Latent Chain-of-Thought World Modeling for End-to-End Autonomous Driving Appendix

Shuhan Tan[1,2*] Kashyap Chitta[2] Yuxiao Chen[2] Ran Tian[2] Yurong You[2] Yan Wang[2]
Wenjie Luo[2] Yulong Cao[2] Philipp Krähenbühl[1] Marco Pavone[2,3] Boris Ivanovic[2]
[1]UT Austin [2]NVIDIA [3]Stanford University

A. Additional Implementation Details

A.1. Latent World Model Encoder

Our latent world model (LWM) encodes the surrounding agents around the ego vehicle (*excluding* the ego vehicle) into a compact set of tokens for latent chain-of-thought reasoning. Concretely, each LWM state summarizes a fixed $1.0\,\mathrm{s}$ window at $10\,\mathrm{Hz}$ in an ego-centric frame.

Per-agent temporal encoder. For each clip, we select the N nearest agents (based on distance in the current frame). The raw per-timestep state of each agent includes position, heading, dimensions, velocity, and other kinematic attributes. We stack these over a $1.0\,\mathrm{s}$ window (10 frames) to obtain

$$\texttt{agent\_state} \in \mathbb{R}^{B\times N\times T\times F},$$

where B is the batch size, N the number of agents, $T{=}10$ the number of timesteps, and F the number of input features. We first augment the state with 4 oriented corner points of the 3D bounding box (projected to BEV), resulting in 8 additional normalized features per timestep. A linear layer projects the concatenated features from dimension $(F{+}8)$ to a latent dimension d_{lwm}, after which we apply: 1) a learned timestep embedding added along the temporal axis; 2) an agent-type embedding (shared over timesteps) added per agent; 3) a stack of MLP residual blocks along the feature dimension. This produces a sequence of per-agent, per-timestep features of shape $\mathbb{R}^{B\times N\times T\times d_{\mathrm{lwm}}}$.

Temporal pooling per agent. To summarize the $T{=}10$ timesteps into a single feature per agent, we use a learnable query vector and a cross-attention layer along the time axis. The query attends to the T timestep features with an attention mask that ignores invalid timesteps, yielding one vector per agent:

$$\texttt{LWM\_agent} \in \mathbb{R}^{B\times N\times d_{\mathrm{lwm}}}.$$

Two-token LWM summarization. The latent world model state LWM_t used in LCDrive is a compact summary of all agents in the $1.0\,\mathrm{s}$ window. We train an additional attention layer with $M << N$ learnable query tokens, each of dimension d_{lwm}, to attend over the N agent features:

$$\mathrm{LWM}_t = \texttt{Attn}\big(Q_M,\ \texttt{LWM\_agent}\big) \in \mathbb{R}^{B\times M\times d_{\mathrm{lwm}}}.$$

These M tokens keep the LWM interface extremely compact for latent reasoning. In this paper, we use $N = 64$ and $M = 2$, maintaining a compact representation of LWM while capturing rich agent state information.

A.2. Stage 1: CoT Cold Start

In Stage 1, we teach the model the structure of latent chain-of-thought (CoT) by *teacher forcing* both the action-proposal tokens and the corresponding latent world model (LWM) tokens. Here we focus on how we construct the supervised reasoning sequence.

Action proposals from a frozen GT-LWM model. We start from the LWM_0-only model with ground-truth LWM inputs (Row 1 of Tab. 1 in the main paper). This model is trained without latent reasoning and serves as a strong teacher that produces full $6.4\,\mathrm{s}$ trajectories. Given sensor inputs $(o_{\text{image}}, o_{\text{ego}})$ and the history latent state LWM_0, the frozen teacher π_0 autoregressively samples discrete trajectory tokens

$$a_{1:64} \sim \pi_0(\cdot \mid o_{\text{image}}, o_{\text{ego}}, \mathrm{LWM}_0).$$

For each training clip, we draw B such trajectories $\{a^{(i)}_{1:64}\}_{i=1}^{B}$ using top-p sampling (temperature 0.6, $p = 0.98$). Each sampled trajectory is then sliced into K non-overlapping $1.0\,\mathrm{s}$ action blocks of length 10:

$$A^{(i)}_t := \big(a^{(i)}_{10t+1}, \ldots, a^{(i)}_{10(t+1)}\big), \qquad t = 0, \ldots, K{-}1.$$

These blocks define the *target* action-proposal tokens that our latent CoT policy imitates during cold start.

