LATENT MEMORY PALACE: REASONING FOR CONTROL AS AUTOREGRESSIVE VARIATIONAL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Human decision-making is highly flexible – some actions are taken immediately; others require longer deliberation. Language models have exhibited a similar capacity for adaptive "reasoning." However, transferring this capability to continuous control policies has been challenging, as directly reasoning in language space may lack the granularity for spatial understanding and precise motions. In this work, we show that reasoning for control policies can emerge by organizing information in an autoregressive *latent* space reminiscent of a memory palace, where retrieval is iterative and adaptive. Our method, Latent Memory Palace (LMP), formulates reasoning as *variational inference with an autoregressive latent distribution*. We derive a latent-space reinforcement learning technique to tractably optimize its variational lower bound. The resulting policy, LMP-π, achieves strong empirical performance in simulation and real-world domains while exhibiting interpretable, adaptive allocation of test-time compute. We further show that the same framework yields a variable-length action tokenizer, LMP-`tok`, which significantly improves the performance of downstream autoregressive policies. Together, these results present a new perspective on latent reasoning for control through the lens of variational inference. Videos are available at: `https://latentmemorypalace.github.io/`

1 INTRODUCTION

Human decision-making ranges from the reflexive (e.g. avoiding obstacles) to the deliberate (e.g. playing chess), with variability in the time and effort devoted to each decision. This process is both *iterative* and *adaptive*, proceeding through a chain of steps that scales with the complexity of the decision. Modern large language models (LLMs) exhibit a similar pattern: a broad class of "reasoning" LLMs achieves substantially improved performance by generating intermediate tokens before the final answer (Wei et al., 2022; DeepSeek-AI, 2025; OpenAI, 2024).

We ask: can iterative, adaptive computation benefit sequential decision making problems such as robotics? For robotic policies, we posit that the benefits are twofold: efficient decision making under task variability, and improved generalization. However, directly reasoning in language space is unlikely to suffice, as language tokens may not capture the nuances required for spatial understanding and precise motions (Chen et al., 2024). We therefore propose that robotic policies reason iteratively in a *latent space* learned end-to-end for action prediction. This preserves the benefits of iterative, adaptive computation while allowing the intermediate representations to capture control-relevant information at the appropriate granularity.

To enable latent-space reasoning in robotic policies, we build on variational inference, a framework for learning latent-variable models by approximating posterior distributions over latent variables (Kingma & Welling, 2014). Applied to control, this perspective interprets an action not as a direct prediction from an observation, but as the outcome of an intermediate latent computation. However, standard variational inference typically uses fixed-size latent variables, unable to adapt their computation to each input. To address this, we propose Latent Memory Palace (LMP), a formulation of reasoning as *variational inference with an autoregressive latent distribution*. As shown in Fig. 1(a), LMP encodes control-relevant information in an autoregressive latent space, where an `EOS` token allows information to be represented as variable-length paths. This structure resembles the memory palace, a mnemonic where information is associated with locations along a spatial route, and later recalled

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

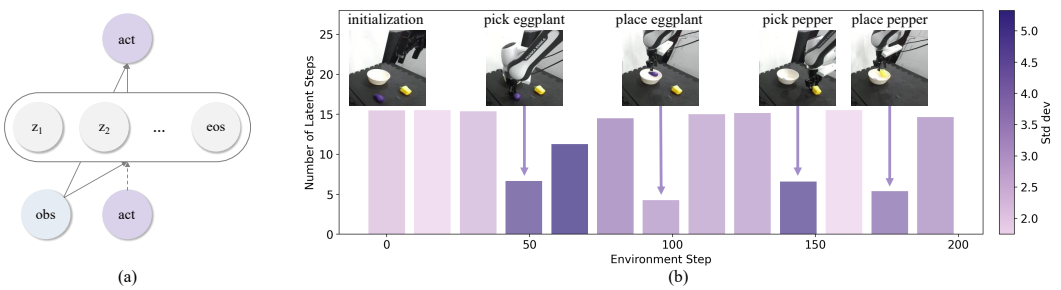


Figure 1: (a) Latent Memory Palace (LMP) formulates iterative and adaptive reasoning as variational inference with a variable-length autoregressive latent distribution. (b) Applying LMP to decision-making results in control policies that adaptively allocate their test-time computation.

by mentally traversing that route one location at a time. In LMP, the autoregressive latent space is the palace, each latent token a location along a path, and generating the latent sequence amounts to traversing that path to retrieve the control-relevant information. Crucially, the `EOS` token lets the policy walk only as far as needed, giving rise to the iterative, adaptive retrieval that motivates our approach.

As in standard variational inference, we learn the latent tokens by maximizing a variational lower bound of the maximum likelihood objective. This involves training a *posterior* to encode each observation–action pair into a sequence of latent tokens with two goals: (i) reconstructing the action, and (ii) keeping the posterior close to a *prior* conditioned on the observation alone. This prior, paired with the action decoder, precisely defines a policy LMP-π. At test time, it generates latent "reasoning" tokens from the observation and decodes them into the action, adaptively allocating computation to each decision (Fig. 1(b)). To tractably optimize through the variable-length autoregressive sampling process, we convert the variational lower bound into a reinforcement learning (RL) objective over latent trajectories and apply RL techniques. We further show that by dropping the observation conditioning, the same framework instantiates a variable-length sequential action tokenizer, LMP-`tok`, which compresses continuous actions into discrete latent tokens used as output space for downstream autoregressive policies.

Experimentally, LMP-π achieves strong performance across simulated and real-world robot learning tasks. Our analyses indicate that (i) variable-length latents outperform fixed-length latent-variable policies, (ii) additional latent steps let the policy express more behavioral modes, which is important for recovery behaviors, and (iii) parsimonious allocation of latents, i.e., compression, improves performance and results in fewer latent steps on actions such as gripper movements. The action tokenizer, LMP-`tok`, demonstrates substantial improvement over alternative tokenizers when evaluated with the same downstream autoregressive policy. Overall, these results establish autoregressive variational inference as a practical framework for iterative and adaptive computation in robot learning.

2 Related Work

Reasoning for Foundation Models: Reasoning has been shown to improve the output of foundation models via *iterative inference* and *adaptive test-time compute*. Chain-of-thought prompting improves accuracy in multi-step problems by generating intermediate steps (Wei et al., 2022; Yao et al., 2023). RL-based reasoning models elicit and amortize iterative behaviors into inference (OpenAI, 2024; DeepSeek-AI, 2025). Test-time scaling laws show that allocating additional test-time compute can systematically improve model performance (Snell et al., 2024; Ma et al., 2025). Latent reasoning shifts multi-step inference from explicit token-level traces to implicit representations, potentially increasing bandwidth and reducing latency (Zhu et al., 2025b). COCONUT progressively replaces explicit reasoning tokens with continuous embeddings via a training curriculum (Hao et al., 2025). CODI replaces this curriculum with one-step distillation (Shen et al., 2025). Extensions to multimodal settings similarly interleave latent visual representations with language tokens for visual reasoning (Yang et al., 2025). Most closely related is recent work on variational latent reasoning, which formulates reasoning as posterior inference over latent variables (Zhou et al., 2025; Wang et al., 2026; Lin et al., 2025b). Our work motivates latent reasoning through a different lens: robotic policies

lack a natural reasoning medium such as language. Therefore, reasoning in latent space provides a principled framework to bring the benefits of iterative and adaptive computation to robotics.

Behavior Cloning via Generative Control Policies: Behavior cloning (BC) is a standard approach for learning robot policies from demonstrations (Pomerleau, 1988), increasingly cast as *generative modeling* over actions. IBC represents action distributions with energy-based models to handle discontinuous or set-valued mappings (Florence et al., 2022). Diffusion Policy models actions using iterative denoising processes, yielding expressive multimodal behavior with stable training (Chi et al., 2024). BeT and VQ-BeT enable multiple behavioral modes within transformer policies via discrete action tokens (Shafiullah et al., 2022; Lee et al., 2024b). Our method is a discrete latent-variable model capable of expanding computation as action prediction demands. As our experiments show, this approach achieves strong empirical performance while retaining the multimodal expressivity in prior generative control policies.

Reasoning for Robotics: Recent work brings iterative reasoning and test-time compute to embodied decision making through several complementary directions. First, some methods utilize *explicit* intermediate reasoning traces to structure decision making: ECoT generates grounded sub-steps (e.g., sub-tasks) before acting (Zawalski et al., 2025), ECoT Lite refines this recipe with a latency-aware design (Chen et al., 2025), and OneTwoVLA combines fast reactive control with deliberative "System 1/2" computation (Lin et al., 2025a). Second, some works introduce *structured intermediate representations* to bridge perception and control, such as spatial abstractions or hierarchical guidance (Lee et al., 2025; Li et al., 2025). Concurrently, RD-VLA (Tur et al., 2026) instantiates latent reasoning using a recurrent-depth transformer action head, halting when changes in predicted actions fall below a threshold. In contrast, our method reasons in an autoregressive latent space trained directly for action prediction, avoiding manually designed reasoning interfaces or stopping criteria.

3 Latent Memory Palace: Reasoning as Autoregressive Variational Inference

We consider the imitation learning setting, where the goal is to learn a control policy $\pi_\theta : \mathcal{O} \to \mathcal{A}$ from a dataset of expert demonstrations, $\mathcal{D} = \{(o_i, a_i)\}_{i=1}^n$, with observations o_i and actions a_i. For the sake of practicality, o_i may be a stacked history of observations with language instructions, and a_i may be a chunk of actions. The behavior cloning objective maximizes the likelihood of expert actions given observations: $J(\theta) = \mathbb{E}_{(o_i,a_i)\sim\mathcal{D}} [\log p_\theta(a_i|o_i)]$. We introduce a new parameterization of the policy π_θ using a variable-length latent autoregressive model.

