Policy-Only Power Sampling for Vision-Language-Action Control

Jimin Park [1] **Wonjeong Choi** [1] **Jaekyun Moon** [1]

Abstract

We study whether training-free power-distribution sampling, originally proposed for large language model reasoning, can improve vision-language-action (VLA) control under distribution shift. In closed-loop control, actions affect future observations, so the powered next-decision conditional contains a future-correction term. We study this issue in a chunked-control regime used by high-performance VLAs, where the mismatch between local temperature scaling and trajectory-level power sampling persists at the chunk level. Because exact trajectory powering also sharpens simulator and observation randomness, we introduce *policy-only power sampling*, which powers only the policy while leaving rollout stochasticity unchanged. We then adapt Power-SMC to chunked replanning, where particles represent imagined action-chunk rollouts and importance weights depend only on policy log-probabilities. On ManiSkill3 pick-and-place out-of-distribution benchmarks with PPO-trained OpenVLA-OFT checkpoint, chunk-level Power-SMC more often reaches a successful state, more reliably remains successful through episode end, and does so with fewer executed actions until success. These results suggest that policy-only power sampling is a promising inference-time adaptation mechanism for VLA control under distribution shift.

1. Introduction

Vision-language-action (VLA) models increasingly benefit from post-training methods, including optimized fine-tuning and reinforcement-learning-based adaptation (Kim et al., 2025a; Zang et al., 2026; Liu et al., 2026). Many modern deployments also use action chunking to improve throughput, latency, or responsiveness (Kim et al., 2025a; Song et al., 2026; Black et al., 2026; Liang et al., 2026). Yet under new deployment shifts, performance improvements still require optimization, policy-training rollouts, or task-specific fine-tuning. In robotics this can be expensive or unsafe, especially when the shift is discovered at test time.

Recent LLM work has shown that *training-free* power-distribution sampling can improve generation quality beyond local temperature scaling, without updating model parameters (Karan & Du, 2026; Azizi et al., 2026). This raises a natural question for robotics: **can the same idea improve VLA control under distribution shift?**

The transfer is not immediate. In autoregressive LLMs, low-temperature decoding is already known not to sample from the sequence-level power target, motivating suffix-regeneration MH, Scalable Power Sampling, and Power-SMC (Karan & Du, 2026; Ji et al., 2026; Azizi et al., 2026). We ask whether the same power-distribution viewpoint remains useful in closed-loop VLA control.

In autoregressive LLMs, the model defines a prompt-conditioned distribution over token sequences, so future conditioning variables are generated tokens appended to the prefix. In many VLA settings, the policy instead defines a distribution over actions conditioned on the current observation and instruction, and after execution the next conditioning variable comes from the environment. This makes the natural power target a closed-loop trajectory distribution rather than a distribution over policy outputs alone.

Our starting point is therefore the exact power target over *VLA trajectories*. This implies a future-correction factor in the next-decision conditional. Our main idea is *policy-only power sampling*: because the exact target also sharpens transition and observation stochasticity, we instead work with a policy-only powered target that sharpens only the policy while leaving rollout stochasticity unchanged.

We instantiate this viewpoint with OpenVLA-OFT, a representative chunked VLA policy, in simulation. A suffix-regeneration Metropolis-Hastings (MH) sampler (Metropolis et al., 1953; Hastings, 1970) serves as an appendix proof of viability, while the main paper focuses on a chunk-level adaptation of Power-SMC, i.e., a Sequential Monte Carlo sampler for powered VLA planning (Azizi et al., 2026; Del Moral, 2004; Del Moral et al., 2006).

[1]Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea. Correspondence to: Jimin Park <gpfzl2@kaist.ac.kr>.

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning., Copyright 2026 by the author(s).

Our contributions are:

- We formulate exact trajectory-level power sampling for chunked VLA trajectories and derive the induced next-chunk conditional, which exposes the future-correction absent from pure low-temperature chunk sampling.
- We introduce *policy-only power sampling*, which sharpens only policy probabilities while leaving rollout stochasticity unchanged, and adapt Power-SMC to chunked OpenVLA-OFT planning.
- We provide simulator-based evidence that policy-only power sampling helps the policy find successful states more often, maintain success more reliably through episode end, and reach success with fewer actions. We use MH as an appendix proof-of-viability baseline and Power-SMC as the main evaluation method.

