**98.6**	**98.7**	95.5	**100.0**	86.8	**91.7**
ReCogDrive [Li et al., 2025d]	97.9	97.3	94.9	**100.0**	**87.3**	90.8
RAPDrive + GRPO	98.1	96.9	95.1	99.9	86.4	90.3

Bench2Drive results. Table 3 reports the available Bench2Drive comparison. RAPDrive reaches a driving score of 79.82, exceeding AutoVLA and ReCogDrive in the collected table while remaining below DynVLA. Its success rate does not lead the table, indicating that the closed-loop gains are less uniform across metrics.

Table 3: Bench2Drive comparison. A dash indicates that the metric is not reported in the collected source table.

Method	Driving score↑	Success rate↑	Efficiency↑	Comfort↑
AD-MLP [Jia et al., 2024]	18.05	0.00	48.45	22.63
VAD [Jiang et al., 2023]	42.35	15.00	**157.94**	46.01
UniAD-T [Hu et al., 2023]	40.73	13.18	123.92	**47.04**
UniAD-B [Hu et al., 2023]	45.81	16.36	129.21	43.58
TCP-traj [Wu et al., 2022]	59.90	30.00	76.54	18.08
DriveAdapter [Jia et al., 2023a]	64.22	33.08	70.22	16.01
ThinkTwice [Jia et al., 2023b]	62.44	3.13	76.93	16.22
DriveTransformer-L [Jia et al., 2025]	63.46	35.01	100.64	46.01
Orion [Fu et al., 2025]	77.74	54.62	151.48	17.38
AutoVLA [Zhou et al., 2025]	78.84	57.73	146.93	39.33
ReCogDrive [Li et al., 2025d]	71.36	45.45	138.18	17.45
DynVLA [Shang et al., 2026]	**88.34**	**72.73**	–	–
RAPDrive	79.82	49.55	147.81	25.22

4.3 Ablation Studies

We next isolate the main design choices in RAPDrive under the same NAVSIM setting as the base model. Detailed component-wise ablation tables are provided in Appendix A.6. Across these ablations, the full model outperforms autoregressive future decoding, single-pass masked prediction, detached planning, removing reasoning conditioning, and pure continuous decoding. Removing rollout/keypose supervision causes the largest drop, while removing text-state alignment yields a smaller but visible degradation. Refinement-step analysis further improves from $N = 1$ to $N = 4$ and then slightly decreases at $N = 8$, supporting the default four-round schedule.

Scene-wise analysis further shows larger gains from reasoning conditioning and shared-latent planning in turning, intersections, close following, dense traffic, and traffic-light scenes than in straightforward lane following. Table 9 reports the detailed comparison across six scene categories.

Summary. Overall, the benchmark comparisons and ablations support two main observations. First, RAPDrive strengthens the supervised/base driving VLA setting on NAVSIM. Second, GRPO-based post-training further improves RAPDrive. The ablations support masked future-slot refinement, shared-latent planning, the endpoint-contour motion interface, and rollout/keypose supervision.

5 Limitations

RAPDrive is currently evaluated in a camera-centric setting with RGB observations, ego status, and recent motion history. Extending the shared-latent formulation to LiDAR, radar, BEV/occupancy representations, HD-map-centric scene features, or other explicit 3D inputs remains future work. RAPDrive also focuses on trajectory-centered reasoning and planning rather than dense future visual reconstruction, leaving the combination of shared planning slots with world-model-style supervision unexplored. The full-reasoning inference pipeline takes an average of 8.512 s per scene and does not yet meet real-time driving requirements. Reducing the latency of autoregressive reasoning before motion refinement remains an important step toward real-time operation.

6 Conclusion

We presented RAPDrive, a shared-latent hybrid-decoding framework for end-to-end driving VLA. RAPDrive couples reasoning and trajectory planning within one transformer sequence while using autoregressive text generation, iterative masked motion-token refinement, and continuous local-delta realization for trajectory decoding. Experiments on NAVSIM and Bench2Drive show that RAPDrive improves the supervised/base driving VLA setting relative to the listed non-world-model baselines, and controlled ablations support the main design choices. Optional GRPO-based post-training further improves RAPDrive, indicating compatibility with planner-centric reward refinement.

Acknowledgments and Disclosure of Funding

This work was supported by the National Key R&D Program of China under Grant No. 2024YFC3015801, the National Natural Science Foundation of China under Grant Nos. 62361166670, U24A20330, 62306238, and 62306155, and the Fundamental Research Funds for the Central Universities.

Competing Interests

The authors declare no competing interests.

References

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL `https://arxiv.org/abs/2212.06817`.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric M. Wolff, Alex H. Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021. URL `https://arxiv.org/abs/2106.11810`.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11305–11315, 2022. doi: 10.1109/CVPR52688.2022.01103.

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10164–10183, 2024. doi: 10.1109/TPAMI.2024.3435937.

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. URL `https://dblp.org/rec/journals/pami/ChittaPJYRG23`.

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. URL https://dblp.org/rec/journals/corr/abs-2406-15349.

