2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Fast ECoT: Efficient Embodied Chain-of-Thought via Thoughts Reuse

Zhekai Duan[1], Yuan Zhang[2], Shikai Geng[1], Gaowen Liu[3], Joschka Boedecker[2], Chris Xiaoxuan Lu*[1]

***Abstract*— Embodied Chain-of-Thought (ECoT) reasoning enhances vision-language-action (VLA) models by improving performance and interpretability through intermediate reasoning steps. However, its sequential autoregressive token generation introduces significant inference latency, limiting real-time deployment. We propose Fast ECoT, an inference-time acceleration method that exploits the structured and repetitive nature of ECoT to (1) cache and reuse high-level reasoning across timesteps and (2) parallelise the generation of modular reasoning steps. Additionally, we introduce an asynchronous scheduler that decouples reasoning from action decoding, further boosting responsiveness. Fast ECoT requires no model changes or additional training and easily integrates into existing VLA pipelines. Experiments in both simulation (LIBERO) and real-world robot tasks show up to a 7.5× reduction in latency with comparable or improved task success rate and reasoning faithfulness, bringing ECoT policies closer to practical real-time deployment. Code is available at `https://github.com/kevinDuan1/Fast-ECoT`.**

I. INTRODUCTION

Large-scale vision-language-action (VLA) models have recently advanced robotic control by leveraging internet-scale visual and textual knowledge [1], [2], [3]. By combining pre-trained vision-language backbones with policy learning, these models exhibit impressive generalisation across open-world tasks. Among their most powerful capabilities is *chain-of-thought* (CoT) reasoning—the ability to iteratively generate intermediate reasoning steps before taking an action. In robotics, *Embodied Chain-of-Thought* (ECoT) [4] extends this concept by enabling robots to "think out loud"—generating step-by-step textual reasoning traces (e.g., plans, subgoals, grounded visual inferences) at each time step, explicitly encoding the robot's thought process before emitting an action. This augmentation improves model interpretability and boosts success rates.

While ECoT offers these benefits, they come at a steep computational cost. Generating reasoning traces involves producing dozens of tokens *autoregressively* at each time step, resulting in sequential generation delays. As tasks grow more complex, the length of these reasoning chains increases, compounding the latency. In real-time robotic control, policies must react quickly to new observations; the inference overhead introduced by ECoT can slow the control loop to impractical speeds. In other words, the robot idles much of its time "thinking" rather than acting. Reducing this latency without sacrificing reasoning quality or task performance is essential for making ECoT viable in real-world deployments.

* Corresponding author. Email: xiaoxuan.lu@ucl.ac.uk
[1] Department of Computer Science, University College London, UK
[2] Department of Computer Science, University of Freiburg, Germany
[3] Cisco Research, USA

In this work, we address this inference bottleneck by proposing `Fast ECoT`, a novel method for accelerating embodied chain-of-thought reasoning through thought reuse and parallelised reasoning. Our key insight is that ECoT outputs structurally and exhibits a high degree of temporal locality: many reasoning steps—such as recalling the task goal or rechecking the state of a target object—are repeated across time steps. Rather than regenerating the full reasoning trace at every step, we identify and cache recurring reasoning segments, reusing them in subsequent time steps. This reduces redundant computation while preserving the structure and interpretability of the thought process.

Building on this reuse, we introduce a partial parallelisation strategy that transforms the traditionally sequential reasoning process into a batched one. By branching only when necessary to handle novel information, Fast ECoT enables multiple reasoning steps to be generated in parallel—substantially reducing inference latency. We further introduce an *asynchronous scheduling* mechanism that decouples action and reasoning generation. Recognising that robot actions evolve faster than reasoning traces, we prioritise fast action decoding while allowing reasoning traces to update asynchronously in the background. This design reduces latency without compromising decision quality, as the cached reasoning remains stable over short time horizons.

II. RELATED WORK

Foundation Models for Robotic Manipulation. Recent advances in robot learning have produced large-scale generalist policies that excel at robotic manipulation tasks, by first pre-training on diverse multimodal datasets [1], [2], [3], and further fine-tuning on extensive robot-specific data collections [5], [6]. Vision-language-action (VLA) architectures [1], [2], [3]—which integrate vision-language models pretrained on internet-scale corpora [7], [8]—unify perception, language, and control into a single transformer-based policy, achieving state-of-the-art performance.

Current VLA methods can be broadly categorized into two families: (1) *monolithic models*, which integrate perception, language, and action within either single- or dual-system architectures [1], [2], [3]; and (2) *hierarchical models*, which explicitly decouple planning from policy execution by producing interpretable intermediate representations such as subtasks, keypoints, or programs [9], [10]. While monolithic approaches highlight simplicity and end-to-end generalisation, hierarchical designs emphasise interpretability and modularity for long-horizon control.