2. Related Work

Vision-language-action models. Robot foundation policies such as RT-1, RT-2, Octo, OpenVLA, and π_0 jointly model perception, language, and control (Brohan et al., 2023; Zitkovich et al., 2023; Team et al., 2024; Kim et al., 2025b; Black et al., 2025). We build on OpenVLA-OFT, a strong post-trained OpenVLA variant whose chunked action head supports efficient inference (Kim et al., 2025a).

Adaptation and inference-time scaling for VLA control. Recent work adapts VLAs under distribution shift through RL-based policy updates (Zang et al., 2026; Li et al., 2026; Liu et al., 2026), changing the policy itself through additional optimization. Concurrent work also studies inference-time scaling for VLA control using test-time signals such as learned verification and model-internal uncertainty (Kwok et al., 2025; Jang et al., 2026), re-ranking or filtering candidate actions with auxiliary inference-time scores. Like those inference-time approaches, our method keeps model weights fixed and acts only at inference time, but it instead reshapes the policy's own sequence distribution through training-free power sampling over chunked trajectories.

Power sampling for LLMs. Reasoning with Sampling introduced training-free power-distribution decoding for LLMs, and follow-up work developed faster approximations to the same target, including Scalable Power Sampling and Power-SMC (Karan & Du, 2026; Ji et al., 2026; Azizi et al., 2026). We adapt this line from autoregressive text generation to chunked closed-loop VLA control.

3. Methodology

Bars denote replanning-boundary quantities (for example, $\bar{s}_n$ and $\bar{x}_n$), tildes denote the policy-only powered target, and superscripts (i) index SMC particles. Let A_n denote the action chunk sampled at replanning boundary n, let ξ_n denote the resulting rollout segment, and let $T_n(\xi_n \mid \bar{s}_n, A_n, c)$ be the conditional density of that segment under the simulator dynamics and observation process.

3.1. Theoretical Formulation of Power Targets

Trajectory model. Given instruction c, we write a chunked VLA trajectory as

$$\tau = (\bar{s}_0, \bar{x}_0, A_0, \xi_0, \ldots, A_{N-1}, \xi_{N-1}),$$

with closed-loop density

$$p_\theta(\tau \mid c) = \rho(s_0)\, O(x_0 \mid s_0, c) \times \prod_{n=0}^{N-1} q_\theta(A_n \mid \bar{x}_n, c) T_n(\xi_n \mid \bar{s}_n, A_n, c).$$

Here $\rho(s_0)$ is the initial-state distribution, $O(x_0 \mid s_0, c)$ is the initial observation model given state s_0 and instruction c, and $q_\theta(A_n \mid \bar{x}_n, c)$ is the OpenVLA-OFT chunk policy over action chunks conditioned on the boundary observation. Equivalently, T_n absorbs the state transitions and intermediate observations generated while executing A_n from boundary state $\bar{s}_n$, so O denotes exogenous sensing while q_θ is the learned control policy.

Power targets. Conceptually, the exact trajectory-level power target is

$$\Pi_\alpha(\tau \mid c) \propto p_\theta(\tau \mid c)^\alpha,$$

which sharpens not only the policy but also rollout and observation stochasticity. In the main text, we instead use the policy-only powered target

$$\widetilde{p}_{\theta,\alpha}(\tau \mid c) := \rho(s_0) O(x_0 \mid s_0, c) \times \prod_{n=0}^{N-1} q_\theta(A_n \mid \bar{x}_n, c)^\alpha T_n(\xi_n \mid \bar{s}_n, A_n, c).$$

which sharpens only the policy terms and leaves rollout stochasticity unchanged. This matters because the exact target would reward not only policy choices but also favorable simulator realizations and observation noise. The policy-only target instead asks which chunks remain preferable after integrating over unchanged rollout stochasticity, so inference-time sharpening acts on controllable policy randomness rather than exogenous randomness.