Thierry Deruyttere, Simon Vandenhende, Dusan Grujicic, Luc Van Gool, and Marie-Francine Moens. Talk2Car: Taking control of your self-driving car. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2088–2098, 2019. URL https://aclanthology.org/D19-1215/.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio M. Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the Conference on Robot Learning*, pages 1–16, 2017. URL https://dblp.org/rec/conf/corl/DosovitskiyRCLK17.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. ORION: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24823–24834, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Fu_ORION_A_Holistic_End-to-End_Autonomous_Driving_Framework_by_Vision-Language_Instructed_ICCV_2025_paper.html.

Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. VISTA: A generalizable driving world model with high fidelity and versatile controllability. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://dblp.org/rec/conf/nips/GaoY0CQ0Z024.

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. doi: 10.1109/CVPR52729.2023.01712.

Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, James Guo, Dragomir Anguelov, and Mingxing Tan. EMMA: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024. URL https://arxiv.org/abs/2410.23262.

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. URL https://openaccess.thecvf.com/content/ICCV2023/html/Jia_DriveAdapter_Breaking_the_Coupling_Barrier_of_Perception_and_Planning_in_ICCV_2023_paper.html.

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. URL https://openaccess.thecvf.com/content/CVPR2023/html/Jia_Think_Twice_Before_Driving_Towards_Scalable_Decoders_for_End-to-End_Autonomous_CVPR_2023_paper.html.

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, 2024. URL https://dblp.org/rec/conf/nips/JiaYLZY24.

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*, 2025. URL https://openreview.net/forum?id=M42KR4W9P5.

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 8306–8316, 2023. URL https://dblp.org/rec/conf/iccv/JiangCXLCZZ0HW23.

Bo Jiang, Shaoyu Chen, Bencheng Liao, Xingyu Zhang, Wei Yin, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Senna: Bridging large vision-language models and end-to-end autonomous driving. *arXiv preprint arXiv:2410.22313*, 2024. URL https://arxiv.org/abs/2410.22313.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0a4dA6eUHN.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Yingyan Li, Lue Fan, Jiawei He, Yuqi Wang, Yuntao Chen, Zhaoxiang Zhang, and Tieniu Tan. Enhancing end-to-end autonomous driving with latent world model. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=fd2u60ryG0.

Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, Lu Hou, Lue Fan, and Zhaoxiang Zhang. DriveVLA-W0: World models amplify data scaling law in autonomous driving. *arXiv preprint arXiv:2510.12796*, 2025b. URL https://arxiv.org/abs/2510.12796.

Yingyan Li, Yuqi Wang, Yang Liu, Jiawei He, Lue Fan, and Zhaoxiang Zhang. End-to-end driving with online trajectory evaluation via BEV world model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27137–27146, 2025c. URL https://openaccess.thecvf.com/content/ICCV2025/html/Li_End-to-End_Driving_with_Online_Trajectory_Evaluation_via_BEV_World_Model_ICCV_2025_paper.html.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, 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*, 2025d. URL https://arxiv.org/abs/2506.08052.

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, Yu-Gang Jiang, and Jose M. Alvarez. Hydra-MDP: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*, 2024. URL https://arxiv.org/abs/2406.06978.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. DiffusionDrive: Truncated diffusion model for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12037–12047, 2025. doi: 10.1109/CVPR52734.2025.01124.

Yuechen Luo, Fang Li, Shaoqing Xu, Zhiyi Lai, Lei Yang, Qimao Chen, Ziang Luo, Zixun Xie, Shengyin Jiang, Jiaxin Liu, Long Chen, Bing Wang, and Zhi-xin Yang. AdaThinkDrive: Adaptive thinking via reinforcement learning for autonomous driving. *arXiv preprint arXiv:2509.13769*, 2025. URL https://arxiv.org/abs/2509.13769.

Yingzi Ma, Yulong Cao, Jiachen Sun, Marco Pavone, and Chaowei Xiao. Dolphins: Multimodal language model for driving. In *European Conference on Computer Vision*, pages 403–420, 2024. URL https://arxiv.org/abs/2312.00438.

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

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, Elahe Arani, and Oleg Sinavski. LingoQA: Visual question answering for autonomous driving. In *European Conference on Computer Vision*, pages 252–269, 2024. URL https://arxiv.org/abs/2312.14115.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. URL https://arxiv.org/abs/2212.09748.

Tianwen Qian, Jingjing Chen, Linhai Zhuo, Yang Jiao, and Yu-Gang Jiang. NuScenes-QA: A multi-modal visual question answering benchmark for autonomous driving scenario. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4542–4550, 2024. URL https://arxiv.org/abs/2305.14836.

Shuyao Shang, Bing Zhan, Yunfei Yan, Yuqi Wang, Yingyan Li, Yasong An, Xiaoman Wang, Jierui Liu, Lu Hou, Lue Fan, Zhaoxiang Zhang, and Tieniu Tan. DynVLA: Learning world dynamics for action reasoning in autonomous driving. *arXiv preprint arXiv:2603.11041*, 2026. URL https://arxiv.org/abs/2603.11041.

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, 2024a. doi: 10.1109/CVPR52733.2024.01432.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024b. URL https://arxiv.org/abs/2402.03300.