3.1 Reasoning as Variational Inference

A natural way to introduce reasoning capabilities to policies is to have them first output a latent "trace" z conditioned on the current observation o. The trace can then be used to output actions a conditioned on (o, z). This process can be formalized as a probabilistic latent-variable model (Fig. 1(a)), where an action a is generated from observation o and latent variable z. Under this model, direct maximum likelihood estimation of $p(a \mid o)$ involves an intractable marginalization over z. Instead, we approximate the true posterior $p(z \mid o, a)$ with a variational distribution $q_\theta(z \mid o, a)$ and maximize the evidence lower bound (ELBO):

$$\log p(a \mid o) \geq \mathbb{E}_{z\sim q_\theta(z|o,a)} [\log p_\phi(a \mid o, z)] - D_{\mathrm{KL}}(q_\theta(z \mid o, a) \,\|\, p_\theta(z \mid o)) \,, \tag{1}$$

where θ parametrizes the learned posterior and prior, and ϕ parametrizes the action decoder. *The learned prior distribution $p_\theta(z \mid o)$ and the decoder $p_\phi(a \mid o, z)$ fully parametrize a policy, where actions are generated by first sampling $z \sim p_\theta(z \mid o)$ and then sampling $a \sim p_\phi(a \mid o, z)$.*

However, this still leaves open the important question of what form the latent variable z and the corresponding generative models should take. The transition from standard variational inference (Kingma & Welling, 2014; Lee et al., 2024b) to *latent reasoning* happens when we parameterize the latent space with a *variable-length autoregressive model*. In particular, we will represent z not as a single variable, but as a sequence of latent variables $z = (z_1, z_2, \ldots, z_{T(z)})$ from a vocabulary $\mathcal{V} \cup \{\texttt{EOS}\}$ with termination time $T(z) \coloneqq \min\{t \geq 1 : z_t = \texttt{EOS}\}$. We parameterize the prior and the posterior with a factorized autoregressive family, $p_\theta(z \mid o) = \prod_{t=1}^{T(z)} p_\theta(z_t \mid z_{<t}, o)$, $q_\theta(z \mid o, a) = \prod_{t=1}^{T(z)} q_\theta(z_t \mid z_{<t}, o, a)$, and the likelihood with a full-sequence decoder $p_\phi(a \mid o, z) = p_\phi(a \mid o, z_{1:T(z)})$. This

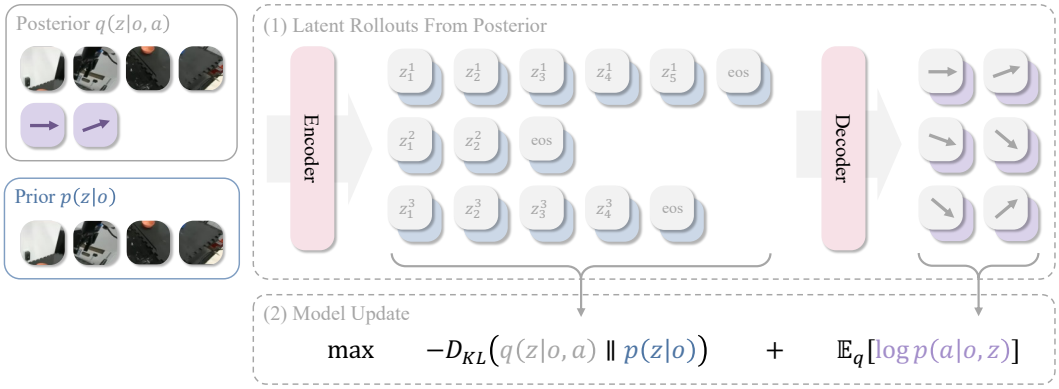


Figure 2: Training LMP-π alternates between two stages. (1) **Latent rollout:** the posterior generates a buffer of latent trajectories. (2) **Model update:** the model optimizes a clipped-surrogate of the variational lower bound, computed from the posterior, the prior, and the decoder.

enables the model to iteratively generate sequences of latent tokens, terminating at an EOS token. Together, the autoregressive factorization and the EOS token allow for *iterative* and *adaptive* computation. Under this factorization, Eq. 1 becomes:

$$\log p(a \mid o) \;\geq\; \mathbb{E}_{z_{1:T(z)} \sim q_\theta(\cdot|o,a)} \left[\log p_\phi(a \mid o, z_{1:T(z)})\right] - D_{\mathrm{KL}}\left(q_\theta(z_{1:T(z)} \mid o, a) \,\middle\|\, p_\theta(z_{1:T(z)} \mid o)\right). \tag{2}$$

Adaptive Reasoning via Compression While the autoregressive factorization in principle allows for adaptive computation, Eq. 2 does not specify when the computation should stop. To elicit adaptive behavior, we introduce compression so that the model only uses more steps as needed. We parameterize the decoder with an isotropic Gaussian distribution whose variance decays with latent length $p_\phi(a \mid o, z) = \mathcal{N}\left(\mu_\phi(o, z), \sigma^2(T(z))I\right)$, $\sigma(T(z)) = \gamma^{T(z)}\sigma_0$, $\gamma \in (0, 1)$. Intuitively, this imposes a length penalty: each additional latent step sharpens the Gaussian, so the model can only afford another step if the predicted actions are precise enough (formalized in Appendix B.2). As a result, the model spends more compute when actions can be precisely predicted from the observation, and less when the remaining uncertainty is hard to reduce. For example, teleoperators close the gripper at slightly different times across demonstrations, so additional computation cannot pin down the exact timing. Empirically, we find that this design choice leads to adaptive reasoning and contributes to the overall performance (Sec. 4.3).

3.2 Tractable Optimization via Latent-Space Reinforcement Learning

Optimizing Eq. 2 involves estimating the gradients through a non-differentiable autoregressive sampling process. To this end, we derive a tractable optimization objective using reinforcement learning. Note that Eq. 2 can be viewed as a reinforcement learning objective over latent-token trajectories. Fixing (o, a), define a latent "episode" that starts from the empty prefix and at each step samples a token z_t from the *variational posterior* $q_\theta(z_t \mid z_{<t}, o, a)$ until EOS. This defines a variable-horizon rollout with "policy" $q_\theta(\cdot \mid o, a)$ and termination at $T(z)$. A trajectory $z_{1:T(z)}$ is then scored by a return consisting of: (i) a reconstruction reward $\log p_\phi(a \mid o, z_{1:T(z)})$, and (ii) a penalty on the KL between the posterior $q_\theta(z_{1:T(z)} \mid o, a)$ and the prior $p_\theta(z_{1:T(z)} \mid o)$. Note that the latent "policy" is different from the executable policy used to interact with the environment, which is sampled from $z \sim p_\theta(\cdot \mid o), a \sim p_\phi(\cdot \mid o, z)$. *RL is used only to optimize inference over latent "reasoning" chains, rather than interacting with the physical environment directly.*

Variance reduction via stepwise KL decomposition. A naive policy-gradient estimator of Eq. 2 has high variance because the return is only provided at the end of the rollout. We leverage the autoregressive structure of the KL term to decompose the episodic return into per-step rewards. Let $\mathcal{P}_{\leq t} := \{z_{\leq t} : z_i \neq \mathrm{EOS}, \forall i < t\}$ denote the set of length-t prefixes that have not terminated before step t. Let $q_\theta(z_{\leq t} \mid o, a)$ denote the prefix marginal mass over $z_{\leq t} \in \mathcal{P}_{\leq t}$, thus defining a subprobability measure whose total mass equals $\Pr_{q_\theta}[T(z) \geq t]$. Let $\tilde{\mathbb{E}}$ denote the expectation under

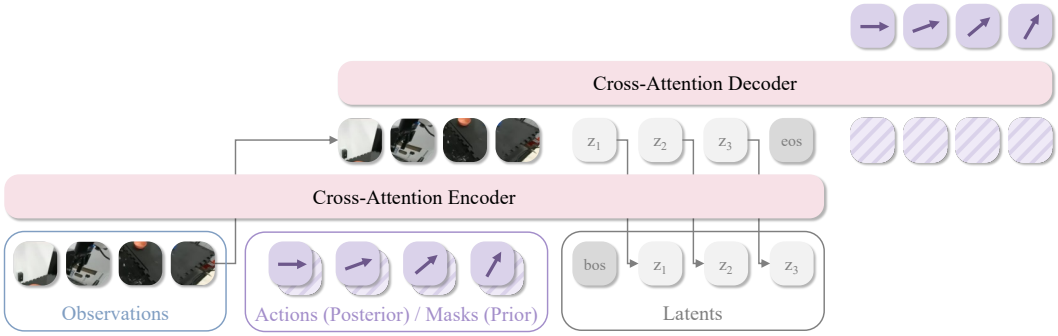


Figure 3: LMP-π architecture. The model consists of a latent encoder and an action decoder. The encoder is a causal transformer cross-attending to observations and actions. The decoder is a bidirectional transformer cross-attending to observations and latents.

this unnormalized measure. As shown in Appendix B.1, the KL term admits the decomposition

$$D_{\mathrm{KL}}(q_\theta(z \mid o,a) \,\|\, p_\theta(z \mid o)) = \sum_{t=1}^{\infty} \tilde{\mathbb{E}}_{z_{\le t}\sim q_\theta(\cdot|o,a)} \left[\log \frac{q_\theta(z_t \mid z_{<t}, o, a)}{p_\theta(z_t \mid z_{<t}, o)} \right]. \tag{3}$$

Optimization with a PPO-style clipped surrogate. To improve sample efficiency, we store on-policy latent rollouts in a buffer and optimize the objective using a trust-region style surrogate (Schulman et al., 2017) as shown in Fig. 2. Let $q_{\theta_{\text{old}}}$ denote the rollout policy. Define likelihood ratio $r(z) := \frac{q_\theta(z|o,a)}{q_{\theta_{\text{old}}}(z|o,a)}$, length-aware clipped likelihood ratio $\bar{r}(z) = \mathrm{clip}\big(r(z),\ (1-\epsilon)^{T(z)},\ (1+\epsilon)^{T(z)}\big)$, and clipped surrogate operator $\mathrm{CS}(r, \bar{r}; x) := \min(r\,x,\ \bar{r}\,\mathrm{stopgrad}(x))$ which updates x only when r is within the trust region.[1] We maximize the surrogate objective

$$\begin{aligned} J_{\mathrm{rec}}(\theta,\phi) &= \mathbb{E}_{z\sim q_{\theta_{\text{old}}}(\cdot|o,a)} \left[\mathrm{CS}\left(r(z), \bar{r}(z);\ \log p_\phi(a \mid o, z)\right)\right], \\ J^t_{\mathrm{kl}}(\theta) &= \tilde{\mathbb{E}}_{z_{\le t}\sim q_{\theta_{\text{old}}}(\cdot|o,a)} \left[\mathrm{CS}\left(r(z_{\le t}), \bar{r}(z_{\le t});\ \log \frac{p_\theta(z_t \mid z_{<t}, o)}{q_\theta(z_t \mid z_{<t}, o, a)}\right)\right], \\ J(\theta,\phi) &= J_{\mathrm{rec}}(\theta,\phi) + \sum_{t=1}^{\infty} J^t_{\mathrm{kl}}(\theta). \end{aligned} \tag{4}$$

As in prior literature (Higgins et al., 2017), we introduce additional stabilization techniques to prevent latent collapse and to balance the reconstruction and KL objectives. The details are discussed in Appendix A.1.

3.3 Connection to Action Tokenizer

In LMP, decisions are represented as variable-length paths through a discrete codebook. This representation naturally lends itself to the problem of action tokenization. In Appendix A.2, we show that by dropping the observation conditioning from the variational inference framework, we can instantiate a variable-length sequential action tokenizer LMP-`tok`. The tokenizer consists of an encoder $q_\theta(z_{1:T(z)} \mid a)$ which autoregressively generates tokens until `EOS`, and a decoder $p_\phi(a \mid z_{1:T(z)})$ which converts token sequences into continuous actions. LMP-`tok` can be used as the output space of downstream autoregressive models for learning expressive policies. Unlike FAST (Pertsch et al., 2025), whose partial token sequences do not form a valid DCT matrix, LMP-`tok` decodes a latent sequence of any length into a valid action.

3.4 Practical Instantiation

As shown in Fig. 3, we parameterize the prior $p_\theta(z|o)$ and the posterior $q_\theta(z|o,a)$ using a shared causal transformer with cross-attention conditioning. The posterior $q_\theta(z|o,a)$ is conditioned on image observation embeddings and action embeddings. The prior $p_\theta(z|o)$ is only conditioned on the

[1] Other equivalent PPO-style forms are possible; we found the above stable for variable-length rollouts.



(a) DROID (b) LIBERO (c) D3IL (d) Robomimic

Figure 4: Visualization of evaluation domains.