This choice is also computationally useful. Once rollout stochasticity is left unpowered, the sampler only needs policy probabilities to reweight particles, rather than likelihood ratios for simulator randomness or observation noise. That makes the powered target much easier to approximate within an existing closed-loop evaluation stack.

Next-chunk conditional under the policy-only target. The policy-only target induces the next-chunk conditional

$$\widetilde{\Pi}_\alpha(A_n \mid \bar{s}_n, \bar{x}_n, c) \propto q_\theta(A_n \mid \bar{x}_n, c)^\alpha \widetilde{\Gamma}_n^\alpha(\bar{s}_n, \bar{x}_n, A_n, c),$$

where $\widetilde{\Gamma}_n^\alpha$ is a one-chunk lookahead correction over rollout-consistent future continuations. Intuitively, this term measures how much powered future trajectory mass remains after executing chunk A_n and reaching the next replanning boundary. Hence low-temperature chunk sampling is not exact trajectory-level power sampling: it sharpens the local policy term but omits the future correction. Exact sampling from $\widetilde{\Pi}_\alpha$ is still intractable because computing $\widetilde{\Gamma}_n^\alpha$ requires integrating over all future closed-loop continuations. Appendix A.1 gives the detailed derivation of this conditional.

3.2. Chunk-Level Power-SMC

Proposal choice. We therefore study approximate inference-time samplers. We keep suffix-regeneration MH as a supplementary proof-of-viability baseline; Appendix B and Appendix D give its proposal ratio and procedure, and Appendix J reports the broader sweep. The main text instead focuses on chunk-level Power-SMC because one OpenVLA-OFT policy call already produces a full action chunk.

For the current quantized OpenVLA-OFT head, one chunk contains K low-level actions of dimension d_{act}, so it can be flattened into $J := Kd_{\text{act}}$ quantized action-token slots. Writing $A_m = a = (a_1, \ldots, a_J)$ for a candidate chunk, where a_j is the discrete token chosen at slot j, and $z_{m,j}(\cdot \mid \bar{x}_m, c)$ for the corresponding pre-softmax logits at boundary m, the untruncated chunk pmf factorizes as

$$q_\theta(A_m = a \mid \bar{x}_m, c) = \prod_{j=1}^{J} \operatorname{softmax}\big(z_{m,j}(\cdot \mid \bar{x}_m, c)\big)(a_j).$$

Raising this pmf to the power α preserves the factorization and rescales logits by α. Hence the locally variance-minimizing proposal is exactly the temperature-scaled chunk policy with proposal temperature $\beta = 1/\alpha$ before any top-k or top-p truncation; Appendix A.3 gives the quantized chunk-head parameterization, Appendix A.4 derives the local-optimality claim, and Appendix A.5 shows why it takes this temperature form.

Prefix target. Fix the replanning boundary and re-index the local planning problem so that it is step 0. For a P-chunk planning horizon, define the particle prefix

$$\Omega_{0:m-1} := \big((A_0, \xi_0), \ldots, (A_{m-1}, \xi_{m-1})\big),$$

and the corresponding unnormalized Power-SMC target

$$\widetilde{\gamma}_m(\Omega_{0:m-1} \mid c) := \prod_{j=0}^{m-1} q_\theta(A_j \mid \bar{x}_j, c)^\alpha T_j(\xi_j \mid \bar{s}_j, A_j, c),$$

with $\widetilde{\gamma}_0 := 1$. Let

$$M_m(A_m, \xi_m \mid \bar{s}_m, \bar{x}_m, c) := q_{\theta,\beta}(A_m \mid \bar{x}_m, c) \times T_m(\xi_m \mid \bar{s}_m, A_m, c).$$

Here $q_{\theta,\beta}$ is the tempered chunk proposal and M_m is the one-step proposal kernel that extends a particle by one chunk.