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, 2024. URL https://arxiv.org/abs/2312.14150.

Chonghao Sima, Kashyap Chitta, Zhiding Yu, Shiyi Lan, Ping Luo, Andreas Geiger, Hongyang Li, and Jose M. Alvarez. Centaur: Robust end-to-end autonomous driving with test-time training. *arXiv preprint arXiv:2503.11650*, 2025. URL https://arxiv.org/abs/2503.11650.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. DriveVLM: The convergence of autonomous driving and large vision-language models. In *Proceedings of the Conference on Robot Learning*, pages 4698–4726, 2024. URL https://dblp.org/rec/conf/corl/TianGLLWZZJLZ24.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a. URL https://arxiv.org/abs/2409.12191.

Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and Jose M. Alvarez. OmniDrive: A holistic LLM-agent framework for autonomous driving with 3d perception, reasoning and planning. *arXiv preprint arXiv:2405.01533*, 2024b. URL https://arxiv.org/abs/2405.01533.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022. URL https://dblp.org/rec/conf/nips/WeiWSBXCL022.

Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. PARA-Drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15449–15458, 2024. doi: 10.1109/CVPR52733.2024.01463.

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. URL https://dblp.org/rec/conf/nips/WuJCYLQ22.

Li Xu, He Huang, and Jun Liu. SUTD-TrafficQA: A question answering benchmark and an efficient network for video reasoning over traffic events. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9878–9888, 2021. URL https://arxiv.org/abs/2103.15538.

Chengran Yuan, Zhanqi Zhang, Jiawei Sun, Shuo Sun, Zefan Huang, Christina Dao Wen Lee, Dongen Li, Yuhang Han, Anthony Wong, Keng Peng Tee, and Marcelo H. Ang. DRAMA: An efficient end-to-end motion planner for autonomous driving with mamba. *arXiv preprint arXiv:2408.03601*, 2024. URL https://arxiv.org/abs/2408.03601.

Zhenlong Yuan, Chengxuan Qian, Jing Tang, Rui Chen, Zijian Song, Lei Sun, Xiangxiang Chu, Yujun Cai, Dapeng Zhang, and Shuo Li. AutoDrive-R²: Incentivizing reasoning and self-reflection capacity for VLA model in autonomous driving. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KVWaCzJrrq.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. *arXiv preprint arXiv:2503.22020*, 2025. URL https://arxiv.org/abs/2503.22020.

Wenzhao Zheng, Ruiqi Song, Xianda Guo, Chenming Zhang, and Long Chen. GenAD: Generative end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 87–104, 2024. URL https://arxiv.org/abs/2402.11502.

Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation*, pages 3560–3566, 2023. doi: 10.1109/ICRA48891.2023.10161463. URL https://arxiv.org/abs/2210.17366.

Zewei Zhou, Tianhui Cai, Seth Z. Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. AutoVLA: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025. URL https://arxiv.org/abs/2506.13757.

Jinguo Zhu et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. URL https://arxiv.org/abs/2504.10479.

Sicheng Zuo, Yuxuan Li, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Vega: Learning to drive with natural language instructions. *arXiv preprint arXiv:2603.25741*, 2026. URL https://arxiv.org/abs/2603.25741.

A Appendix

This appendix provides additional details on motion tokenization, masked motion refinement, supervised and post-training objectives, implementation settings, and supplementary qualitative and quantitative results.

A.1 Motion-Token Reconstruction for Diagnostics

Section 3.2 defines the endpoint-contour tokenization pipeline. Here we give the token-to-contour reconstruction used for tokenizer diagnostics. Given a decoded codeword $\boldsymbol{e}_{z_t}$ and the training-set statistics $(\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi)$, the corresponding endpoint-contour feature is recovered by

$$\hat{\boldsymbol{\phi}}_t = \boldsymbol{\mu}_\phi + (\boldsymbol{\sigma}_\phi + \epsilon \mathbf{1}) \odot \boldsymbol{e}_{z_t}. \tag{21}$$

After reshaping $\hat{\boldsymbol{\phi}}_t$ into four corners $\{\hat{\boldsymbol{b}}_{t,m}\}_{m=1}^4$, define the front and rear centers

$$\hat{\boldsymbol{f}}_t = \frac{\hat{\boldsymbol{b}}_{t,1} + \hat{\boldsymbol{b}}_{t,2}}{2}, \qquad \hat{\boldsymbol{r}}_t = \frac{\hat{\boldsymbol{b}}_{t,3} + \hat{\boldsymbol{b}}_{t,4}}{2}. \tag{22}$$

The local endpoint heading is then recovered as

$$\widehat{\Delta\theta}_t^{\mathrm{rec}} = \operatorname{atan2}\big((\hat{\boldsymbol{f}}_t - \hat{\boldsymbol{r}}_t)_y, (\hat{\boldsymbol{f}}_t - \hat{\boldsymbol{r}}_t)_x\big). \tag{23}$$