Reasoning for Robotic Control. Chain-of-thought (CoT) prompting [11], [12] has proven effective in enhancing large

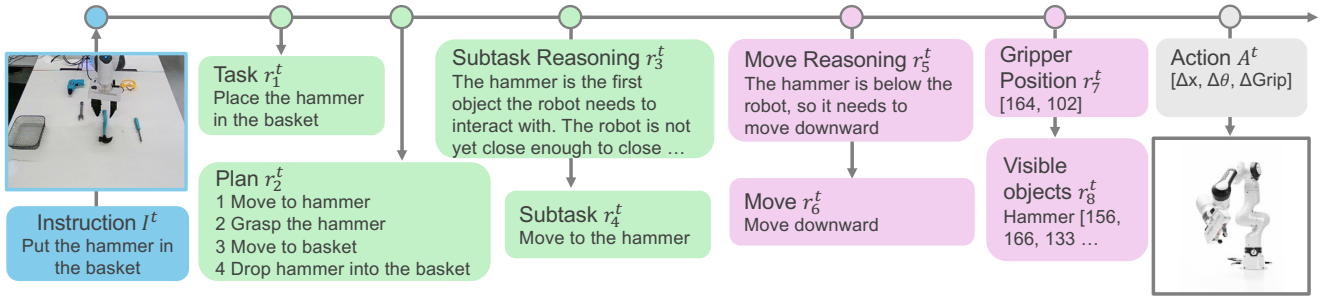


Fig. 1: ECoT [4] reasoning autoregressively generates high-level **(green)** and low-level **(purple)** reasoning steps to enhance VLA performance.

language models by encouraging step-by-step reasoning. Prior methods employ pre-trained language models for high-level planning [13], [14], [15], often requiring separate low-level controllers for execution. Recent work begins to integrate explicit reasoning end-to-end: ECoT introduces embodied chain-of-thought traces grounded in observations [4], CoT-VLA predicts visual subgoal observations [16], and RAD [17] and ThinkAct [18] curate or align language reasoning for low-level action; however, these often rely on latent embeddings, generated sub-goals, or textual descriptions that are hard to precisely ground for manipulation. EMMAX [19] embeds reasoning as subtasks and predicted gripper states (2D/3D), but leverages limited full-scene context. In contrast, MolmoAct [20] performs "reasoning in space," explicitly grounding each step in the scene so it can be decoded and visualised on the image plane and within the 3D environment, improving explainability and action prediction. Orthogonal to reasoning design, efficiency-oriented VLAs such as Spec-VLA [21] and FlashVLA [22] boost responsiveness by replacing strictly sequential decoding with speculative, parallel, or retraining-free acceleration. Our work builds on the ECoT but targets its chief bottleneck—autoregressive latency—via a partially parallelised reasoning framework that preserves interpretability while substantially reducing inference time.

Inference Optimisation in Language and Multimodal Models. A wide range of techniques has been proposed to speed up inference in autoregressive models. Speculative decoding [23], [24] accelerates generation by predicting tokens with a lightweight draft model and verifying them with the full model. Non-autoregressive and parallel decoding strategies have also been adopted, particularly in machine translation and, more recently, in robotic control [9], [25]. Quantisation methods [26], [27], [28] are widely used to reduce model precision for faster computation. In contrast to these methods, which typically operate at the token or model level, our work focuses on *reasoning-level acceleration*—breaking CoT generation into reusable, semantically meaningful segments that can be cached and executed in parallel. This approach is model-agnostic and does not rely on auxiliary models, fine-tuning, or precision reduction. By enabling efficient reuse and branching during reasoning, our method offers a lightweight and integrable solution for speeding up VLA policies with CoT reasoning, without compromising interpretability or task success.

III. PRELIMINARIES

A. Embodied Chain-of-Thought Reasoning (ECoT)

Vision-Language-Action (VLA) models [1] build on large pre-trained vision-language models, fine-tuning them to map a natural language instruction I^t and image observation O^t directly to low-level robot actions A^t via autoregressive token prediction at each control step t. Embodied Chain-of-Thought (ECoT) [4] reasoning augments this reactive paradigm by supervising the model to generate a structured sequence of N intermediate reasoning steps $R^t = \{r_i^t \mid i = 1, 2, \ldots, N\}$, e.g. rephrased task, high-level plan, grounded sub-task, low-level move command, visual features, before emitting the final action A^t. The reasoning steps are separated into high- and low-level steps (see Fig. 1).