Weighting and resampling. The incremental importance weight for that extension is

$$\omega_m(A_m, \xi_m) := \frac{\widetilde{\gamma}_{m+1}(\Omega_{0:m} \mid c)}{\widetilde{\gamma}_m(\Omega_{0:m-1} \mid c)\, M_m} = \frac{q_\theta(A_m \mid \bar{x}_m, c)^\alpha}{q_{\theta,\beta}(A_m \mid \bar{x}_m, c)}.$$

The rollout terms cancel because the policy-only target leaves T_m unpowered. Therefore particle i updates its unnormalized weight by

$$\widetilde{W}_{m+1}^{(i)} = \widetilde{W}_m^{(i)} \cdot \frac{q_\theta(A_m^{(i)} \mid \bar{x}_m^{(i)}, c)^\alpha}{q_{\theta,\beta}(A_m^{(i)} \mid \bar{x}_m^{(i)}, c)}.$$

This cancellation is practically important: chunk-level Power-SMC can treat the simulator only as a forward generator of imagined rollouts, without evaluating transition or observation densities, while the reweighting factors depend only on base-policy and tempered-policy log probabilities. Appendix A.2 compares this incremental weight with the corresponding full-target update. Thus each chunk is reweighted only by a policy log-probability ratio.

With N particles, we normalize weights, write $W_m^{(i)}$ for the normalized weight of particle i after m chunks, monitor $\text{ESS}_m := \left(\sum_i (W_m^{(i)})^2\right)^{-1}$ at scheduled checks, and resample when the monitored ESS falls below the threshold ratio κN. We also use exact alpha-ramping: a stage index ℓ first propagates particles under milder exponents $\alpha^{(\ell)}$ and then reweights them exactly to the sharper target, reducing early particle collapse when the powered policy is much sharper than the base policy; Appendix A.6 gives the exact update. The final plan is drawn from the terminal weighted particle set. Figure 1 illustrates the planning loop, and Appendix C gives pseudocode.

4. Experiments

4.1. Setup

We evaluate on the ManiSkill3 pick-and-place benchmark used in RLinf-VLA (Liu et al., 2026; Zang et al., 2026; Tao et al., 2025). The benchmark contains one in-distribution training setting and three out-of-distribution (OOD) axes that stress different failure modes of VLA control. The *vision* axis includes held-out table appearances together with dynamic texture and dynamic noise corruptions. The *semantic* axis includes unseen objects, unseen receptacles, unseen instruction phrasings, and multi-object or multi-receptacle compositions. The *execution* axis perturbs physical rollout conditions through unseen placements, randomized robot initial poses, and mid-episode object repositioning.

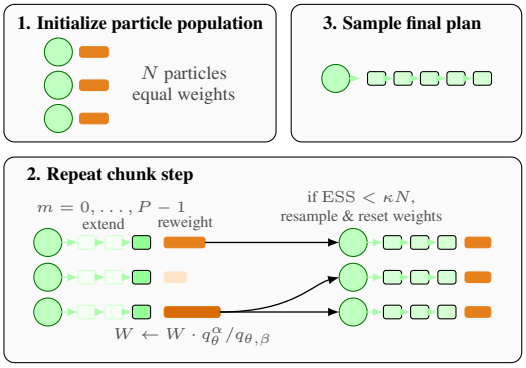

Figure 1. Chunk-level Power-SMC at one replanning boundary. The sampler initializes a weighted particle population, repeatedly extends every particle by one chunk, updates weights, and resamples whenever a scheduled ESS check falls below threshold. Only after the P-chunk loop ends is one final plan sampled from the weighted particle set. Appendix C gives the precise pseudocode.

Our base policy is the PPO-trained OpenVLA-OFT checkpoint released with RLinf-VLA. Following the RLinf-VLA evaluation protocol of Liu et al. (2026), baseline decoding uses evaluation temperature 0.6. We report four metrics: `success_once`, the fraction of episodes that ever reach success and therefore measures whether the policy can find a successful state at all; `success_at_end`, the fraction that remain successful at the final timestep and distinguishes stable completion from transient success; `Actions`, the average number of executed actions until the first success event, so lower values indicate more efficient success; and `Latency`, the average per-episode wall-clock time normalized so that baseline decoding equals 1.0, so lower values indicate lower inference overhead. This separation matters because the benchmark allows transient success.