Let

$$\bar{\boldsymbol{q}} = \frac{1}{4} \sum_{m=1}^{4} \boldsymbol{q}_m. \tag{24}$$

The local translation is recovered from the contour mean:

$$\hat{\boldsymbol{d}}_t = \frac{1}{4} \sum_{m=1}^{4} \hat{\boldsymbol{b}}_{t,m} - \mathrm{Rot}(\widehat{\Delta\theta}_t^{\mathrm{rec}}) \bar{\boldsymbol{q}}. \tag{25}$$

The reconstructed local motion is

$$\hat{\boldsymbol{\Delta}}_t^{\mathrm{rec}} = (\hat{d}_{t,x}, \hat{d}_{t,y}, \widehat{\Delta\theta}_t^{\mathrm{rec}})^\top. \tag{26}$$

Sequentially integrating $\hat{\boldsymbol{\Delta}}_t^{\mathrm{rec}}$ from $\boldsymbol{p}_0 = (0,0,0)^\top$ yields reconstructed keyposes for measuring tokenizer reconstruction error. The superscript rec distinguishes diagnostic reconstruction from the continuous decoder's prediction $\hat{\boldsymbol{\Delta}}_t$. The planned trajectory used by RAPDrive is produced by the continuous local-delta decoder described in Section 3.5.

A.2 Training-Time Masked States for Motion Denoising

Section 3.4 defines the inference-time four-round commit-two refinement schedule. During supervised training, RAPDrive samples partial masked states that match this schedule. Specifically, we sample a refinement stage $n \sim \mathrm{Uniform}\{0,1,2,3\}$, reveal $m_n = 2n$ ground-truth slots, and mask the remaining $r_n = T - m_n$ slots. A mask set $\mathcal{M}_n$ is sampled uniformly without replacement with $|\mathcal{M}_n| = r_n$, producing

$$\tilde{z}_t^{(n)} = \begin{cases} [\mathrm{MASK}], & \text{if } t \in \mathcal{M}_n, \\ z_t^\star, & \text{otherwise.} \end{cases} \tag{27}$$

The denoising loss is applied only to positions in $\mathcal{M}_n$, matching the partial reveal pattern used by the inference-time schedule.

A.3 Detailed Supervised Objectives

The native text path uses the pretrained language-model computation over teacher-forced text tokens with standard causal attention. Let $\mathcal{L}_{\mathrm{text}}$ denote the causal language-modeling loss. RAPDrive also computes hybrid text states from the multimodal sequence and aligns them with native-path text hidden states over valid non-padding text positions:

$$\mathcal{L}_{\mathrm{align}} = \frac{1}{|\Omega_Y|} \sum_{\ell \in \Omega_Y} \left(1 - \frac{\langle \boldsymbol{h}_\ell^{\mathrm{hybrid}}, \boldsymbol{h}_\ell^{\mathrm{native}} \rangle}{\|\boldsymbol{h}_\ell^{\mathrm{hybrid}}\|_2 \|\boldsymbol{h}_\ell^{\mathrm{native}}\|_2}\right). \tag{28}$$

A small hybrid-path text cross-entropy term $\mathcal{L}_{\mathrm{hyb}}$ is also included to make hybrid text states readable by the language head.

Motion-token denoising. Using the training-time masked states defined in Appendix A.2, the motion-token loss is applied only on masked positions:

$$\mathcal{L}_{\text{token}} = \frac{1}{|\mathcal{M}_n|} \sum_{t \in \mathcal{M}_n} -\log p_\theta(z_t^\star \mid \tilde{z}^{(n)}, I, \boldsymbol{s}, H, Y). \tag{29}$$

Continuous local-delta and rollout losses. The final trajectory realization is produced by a continuous local-delta decoder operating on refined future-slot hidden states. Although masked refinement is trained in a discrete endpoint-contour token space, RAPDrive does not output the reconstructed codebook trajectory directly. Instead, after the final refinement round, each future-slot hidden state is passed to a regression head that predicts an ego-local motion primitive

$$\hat{\boldsymbol{\Delta}}_t = (\widehat{\Delta x}_t, \widehat{\Delta y}_t, \widehat{\Delta \theta}_t)^\top, \qquad t = 1, \ldots, T. \tag{30}$$

These local deltas are defined between consecutive future keyposes and are sequentially integrated from $\boldsymbol{p}_0 = (0, 0, 0)^\top$ to obtain the final 4 s trajectory. The local regression target is the direct ego-local delta sequence derived from ground-truth future keyposes:

$$\mathcal{L}_{\text{delta}} = \sum_{t=1}^{T} \text{SmoothL1}(\hat{\boldsymbol{\Delta}}_t, \boldsymbol{\Delta}_t^\star). \tag{31}$$

After integrating predicted local deltas into future keyposes $\hat{P}$, RAPDrive applies rollout-level supervision on position and heading:

$$\mathcal{L}_{\text{keypose}} = \mathcal{L}_{\text{xy}}(\hat{P}, P^\star) + \lambda_\theta \mathcal{L}_{\text{heading}}(\hat{P}, P^\star), \tag{32}$$

where heading error is wrapped before applying the loss.