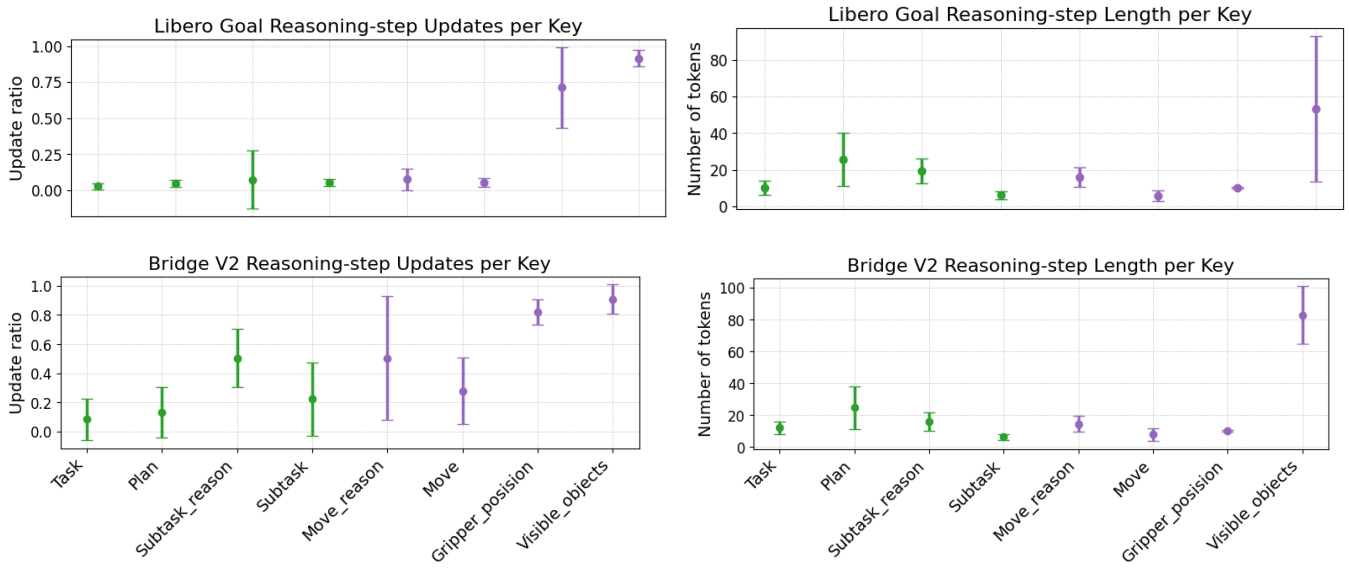
B. Continuous Batching

In autoregressive models, batching improves throughput, but static batching can be inefficient when sequence lengths vary—shorter sequences must wait for longer ones, wasting compute on padding. Continuous batching [29] addresses this by dynamically replacing completed sequences with new ones, maintaining high GPU utilization and minimising idle time. This strategy, adopted by the work [30], [31], has shown 2–4× throughput gains in LLM serving.

IV. METHOD

A. Inference Characteristics of ECoT Reasoning

Unlike traditional chain-of-thought approaches in language modelling [33], [34] and vision-language modelling [35], [36], which yield diverse, dynamic reasoning patterns that rarely *recur*, ECoT consistently follows a structured, periodically recurring workflow: planning, sub-task identification, motion reasoning, and visual-feature processing, before predicting subsequent robot actions. To analyse this behaviour quantitatively, we sample all episodes containing reasoning from the Bridge V2 dataset [5] and compute the average and the standard deviation of the update ratio (percentage of reasoning content updated at the next time



(a) Mean update ratio (± 1 standard deviation) per reasoning step, showing high-level stability versus frequent low-level updates.

(b) Mean length (± 1 standard deviation) of different reasoning steps, showing significant disparities in token counts.

Fig. 2: Statistics illustrating the pattern of ECoT reasoning steps under Libero Goal [32] and Bridge V2 [5].

step) and token length for each reasoning step. As depicted in Fig. 2a, higher-level reasoning components in ECoT (e.g., planning and subtask reasoning) remain relatively similar across multiple time steps within an episode[1]. For example, the planning module exhibits an average update ratio of only 8.4%, meaning 91.6% of its reasoning content is unchanged and can be reused in subsequent inference steps without autoregressive regeneration. Leveraging this temporal locality, we propose to cache ECoT reasoning for reuse in successive time steps, thus potentially enabling the parallel generation of each reasoning step. We observe a similar trend in the simulated environment, LIBERO-Goal [32]: high-level reasoning updates rarely while low-level components refresh more frequently (see Fig. 2); This consistency supports caching high-level reasoning and densely updating low-level content.

B. Parallelising Reasoning and Action Generation

Compared to the original ECoT—which generates the full reasoning sequence sequentially and autoregressively at every timestep (see Fig. 3 left)—our Fast ECoT reformulates each reasoning step r_n^t as a standalone generation task. For each step, we construct the input by combining the current observation O^t, instruction I^t, and the previously generated reasoning steps $R^{t-1} = \{r_i^{t-1} \mid i = 1, 2, \ldots, n-1\}$ from the last timestep as prefix context. This allows all reasoning steps and the action at timestep t to be generated independently and in parallel, rather than waiting for preceding components to finish (see Fig. 3 right).

[1]While profiling the update ratios, we observed substantial variability in the visible detection module. This counterintuitive result stems from the instability inherent in the visual reasoning module of ECoT and the open-vocabulary nature of object labels, whereby identical objects may be assigned varying labels over time.

While conceptually straightforward, this strategy introduces performance overhead. Since each reasoning module prepends a growing context of prior thoughts, the resulting input lengths vary significantly—early steps have short prompts, while later ones accumulate more history. Additionally, output lengths also differ across reasoning steps: low-level steps like gripper commands may require fewer than 20 tokens, while object-grounding reasoning can exceed 120 (see Fig. 2b). Traditional static batching [29] (see Fig. 4) handles such variability by padding all sequences in a batch to match the longest, which leads to substantial inefficiency on modern accelerators. This results in wasted compute on padding tokens, poor GPU utilisation, and offsets the gains from parallel generation.

To address inefficiencies from padding, we adopt continuous batching [29], a dynamic scheduling strategy used in modern LLM serving engines [30], [31]. Instead of padding all sequences to a fixed length, continuous batching maintains a queue where completed sequences are immediately replaced by new ones, allowing variable-length inputs to be processed efficiently. This minimises wasted computation on padding tokens, significantly improves GPU utilisation, and can reduce the total number of tokens processed by up to 6× (see Fig. 4). We use vLLM [31] as our inference backend, and the pseudocode of `Fast ECoT` is shown in Alg. 1.

C. Asynchronous Reasoning and Action Updates

So far we formulate reasoning and action generation at each time step as batched requests that can be processed in parallel. However, reasoning traces typically span hundreds of tokens, while action decoding involves only a few (around 7) tokens. In a synchronised setup, this mismatch causes unnecessary latency: the agent must wait for all reasoning steps to complete before it can act (see Fig. 6).