Each episode has horizon 80. We evaluate 16 settings: 1 training and 15 OOD variants (5 vision, 7 semantic, and 3 execution), with 128 closed-loop episodes each (2,048 total). OOD averages are axis-balanced over the three OOD groups. To avoid per-shift tuning, all Power-SMC results use one fixed configuration across all 16 settings: $P = 10$, $N = 64$, $\alpha = 2.5$, $\kappa = 0.5$, seven alpha-ramping blocks, and ESS checks every 2 blocks. We do not claim this configuration is optimal; compute-quality sweeps over N, α, and planning horizon P are left to future work. Appendix E and Table A1 give the remaining details.

4.2. Main Results

Table 1 shows essentially unchanged in-distribution performance but clear OOD gains. In the training setting, Power-SMC leaves performance close to baseline, suggesting that these OOD improvements are not obtained by sacrificing nominal behavior. On the OOD average, it improves `success_once`/`success_at_end` from 0.782/0.578 to 0.818/0.641 and lowers Actions from 33.75 to 28.97. While all three metrics improve, the strongest aggregate gains appear in `success_at_end` and Actions, suggesting that Power-SMC not only finds successful states more often but also reaches them more stably and efficiently.

Table 1. Main benchmark summary; metrics are defined in Section 4.1. Base/LT values denote decoding temperature; LT denotes low-temperature decoding. LT 0.4 matches Power-SMC's final local proposal temperature ($\beta = 1/\alpha = 0.4$ for $\alpha = 2.5$) but omits closed-loop branching, importance reweighting, and resampling.

Split	Metric	Base 0.6	LT 0.4	LT 0.25	P-SMC
Training (IND)	Success@once ↑	0.984	**0.992**	0.969	0.984
	Success@end ↑	0.844	**0.859**	0.797	0.828
	Actions ↓	23.27	23.79	23.35	**22.16**
	Latency ↓	1.000	**0.937**	0.995	2.975
Vision (OOD)	Success@once ↑	0.936	0.927	0.931	**0.942**
	Success@end ↑	0.725	0.734	0.744	**0.766**
	Actions ↓	24.78	24.34	24.67	**22.90**
	Latency ↓	**1.000**	1.001	1.005	3.073
Semantic (OOD)	Success@once ↑	0.669	0.657	0.642	**0.703**
	Success@end ↑	0.478	0.473	0.455	**0.523**
	Actions ↓	37.75	37.86	37.25	**32.70**
	Latency ↓	1.000	**0.986**	0.988	3.106
Execution (OOD)	Success@once ↑	0.742	0.734	0.719	**0.807**
	Success@end ↑	0.531	0.534	0.563	**0.633**
	Actions ↓	38.71	39.16	38.94	**31.31**
	Latency ↓	1.000	**0.999**	1.004	3.145
OOD average	Success@once ↑	0.782	0.773	0.764	**0.818**
	Success@end ↑	0.578	0.580	0.587	**0.641**
	Actions ↓	33.75	33.79	33.62	**28.97**
	Latency ↓	1.000	**0.995**	0.999	3.108

The largest aggregate gains appear on the semantic and execution OOD axes, where locally plausible chunks more often lead to unstable downstream states and therefore benefit most from future-aware branching and reweighting. Taken together, these OOD trends suggest better trajectory selection rather than mere diversification.

Appendix F provides axis-wise breakdowns and a same-setting different-seed replicate, while Appendix G analyzes when gains are large or limited and Appendix J shows a similar stability-and-efficiency pattern in a supplementary serial MH sweep beyond direct low-temperature decoding. Runtime remains the main drawback at about 3× baseline decoding under the normalized latency metric; Appendix H and Appendix I discuss the tradeoff.