Table 1: DROID results. Mean success rate over 20 trials.

	zero-shot		finetuned			
	block-bowl	marker-mug	peg-hole	clean-table (1/3)	clean-table (2/3)	clean-table (3/3)
OpenVLA	0.30	0.00	0.30	0.85	0.30	0.05
DP	0.40	0.25	0.30	0.80	0.55	0.25
LMP-π	**0.65**	**0.55**	**0.70**	**0.95**	**0.95**	**0.55**

observation embeddings, masking out actions with learned mask tokens. The action decoder is a bidirectional transformer conditioned on the observation embeddings and the sampled latent tokens, outputting action chunks from learned embeddings. During training, we truncate the latent sequences at a fixed length H with shorter sequences padded to the same length. For LMP-π, we define the decoder variance schedule (Sec. 3.1) by specifying $(\sigma_{\text{max}}, \sigma_{\text{min}})$ and computing the corresponding decay factor γ. We additionally let the decoder predict a per-dimension scale $s \in [0, 1]$ applied on top of the variance schedule, so that the model can commit to small variance even when terminating early. For LMP-`tok`, we use a scalar learned variance applied to all timesteps, similar to σ-VAE (Rybkin et al., 2021b).

4 Experiments

We conduct experiments to answer the following research questions. (1) Is LMP-π a performant generative control policy? (2) Is LMP-`tok` an effective action tokenizer? (3) Which factors contribute to LMP's performance? (4) What type of reasoning does LMP show?

Environments We evaluate our method on real-world and simulated visuomotor control benchmarks as shown in Fig. 4. (1) DROID (Khazatsky et al., 2024) is a real-world manipulation platform with multitask training data. We evaluate language-conditioned multitask policies in zero-shot and finetuned settings. For each task, we evaluate on 20 randomly sampled initial conditions. (2) LIBERO (Liu et al., 2023) is a simulated manipulation benchmark featuring diverse task scenarios. We evaluate language-conditioned multitask policies in-distribution. (3) D3IL (Jia et al., 2024) is a simulated benchmark emphasizing multimodality. (4) RoboMimic (Mandlekar et al., 2021) is a simulated benchmark featuring high-precision tasks. Additional environment details are provided in Appendix C.3. All simulation results are averaged over three seeds.

4.1 Is LMP-π a performant generative control policy?

We compare LMP-π with (1) Diffusion Policy (DP) (Chi et al., 2024), a generative control policy parameterized by a diffusion model, and (2) OpenVLA (Kim et al., 2024), an autoregressive generative control policy that discretizes actions into language-model tokens. We instantiate all methods as 1B-parameter, language-conditioned policies (details in Appendix C). Table 1 shows the results on the real-world DROID platform. Evaluated zero-shot on block-bowl and marker-mug, LMP-π outperforms DP and OpenVLA, indicating stronger out-of-the-box generalization. We then finetune the policies on two challenging tasks, peg-hole and clean-table, designed to measure precision and compositional generalization respectively. We find LMP-π to outperform baselines on both

tasks, notably picking up at least 2 objects with 95% success rate in the clean-table task. While DP and OpenVLA can reliably clean up one object, they quickly taper off at two and three objects. These results suggest that LMP-π is capable of precise control and compositional generalization. Table 2 shows the language-conditioned multitask results on LIBERO-90. While LMP-π performs marginally better in terms of overall success rate, a closer examination of the bottom 10 tasks of each method reveals that LMP-π has a significantly higher floor than DP and OpenVLA. This pattern indicates reduced cross-task interference and better retrieval. Fig. 12 provides a detailed comparison of per-task success rates, where we see LMP-π gaining a bigger margin over DP on hard tasks.

Table 2: LIBERO-90 results. Mean success rate $\pm$ standard deviation over 3 seeds.

	all tasks	bottom-10 tasks
OpenVLA	0.358 $\pm$0.024	0.077 $\pm$0.008
DP	0.909 $\pm$0.004	0.463 $\pm$0.011
LMP-π	**0.933** $\pm$**0.003**	**0.645** $\pm$**0.023**

4.2 Is LMP-tok an effective action tokenizer?

We compare LMP-tok with representative action tokenizer baselines: (1) VQ-VAE (Lee et al., 2024a), a 2-layer residual VQ tokenizer, (2) FAST (Pertsch et al., 2025), a variable-length action tokenizer based on DCT transform and byte-pair encoding, and (3) OAT (Liu et al., 2026), a variable-length latent action tokenizer with a coarse-to-fine inductive bias. We evaluate the methods on RoboMimic and LIBERO, each featuring a large pretraining action dataset and multiple downstream tasks. We apply the tokenizers to the same downstream autoregressive transformer policy (with an added EOS token for variable-length methods). As shown in Fig. 5, LMP-tok significantly outperforms baselines in the downstream tasks. Moreover, individual task results in Table 7 show that LMP-tok is the only method achieving nontrivial performance on the high-precision RoboMimic tool-hang task. VQ-VAE lacks sufficient precision for fine-grained manipulation. FAST requires exact reconstruction of DCT matrices, placing a heavy burden on the downstream policy. While OAT belongs to the same class of latent autoregressive action tokenizers, we find it less effective than LMP-tok, likely because it is trained with only reconstruction and not a variational objective to maintain latent diversity.

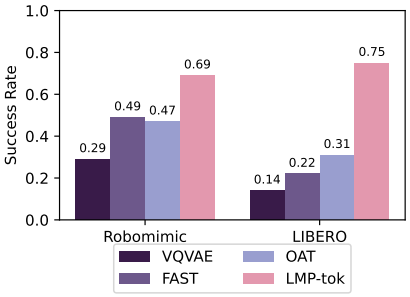


Figure 5: Aggregate mean success rates of autoregressive policies fitted to action tokenizers.

4.3 Analysis

We conduct analytical experiments to ablate the factors that contribute to LMP's performance, and to understand the mechanism behind its test-time compute allocation. We provide additional ablation results in Appendix D.

Does variable-length latent computation outperform fixed-length latent computation? To study the effect of variable-length latent computation, we compare LMP-π to two fixed-length latent-variable baselines, (1) VAE Policy (Kingma & Welling, 2014), which fits a one-step Gaussian VAE to expert actions conditioned on observations, and (2) VQ-BeT (Lee et al., 2024a), which combines a behavior transformer with a residual VQ-VAE tokenizer. Although VQ-BeT uses an autoregressive transformer, its two-level hierarchy effectively fixes the latent length. As shown in Fig. 6, LMP-π consistently outperforms both baselines on D3IL and RoboMimic, indicating that fixed-length latent representations lack the granularity needed for complex manipulation tasks.

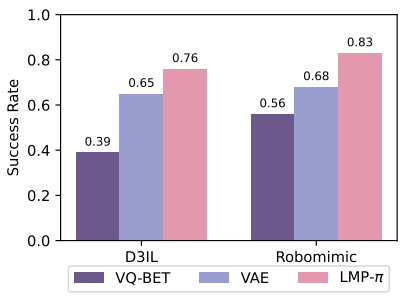


Figure 6: Aggregate mean success rates of single-task policies.

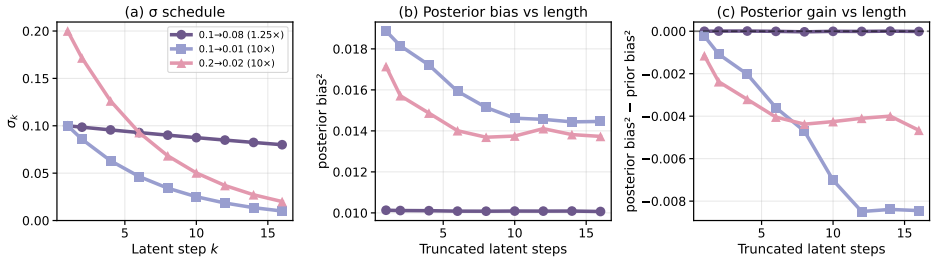


Figure 7: Effect of compression on latent use in DROID. (a) Decoder σ schedules $\sigma_{\max}$-$\sigma_{\min}$. (b) Posterior reconstruction error vs the number of latent steps kept. (c) Posterior minus prior error. Weak compression (0.1-0.08) leads to the decoder ignoring the latents (flat curves), and strong compression (0.1-0.01) raises the error.

Does compression contribute to performance? We ablate the compression strength through the variance schedule. As discussed in Sec. 3.1, the smaller the $\sigma_{\min}$, the faster the variance decays, and the stronger the compression. Table 3 shows LIBERO-90 results with different levels of compression. We find that stronger compression (smaller $\sigma_{\min}$) leads to improved policy performance. This suggests that parsimonious allocation of latent compute can prevent overfitting. Fig. 7 analyzes the effect of compression on reconstruction error and latent utilization using DROID checkpoints. We find that (1) insufficient compression leads to latents being ignored by the decoder (0.1-0.08 vs. 0.2-0.02), and (2) excessive compression forces the model to use shorter traces and thus raises the overall reconstruction error (0.1-0.01 vs 0.2-0.02). Empirically, we find a configuration that balances reconstruction and latent utilization (0.2-0.02) to work well.

Table 3: Ablation of variance schedule. Lower $\sigma_{\min}$ means higher compression. $\sigma_{\max}$ is fixed at 0.1.

$\sigma_{\min}$	all tasks	bottom-10 tasks
0.01	**0.933** ±0.003	**0.645** ±0.023
0.03	0.913 ±0.001	0.567 ±0.006
0.05	0.870 ±0.012	0.450 ±0.045

What type of reasoning does LMP show? In LMP, latent reasoning provides a variable-length representation of the expert action distribution. Under Eq. 2, every latent trace, long or short, is trained to decode an expert action. Hence, "reasoning" in LMP is not iterative refinement of a partial decision. Instead, additional latent steps increase the number of behavioral modes the policy can express. In Fig. 9(a), we visualize the rollouts of a d3il-avoiding policy as the maximum number of latent steps is truncated at *test time*. We find that reducing the number of latent steps progressively drops behavioral modes. The effect of this reduced action diversity on success varies by task. In simulation, a single expert mode suffices to complete the task, so truncation barely affects success (Fig. 9(b)). In DROID zero-shot tasks, truncating the latent steps lowers success from 0.65 to 0.55 on block-bowl and from 0.55 to 0.40 on marker-mug. Zero-shot failures often stem from object confusion, e.g., picking up a different object from the instructed one. While a multimodal policy can recover, a unimodal one may commit to the wrong object. Note that test-time truncation differs from training-time truncation. In Fig. 9(c), we show that reducing the latent space at *training time* brings the success rate closer to deterministic behavior cloning, a sign of mode averaging. In summary, more "reasoning" in LMP means painting a fuller picture of the expert distribution, which affects success indirectly through recovery behaviors.

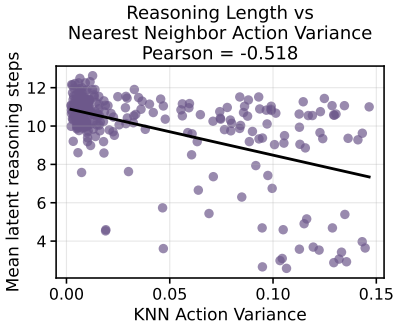


Figure 8: Mean latent steps of LIBERO-90 policy vs action variance at $k = 32$ nearest dataset states.