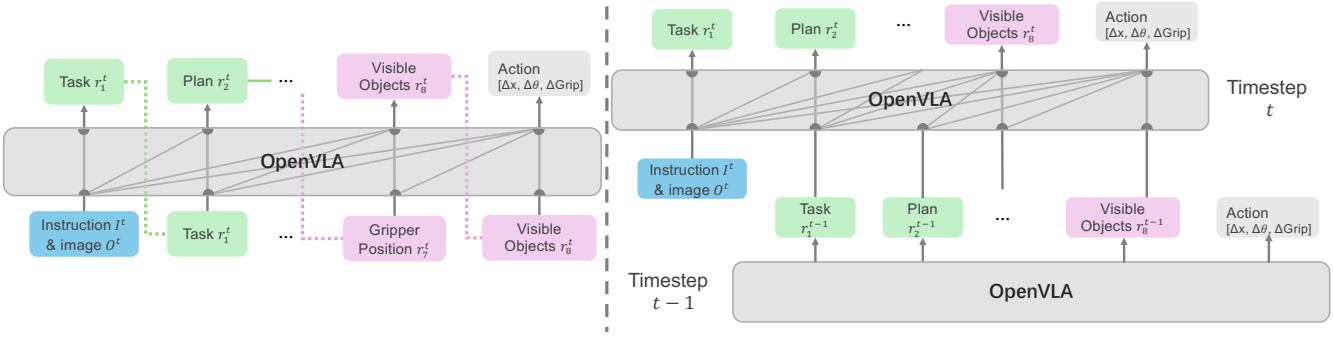


Fig. 3: Comparison between ECoT **(left)** and our proposed `Fast ECoT` **(right)**. Both decompose reasoning into fixed stages (e.g., task, plan, object grounding), but ECoT generates these sequentially at every step, while `Fast ECoT` enables parallel generation and reuses cached higher-level reasoning from previous timesteps as context. The dotted lines coloured in green/magenta represent token copying.

Algorithm 1 Parallel Embodied Chain-of-Thought

Require: Time step t, Instruction I^t, Observation O^t, Last reasoning steps R^{t-1}

$c^t \leftarrow \text{encode}(I^t, O^t)$
$R^t \leftarrow [\,]$
for $i = 1$ **to** $N+1$ **concurrently do**
    Autoregressively sample $r_i^t \sim P\big(r_i^t \mid c^t, R^{t-1}[:i]\big)$
end for
Synchronize
$R^t, A^t \leftarrow \text{decode}(R^t[:-1], R^t[-1])$
return R^t, A^t

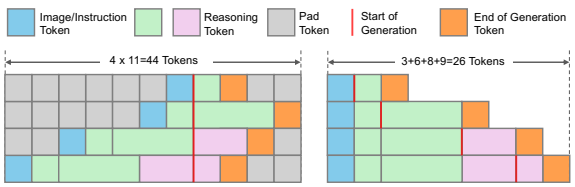


Fig. 4: Illustratively comparing static vs. continuous batching for reasoning generation. **Left:** Static batching pads to the longest sequence, processing 4×11=44 tokens. **Right:** Continuous batching processes only actual tokens (3, 6, 8, 9), adding up to 26 tokens, which reduces padding and improves efficiency.

While ECoT couples reasoning and action for interpretability and causal grounding, this coupling need not be *time-synchronous*. Empirically, high-level elements (e.g., TASK, PLAN) change slowly across steps; their influence on actions persists even with infrequent updates—see rollouts in Fig. 5, where reasoning traces and visible objects remain stable as actions evolve. Inspired by VLA systems that separate a fast action module from a slower VLM (SmolVLA [37]; GR00T [38]), we adopt an asynchronous split: the controller decodes actions from the current observation O^t and a cached high-level reasoning R^c, while R^c is refreshed in the background. As shown in Fig. 6, asynchronous reasoning action generation effectively reduces stalls and increases action throughput without sacrificing interpretability. Full pseudocode of the above is shown in Alg. 2.

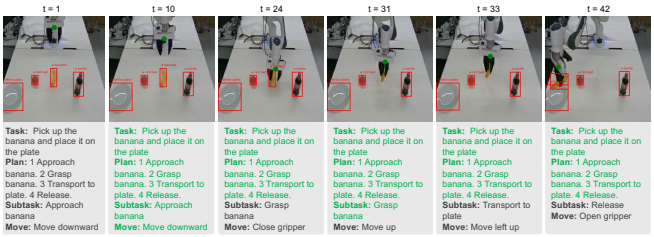


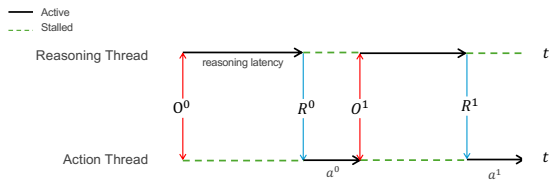
Fig. 5: Generated robot rollouts at successive time steps **(top row)** alongside its reasoning **(bottom row)**. Between frames, a large part of the reasoning remains unchanged **(Green)**. At each timestep (t=1, 10, 24, 31, 33, 42), the Subtask updates intermittently, and the low-level Move command adapts continuously as it picks up the banana and places it on the plate.

V. RESULTS

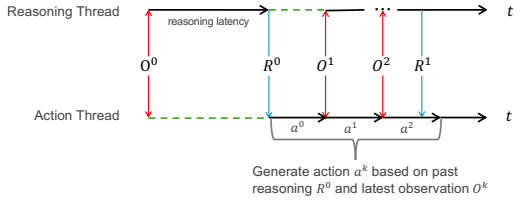
In this section, we conduct experiments to evaluate the effectiveness of `Fast ECoT`. Our evaluation focuses on the following key questions: (1) To what extent does parallel reasoning generation improve computational efficiency? (2) Does the proposed method preserve task performance comparable to the sequential baseline? (3) What is the impact of reasoning step parallelization on the overall quality of reasoning?