Action-conditioned LWM supervision. For each branch i and block index t, we construct an LWM supervision token $\mathrm{LWM}^{(i)}_{t+1}$ that encodes the *future world state conditioned on the proposal* $A^{(i)}_t$.

*Work done during an internship at NVIDIA.

Starting from the ground-truth ego pose at the beginning of the window, we integrate the sequence of 10 motion-primitive codes in $A_t^{(i)}$ to obtain the ego pose trajectory over that 1.0 s interval. At each timestep, we 1) take the ground-truth bounding boxes of all tracked agents from the PhysicalAI-AV dataset; 2) transform these boxes into the ego-centric frame defined by the integrated ego pose (translation and rotation); 3) feed the resulting agent states into the LWM encoder described in the last subsection. The encoder yields a compact latent world-model summary for that 1.0 s window, which we store as the target token $\mathrm{LWM}_{t+1}^{(i)}$. Repeating this for all blocks $t = 0, \ldots, K-1$ produces an interleaved supervision trace

$$R^{(i)} = \left[A_0^{(i)}, \mathrm{LWM}_1^{(i)}, \ldots, A_{K-1}^{(i)}, \mathrm{LWM}_K^{(i)}\right].$$

A.3. Stage 2: Reinforcement Learning

For the reinforcement learning stage of `LCDrive`, we adopt the `cosmos-rl` framework as our RL backbone. All RL experiments are conducted on a single 8-GPU node. We allocate 6 GPUs as rollout actors, each running an independent sampler replica of `LCDrive` in inference mode; 2 GPUs as learners, jointly performing GRPO optimization and broadcasting updated parameters to all actors. This partitioning enables high-throughput rollout while keeping optimization stable and fully GPU-resident.

The learning objective is the GRPO loss described in the main paper, but applied to *all* latent CoT tokens. This allows RL to restructure and refine the latent reasoning process itself, beyond imitation from Stage 1. Empirically, we observe that latent reasoning benefits significantly more from RL than non-reasoning baselines, highlighting the importance of RL-based optimization through the latent world-model interface.

B. Runtime and Compute Details

We report wall-clock inference latency and training compute to substantiate the efficiency claims in the main paper.

Inference latency. We benchmark all methods on a single NVIDIA RTX A5000 (24 GB) with identical decoding settings (temperature 0.6, top-p 0.98) and batch size 1. Tab. 1 breaks the total latency into three phases: input encoding, reasoning, and action decoding. Because the backbone, input pipeline, and action decoder are shared across all methods, the latency difference comes entirely from the reasoning phase. `LCDrive` generates 42 latent reasoning tokens in 1,388 ms compared to 72 text tokens in 2,447 ms for Text CoT, yielding a $1.8\times$ reasoning speedup with the same peak VRAM. A lighter configuration (LCDrive-Light, 24 tokens) further reduces reasoning time to 755 ms ($3.2\times$). Note that AR1 achieves 99 ms on-vehicle latency via distillation, TensorRT export, and optimized hardware [1]; as `LCDrive` shares the same backbone, these optimizations are directly applicable.

Training compute. Tab. 2 summarizes the LCDrive-specific training cost. All runs use a single node with $8\times$NVIDIA A100 (80 GB) GPUs. Stage 0 initializes from the pre-trained AR1 checkpoint (No-CoT). Stage 1 (Latent CoT cold start) takes 34 wall-clock hours (272 GPU-hours) with batch size 128. Stage 2 (RL via GRPO) takes 46 wall-clock hours (369 GPU-hours) with policy optimization batch size 96. The total LCDrive-specific training cost is 641 GPU-hours beyond the AR1 checkpoint.