A.4 GRPO Post-Training Objective

RAPDrive can be refined with GRPO after supervised training. Let $x = (I, \boldsymbol{s}, H)$ be the scene context, $Y = (y_1, \ldots, y_L)$ the generated reasoning text, and $z = (z_1, \ldots, z_T)$ the committed endpoint-contour motion-token sequence. The final trajectory, denoted τ here and P in the main text, is obtained from the refined future-slot hidden states:

$$\tau = \text{Decode}_\theta(\mathbf{H}_{\text{plan}}(x, Y, z)), \tag{33}$$

where $\mathbf{H}_{\text{plan}}$ denotes the planning-slot hidden-state matrix after masked refinement and Decode_θ is the continuous local-delta decoder.

The joint policy factorizes into autoregressive text decisions and committed motion-token decisions:

$$\pi_\theta(Y, z \mid x) = \prod_{\ell=1}^{L} \pi_\theta^{\text{text}}(y_\ell \mid x, y_{<\ell}) \cdot \prod_{n=0}^{N-1} \prod_{j \in C^{(n)}} \pi_\theta^{\text{mot}}(z_j \mid \xi_{n,j}), \tag{34}$$

where $C^{(n)}$ is the set of slots committed at refinement round n, and $\xi_{n,j}$ denotes the corresponding masked-refinement state. Since committed slots remain fixed, each future slot contributes once to the motion-token log probability.

For each scene, GRPO samples a group of candidate reasoning-and-planning outputs from the rollout policy:

$$(Y_i, Z_i) \sim \pi_{\text{old}}(\cdot \mid x), \qquad i = 1, \ldots, G. \tag{35}$$

Here Y_i and Z_i denote complete text and motion-token sequences for candidate i, rather than individual tokens. Each candidate receives a scalar planner-centric reward from its final trajectory:

$$r_i = S_{\text{plan}}(\tau_i). \tag{36}$$

In the NAVSIM post-training experiments reported in this paper, S_{plan} is the official PDMS computed on the decoded trajectory:

$$S_{\text{plan}}(\tau_i) = \text{PDMS}(\tau_i) = \text{NC}_i \cdot \text{DAC}_i \cdot \frac{5\text{TTC}_i + 2\text{C}_i + 5\text{EP}_i}{12}. \tag{37}$$

The component scores are normalized to $[0, 1]$ when computing the reward. Invalid rollouts or outputs that cannot be evaluated are assigned zero reward.

Group-relative advantages are

$$\mu_r = \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \sigma_r = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i - \mu_r)^2}, \tag{38}$$

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon_{\text{adv}}}. \tag{39}$$

With

$$\rho_i(\theta) = \exp\left(\log \pi_\theta(Y_i, Z_i \mid x) - \log \pi_{\text{old}}(Y_i, Z_i \mid x)\right), \tag{40}$$

the clipped GRPO surrogate is

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{G}\sum_{i=1}^{G} \min\left(\rho_i(\theta)\hat{A}_i,\ \text{clip}(\rho_i(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\hat{A}_i\right) - \beta \mathcal{D}_{\text{KL}}^{\text{joint}}(\pi_\theta \,\|\, \pi_{\text{ref}}), \tag{41}$$

where π_{ref} is the supervised reference policy.

A.5 Additional Implementation Details

Section 4.1 reports the backbone, sensor configuration, codebook size, and high-level optimization settings. Here we record the implementation details used in the current training setup.

Backbone and trainable modules. RAPDrive uses InternVL3-2B as the multimodal backbone. The vision encoder is frozen throughout training. We train the multimodal projector, the top four language-model decoder layers, the final normalization layer, and the language head. The RAPDrive-specific trainable modules include the ego-state projector, history projector, future-slot embedding, motion-codebook projector, motion-token prediction head, local-delta regression head, planning-fusion module, planning-context attention blocks, mask-token embedding, and future-slot positional embedding. The endpoint-contour token embedding $\boldsymbol{e}_{\text{tok}}$ is a learned embedding indexed by the committed motion-token id; the codebook itself defines the geometric target used for supervision and tokenizer diagnostics.

Optimization and training system. We optimize with AdamW using weight decay 0.01 and betas $(0.9, 0.95)$. The implementation uses three learning-rate groups: 2×10^{-5} for pretrained language-model parameters, 1×10^{-4} for newly introduced non-trajectory modules, and 2×10^{-5} for trajectory-related modules. The current training script uses a constant learning rate without an enabled scheduler, BF16 mixed precision, gradient clipping at 1.0, and up to 40 training epochs. Training is run with DeepSpeed ZeRO stage 2 on one node with eight H20 GPUs.

Reasoning annotation generation. Reasoning annotations are generated offline by Qwen2.5-VL-72B. For each training scene, the annotator receives camera observations, ego state, the navigation command, and the expert future trajectory as an action hint. The annotation request specifies seven fields: `<Perception>`, `<Traffic>`, `<Signal>`, `<Map>`, `<Influence>`, `<Decision_Logic>`, and `<Current_Action>`. These fields form the semi-structured reasoning sequence used for supervised training.

Annotation processing. Malformed annotation outputs are regenerated or discarded. Annotation generation uses training-split data only, without navtest labels or evaluation scores. The expert future trajectory is used only as an offline annotation hint during training; at inference, reasoning is generated from the available camera observations, ego state, and motion-history context.