A. LIBERO Experiments

Task Setup. We conduct experiments using a Franka Emika Panda robotic arm within the LIBERO environment [32], a widely adopted benchmark for evaluating generalizable robotic policies. To comprehensively assess policy



(a) Synchronous inference.



(b) Asynchronous inference.

Fig. 6: Illustration of inference stacks. Asynchronous overlap reduces stall time (green dashed) of robot operation and increases action throughput within the same time window compared to synchronous execution.

Algorithm 2 Asynchronous Parallel Embodied Chain-of-Thought

Require: Time step t, Instruction I^t, Observation O^t, History Reasoning R^c

$c^t \leftarrow \text{encode}(I^t, O^t)$
for $i = 1$ **to** $N + 1$ **concurrently do**
  Lock R^c and autoregressively sample $r_i^t \sim P\big(r_i^t \mid c^t, R^c[:i]\big)$
  if $i = N + 1$ **then**
    $A^t = r_i^t$
  else
    Lock R^c and update R^c with r_i^t
  end if
end for
wait until A^t is finished
$A^t \leftarrow \text{decode}(A^t)$
return R^c, A^t

generalisation, we select four diverse task suites—LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long—each targeting distinct challenges, including spatial configuration, object manipulation, goal specification, and long-horizon task execution.

Training Data and Training Recipe. ECoT models require fine-tuning on each task for optimal performance [1], [4]. Training data is generated following the ECoT pipeline [4], with modifications for reproduction: we integrate Janus [39] for automated scene description and Deepseek-Reasoner [34] for generating CoT reasoning trajectories from successful demonstrations. We initialise ECoT models from the open-sourced checkpoint [4], which was pre-trained on Bridge V2 [5] and OXE dataset [6]. Then we apply LoRA [40] with rank 32 and train for $200,000$ gradient steps using a batch size of 1 distributed across 4 NVIDIA A6000 GPUs.

Baselines. We compare our proposed method, `Fast ECoT`, against four baselines: ECoT, the original ECoT model that autoregressively generates the full reasoning chain; ECoT (5-step), a variant that updates low-level reasoning at every timestep but updates high-level reasoning only every 5 timesteps; ECoT (async), a variant originally designed to use two GPUs to asynchronously compute high-level and low-level reasoning, which we adapt to run entirely on a single GPU for fair comparison; and ECoT (Quant), a post-training quantized version of ECoT utilizing Huggingface's Bitsandbytes [41], selected due to its best acceleration performance compared to other quantization approaches tested [26], [27]. For fairness, we exclude methods requiring additional training or architectural changes, such as speculative decoding [23], [24], and action chunking [25], [42].

Inference Speedup. As shown in the last column of Tab. I, OpenVLA achieves the lowest latency (184 ± 37 ms) since it directly maps observations to actions without generating intermediate reasoning. In contrast, reasoning-based models such as ECoT incur substantially higher costs. Our method substantially narrows this gap: `Fast ECoT` reduces latency to 2156 ± 353 ms, a $2.3\times$ speedup over ECoT (4997 ± 691 ms), while `Fast ECoT (Async)` further reduces it to 686 ± 412 ms (nearly $7\times$ faster), all while retaining reasoning capability. This demonstrates that parallel reasoning effectively amortises the overhead of structured reasoning, bringing latency closer to non-reasoning policies.

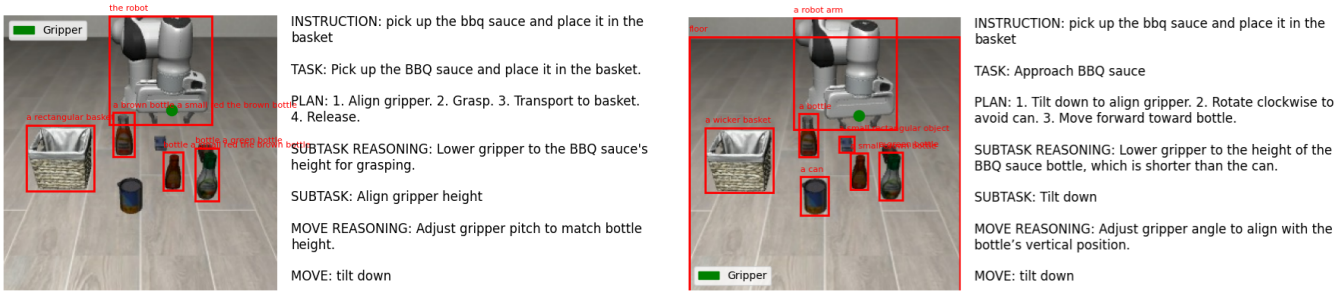
Performance. Our method consistently improves task performance relative to prior baselines. `Fast ECoT` achieves the highest average success rate (80.0%), surpassing all ECoT variants and non-reasoning OpenVLA. We attribute this to our parallel reasoning strategy, which smooths temporal inconsistencies and leverages prior reasoning steps. `Fast ECoT (Async)` remains competitive (77.5%) and achieves the best result on LIBERO-Long (69%), though at some cost in LIBERO-Object due to greater sensitivity to object layouts and temporal mismatch. Interestingly, ECoT (Quant) also improves over vanilla ECoT, likely due to quantisation regularising model behaviour [43]. Fig. 7 shows that both ECoT and `Fast ECoT (Async)` produce coherent, grounded reasoning, but the latter reaches comparable states much quicker.