While the *number* of modes grows with latent length, the *ordering* of the modes is determined by the decoder variance schedule (Sec. 3.1). Modes that can tolerate imprecision are stored in shorter traces with higher decoder variance, while modes that high precision are stored in longer traces with lower decoder variance. Fig. 8 shows that the number of latent steps of a LIBERO-90 policy is negatively correlated with the variance of nearest-neighbor dataset actions. Empirically, this shows up as the model using fewer latent steps during gripper movements, e.g., on DROID clean-table (Fig. 1(b)),

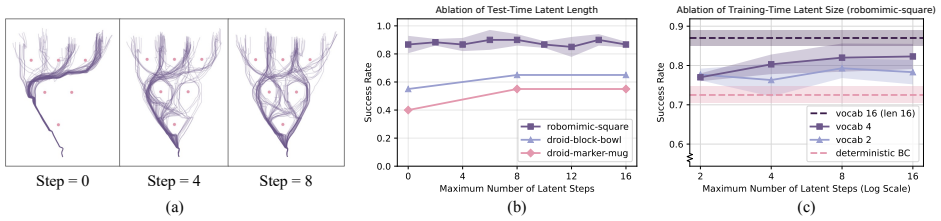


Figure 9: Analysis of reasoning behavior. (a) Visualization of d3il-avoiding rollouts. As the number of latent steps is truncated, the policy exhibits mode-dropping behavior. (b) Success rate vs test-time truncation. Truncating the number of latent steps at test time leads to a performance drop on tasks where recovery actions are important.(c) Success rate vs training-time latent size. Reducing the latent size during training leads to mode averaging.

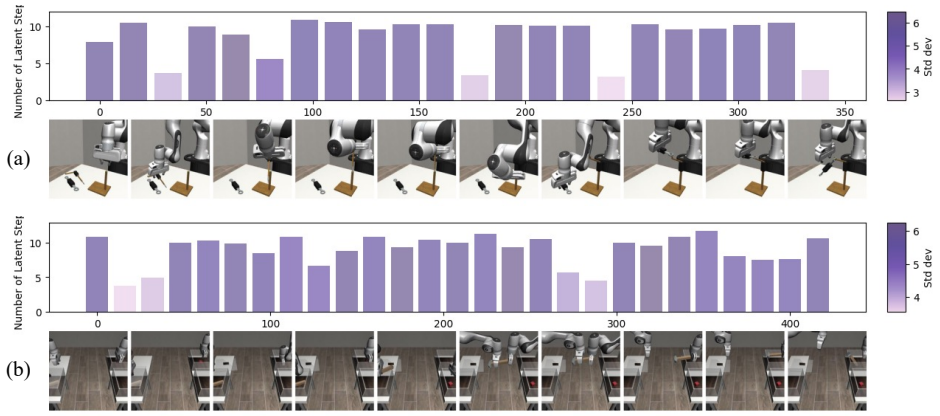


Figure 10: LMP-π exhibits adaptive allocation of latent steps. The length of latent traces is lower during gripper movements, e.g. (a) grasping and releasing, and (b) hand-over.

RoboMimic (Fig. 10), and other simulated and real-world environments (Appendix D). Gripper movement is a canonical high-variance action mode: the gripper timing varies across demonstrations, and the binary open-close action does not require precision. The model stops early since more compute does not bring more value. This is closely related to the value of computation (Sabbata et al., 2024) view of reasoning, where the model spends more compute only when the benefit justifies the cost. In LMP, additional compute is useful only when the improved fit to the expert distribution justifies the cost imposed by the variance schedule.

5 Discussion

We introduce Latent Memory Palace, a formulation of iterative and adaptive reasoning as variational inference with an autoregressive latent distribution. Applied to sequential decision making, this formulation yields a policy class, LMP-π, which exhibits strong empirical performance and adaptive allocation of test-time compute. Our analyses show that variable-length latents and compression both contribute to performance. We also introduce a variable-length action tokenizer, LMP-`tok`, which significantly outperforms existing methods when controlling for the downstream policy. These results establish autoregressive variational inference as a principled and practical route to iterative, adaptive computation for robotic control.

Limitations Since LMP is trained with sampling-based RL techniques, it is sensitive to hyperparameters. For example, the latent distribution is prone to collapse without a sufficiently large rollout buffer or regularization. We leave the exploration of stable and scalable optimization of the proposed framework to future work.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing and proofread mathematical proofs. We did not use generative AI for ideation, retrieval of related work, drafting sections of the paper, generating synthetic datasets, or implementation. We have reviewed all AI-assisted work, e.g. checking the writing and proofs. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We include anonymous code in the supplemental material to reproduce the experiments. All datasets used in the experiments are publicly available. We will release the code upon acceptance.

REFERENCES

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities, 2024. URL https://arxiv.org/abs/2401.12168.

William Chen, Suneel Belkhale, Suvir Mirchandani, Oier Mees, Danny Driess, Karl Pertsch, and Sergey Levine. Training strategies for efficient embodied reasoning, 2025. URL https://arxiv.org/abs/2505.08243.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

DeepSeek-AI. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6pPYRXKPpw.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani,

Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2024. URL https://arxiv.org/abs/2403.12945.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025. URL https://arxiv.org/abs/2508.07917.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024a.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions, 2024b. URL https://arxiv.org/abs/2403.03181.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation, 2025. URL https://arxiv.org/abs/2502.05485.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning, 2025a. URL https://arxiv.org/abs/2505.11917.

Tianqianjin Lin, Xi Zhao, Xingyao Zhang, Rujiao Long, Yi Xu, Zhuoren Jiang, Wenbo Su, and Bo Zheng. Ravr: Reference-answer-guided variational reasoning for large language models, 2025b. URL https://arxiv.org/abs/2510.25206.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, qiang liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=xzEtNSuDJk.

Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. Oat: Ordered action tokenization. In *Proceedings of Robotics: Science and Systems*, 2026.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps. 2025.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

OpenAI. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.012.

Dean A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS'88, pp. 305–313, Cambridge, MA, USA, 1988. MIT Press.

Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. Simple and effective vae training with calibrated decoders. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9179–9189. PMLR, 18–24 Jul 2021a. URL https://proceedings.mlr.press/v139/rybkin21a.html.

Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. Simple and effective vae training with calibrated decoders. In *International conference on machine learning*, pp. 9179–9189. PMLR, 2021b.

C. Nicolò De Sabbata, Theodore Sumers, and Thomas L. Griffiths. Rational metareasoning for large language models. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*, 2024. URL https://openreview.net/forum?id=Otm03OVHzG.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

TRI LBM Team, Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, Naveen Kuppuswamy, Kuan-Hui Lee, Katherine Liu, Dale McConachie, Ian McMahon, Haruki Nishimura, Calder Phillips-Grafflin, Charles Richter, Paarth Shah, Krishnan Srinivasan, Blake Wulfe, Chen Xu, Mengchao Zhang, Alex Alspach, Maya Angeles, Kushal Arora, Vitor Campagnolo Guizilini, Alejandro Castro, Dian Chen, Ting-Sheng Chu, Sam Creasey, Sean Curtis, Richard Denitto, Emma Dixon, Eric Dusel, Matthew Ferreira, Aimee Goncalves, Grant Gould, Damrong Guoy, Swati Gupta, Xuchen Han, Kyle Hatch, Brendan Hathaway, Allison Henry, Hillel Hochsztein, Phoebe Horgan, Shun Iwase, Donovon Jackson, Siddharth Karamcheti, Sedrick Keh, Joseph Masterjohn, Jean Mercat, Patrick Miller, Paul Mitiguy, Tony Nguyen, Jeremy Nimmer, Yuki Noguchi, Reko Ong, Aykut Onol, Owen Pfannenstiehl, Richard Poyner, Leticia Priebe Mendes Rocha, Gordon Richardson, Christopher Rodriguez, Derick Seale, Michael Sherman, Mariah Smith-Jones, David Tago, Pavel Tokmakov, Matthew Tran, Basile Van Hoorick, Igor Vasiljevic, Sergey Zakharov, Mark Zolotas, Rares Ambrus, Kerri Fetzer-Borelli, Benjamin Burchfiel, Hadas Kress-Gazit, Siyuan Feng, Stacie Ford, and Russ Tedrake. A careful examination of large behavior models for multitask dexterous manipulation. 2025. URL https://arxiv.org/abs/2507.05331.

Yalcin Tur, Jalal Naghiyev, Haoquan Fang, Wei-Chuan Tsai, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Recurrent-depth vla: Implicit test-time compute scaling of vision-language-action models via latent iterative reasoning, 2026. URL https://arxiv.org/abs/2602.07845.

Fanmeng Wang, Haotian Liu, Guojiang Zhao, Hongteng Xu, and Zhifeng Gao. Regular: Variational latent reasoning guided by rendered chain-of-thought. *arXiv preprint arXiv:2601.23184*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens, 2025. URL https://arxiv.org/abs/2506.17218.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2407.08693.

Xiangxin Zhou, Zichen Liu, Haonan Wang, Chao Du, Min Lin, Chongxuan Li, Liang Wang, and Tianyu Pang. Variational reasoning for language models. *arXiv preprint arXiv:2509.22637*, 2025.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025a.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, Zhenhe Wu, Yongchi Zhao, Tianyu Liu, Jian Yang, Wangchunshu Zhou, Chujie Zheng, Chongxuan Li, Yuyin Zhou, Zhoujun Li, Zhaoxiang Zhang, Jiaheng Liu, Ge Zhang, Wenhao Huang, and Jason Eshraghian. A survey on latent reasoning, 2025b. URL https://arxiv.org/abs/2507.06203.

A Extended Method Details

A.1 Posterior stabilization

Optimizing Eq. 4 involves jointly training the prior $p_\theta(z \mid o)$ and the posterior $q_\theta(z \mid o, a)$, which is prone to collapse. We mitigate this with two stabilizers.

(i) Uniform-interpolated regularization. We regularize the posterior toward a length-aware uniform prior $u(z_t)$ over $\mathcal{V} \cup \{\texttt{EOS}\}$. We define a geometrically interpolated prior at each step:

$$p^\alpha_{\text{mix}}(z_t \mid z_{<t}, o) \;\propto\; p_\theta(z_t \mid z_{<t}, o)^\alpha \, u(z_t)^{1-\alpha}, \;\; \alpha \in [0, 1],$$

and replace p_θ with p^α_{mix} in the KL term:

$$J^t_{\text{kl}}(\theta, \alpha) = \tilde{\mathbb{E}}_{z_{\le t} \sim q_{\theta_{\text{old}}}(\cdot \mid o, a)} \left[\text{CS}\left(r(z_{\le t}), \bar{r}(z_{\le t}); \; \log \tfrac{p^\alpha_{\text{mix}}(z_t \mid z_{<t}, o)}{q_\theta(z_t \mid z_{<t}, o, a)} \right) \right]. \tag{5}$$

(ii) Per-step free-nats clipping. We cap the strength of the KL "pull" toward the prior at each step using a free-nats threshold, preventing overly aggressive distillation early in training. Let $\rho^t := \Pr_{q_\theta}[T(z) \ge t]$ denote the survival probability and $D^t_{\text{KL}} := D_{\text{KL}}\big(q_\theta(z_t \mid z_{<t}, o, a) \,\|\, p_\theta(z_t \mid z_{<t}, o)\big)$ denote the one-step KL. The per-step KL term in Eq. 3 can be rewritten as survival-weighted expected one-step KL:

$$\tilde{\mathbb{E}}_{z_{\le t} \sim q_\theta(\cdot \mid o, a)} \left[\log \frac{q_\theta(z_t \mid z_{<t}, o, a)}{p_\theta(z_t \mid z_{<t}, o)} \right] = \rho^t \, \mathbb{E}_{z_{<t} \sim q_\theta(\cdot \mid o, a, \, T(z) \ge t)} \left[D^t_{\text{KL}} \right]. \tag{6}$$

We clip the expected one-step KL $\mathbb{E}_{z_{<t} \sim q_\theta(\cdot \mid o, a, \, T(z) \ge t)} [D^t_{\text{KL}}]$ by a free nat budget $\tau \ge 0$. Applying this to the clipped surrogate objective gives:

$$J^t_{\text{kl}}(\theta, \alpha, \tau) = \rho^t_{\text{old}} \min\left(-\tau, \mathbb{E}_{z_{\le t} \sim q_{\theta_{\text{old}}}(\cdot \mid o, a, T(z) \ge t)} \left[\text{CS}\left(r(z_{\le t}), \bar{r}(z_{\le t}); \; \log \tfrac{p^\alpha_{\text{mix}}(z_t \mid z_{<t}, o)}{q_\theta(z_t \mid z_{<t}, o, a)} \right) \right] \right). \tag{7}$$

This effectively applies a soft constraint to the KL contribution at each timestep. In practice, we set $\tau = \beta \log |\mathcal{V} \cup \{\texttt{EOS}\}|, \beta \in [0, 1]$, to scale free nats proportionally to the maximum entropy of a categorical variable.