Geometry convention. For motion-token construction and trajectory supervision, we use the default NAVSIM/nuPlan ego-vehicle footprint convention with Pacifica vehicle parameters and a rear-axle-centered anchor. The endpoint-contour codebook is fit on per-dimension standardized endpoint-corner features computed from the training split only, so tokenizer statistics and token assignments remain aligned with the evaluation geometry.

A.6 Detailed Ablation Tables

Tables 4–8 provide the detailed component-wise ablation results supporting the ablation discussion in the main paper. Bold values mark column-wise maxima, including ties.

Table 4: Future decoding ablation on NAVSIM-style PDMS components.

Method	NC↑	DAC↑	TTC↑	C↑	EP↑	PDMS↑
AR future decoding	97.8	92.2	91.8	**100.0**	79.8	83.1
Single-pass masked prediction	97.4	91.6	93.3	99.9	79.9	84.2
RAPDrive	**98.3**	**94.7**	**94.5**	**100.0**	**81.0**	**86.9**

Table 5: Denoising-step analysis. The default uses $N = 4$.

Refinement steps	PDMS↑
$N = 1$	84.2
$N = 2$	85.5
$N = 4$	**86.9**
$N = 8$	86.6

Table 6: Shared-latent and reasoning-conditioning ablations.

Method	NC↑	DAC↑	TTC↑	C↑	EP↑	PDMS↑
Detached planner	96.4	91.8	92.4	99.9	78.4	82.7
No reasoning conditioning	97.6	92.1	92.2	99.9	79.5	84.1
RAPDrive	**98.3**	**94.7**	**94.5**	**100.0**	**81.0**	**86.9**

Table 7: Discrete motion interface versus pure continuous decoding.

Method	NC↑	DAC↑	TTC↑	C↑	EP↑	PDMS↑
Pure continuous decoder	98.2	94.3	93.2	**100.0**	79.8	85.0
Discrete interface + continuous realization	**98.3**	**94.7**	**94.5**	**100.0**	**81.0**	**86.9**

Table 8: Training-objective ablations.

Method	NC↑	DAC↑	TTC↑	C↑	EP↑	PDMS↑
Without rollout/keypose loss	97.1	90.3	92.5	**100.0**	79.0	82.8
Without text-state alignment	98.1	**96.0**	94.2	99.9	78.4	85.8
RAPDrive	**98.3**	94.7	**94.5**	**100.0**	**81.0**	**86.9**

Scene-wise analysis. Table 9 compares RAPDrive with the no-reasoning and detached-planner variants across six NAVSIM scene categories under the same supervised/base evaluation setting. The gains over the no-reasoning variant increase from 0.8 PDMS in lane following to 2.7–3.7 in the other categories. The corresponding gains over the detached planner increase from 1.6 to 4.5–5.4 PDMS, with the largest gains in dense traffic and interaction scenes.

Table 9: Scene-wise NAVSIM planning results in the supervised/base setting. The last two columns report RAPDrive's gains in PDMS points over the no-reasoning and detached-planner variants.

NAVSIM subset	RAPDrive	Gain over no reasoning	Gain over detached planner
Lane following / low curvature	89.0	+0.8	+1.6
High curvature / turning	85.8	+3.3	+5.2
Intersection	85.3	+3.3	+5.2
Close leading vehicle	84.9	+3.3	+5.2
Dense traffic / interaction	83.8	+3.7	+5.4
Traffic-light relevant	85.5	+2.7	+4.5

A.7 Dual-mode Planning Extension

RAPDrive conditions motion refinement on reasoning tokens in the default inference pipeline. This improves the coupling between semantic reasoning and trajectory planning, but it also introduces autoregressive text-generation latency before motion refinement starts. A natural extension is to train RAPDrive with both a reasoning-conditioned slow mode and a direct-planning fast mode, allowing the model to skip explicit reasoning in simple scenes while retaining the full reasoning pathway when needed.

Dual-mode sequence construction. The slow mode follows the default RAPDrive sequence:

$$\mathbf{X}_{\text{slow}} = [\,\mathbf{X}_{\text{vis}} \mid \boldsymbol{x}_{\text{ego}} \mid \mathbf{X}_{\text{hist}} \mid \mathbf{X}_{\text{text}} \mid \mathbf{X}_{\text{plan}}\,], \tag{42}$$

where $\mathbf{X}_{\text{text}} = \text{Embed}_{\text{text}}(Y)$ embeds teacher-forced reasoning tokens during supervised training and generated reasoning tokens during inference. Future planning slots attend to the multimodal prefix, reasoning states, and all future slots under the hybrid attention mask.

The fast mode removes the long reasoning sequence and inserts a short control token `<FAST_PLAN>`:

$$\mathbf{X}_{\text{fast}} = [\,\mathbf{X}_{\text{vis}} \mid \boldsymbol{x}_{\text{ego}} \mid \mathbf{X}_{\text{hist}} \mid \boldsymbol{x}_{\text{fast}} \mid \mathbf{X}_{\text{plan}}\,]. \tag{43}$$

Here $\boldsymbol{x}_{\text{fast}}$ is the control-token embedding indicating direct planning from the visual, ego-state, history, and prompt context. The planning slots still attend bidirectionally to one another and to the full multimodal prefix, preserving the shared-latent planning mechanism while avoiding long autoregressive reasoning generation.