C. Reasoning Action Analysis

To better understand the behavior of latent chain-of-thought reasoning before/after reinforcement learning, we analyze the relationship between the *proposal actions* generated during the reasoning stage and the *final action* output by the policy. For each validation clip, `LCDrive` generates $B{=}2$ reasoning branches, each producing a 50-step rollout trajectory, decoded from the action proposal tokens $A_t^{(i)}$. The final decoded trajectory has 64 steps; we truncate it to the first 50 steps for consistent comparison.

Let $\hat{\tau}_0$ and $\hat{\tau}_1$ denote the two proposal rollouts, $\hat{\tau}_{\mathrm{final}}$ the final action trajectory (trimmed to 50 steps), and $\tau^\star$ the ground-truth future trajectory. We define four metrics as below. All metrics are reported as Average Displacement Error (ADE) in meters.

1. **Reasoning Diversity:**

$$\text{Diversity} = \mathrm{ADE}\big(\hat{\tau}_0, \hat{\tau}_1\big),$$

   measures how different the two proposal branches are.
2. **Reasoning–Action Alignment:**

$$\text{Alignment} = \min_{k\in\{0,1\}} \mathrm{ADE}\big(\hat{\tau}_{\mathrm{final}}, \hat{\tau}_k\big),$$

   measures how closely the final action aligns with at least one proposal.
3. **Reasoning Quality:**

$$\text{Quality} = \frac{1}{2}\sum_{k\in\{0,1\}} \mathrm{ADE}\big(\hat{\tau}_k, \tau^\star\big),$$

   measures how good the proposals are with respect to the ground-truth trajectory.
4. **Final-Action Quality:**

$$\text{Final-Action} = \mathrm{ADE}\big(\hat{\tau}_{\mathrm{final}}, \tau^\star\big),$$

   the standard ADE of the final action relative to ground truth.

We evaluate `LCDrive` using GT LWM and compare the result with and without RL, and show the result in Tab. 3. We

Method	Input (ms)	Reason (ms)	Action (ms)	Total (ms)	Tokens	Reason tok/s	VRAM (MB)
Text CoT (AR1)	1045.9	2446.6	380.2	3872.7	72	29.0	22931
`LCDrive`	1053.6	1388.1	372.3	2814.1	42	29.5	22931
LCDrive-Light	1054.5	754.8	380.2	2189.5	24	30.5	22931
No CoT	1052.7	0.0	379.8	1432.5	0	–	22931

Table 1. **Detailed wall-clock inference benchmark** on an RTX A5000 (24 GB) with identical decoding settings and batch size 1. The backbone, input pipeline, and final action decoder are shared; the speedup comes from the shorter reasoning phase.

Stage	Hardware	Wall-clock	GPU-hours
Stage 0	AR1 checkpoint	–	–
Stage 1	8×A100 80GB	34 h	272
Stage 2	8×A100 80GB	46 h	369
Total	–	80 h	641

Table 2. **LCDrive-specific training compute.** The total excludes the pre-trained AR1 checkpoint used for initialization.

Metric	No RL	With RL
Reasoning Diversity	0.412	0.353
Reasoning–Action Alignment	0.614	0.581
Reasoning Quality	0.976	0.961
Final-Action Quality	0.784	0.749

Table 3. Reasoning action analysis of `LCDrive` with/without RL training, using GT LWM. All values are ADE (m).

summarize two key aspects of the reasoning behavior: (i) how latent reasoning behaves in general, and (ii) how reinforcement learning further improves it. Together, these results reveal the functional role of latent chain-of-thought reasoning in `LCDrive`. We have the following observations:

1) Final actions improve upon the reasoning proposals. In both settings, we observe that Final-Action Quality $<$ Reasoning Quality. This means that even though the reasoning branches provide two candidate future plans, the decoder does not simply copy a branch. Instead, it selects the more promising proposal and further *refines* it to produce a more accurate final trajectory. This refinement effect becomes even stronger after RL.