A.2 Tokenizer objective

The tokenizer objective is similar to the policy (Eq. 2) but drops the observation conditioning. Let $q_\theta(z_{1:T(z)} \mid a)$ denote the learned posterior and $p(z_{1:T(z)})$ denote a prior where each $p(z_t \mid z_{<t})$ is uniform over $\mathcal{V} \cup \{\texttt{EOS}\}$. LMP-`tok` maximizes the ELBO:

$$\log p(a) \;\ge\; \mathbb{E}_{z_{1:T(z)} \sim q_\theta(\cdot \mid a)} \left[\log p_\phi(a \mid z_{1:T(z)}) \right] - D_{\text{KL}}\big(q_\theta(z_{1:T(z)} \mid a) \,\big\|\, p(z_{1:T(z)}) \big). \tag{8}$$

The clipped surrogate objective is:

$$\begin{aligned} J_{\text{rec}}(\theta, \phi) &= \mathbb{E}_{q^{\text{old}}_\theta(z \mid a)} \left[\text{CS}\left(r(z), \bar{r}(z); \log p_\phi(a \mid z) \right) \right], \\ J^t_{\text{kl}}(\theta) &= \tilde{\mathbb{E}}_{q^{\text{old}}_\theta(z_{\le t} \mid a)} \left[\text{CS}\left(r(z_{\le t}), \bar{r}(z_{\le t}); \log \frac{p(z_t)}{q_\theta(z_t \mid z_{<t}, a)} \right) \right], \\ J(\theta, \phi) &= J_{\text{rec}}(\theta, \phi) + \sum_{t=1}^{\infty} J^t_{\text{kl}}(\theta). \end{aligned} \tag{9}$$

A.3 Instantiation as World Action Model

We can extend LMP-π to a world action model (Zhu et al., 2025a; Ye et al., 2026) by noticing that under deterministic and invertible dynamics, actions and future observations are equivalent given current observations: $(o_t, a_t) \sim (o_t, o_{t+1})$. This allows us to instantiate a world action model, LMP-`wam`, by replacing the encoder's input actions a_t with future observations o_{t+1}. The model jointly predicts future observations in the latent space (via the KL objective) and actions explicitly (via the reconstruction objective). In Table 4, we show that this formulation achieves comparable performance to LMP-π on the LIBERO-90 multitask benchmark. We leave exploration of the world action model formulation to future work.

Table 4: Full Evaluation Results Across LIBERO-90 Tasks.

	DP	LMP-π	LMP-`wam`
all tasks	0.909 $\pm$0.004	**0.933** $\pm$**0.003**	**0.928** $\pm$**0.002**
bottom-10 tasks	0.463 $\pm$0.011	**0.645** $\pm$**0.023**	**0.635** $\pm$**0.005**

B Derivations

B.1 Derivation of Per-Step KL Objective

Optimizing $D_{\mathrm{KL}}\big(q(z \mid o,a) \,\|\, p(z \mid o)\big)$ in Eq. 2 is subject to high variance. To stabilize training, we decompose the sequence-level KL into per-step contributions, effectively breaking an episodic return into per-step rewards. For clarity, we omit the conditioning on (o,a).

Assume (i) $q(z)$ and $p(z)$ are defined over finite `EOS`-terminated sequences, i.e. $\Pr_q[T(z)<\infty]=1$ and $\Pr_p[T(z)<\infty]=1$, (ii) $q(z)$ and $p(z)$ have compatible support, i.e. $q(z)>0 \Rightarrow p(z)>0$, and (iii) the sequence-level KL is finite, i.e. $D_{\mathrm{KL}}\big(q(z) \,\|\, p(z)\big)<\infty$. We first carry out the derivation on a finite horizon H and then pass to the limit. To compare sequences of different lengths, we extend each sequence that has terminated by step H with deterministic null tokens after `EOS`, where the padded positions satisfy $q(z_t \mid z_{<t}) = p(z_t \mid z_{<t}) = 1$ and contribute zero log-ratio. Writing $z_{\le H}$ for the padded length-H prefix and applying the autoregressive factorization, we have:

$$\begin{aligned} D_{\mathrm{KL}}\big(q(z_{\le H}) \,\|\, p(z_{\le H})\big) &= \sum_{z_{\le H}} q(z_{\le H}) \log \frac{q(z_{\le H})}{p(z_{\le H})} \\ &= \sum_{z_{\le H}} q(z_{\le H}) \sum_{t=1}^{H} \log \frac{q(z_t|z_{<t})}{p(z_t|z_{<t})} \\ &= \sum_{t=1}^{H} \sum_{z_{\le H}} q(z_{\le H}) \log \frac{q(z_t|z_{<t})}{p(z_t|z_{<t})} \quad \text{(exchanging finite sums).} \end{aligned}$$

Since the padded positions after `EOS` contribute zero log-ratio, only prefixes that have not yet terminated remain. Let $\mathcal{P}_{<t} \coloneqq \{z_{<t} : z_i \neq \texttt{EOS}, \forall i < t\}$ denote the set of length-$(t-1)$ prefixes that have not terminated before step t. We have:

$$\begin{aligned} D_{\mathrm{KL}}\big(q(z_{\le H}) \,\|\, p(z_{\le H})\big) &= \sum_{t=1}^{H} \sum_{z_{<t} \in \mathcal{P}_{<t}} \sum_{z_t} q(z_{<t})\, q(z_t \mid z_{<t}) \log \frac{q(z_t \mid z_{<t})}{p(z_t \mid z_{<t})} \\ &= \sum_{t=1}^{H} \sum_{z_{<t} \in \mathcal{P}_{<t}} q(z_{<t}) D_{\mathrm{KL}}\big(q(z_t \mid z_{<t}) \,\|\, p(z_t \mid z_{<t})\big). \end{aligned}$$

This expresses the finite-horizon KL as a sum of nonnegative per-step terms. By assumption (i), the length-H prefix determines the full sequence in the limit, so the left hand side converges to $D_{\mathrm{KL}}\big(q(z) \,\|\, p(z)\big)$. The right hand side is a series of nonnegative per-step partial sums, whose limit exists by monotone convergence. Taking the limit of both sides gives:

$$D_{\mathrm{KL}}\big(q(z) \,\|\, p(z)\big) = \sum_{t=1}^{\infty} \sum_{z_{<t} \in \mathcal{P}_{<t}} q(z_{<t}) D_{\mathrm{KL}}\big(q(z_t \mid z_{<t}) \,\|\, p(z_t \mid z_{<t})\big).$$

Let $\mathcal{P}_{\le t} \coloneqq \{z_{\le t} : z_i \neq \texttt{EOS}, \forall i < t\}$ denote the set of length-t prefixes that have not terminated before step t. Let $q(z_{\le t})$ denote the marginalized prefix mass over $z_{\le t} \in \mathcal{P}_{\le t}$, thus defining a subprobability measure whose total mass equals $\Pr_q[T(z) \ge t]$. Let $\tilde{\mathbb{E}}$ denote the expectation under this unnormalized measure. The expression above can be equivalently written as:

$$D_{\mathrm{KL}}\big(q(z) \,\|\, p(z)\big) = \sum_{t=1}^{\infty} \tilde{\mathbb{E}}_{z_{\le t} \sim q(\cdot)} \left[\log \frac{q(z_t \mid z_{<t})}{p(z_t \mid z_{<t})} \right].$$

B.2 Derivation of Stopping Criterion

We analyze the stopping criterion for the autoregressive process under the decoder variance schedule from Sec. 3.1. We isolate the reconstruction objective and omit the KL term to provide a heuristic understanding of the latent dynamics. Let D denote the action dimension, $\mu_L := \mu_\phi(o, z_{1:L})$ the decoder mean at length L, $\varepsilon_L := ||a - \mu_L||^2/D$ the per-dimension decoder mean squared error at length L, and $\sigma(L) = \gamma^L \sigma_0$ the decoder standard deviation with discount $\gamma \in (0, 1)$. The action log-likelihoods at length L and $L + 1$ are:

$$\ell(L) = -\frac{D}{2}\log(2\pi) - D\log\sigma(L) - \frac{D\,\varepsilon_L}{2\,\sigma^2(L)}$$
$$\ell(L+1) = -\frac{D}{2}\log(2\pi) - D\log\sigma(L+1) - \frac{D\,\varepsilon_{L+1}}{2\,\sigma^2(L+1)}$$

Taking their difference and assuming the reconstruction error has saturated, i.e. $\varepsilon_L = \varepsilon_{L+1} = \varepsilon$, we have

$$\begin{aligned}\Delta\ell &= -D\log\frac{\sigma(L+1)}{\sigma(L)} + \frac{D\,\varepsilon_L}{2\,\sigma^2(L)} - \frac{D\,\varepsilon_{L+1}}{2\,\sigma^2(L+1)} \\ &= -D\log\gamma - \frac{D\,\varepsilon}{2\,\sigma^2(L)}\left(\frac{1}{\gamma^2} - 1\right)\end{aligned}$$

A crossover happens when the delta action log-likelihood $\Delta\ell = 0$. Solving for the corresponding ε_L^*:

$$-D\log\gamma = \frac{D\,\varepsilon}{2\,\sigma^2(L)}\left(\frac{1}{\gamma^2} - 1\right)$$
$$\varepsilon_L^* = \frac{-2\,\sigma^2(L)\,\log\gamma}{1/\gamma^2 - 1}$$

For small $|\log\gamma|$, we have $1/\gamma^2 - 1 = e^{-2\log\gamma} - 1 \approx -2\log\gamma$. Therefore:

$$\varepsilon_L^* \approx \frac{-2\,\sigma^2(L)\,\log\gamma}{-2\log\gamma} = \sigma^2(L)$$

This suggests that the autoregressive process stops when the minimum achievable mean squared error at step L is on the order of the decoder variance $\sigma^2(L)$.

The regularization effect of the KL term Without the KL term, the reconstruction objective alone admits a degenerate solution: the posterior could "splat" a multimodal action distribution with many small Gaussians, pushing the reconstruction error arbitrarily low and maxing out the latent steps. The KL term counteracts this degeneracy. While the posterior sees the input action and can route each sample to a dedicated latent path, the prior sees only the observation and must spread its probability mass to cover all paths. Under the splatting solution, the prior mass is stretched thin, and each routed sample incurs a KL cost that offsets the log-likelihood gain of the sharper Gaussians. This pulls the posterior back towards covering the action distribution with fewer, broader components, i.e., shorter latent traces. The characterization in Sec. 3.1 can therefore be stated more precisely: the model spends less compute when the remaining uncertainty cannot be reduced without a commensurate KL cost. We retain the simpler statement in the main text for clarity.