Mixed-mode supervised training. For each training scene, we construct either a slow-mode sample or a fast-mode sample. In the slow mode, RAPDrive uses the standard supervised objective with text supervision, text-state alignment, motion-token denoising, local-delta regression, and rollout-level keypose supervision:

$$\mathcal{L}_{\text{slow}} = \mathcal{L}_{\text{text}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}} + \lambda_{\text{hyb}}\mathcal{L}_{\text{hyb}} + \lambda_{\text{token}}\mathcal{L}_{\text{token}} + \lambda_{\text{delta}}\mathcal{L}_{\text{delta}} + \lambda_{\text{key}}\mathcal{L}_{\text{keypose}}. \tag{44}$$

In the fast mode, the model skips long-form reasoning supervision and optimizes only the planning-related objectives:

$$\mathcal{L}_{\text{fast}} = \lambda_{\text{token}}\mathcal{L}_{\text{token}} + \lambda_{\text{delta}}\mathcal{L}_{\text{delta}} + \lambda_{\text{key}}\mathcal{L}_{\text{keypose}}. \tag{45}$$

The mixed-mode training objective is

$$\mathcal{L}_{\text{mix}} = \mathbb{I}\{\eta = \text{slow}\}\mathcal{L}_{\text{slow}} + \mathbb{I}\{\eta = \text{fast}\}\mathcal{L}_{\text{fast}}, \tag{46}$$

where $\eta \in \{\text{fast}, \text{slow}\}$ is the selected mode and $\mathbb{I}$ is the indicator function, so exactly one term is active per example. The mode can be sampled randomly or assigned by scene complexity. A simple implementation uses a fixed probability p_{fast} to sample fast-mode examples and $1 - p_{\text{fast}}$ to sample slow-mode examples. In practice, a conservative schedule $p_{\text{fast}} = 0.25$ is used initially to reduce latency while preserving the reasoning-conditioned pathway.

Scene-dependent mode selection. The fast mode is most suitable for straightforward scenes, such as lane following with low traffic density and low trajectory curvature. The slow mode is more appropriate for complex intersections, dense traffic, close leading vehicles, traffic-light reasoning, or uncertain planning states. A simple adaptive policy can therefore select the mode using model uncertainty:

$$\eta = \begin{cases} \text{fast}, & \text{if the scene is simple and planning confidence is high,} \\ \text{slow}, & \text{otherwise.} \end{cases} \tag{47}$$

Specifically, the model first runs fast planning and measures motion-token entropy or commit confidence. If confidence is low, it can rerun the scene in slow mode with explicit reasoning.

Latency-quality trade-off. Table 10 summarizes the currently measured latency and planning-quality results for three reasoning modes. The *skip reasoning* setting measures only the runtime of the direct-planning fast pathway, which reduces latency to 0.624 s. The *adaptive reasoning* setting is currently evaluated only in terms of planning quality and reaches 81.5 PDMS. The *full reasoning* setting corresponds to the default RAPDrive inference pipeline used in the main paper and reports both runtime and quality, with 8.512 s average runtime and 86.9 PDMS. These results indicate that removing or reducing explicit reasoning can improve efficiency, while the default full-reasoning pathway remains the strongest among the reported settings in planning quality.

Table 10: Latency and planning-quality measurements for the dual-mode planning extension. Runtime is reported in seconds, PDMS is reported in percentage form, and "–" indicates that the corresponding metric is not measured in the current comparison.

Model setting	Avg. runtime (s)	PDMS
Skip reasoning	0.624	–
Adaptive reasoning	–	81.5
Full reasoning	8.512	86.9

A.8 Qualitative NAVSIM Visualizations

Figures 4 and 5 provide qualitative NAVSIM examples from the navtest split. These visualizations complement the quantitative planning results in Section 4.2.



Figure 4: Qualitative NAVSIM visualization from the navtest split. Green denotes the ground-truth trajectory, and red denotes the predicted trajectory.



Figure 5: Additional qualitative NAVSIM visualization from the navtest split. Green denotes the ground-truth trajectory, and red denotes the predicted trajectory.

A.9 Qualitative Reasoning Example

Figure 6 provides a qualitative NAVSIM navtest example of the semi-structured reasoning format used by RAPDrive.