B. Real-world Experiments

Setup. We validate `Fast ECoT` on a physical Franka Emika Panda robotic arm equipped with a RealSense D455 camera, providing third-person RGB-D observations. We design six manipulation tasks representative of common household scenarios. The evaluation includes both in-distribution and out-of-distribution tasks featuring unseen objects and instructions.We collect 50 expert demonstrations via Droid teleoperation pipeline [44] to fine-tune ECoT models. We follow the same training data generation process, training recipe, and baselines used in Sec. V-A.

Method	LIBERO-Object SR (%) ↑	LIBERO-Spatial SR (%) ↑	LIBERO-Goal SR (%) ↑	LIBERO-Long SR (%) ↑	Average SR (%) ↑	Latency per Step (ms) ↓
OpenVLA	**87**	83	74	55	75.3	**184 ± 37**
ECoT	77	84	75	57	73.3	4997 ± 691
ECoT (5-step)	79	75	72	56	70.5	3514 ± 969
ECoT (Async)	70	83	80	47	70.0	3655 ± 773
ECoT (Quant)	82	82	**84**	57	76.3	2180 ± 207
`Fast ECoT`	83	**85**	83	**69**	**80.0**	2156 ± 353
`Fast ECoT (Async)`	75	83	83	**69**	77.5	686 ± 412

TABLE I: LIBERO simulation experimental results. SR = Success Rate.



(a) ECoT at $t = 143.7$s.

(b) `Fast ECoT (Async)` at $t = 21.4$s.

Fig. 7: Qualitative examples in LIBERO experiments. `Fast ECoT (Async)` reaches comparable task states much quicker.

Method	ID SR (%) ↑	OOD Objects SR (%) ↑	OOD Instruction SR (%) ↑	Average SR (%) ↑	Latency per Step (ms) ↓
OpenVLA	**83.3**	68.3	42.7	64.7	196 ± 32
ECoT	78.3	73.3	40.0	64.0	5556 ± 384
ECoT (5-step)	56.6	51.7	35.0	47.8	4327 ± 619
ECoT (Async)	76.6	70.0	41.7	62.8	4206 ± 323
ECoT (Quant)	75.0	63.3	48.3	62.2	2437 ± 171
`Fast ECoT`	81.6	73.3	**50.0**	**68.3**	2479 ± 520
`Fast ECoT (Async)`	78.3	**75.0**	42.7	65.3	**716 ± 529**

TABLE II: Real-world experimental results on selected household tasks. SR = Success Rate. ID = In distribution. OOD = Out of distribution.

Results. Tab. II summarises the performance of our method against baseline approaches on real-world manipulation tasks. `Fast ECoT` achieves the highest overall success rate (68.3%), with strong improvements in OOD instruction tasks (50.0%, +10 points over OpenVLA). Meanwhile, the Async version of `Fast ECoT` yields the best trade-off between performance and efficiency: although its average success rate is slightly lower (65.3%), it delivers the lowest latency of 716 ± 529 ms per step—a $7.7\times$ speedup compared to the original ECoT baseline (5556 ± 384 ms). This highlights the benefit of asynchronous parallel reasoning in real-world deployment, where rapid, low-variance inference is critical. In contrast, prior ECoT variants either sacrifice task performance (ECoT 5-step) or still suffer from high inference latency (ECoT and ECoT-Async), making them less practical for on-robot use. Consistent with the simulation examples, Fig. 8 shows that both methods produce coherent, grounded reasoning; again, the asynchronous variant reaches comparable task states much quicker.

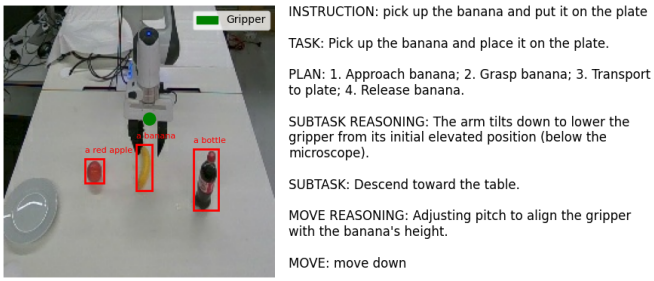
C. AutoEval Real-world Experiments

Setup. We use AutoEval [45], an online platform for standardised real-robot policy evaluation. We choose AutoEval both for reproducible, third-party benchmarking and because its tasks/environments are drawn from BridgeData V2 [5]—the same distribution on which ECoT [4] was trained. This lets us run the *vanilla* ECoT *without any finetuning* and directly apply our accelerations (`Fast ECoT` and `Fast ECoT (Async)`) as drop-in replacements. Concretely, we evaluate four tabletop tasks in two accessible environment (*Drawer*, *Sink*) with 10 trials per task.

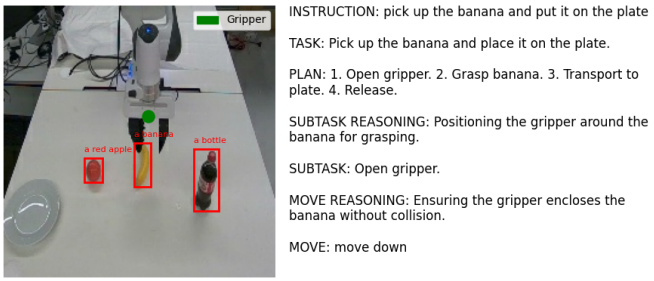
Method	Drawer SR (%) ↑	Sink SR (%) ↑	Latency (ms) ↓
ECoT	**30**	30	4030 ± 270
`Fast ECoT`	**30**	**35**	2105 ± 324
`Fast ECoT (Async)`	20	25	**790 ± 331**

TABLE III: AutoEval results on BridgeData V2 tabletop tasks. SR = Success Rate.