C Implementation Details

C.1 Model Architecture

Our implementation of LMP-π consists of a shared latent encoder parameterizing the posterior $q_\theta(z \mid o, a)$ and the prior $p_\theta(z \mid o)$, and an action decoder parameterizing $p(a \mid o, z)$. o is a stack of h_o frames from n_c camera views, each encoded via an image encoder into an $n_{\texttt{dim}}$-dimensional token. When provided, low-dimensional observations and language instructions are encoded using separate encoders into $n_{\texttt{dim}}$-dimensional tokens and concatenated with image tokens. a is a chunk of h_a actions, each encoded via a 2-layer MLP into an $n_{\texttt{dim}}$-dimensional action token. For the posterior

Table 5: Hyperparameters

Model Parameter	Value
LMP-π Shared	
Observation History h_o	1
Action Chunk h_a	16
Latent Sequence Length H	16
Num Registers	1
MLP Ratio	4
LMP-π Multi-Task	
Latent Vocab Size $n_{\texttt{vocab}}$	DROID: 64 LIBERO: 16
Observation Encoder	SigLip (from Pi-0.5)
Language Encoder	SigLip 2
Embed Dim $n_{\texttt{dim}}$	768
Encoder Depth	12
Decoder Depth	10
Num Heads	12
Variance Range	DROID: 0.2-0.02 LIBERO: 0.1-0.01
LMP-π Single-Task	
Latent Vocab Size $n_{\texttt{vocab}}$	16
Observation Encoder	ResNet-18
Embed Dim $n_{\texttt{dim}}$	384
Encoder Depth	8
Decoder Depth	6
Num Heads	8
Variance Range	0.1-0.01
LMP-`tok`	
Latent Sequence Length H	8
Latent Vocab Size $n_{\texttt{vocab}}$	64
Initial Variance	1.0
Autoregressive Policy	
Observation Encoder	ResNet-18
Embed Dim $n_{\texttt{dim}}$	1024
Depth	12
Num Heads	8
MLP Ratio	4

Training Parameter	Value ($\times$ GPUs)
LMP Shared	
Training Epochs	2
Clip Range	0.01
Prior Mixture Coef	0.1
LMP-π Multi-Task	
Num Samples	5000000 $\times$ 32
Rollout Buffer Size	3200 $\times$ 32
Batch Size	64 $\times$ 32
Free Nats Ratio	0.05
KL Loss Coef	1.0
Reconstruction Loss Coef	0.01
LMP-π Single-Task	
Num Samples	25000000 $\times$ 4
Rollout Buffer Size	25000 $\times$ 4
Batch Size	500 $\times$ 4
Free Nats Ratio	0.05
KL Loss Coef	1.0
Reconstruction Loss Coef	0.1
LMP-`tok`	
Free Nats Ratio	0.2
KL Loss Coef	0.1
Reconstruction Loss Coef	1.0
Autoregressive Policy	
Number of gradient steps	50000 $\times$ 4
Batch Size	1024 $\times$ 4
Focal loss γ	1.0
Optimization	
Optimizer	AdamW
Learning Rate	$1e^{-4}$
Weight Decay	$1e^{-6}$
Betas	[0.9, 0.999]
Epsilon	$1e^{-8}$

$q_\theta(z \mid o, a)$, we concatenate observation and action tokens along the sequence dimension and pass them as cross-attention context to the causal transformer. For the prior $p_\theta(z \mid o)$, we mask out the action tokens with learned mask tokens. The latent encoder learns an $n_{\texttt{vocab}} + 1$ size codebook where the additional token denotes `EOS`. The autoregressive generation begins with a learnable root token `ROOT` and iteratively generates the next token until `EOS` is generated or a maximum sequence length H is reached. The output sequence is then concatenated again with the observation embeddings and passed into the action decoder as cross-attention context. The action decoder is a bidirectional transformer with h_a learnable query tokens. The corresponding output tokens are then decoded into action chunks via a 2-layer MLP. We use learned positional embeddings for both the encoder and the decoder.

The action tokenizer LMP-`tok` uses the same parameterization but drops the observation conditioning from both the encoder and the decoder. The downstream policy for the action tokenizer is an autoregressive transformer with an `EOS` token for early termination. Following (Lee et al., 2024b), we train the autoregressive policy with a focal loss.

Detailed hyperparameters for LMP-π, LMP-`tok`, and autoregressive policy are presented in Table 5.

C.2 Baseline Details

In this section, we describe the details of baseline methods. All policy baselines use an observation history of 1 and action chunk of 16. We train all baselines for the same number of gradient steps as our method.

Diffusion Policy (Chi et al., 2024) & Large Behavior Model (Team et al., 2025) are generative control policies that model the expert's action distribution using a denoising diffusion process. Diffusion Policy is the single-task variant, and Large Behavior Model is the language-conditioned multi-task variant. We parameterize the noise prediction network as a bidirectional transformer with cross-attention conditioning on observations and language instructions. We use a 1B transformer and SigLip vision and language encoders. Training use 100 diffusion steps and inference uses 10 diffusion steps with DDIM sampler.

OpenVLA (Kim et al., 2024) is a generative control policy derived from Vision-Language Models (VLMs). Actions are discretized by binning and autoregressively generated conditioned on observations and language instructions. To ensure controlled comparison, we instantiate OpenVLA with a SigLip vision encoder and 1B Llama 3.2 LLM backbone with no vision-language pretraining, and finetune end-to-end on the robot datasets.

VQ-BeT (Lee et al., 2024b) is a generative control policy consisting of a residual VQ action tokenizer and an autoregressive transformer policy. We use the 2-layer residual VQ tokenizer from the official implementation of (Lee et al., 2024b) with a codebook size of 32 per layer. We then implement a 10-layer autoregressive transformer policy to regress the action tokens with cross-attention observation conditioning. Following (Lee et al., 2024b), we train an additional residual policy conditioned on the continuous output embedding (used to decode the first-layer action token) to correct errors from the quantized actions.

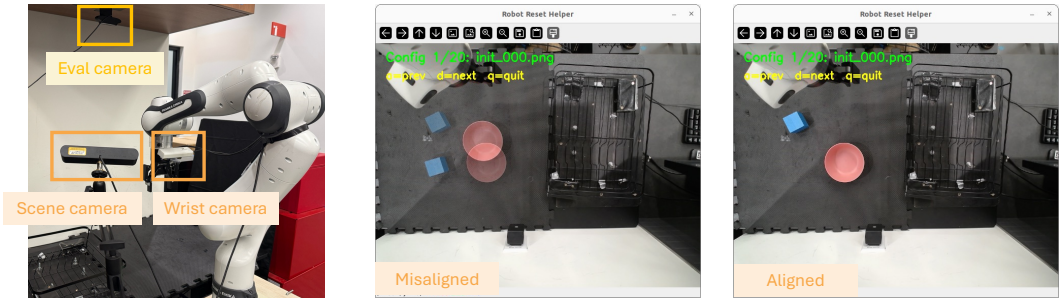
VAE Policy is a conditional Gaussian VAE (Kingma & Welling, 2014) used as a non-iterative generative control policy baseline. The encoder and decoder are parameterized by bidirectional transformers with 8 and 6 layers respectively, and the observation conditioning is passed in as cross-attention context. The encoder predicts a continuous mean and log standard deviation of a Gaussian distribution. The action decoder outputs the means and a learned scalar variance as in (Rybkin et al., 2021a). The model is trained end-to-end via reparameterization to optimize the ELBO. Instead of setting a fixed KL coefficient, we use a free-nats approach to constrain the KL within a set threshold of 0.1.

FAST (Pertsch et al., 2025) is a variable-length discrete action tokenizer. The tokenization process (1) performs a Discrete Cosine Transform to map an action chunk to a frequency domain matrix, (2) quantizes the matrix into integers, (3) flattens the matrix into a sequence, and (4) applies byte pair encoding to chunk the sequence into tokens. We use the official implementation of (Pertsch et al., 2025) with vocab size 1024 for the byte-pair encoding.

OAT (Liu et al., 2026) is a variable-length latent action tokenizer which encodes continuous action chunks autoregressively into sequences of discrete tokens. The method injects a coarse-to-fine inductive bias by randomly dropping out suffix tokens and forcing early tokens to reconstruct the full actions. We use the official implementation of (Liu et al., 2026) with vocab size 1000 and sequence length 8.

C.3 Environment Details

DROID (Khazatsky et al., 2024) (Fig. 11) is a single-arm manipulation platform based on a 7-DoF Franka robot. The image observations consist of two (180, 320) images from a scene camera and a wrist camera. The low-dimensional observations consist of 6-dimensional absolute joint positions and a 1-dimensional absolute gripper position, represented as a continuous value in [0, 1]. The actions consist of 6-dimensional delta joint positions and a 1-dimensional absolute gripper position. The controller runs at 10 Hz. To ensure reliable comparisons, we mount an overhead camera and use a program to track 20 initial configurations for each task. The program alpha-blends the pre-captured



(a) DROID Setup (b) Evaluation Tracker

Figure 11: Real-world experiment setup. We run real-world experiments on the DROID Franka platform, with one external camera and one wrist camera. We mount an overhead camera to track the initializations for a controlled comparison between methods.

snapshot and the live frame so that objects can be reset by aligning them. We evaluate on two zero-shot tasks roughly within the training distribution of the DROID dataset, and two challenging tasks where we collect 100 demonstrations each and finetune the policies.

1. **block-bowl (zero-shot)** involves picking up a cube block and placing it in a bowl. The objects are randomly initialized across the workspace. An episode is considered successful if the gripper releases the block into the bowl. Language instruction: "Pick up the block and put it in the bowl."
2. **marker-mug (zero-shot)** involves picking up a marker and dropping it in a mug. The objects are randomly initialized across the workspace. An episode is considered successful if the gripper releases the marker into the mug. Language instruction: "Pick up the marker and put it in the mug."
3. **peg-hole (finetuned)** is a high-precision manipulation task which involves inserting a square peg into a base with a square hole. The base is randomly initialized in a square region near the center of the workspace, and the peg is randomly initialized outside the square region. An episode is considered successful if the gripper inserts the peg securely into the base. Language instruction: "Insert the peg into the hole."
4. **clean-table (finetuned)** is a compositional manipulation task which involves placing 3 objects scattered around the workspace into a bowl. We collect demonstrations by randomly choosing 3 objects from a 7-object suite. We measure partial success of cleaning up at least 1, 2, and 3 objects. Language instruction: "Clean up the table."

LIBERO (Liu et al., 2023) is a simulated robotic benchmark designed to evaluate lifelong learning algorithms. The benchmark involves controlling a 7-DoF Franka Panda robot to complete various tasks across different scenes. The LIBERO-100 suite consists of 100 tasks distributed across three scenes (kitchen, living room, study), split into 90 training tasks (LIBERO-90) and 10 held-out tasks (LIBERO-10). The observations are (128, 128) images from a scene camera and a wrist camera, and the actions are 7-dimensional delta endeffector poses.

D3IL (Jia et al., 2024) is a simulated robotic benchmark designed to evaluate multimodal behaviors. The benchmark features a Franka Panda robot with task-specific endeffectors. We use tasks that provide vision data for our experiments. The observations are (96, 96) images from a scene camera and a wrist camera. The actions are absolute endeffector poses whose dimensionality varies across tasks. We describe the individual tasks below.