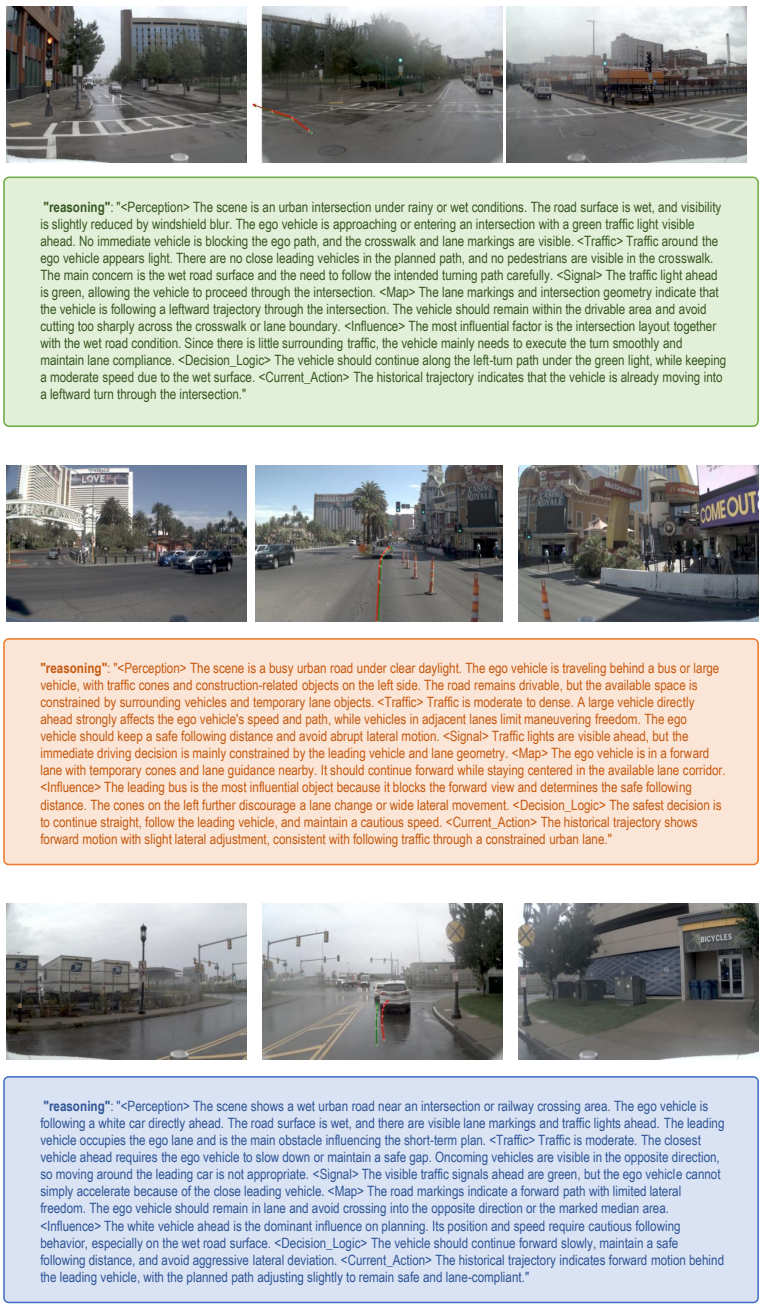

Figure 6: Qualitative reasoning example from the NAVSIM navtest split. The example illustrates the semi-structured reasoning format used before masked motion refinement.

B Broader Impact

RAPDrive is evaluated only as a research prototype on established autonomous-driving benchmarks and is not intended for direct deployment. Potential positive impacts include improving reasoning-aware trajectory planning and providing a clearer testbed for studying language-conditioned driving policies. Potential risks include over-reliance on generated reasoning, unsafe behavior under distribution shift, and misuse of benchmark performance as evidence of deployment readiness. Any real-world use would require extensive closed-loop validation, safety monitoring, redundancy, and compliance with local regulations.

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 shared-latent hybrid-decoding formulation, the endpoint-contour motion interface, and optional GRPO post-training, and these claims are supported by the method, experiments, results, and limitations sections.

   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: Section 5 discusses the camera-centric setting, the absence of explicit 3D modalities such as LiDAR, and the current focus on trajectory-centered supervision rather than world-model-style future reconstruction.

   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 theorems or proof-based theoretical results; it uses equations to define the model architecture, objectives, and training procedures.

   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: Sections 3–4 and the appendix describe the architecture, objectives, tokenizer, evaluation setup and hyperparameters.

   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 cited pretrained assets, and the authors intend to release the codes and dataset to facilitate community research.

   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: Section 4.1 specifies the datasets, metrics, sensor configuration, horizon, backbone, codebook size, optimizer, precision, and GPU count, while Section 3 and the appendix describe the training objectives and inference procedure needed to interpret the reported results.

   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: [No]

   Justification: The paper reports benchmark scores from the evaluated runs, consistent with prior work on this task.

   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: The paper provides necessary details on computational resources in Section 4.1.

   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 an academic study on established public benchmarks, does not involve human subjects or crowdsourcing, and we are not aware of any aspect that conflicts with the NeurIPS Code of Ethics.

   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: Appendix discusses potential benefits and risks of reasoning-aware autonomous-driving policies, including safety-critical deployment concerns, distribution shift, and the need for extensive validation before real-world use.

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 current submission does not describe the release of a model, dataset, or interactive system that would require a responsible-release protocol.

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 original creators and asset owners are appropriately credited, and licenses and terms of use are explicitly stated and duly respected.

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: [Yes]

    Justification: The authors will release the code, model weights, and datasets to facilitate research in the domain.

    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 subjects research.

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: Sections 3 and 4 explicitly describe the multimodal backbone, language-model components, autoregressive reasoning generation, and the role of language tokens in the core method.

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.