2) Strong alignment between reasoning proposals and the final action. Across both models, the Reasoning–Action Alignment score remains small, indicating that the final trajectory lies close to at least one of the proposal branches. This shows that the proposal actions are actively used. After RL, the alignment improves ($0.614 \rightarrow 0.581$), indicating that RL strengthens the integration between proposals and the final action. Note that the Reasoning-Action Alignment score is consistently lower than the Reasoning Quality score. This means that the final action lies *closer to one of the reasoning proposals* than either proposal lies to the ground truth. Thus, the final plan is strongly aligned with the latent reasoning process, showing that `LCDrive` relies on and refines the reasoning rollouts when producing its final trajectory.

3) Reasoning branches maintain meaningful diversity. The Diversity score for both models indicates the two branches represent distinct motion hypotheses. This is essential in multi-agent driving scenarios with inherent uncertainty. RL slightly reduces diversity ($0.412 \rightarrow 0.353$), but the branches remain significantly different. In other words, RL makes exploration more targeted towards better proposal quality ($0.976 \rightarrow 0.961$).

Overall, we find that the final action trajectory is tightly aligned with the latent reasoning proposals, yet still achieves clearly lower ADE to the ground truth than the proposals themselves, showing that the model both uses and refines the proposed futures. Compared to the latent CoT model without RL, RL post-training further reduces both proposal and final-action errors and strengthens the alignment between proposals and the final decision.

D. Inference Efficiency Study

D.1. Ablation Study on Reasoning Depth

In this section, we study the trade-off between the reasoning token budget and trajectory accuracy by varying the reasoning depth K and branch factor B of `LCDrive` (GT LWM, Non-RL). For each variant, we construct the CoT supervision target in Stage 1 CoT Cold start stage with different settings of K and B. Then, we train the model with teacher forcing with different reasoning depths and branch factors, keeping all other components and hyperparameters fixed across runs. Importantly, we *do not* apply RL fine-tuning and we *do not* use predicted LWM tokens in this study, since our goal here is to quantify the tradeoff of reasoning cost and final action performance of latent CoT.

We then evaluate each model on the validation dataset and compare the performance in Fig. 1. We also compare them with the non-reasoning baseline (LWM_0-only with GT LWM). The horizontal axis plots the number of reasoning tokens generated per input clip, and the vertical axis shows the resulting ADE (lower is better). We have the following observations:

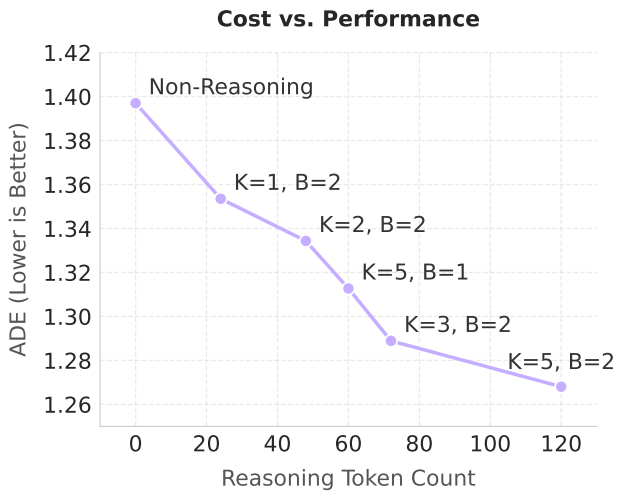

Figure 1. **Efficiency Curve**. We train different variants of `LCDrive` with different reasoning depth K and branch factor B.

1) Latent CoT provides consistent improvements over the baseline The leftmost point corresponds to the non-reasoning model. Introducing even a minimal amount of latent reasoning (e.g., K=1, B=2 with 24 tokens) produces a clear reduction in ADE. This demonstrates that a small number of interleaved action-proposal and latent world-model tokens already provides useful counterfactual context for the final trajectory prediction.