Results. `Fast ECoT` and `Fast ECoT (Async)` achieve performance comparable to ECoT under AutoEval



(a) ECoT at $t = 84.8$s.



(b) `Fast ECoT (Async)` at $t = 18.7$s.

Fig. 8: Real-world qualitative examples. `Fast ECoT (Async)` reaches comparable task states much quicker.

(Tab. III): `Fast ECoT` matches ECoT on *Drawer* (30%) and slightly exceeds it on *Sink* (35%), while the asynchronous variant is modestly lower (20–25%) but offers a favourable accuracy–efficiency trade-off. Note that AutoEval's environment reset policy introduces distribution shift—e.g., altered object poses/orientations, handle angles, and initial camera/scene configurations—relative to the canonical starts used in the original evaluation. These OOD initialisations make perception and short-horizon control more volatile, reducing absolute SRs across all methods (including ECoT) even though their relative behaviour remains similar.

D. Updating Frequency of Across-level Reasoning

Reusing past reasoning steps yields a temporal smoothing effect that dampens noisy fluctuations in planning and stabilises control. We perform a controlled ablation on LIBERO-Object (see Tab. IV) to examine performance differences resulting from varying the update frequency of high-level and low-level reasoning steps. When *high-level* reasoning is refreshed less frequently (every 5 frames), while *low-level* reasoning remains updated at every step, success rates increase from 77% (baseline with updates at every step) to 79%. In `Fast ECoT`, each reasoning step similarly leverages current visual observations combined with cached reasoning from previous steps, achieving an even higher success rate of 83%. **However, excessively infrequent updates reduce performance**. Updating both high-level and low-level reasoning every 5 frames lowers success to 70%, and completely infrequent updates (∞/∞) dramatically drop success to 35%. Common failures in such scenarios involve delayed task transitions—for instance, after grasping an object, the robot may stall indefinitely rather than moving promptly to the placement step. This underscores the critical balance between beneficial temporal smoothing and necessary action responsiveness.

Our findings (see Tab. IV) align with the results in SmolVLA [37] and Tab. 2 of the original ECoT work, where the reuse of historical reasoning similarly improved task performance. Thus, moderate reuse of recent reasoning can maintain or even enhance robot performance.

High-level step	Low-level step	Success (%) ↑
1	1	77
5	1	79
5	2	78
5	3	75
5	5	70
∞	∞	35
`Fast ECoT`	`Fast ECoT`	**83**

TABLE IV: Ablation on update frequency.

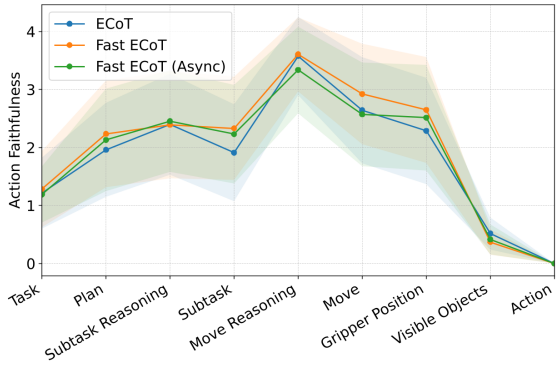


Fig. 9: Action faithfulness (AF) on LIBERO tasks. All graphs are plotted with mean and standard deviation shading across 1000 timesteps.

E. Action Faithfulness in ECoT Reasoning

To validate that the generated chain-of-thought (CoT) not only improves performance but also *faithfully explains* the model's decision-making process, we build on the faithfulness criteria for language models [46] and introduce a novel quantitative metric - *action faithfulness (AF)* - to measure the faithfulness of the CoT reasoning for robotic tasks. Formally, given a complete CoT consisting of N intermediate reasoning steps that culminate in a final robot action A, we enforce the model to *directly* predict an action A_i after generating only the first i reasoning steps, where $i = 0, 1, \ldots, N$. We then compute the L1 distance between A_i and A as the faithfulness score $AF_i = \|A_i - A\|_1$. Higher L1 distances indicate a greater dependence on subsequent reasoning steps, thereby suggesting higher CoT faithfulness.

We plot action faithfulness scores of `Fast ECoT` and ECoT for all reasoning steps $\{AF_i\}$ in Fig. 9. `Fast ECoT`

preserves the faithfulness of the base ECoT model during parallel reasoning generation for most reasoning steps. The action faithfulness without predicting "Visible Objects" is slightly lower for `Fast ECoT`, since the high update ratio and one-step delay in visual features might increase its chances to be post-hoc. Meanwhile, the asynchronous variant of `Fast ECoT` generally exhibits a lower faithfulness, probably due to the larger temporal mismatch introduced in async reasoning. Faithfulness is notably low when no reasoning steps are generated (i.e., at "Task"), likely due to fine-tuning from the OpenVLA [1] checkpoint, which did not require reasoning steps.

VI. CONCLUSION

We present `Fast ECoT`, an inference-time acceleration method for Embodied Chain-of-Thought (ECoT) reasoning in VLA models. By exploiting structural and temporal locality, `Fast ECoT` enables (1) reuse of cached high-level reasoning and (2) parallel generation of modular steps. An asynchronous scheduler further decouples reasoning from action decoding, enhancing real-time responsiveness. Requiring no architectural changes or retraining, `Fast ECoT` integrates easily into existing VLA pipelines. Across simulations and real-world tests, it reduces inference latency by up to 7.5× while maintaining task performance and interpretability, advancing ECoT toward real-time deployment.