5. Conclusion

Policy-only power sampling improves chunked VLA control under fixed policy weights: the policy-only target and chunk-level Power-SMC improve reachability, end-state robustness, and efficiency in ManiSkill3 beyond local temperature tuning, without retraining or external verification. The method uses simulator rollouts only as inference-time imagined futures rather than policy-training rollouts or external reward labels, so direct real-world deployment would require a reliable world model or another rollout surrogate; future work should approximate future correction, reduce simulator reliance (Wan et al., 2025; Jiang et al., 2026), and develop better-matched post-training objectives.

Impact Statement

This paper studies training-free inference-time control adaptation for simulated VLA models. Potential positive impacts include improving robustness under distribution shift without additional policy training, robot data collection, or repeated environment interaction, which could lower the cost of adapting VLA policies after deployment. Potential risks arise if sharper inference-time action selection is deployed in real robots without safety constraints, because overconfident chunk selection under partial observability could amplify failures and reduce opportunities for conservative fallback behavior. Our results are limited to simulation and should not be interpreted as evidence for unconstrained real-world deployment; any practical use would require task-specific safety checks, human oversight, and evaluation beyond the present benchmark.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (Nos. RS-2024-00340966 and RS-2024-00408003), and by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00444862).

References

Azizi, S., Potraghloo, E. B., Ahmadi, M., Kundu, S., and Pedram, M. Power-smc: Low-latency sequence-level power sampling for training-free llm reasoning. *arXiv preprint arXiv:2602.10273*, 2026.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M. R., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L. X., Smith, L., Tanner, J., Vuong, Q., Walling, A., Wang, H., and Zhilinsky, U. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Black, K., Galliker, M. Y., and Levine, S. Real-time execution of action chunking flow policies. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=UkR2zO5uww.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2023.

Del Moral, P. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer, New York, NY, 2004. ISBN 978-0-387-20268-6. doi: 10.1007/978-1-4684-9393-1.

Del Moral, P., Doucet, A., and Jasra, A. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006. doi: https://doi.org/10.1111/j.1467-9868.2006.00553.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2006.00553.x.

Hastings, W. K. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57 (1):97–109, 1970. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2334940.

Jang, S., Kim, D., Kim, C., Kim, Y., and Shin, J. Verifier-free test-time sampling for vision-language-action models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=UD4Rw8MOEK. Poster.

Ji, X., Tutunov, R., Zimmer, M., and Ammar, H. B. Scalable power sampling: Unlocking efficient, training-free reasoning for llms via distribution sharpening. *arXiv preprint arXiv:2601.21590*, 2026.

Jiang, Z., Zhou, S., Jiang, Y., Huang, Z., Wei, M., Chen, Y., Zhou, T., Guo, Z., Lin, H., Zhang, Q., Wang, Y., Li, H., Yu, C., and Zhao, D. WoVR: World models as reliable simulators for post-training vla policies with rl. *arXiv preprint arXiv:2602.13977*, 2026.

Karan, A. and Du, Y. Reasoning with sampling: Your base model is smarter than you think. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Vsgq2ldr4K. Oral.

Kim, M. J., Finn, C., and Liang, P. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025a.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E. P., Sanketi, P. R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., and Finn, C. Openvla: An open-source vision-language-action model. In Agrawal, P., Kroemer, O., and Burgard, W. (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Kwok, J., Agia, C., Sinha, R., Foutter, M., Li, S., Stoica, I., Mirhoseini, A., and Pavone, M. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In Lim, J., Song, S., and Park, H.-W. (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3200–3217. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/kwok25a.html.

Li, H., Zuo, Y., Yu, J., Zhang, Y., Zhaohui, Y., Zhang, K., Zhu, X., Zhang, Y., Chen, T., Cui, G., Wang, D., Luo, D., Fan, Y., Sun, Y., Zeng, J., Pang, J., Zhang, S., Wang, Y., Mu, Y., Zhou, B., and Ding, N. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TQhSodCM4r. Poster.

Liang, Y., Wang, X., Wang, K., Wang, S., Peng, X., Chen, H., Chua, D. K. H., and Vadakkepat, P. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

Liu, J., Gao, F., Wei, B., Chen, X., Liao, Q., Wu, Y., Yu, C., and Wang, Y. What can RL bring to VLA generalization? an empirical study. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=qmBMPInbZC. Poster.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114.