1. **avoiding** is an analytical task where the robot needs to navigate from one end of the table to the other while avoiding the rods arranged in a pyramidal shape. The gaps between the rods naturally create multimodality. The action space is 2-dimensional xy coordinates.
2. **aligning** involves aligning a hollow square shape to a target pose with a rod endeffector, where colored faces need to match. The action space is 3-dimensional xyz coordinates. The

strategy is to either push from outside the square, or insert the rod inside the square and pull. The dataset contains both pushing and pulling demonstrations.

3. **stacking** involves stacking three cubes (red, blue, green) in a target region using a parallel-jaw gripper. The action space is 8-dimensional endeffector pose. The dataset contains demonstrations of stacking the objects in different orders.
4. **sorting (2, 4, 6)** involves pushing red and blue colored blocks into corresponding red and blue colored bins using a rod endeffector. The 2, 4, and 6 variants have 1, 2, and 3 blocks for each color. The action space is 2-dimensional xy coordinates. The dataset contains demonstrations of sorting the objects in different orders.

Robomimic (Mandlekar et al., 2021) is a simulated robotic benchmark which involves controlling a Franka Panda arm to complete tasks of varying difficulty. We use the ph dataset for all our experiments. The actions are 7-dimensional delta endeffector poses for all single-arm tasks and 14-dimensional for bimanual tasks. The observations vary across tasks. We describe the individual tasks below.

1. **lift** involves lifting a cube off the table. The observations are (84, 84) images from a scene camera and a wrist camera.
2. **can** involves picking up a can from one table and putting it in the correct cell in the adjacent table. The observations are (84, 84) images from a scene camera and a wrist camera.
3. **square** involves picking up a "frame" piece with a square hole and dropping it on a square peg. The observations are (84, 84) images from a scene camera and a wrist camera.
4. **tool-hang** is a high-precision task that involves inserting a hanger on a narrow stand and hanging a wrench on the hanger. The observations are (240, 240) images from a scene camera and a wrist camera.
5. **transport** is a bimanual task involving a sequence of operations: the left arm lifts up the lid and hands the hammer to the right arm, while the right arm first moves a block from one bin to another, then receives the hammer and places it into a bin. The observations are (84, 84) images from one scene camera and two wrist cameras.

D Additional Experiments

D.1 Distribution of task success rates across LIBERO-90

In Fig. 12, we plot the per-task success rates over all 90 tasks in LIBERO-90 over 3 seeds. While both LMP-π and DP achieve near-perfect success in a majority of tasks, LMP-π tapers off at a slower rate, marking better generalization across task semantics and less cross-task interference.

D.2 Full single-task results

Table 6 presents the full numerical results on single-task benchmarks from Sec. 4.3. As discussed in Sec. 4.3, non-iterative baselines using a similar latent-space formulation fail to capture the granularity required to solve complex manipulation tasks. We additionally compare to Diffusion Policy (Chi et al., 2024), an iterative baseline which models the action generation process with a denoising diffusion process. We find DP to achieve comparable performance to LMP-π in single-task domains.

D.3 Full tokenizer results

Table 7 shows the full numerical results of tokenizer experiments on Robomimic and LIBERO tasks over 3 seeds. As discussed in Sec. 4.2, we pretrain the methods on a multi-task dataset and use them to train downstream autoregressive policies. For Robomimic, we train on the combined dataset of single-arm tasks: lift, can, square, tool-hang, and evaluate on the same tasks. For LIBERO, we train on the combined dataset of LIBERO-90 and evaluate on LIBERO-10. When controlling for the downstream policy, LMP-`tok` achieves significantly higher success rates than baselines.

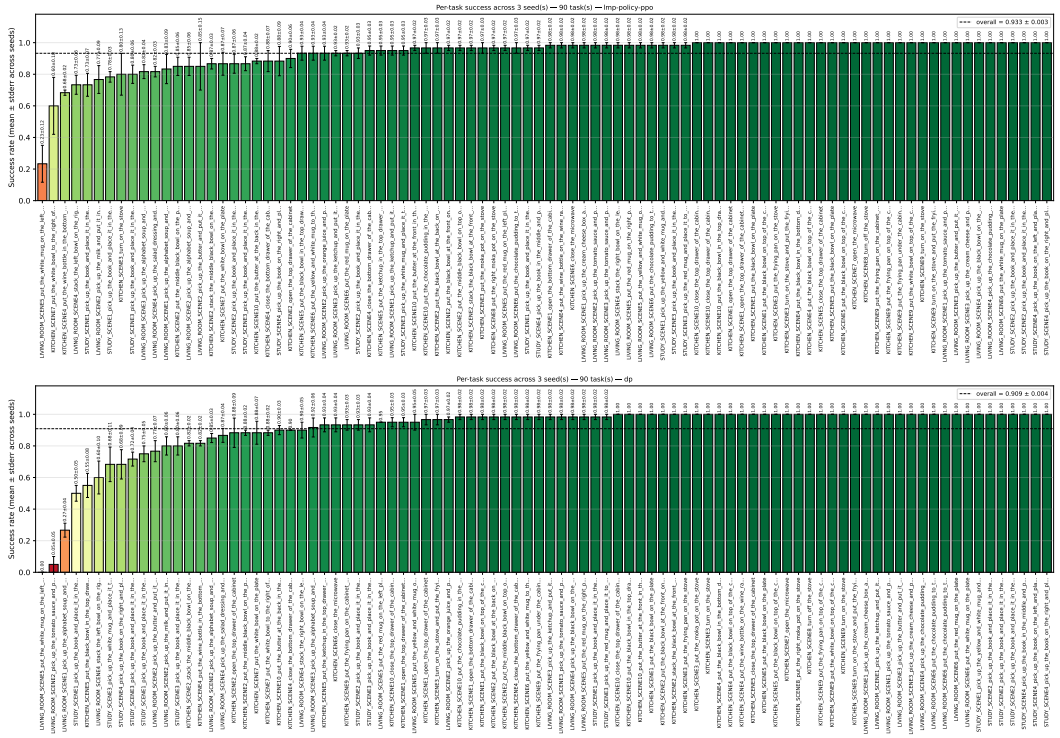


Figure 12: Distribution of single-task success rates over LIBERO-90. LMP-π (top) shows improved generalization across tasks compared to DP (bottom).

Table 6: Evaluation Results Across Robomimic and D3IL Tasks

Task	VQ-BET	VAE	DP	LMP-π (Ours)
d3il-aligning	0.23 $\pm$0.03	0.51 $\pm$0.07	**0.74 $\pm$0.04**	**0.71 $\pm$0.04**
d3il-stacking	0.53 $\pm$0.03	0.71 $\pm$0.07	0.81 $\pm$0.01	**0.86 $\pm$0.04**
d3il-sorting-2	0.46 $\pm$0.02	0.67 $\pm$0.13	0.75 $\pm$0.01	**0.80 $\pm$0.01**
d3il-sorting-4	0.41 $\pm$0.01	**0.68 $\pm$0.04**	**0.69 $\pm$0.05**	**0.68 $\pm$0.02**
d3il-sorting-6	0.32 $\pm$0.04	**0.70 $\pm$0.04**	0.68 $\pm$0.03	**0.75 $\pm$0.04**
d3il average	0.39	0.65	0.73	**0.76**
robomimic-lift	0.96 $\pm$0.00	0.97 $\pm$0.01	**1.00 $\pm$0.00**	**1.00 $\pm$0.00**
robomimic-can	0.83 $\pm$0.04	0.78 $\pm$0.07	**1.00 $\pm$0.00**	**1.00 $\pm$0.00**
robomimic-square	0.58 $\pm$0.06	0.79 $\pm$0.08	**0.92 $\pm$0.00**	0.87 $\pm$0.02
robomimic-tool-hang	0.15 $\pm$0.04	0.29 $\pm$0.06	**0.53 $\pm$0.07**	**0.51 $\pm$0.08**
robomimic-transport	0.26 $\pm$0.05	0.58 $\pm$0.12	**0.80 $\pm$0.10**	**0.75 $\pm$0.08**
robomimic average	0.56	0.68	**0.85**	0.83

D.4 Additional qualitative analysis of test-time compute

In Fig. 13, we construct an analytical experiment in d3il-stacking to qualitatively understand the allocation of test-time compute. We keep the gripper in an open position and slide a block through the open gripper, while continuously querying the policy for the average number of latent steps. As the block moves, the number of latent steps reaches the lowest point the moment the block is right underneath the gripper. This qualitatively shows a correlation between reasoning steps and potential gripper movement.

We provide additional visualizations of episodic reasoning traces in Fig. 16 and 17. We find that across a number of simulated and real-world tasks, the number of latent steps is low during grasping and releasing, and high during aligning and other movements. We did not observe as much latent

Table 7: Tokenizer Evaluation Results Across Robomimic and LIBERO Tasks.

Task	VQVAE	FAST	OAT	LMP-tok
robomimic-lift	0.82 ±0.06	0.98 ±0.02	0.97 ±0.02	**1.00 ±0.00**
robomimic-can	0.32 ±0.08	0.57 ±0.12	0.57 ±0.08	**0.93 ±0.02**
robomimic-square	0.02 ±0.02	0.42 ±0.10	0.32 ±0.08	**0.67 ±0.10**
robomimic-tool-hang	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	**0.17 ±0.02**
robomimic average	0.29	0.49	0.47	**0.69**
libero-soup-cheese	0.18 ±0.05	0.22 ±0.13	0.15 ±0.04	**0.92 ±0.05**
libero-mug-mug	0.28 ±0.02	0.10 ±0.04	0.23 ±0.02	**0.88 ±0.06**
libero-moka-moka	0.00 ±0.00	0.02 ±0.02	0.17 ±0.08	**0.63 ±0.08**
libero-bowl-drawer	0.02 ±0.02	0.37 ±0.09	0.60 ±0.07	**0.75 ±0.08**
libero-book-caddy	0.23 ±0.06	0.40 ±0.07	0.42 ±0.10	**0.57 ±0.06**
libero average	0.14	0.22	0.31	**0.75**

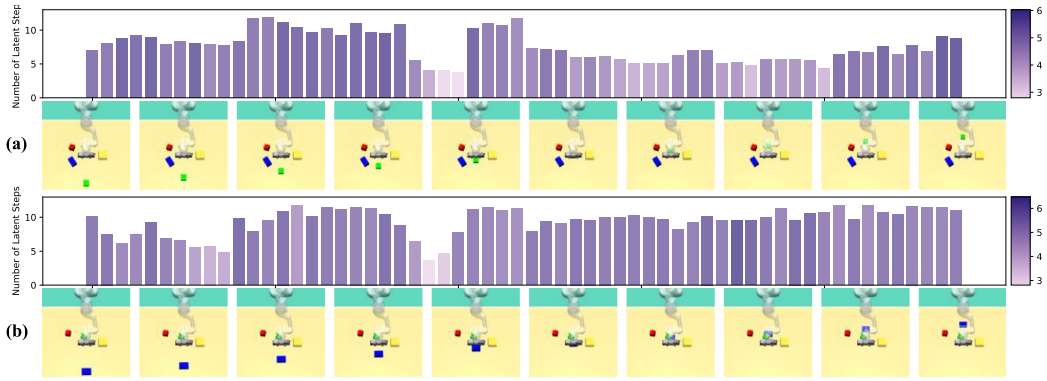


Figure 13: Analysis of test-time compute allocation. We manually move a block through an open gripper while querying the model at each observation. The number of latent steps reaches the lowest point as the block passes through the gripper.

step variation in simpler tasks such as Robomimic lift and can, or D3IL aligning and sorting where the endeffector is a peg.