REFERENCES

[1] M. J. Kim and e. a. Pertsch, "Openvla: An Open-Source Vision-Language-Action Model," in *8th Annual Conference on Robot Learning*, vol. abs/2406.09246, 2024.

[2] K. Black and N. B. et al., "π_0: A vision-language-action flow model for general robot control," 2024.

[3] NVIDIA, "Gr00t n1: An open foundation model for generalist humanoid robots," 2025.

[4] M. Zawalski and W. e. a. Chen, "Robotic Control via Embodied Chain-of-Thought Reasoning," in *8th Annual Conference on Robot Learning*, vol. abs/2407.08693, 2024.

[5] H. Walke and K. e. a. Black, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning (CoRL)*, 2023.

[6] A. O'Neill and A. R. et al., "Open x-embodiment: Robotic learning datasets and rt-x models," 2024.

[7] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[8] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh, "Prismatic vlms: Investigating the design space of visually-conditioned language models," 2024.

[9] W. Song, J. Chen, P. Ding, H. Zhao, W. Zhao, Z. Zhong, Z. Ge, J. Ma, and H. Li, "Accelerating vision-language-action model integrated with action chunking via parallel decoding," 2025.

[10] Y. Li and Y. D. et al., "Hamster: Hierarchical action models for open-world robot manipulation," 2025.

[11] J. Wei and X. W. et al., "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[12] X. Ning and Z. L. et al., "Skeleton-of-thought: Prompting llms for efficient parallel generation," 2024.

[13] M. Ahn and A. e. a. Brohan, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," in *Conference on Robot Learning*, 2022, pp. 287–318.

[14] A. Zeng and M. A. et al., "Socratic models: Composing zero-shot multimodal reasoning with language," 2022.

[15] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," 2023.

[16] Q. Zhao and Y. e. a. Lu, "Cot-vla: Visual chain-of-thought reasoning for vision-language-action models," *arXiv preprint arXiv:2503.22020*, 2025.

[17] H. Gao and S. C. et al., "Rad: Training an end-to-end driving policy via large-scale 3dgs-based reinforcement learning," 2025.

[18] C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang, "Thinkact: Vision-language-action reasoning via reinforced visual latent planning," 2025.

[19] Q. Sun, P. Hong, T. D. Pala, V. Toh, U.-X. Tan, D. Ghosal, and S. Poria, "Emma-x: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning," 2024.

[20] J. Lee and J. D. et al., "Molmoact: Action reasoning models that can reason in space," 2025.

[21] S. Wang, R. Yu, Z. Yuan, C. Yu, F. Gao, Y. Wang, and D. F. Wong, "Spec-vla: Speculative decoding for vision-language-action models with relaxed acceptance," 2025.

[22] X. Tan, Y. Yang, P. Ye, J. Zheng, B. Bai, X. Wang, J. Hao, and T. Chen, "Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models," 2025.

[23] Y. L. et al., "Fast inference from transformers via speculative decoding," 2023.

[24] S. Kim and K. M. et al., "Speculative decoding with big little decoder," 2023.

[25] M. J. Kim, C. Finn, and P. Liang, "Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success," *arXiv.org*, vol. abs/2502.19645, 2025.

[26] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," 2024.

[27] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and Efficient Post-Training Quantization for Large Language Models." in *International Conference on Machine Learning (ICML)*, 2023, pp. 38 087–38 099.

[28] huggingface, "Bitsandbytes," 2025. [Online]. Available: https://huggingface.co/docs/bitsandbytes/main/en/index

[29] G.-I. Yu and J. S. J. et al., "Orca: A distributed serving system for Transformer-Based generative models," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. Carlsbad, CA: USENIX Association, July 2022, pp. 521–538.

[30] Nvidia, "Tensorrt-llm." 2025.

[31] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," 2023.

[32] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *arXiv preprint arXiv:2306.03310*, 2023.

[33] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." in *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[34] DeepSeek-AI, D. Guo, and D. Y. et al., "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025.

[35] P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y. N. Wu, S.-C. Zhu, and J. Gao, "Chameleon: Plug-and-play compositional reasoning with large language models," 2023.

[36] G. Xu, P. Jin, H. Li, Y. Song, L. Sun, and L. Yuan, "Llava-cot: Let vision language models reason step-by-step," 2025.

[37] M. S. et al., "Smolvla: A vision-language-action model for affordable and efficient robotics," 2025.

[38] NVIDIA, :, and J. B. et al., "Gr00t n1: An open foundation model for generalist humanoid robots," 2025.

[39] X. Chen, Z. Wu, X. Liu, Z. Pan, W. Liu, Z. Xie, X. Yu, and C. Ruan, "Janus-pro: Unified multimodal understanding and generation with data and model scaling," *arXiv preprint arXiv:2501.17811*, 2025.

[40] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[41] huggingface, "Bitsandbytes," 2025.

[42] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023.

[43] S. K. et al., "QuaRL: Quantization for fast and environmentally sustainable reinforcement learning," *Transactions on Machine Learning Research*, 2022.

[44] A. Khazatsky and K. P. et al., "Droid: A large-scale in-the-wild robot manipulation dataset," 2025.

[45] Z. Zhou and P. A. et al., "Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world," 2025.

[46] Q. Lyu, S. Havaldar, A. Stein, L. Zhang, D. Rao, E. Wong, M. Apidianaki, and C. Callison-Burch, "Faithful chain-of-thought reasoning," *ArXiv*, vol. abs/2301.13379, 2023.