Song, W., Chen, J., Ding, P., Zhao, H., Zhao, W., Zhong, Z., Ge, Z., Li, Z., Wang, D., Ma, J., Wang, L., and Li, H. Pd-vla: Accelerating vision-language-action model integrated with action chunking via parallel decoding. *arXiv preprint arXiv:2503.02310*, 2026.

Tao, S., Xiang, F., Shukla, A., Qin, Y., Hinrichsen, X., Yuan, X., Bao, C., Lin, X., Liu, Y., kai Chan, T., Gao, Y., Li, X., Mu, T., Xiao, N., Gurha, A., Rajesh, V. N., Choi, Y. W., Chen, Y.-R., Huang, Z., Calandra, R., Chen, R., Luo, S., and Su, H. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2025.

Team, O. M., Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y. L., Chen, L. Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.-W., Chen, D., Yu, F., Zhao, H., Yang, J., Zeng, J., Wang, J., Zhang, J., Zhou, J., Wang, J., Chen, J., Zhu, K., Zhao, K., Yan, K., Huang, L., Feng, M., Zhang, N., Li, P., Wu, P., Chu, R., Feng, R., Zhang, S., Sun, S., Fang, T., Wang, T., Gui, T., Weng, T., Shen, T., Lin, W., Wang, W., Wang, W., Zhou, W., Wang, W., Shen, W., Yu, W., Shi, X., Huang, X., Xu, X., Kou, Y., Lv, Y., Li, Y., Liu, Y., Wang, Y., Zhang, Y., Huang, Y., Li, Y., Wu, Y., Liu, Y., Pan, Y., Zheng, Y., Hong, Y., Shi, Y., Feng, Y., Jiang, Z., Han, Z., Wu, Z.-F., and Liu, Z. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Zang, H., Wei, M., Xu, S., Wu, Y., Guo, Z., Wang, Y., Lin, H., Wang, P., Shi, L., Xie, Y., Xu, Z., Liu, Z., Chen, K., Tang, W., Zhang, Q., Zhang, W., Yu, C., and Wang, Y. Rlinf-vla: A unified and efficient framework for reinforcement learning of vision-language-action models. *arXiv preprint arXiv:2510.06710*, 2026.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P. R., Salazar, G., Ryoo, M. S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T.-W. E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N. J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K. A., Driess, D., Ding, T., Choromanski, K. M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M. G., and Han, K. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Tan, J., Toussaint, M., and Darvish, K. (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A. Power-SMC Theory and Implementation

A.1. Full and Policy-Only Next-Chunk Conditionals

Here Π_α denotes the normalized full-trajectory power target and $Z_\alpha(c)$ its normalizing constant. The full chunked trajectory density is

$$p_\theta(\tau \mid c) = \rho(s_0)\, O(x_0 \mid s_0, c) \prod_{n=0}^{N-1} q_\theta(A_n \mid \bar{x}_n, c) T_n(\xi_n \mid \bar{s}_n, A_n, c).$$

The corresponding full trajectory-level power target is

$$\Pi_\alpha(\tau \mid c) = \frac{p_\theta(\tau \mid c)^\alpha}{Z_\alpha(c)}.$$

For a replanning boundary n, define the tail trajectory

$$\tau_{n:}^{\text{chunk}} := (A_n, \xi_n, A_{n+1}, \xi_{n+1}, \ldots, A_{N-1}, \xi_{N-1}).$$

Under the full trajectory-level target, the next-chunk conditional is proportional to

$$q_\theta(A_n \mid \bar{x}_n, c)^\alpha \Gamma_n^\alpha(\bar{s}_n, \bar{x}_n, A_n, c),$$

where the future power value at the next boundary is

$$V_{n+1}^\alpha(\bar{s}_{n+1}, \bar{x}_{n+1}, c) := \int p_\theta^\alpha(\tau_{n+1:}^{\text{chunk}} \mid \bar{s}_{n+1}, \bar{x}_{n+1}, c)\, d\tau_{n+1:}^{\text{chunk}},$$

and the resulting one-chunk lookahead correction is

$$\Gamma_n^\alpha(\bar{s}_n, \bar{x}_n, A_n, c) := \int \left[\prod_{k=0}^{K-1} P_1^\alpha(s_{nK+k+1} \mid s_{nK+k}, a_{nK+k}) O^\alpha(x_{nK+k+1} \mid s_{nK+k+1}, c) \right] V_{n+1}^\alpha(\bar{s}_{n+1}, \bar{x}_{n+1}, c)\, d\xi_n.$$