D.5 Visualization of Training Dynamics

To understand the training dynamics of LMP-π, we plot the average number of reasoning steps against the number of gradient steps in Fig. 14. At the beginning of training, the model is randomly initialized, resulting in a uniform latent distribution with a high average length. Early in training, the model is dominated by the reconstruction objective, collapsing to short reasoning traces to myopically maximize the action log likelihood. As training progresses, the length of latent trace gradually increases as the model learns to utilize more latent steps to effectively reconstruct the action distribution. Notably, the training curve resembles DeepSeek R1 (DeepSeek-AI, 2025), where reasoning length increases over training. We also find that more aggressive decoder variance decay leads to shorter average reasoning lengths, confirming the effectiveness of compression as a length penalty.

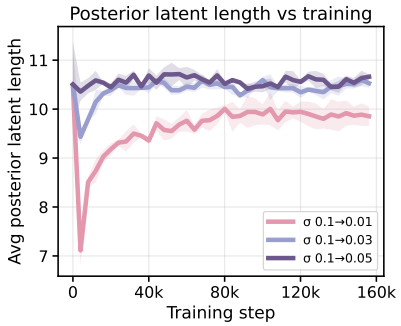


Figure 14: Average latent length vs training steps. The model uses more latent steps as training progresses.

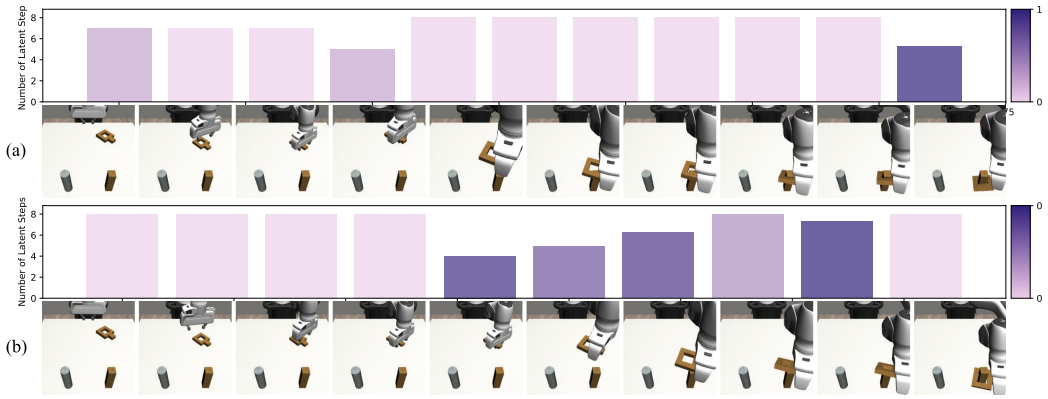


Figure 15: Allocation of action tokens in autoregressive policies trained on LMP-`tok`. (a) When LMP-`tok` is trained with a length penalty, the downstream autoregressive policy shows similar "reasoning" behavior to LMP-π where gripper movements correspond to fewer action tokens. (b) Removing the length penalty results in more arbitrary allocation of action tokens.

D.6 Token allocation in autoregressive policies trained on LMP-`tok`

In Fig. 15, we visualize the allocation of action tokens in autoregressive policies trained on LMP-`tok`. When LMP-`tok` is trained with a length penalty (σ decaying from 1.0-0.1), the downstream autoregressive policy behaves similarly to LMP-π, where gripper movements are associated with fewer action tokens. However, the allocation of action tokens is rather arbitrary when the tokenizer is trained without a length penalty (learned scalar σ). Since we do not find the length penalty to improve policy performance, we use the simpler scalar variance for our experiments in Sec. 4.2.

E Pseudocode

We provide the pseudocode for LMP-π in Alg. 1 and LMP-`tok` in Alg. 2.

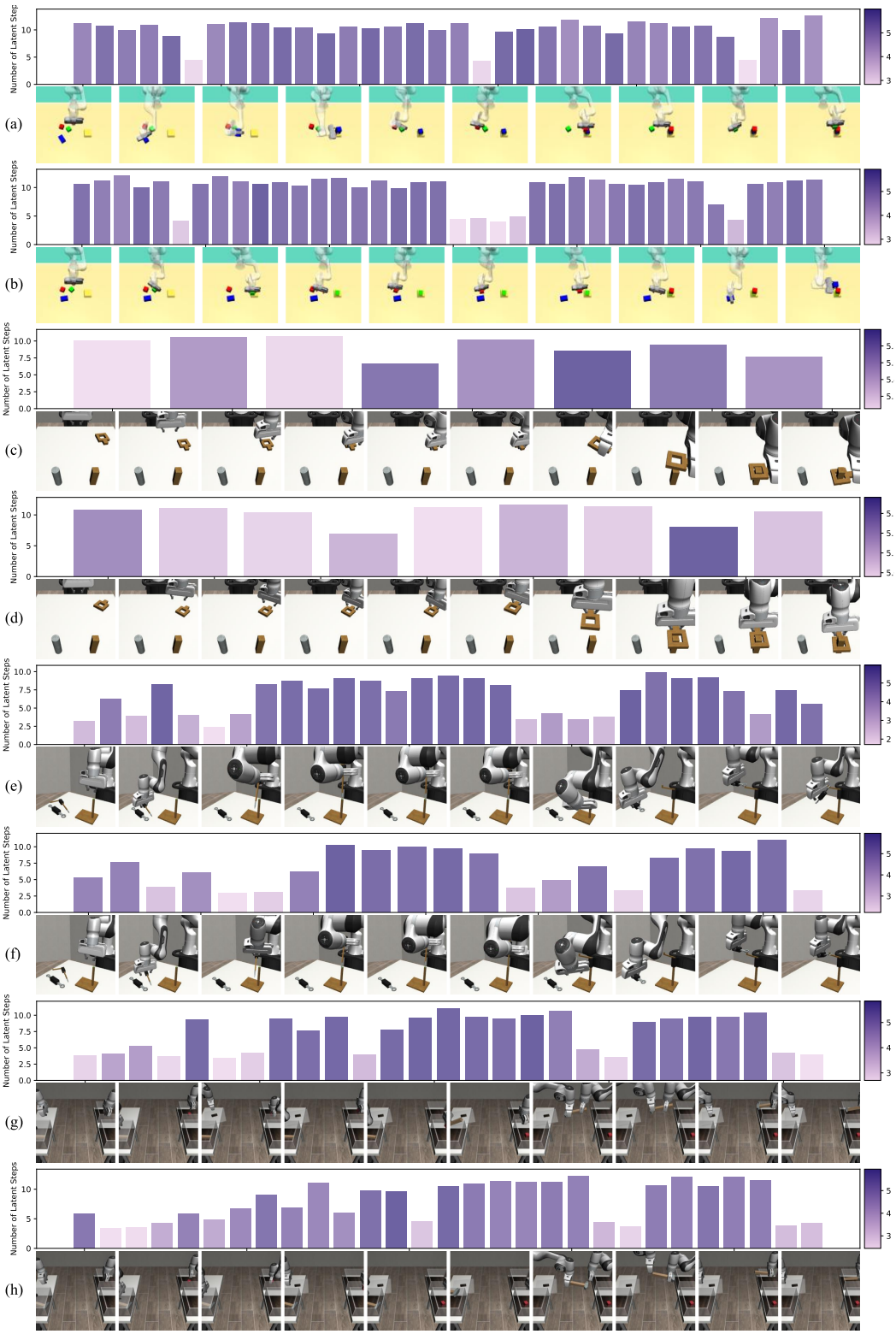


Figure 16: Visualization of test-time compute allocation in simulation tasks. Each timestep is averaged over 64 model queries. (a)-(b): d3il-stacking. (c)-(d): robomimic-square. (e)-(f) robomimic-tool-hang. (g)-(h): robomimic-transport.



Figure 17: Visualization of test-time compute allocation in real robot tasks. Each timestep is averaged over 64 model queries. (a)-(b) clean-table (finetuned). (c)-(d): peg-hole (finetuned). (e)-(f) block-bowl (zero-shot). (g)-(h): marker-mug (zero-shot).

Algorithm 1 Latent Memory Palace Policy (LMP-π)

input Expert dataset $\mathcal{D}_{\texttt{exp}}$
Initialize neural network parameters θ, ϕ.
while not converged **do**
  `// Collect on-policy rollouts`
  Draw $|\mathcal{D}_{\texttt{buf}}|$ expert samples $\{(o_i, a_i)\}_{i=1}^{|\mathcal{D}_{\texttt{buf}}|} \sim \mathcal{D}_{\texttt{exp}}$.
  for $i = 1 \ldots |\mathcal{D}_{\texttt{buf}}|$ **do**
    Sample latent traces from the posterior $z_i \sim q_\theta(\cdot|o_i, a_i)$
    Push z_i and the on-policy log likelihood $\log q_\theta^{\texttt{old}}(z_i|o_i, a_i)$ to buffer $\mathcal{D}_{\texttt{buf}}$.
  end for
  `// Update model`
  for epoch $e = 1 \ldots N$ **do**
    for minibatch $\{(o_j, a_j, z_j, \log q_\theta^{\texttt{old}}(z_j|o_j, a_j))\}_{j=1}^{\mathcal{B}} \sim \mathcal{D}_{\texttt{buf}}$ **do**
      Forward pass through the latent encoder to compute $\log q_\theta^{\texttt{new}}(z_j|o_j, a_j)$ and $\log p_\theta(z_j|o_j)$
      Forward pass through the action decoder to obtain $\log p_\phi(a_j|o_j, z_j)$
      Compute objective $J(\theta, \phi)$ from Eq. 4
      Update θ and ϕ using gradient descent.
    end for
  end for
end while

Algorithm 2 Latent Memory Palace Action Tokenizer (LMP-`tok`)

input Action dataset $\mathcal{D}_{\texttt{act}}$
Initialize neural network parameters θ, ϕ.
while not converged **do**
  `// Collect on-policy rollouts`
  Draw $|\mathcal{D}_{\texttt{buf}}|$ action samples $\{(a_i)\}_{i=1}^{|\mathcal{D}_{\texttt{buf}}|} \sim \mathcal{D}_{\texttt{act}}$.
  for $i = 1 \ldots |\mathcal{D}_{\texttt{buf}}|$ **do**
    Sample latent traces from the posterior $z_i \sim q_\theta(\cdot|a_i)$
    Push z_i and the on-policy log likelihood $\log q_\theta^{\texttt{old}}(z_i|a_i)$ to buffer $\mathcal{D}_{\texttt{buf}}$.
  end for
  `// Update model`
  for epoch $e = 1 \ldots N$ **do**
    for minibatch $\{(o_j, a_j, z_j, \log q_\theta^{\texttt{old}}(z_j|a_j))\}_{j=1}^{\mathcal{B}} \sim \mathcal{D}_{\texttt{buf}}$ **do**
      Forward pass through the latent encoder to compute $\log q_\theta^{\texttt{new}}(z_j|a_j)$
      Forward pass through the action decoder to obtain $\log p_\phi(a_j|z_j)$
      Compute objective $J(\theta, \phi)$ from Eq. 9
      Update θ and ϕ using gradient descent.
    end for
  end for
end while