2) Increasing reasoning budget yields meaningful gains As we increase (K, B), performance improves smoothly, indicating that deeper latent reasoning enables the model to explore more steps into the future and produce better action plans based on that. The largest gains are obtained when moving from shallow reasoning (e.g., K=1, 2) to larger reasoning depth (K=3–5). Beyond this range, improvements are smaller but still positive, showing that `LCDrive` remains effective with different levels of token budgets.

3) Branching (B) leads to complementary improvements to depth (K) Branches encourage diverse counterfactual futures. Models with multiple branches (e.g., K=5, B=2) outperform the one with the same depth but fewer branches (e.g., K=5, B=1). This aligns with our diversity analysis: exploring alternative counterfactual futures provides richer reasoning signals for the final policy.

Overall, this curve indicate that latent reasoning offers a highly effective cost-performance tradeoff: a modest reasoning budget (120 tokens) achieves strong trajectory accuracy while remaining relatively cheap. These results demonstrate that `LCDrive` can flexibly trade inference cost for planning quality. Even lightweight latent CoT substantially enhances the end-to-end driving performance.

D.2. Inference Cost Analysis

We next compare the inference cost of latent chain-of-thought (Latent CoT) reasoning in `LCDrive` with a text-based CoT baseline.

Latent CoT inference cost. In `LCDrive`, each reasoning step $k \in \{1, \ldots, K\}$ simulates a 1.0 s future window and produces: (i) 10 discrete action tokens (representing the ego trajectory at 10 Hz), and (ii) 2 latent world model (LWM) tokens. For a model with reasoning depth K and branch factor B, the total number of latent reasoning tokens is therefore

$$N_{\text{latent}} \approx (10 + 2) \times K \times B,$$

plus a small constant overhead for the special tokens. At inference time, the inference cost of latent reasoning scales linearly with N_{latent}.

Text CoT baseline cost. For comparison, we tokenize the text CoT reasoning produced by the text-CoT baseline and compute the statistics over the validation dataset. Over this dataset we obtain an average length of 71.8 tokens, a 75-th percentile of 80 tokens, and a long tail up to 252 tokens per clip. Thus, a typical text-CoT explanation requires on the order of 70–80 additional tokens at inference time.

From the cost–performance curve in Fig. 1, we find that `LCDrive` already achieves *significant* improvements over the non-reasoning baseline using only a small, fixed latent budget of roughly 20–60 tokens (e.g., shallow configurations such as $(K, B) = (1, 2)$, $(2, 2)$, or $(3, 2)$). These settings use comparable or fewer tokens than typical text CoT, showing that compact latent reasoning is very cost-effective. As we increase the latent reasoning depth and branch factor, the model consistently achieves better trajectory accuracy, and remains *superior* to the text-CoT baseline (as shown in Table 1 of our paper) when using similar total tokens. This suggests that latent world-model rollouts provide more actionable planning signal per token than free-form natural language reasoning.

Potential for further latent reasoning. Our current action tokenizer produces 10 tokens per second of motion. A promising next step is to design a more aggressive motion tokenizer (e.g., fewer tokens per second or multi-step primitives), which would *linearly* reduce the latent reasoning token count for a fixed (K, B). Because these tokens are structured and low-entropy compared to text, they are much easier to compress than natural-language CoT, indicating significant room for future latency and cost reductions while preserving the benefits of latent reasoning.

References

[1] NVIDIA, Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, Liang Feng, Greg Heinrich, Jack Huang, Peter Karkus, Boyi Li, Pinyi Li, Tsung-Yi Lin, Dongran Liu,

Ming-Yu Liu, Langechuan Liu, Zhijian Liu, Jason Lu, Yunxiang Mao, Pavlo Molchanov, Lindsey Pavao, Zhenghao Peng, Mike Ranzinger, Ed Schmerling, Shida Shen, Yunfei Shi, Sarah Tariq, Ran Tian, Tilman Wekel, Xinshuo Weng, Tianjun Xiao, Eric Yang, Xiaodong Yang, Yurong You, Xiaohui Zeng, Wenyuan Zhang, Boris Ivanovic, and Marco Pavone. Alpamayo-R1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail. *arXiv preprint arXiv:2511.00088*, 2025. 2