This is the exact full-trajectory analogue of power sampling, but it also sharpens the transition and observation terms. In the main text we instead use the policy-only target

$$\widetilde{p}_{\theta,\alpha}(\tau \mid c) := \rho(s_0) O(x_0 \mid s_0, c) \prod_{n=0}^{N-1} q_\theta(A_n \mid \bar{x}_n, c)^\alpha T_n(\xi_n \mid \bar{s}_n, A_n, c),$$

which is more natural when observation stochasticity is treated as exogenous nuisance randomness. The corresponding policy-only future value is

$$\widetilde{V}_{n+1}^\alpha(\bar{s}_{n+1}, \bar{x}_{n+1}, c) := \int \widetilde{p}_{\theta,\alpha}(\tau_{n+1:}^{\text{chunk}} \mid \bar{s}_{n+1}, \bar{x}_{n+1}, c)\, d\tau_{n+1:}^{\text{chunk}},$$

and the policy-only correction becomes

$$\widetilde{\Gamma}_n^\alpha(\bar{s}_n, \bar{x}_n, A_n, c) := \int T_n(\xi_n \mid \bar{s}_n, A_n, c)\, \widetilde{V}_{n+1}^\alpha(\bar{s}_{n+1}, \bar{x}_{n+1}, c)\, d\xi_n.$$

Hence

$$\widetilde{\Pi}_\alpha(A_n \mid \bar{s}_n, \bar{x}_n, c) \propto q_\theta(A_n \mid \bar{x}_n, c)^\alpha \widetilde{\Gamma}_n^\alpha(\bar{s}_n, \bar{x}_n, A_n, c).$$

This makes explicit why low-temperature chunk sampling is not exact trajectory-level power sampling: it sharpens the local policy term but omits the future correction.

A.2. Full and Policy-Only Incremental Weights

To isolate the computational effect of policy-only powering, fix the same local planning re-indexing as in the main text and write

$$\Omega_{0:m-1} := \big((A_0, \xi_0), \ldots, (A_{m-1}, \xi_{m-1})\big).$$

Consider the common one-step proposal

$$M_m(A_m, \xi_m \mid \bar{s}_m, \bar{x}_m, c) := q_{\theta,\beta}(A_m \mid \bar{x}_m, c)\, T_m(\xi_m \mid \bar{s}_m, A_m, c),$$

which first samples a chunk from the tempered policy and then rolls the simulator forward for one chunk.

If we target the exact full-trajectory power distribution, the corresponding unnormalized prefix target is

$$\gamma_m^{\alpha}(\Omega_{0:m-1} \mid c) := \prod_{j=0}^{m-1} q_\theta(A_j \mid \bar{x}_j, c)^{\alpha} T_j(\xi_j \mid \bar{s}_j, A_j, c)^{\alpha}.$$

The resulting incremental importance weight is

$$\begin{aligned}
\omega_m^{\text{full}}(A_m, \xi_m) &:= \frac{\gamma_{m+1}^{\alpha}(\Omega_{0:m} \mid c)}{\gamma_m^{\alpha}(\Omega_{0:m-1} \mid c)\, M_m} \\
&= \frac{q_\theta(A_m \mid \bar{x}_m, c)^{\alpha} T_m(\xi_m \mid \bar{s}_m, A_m, c)^{\alpha}}{q_{\theta,\beta}(A_m \mid \bar{x}_m, c)\, T_m(\xi_m \mid \bar{s}_m, A_m, c)} \\
&= \frac{q_\theta(A_m \mid \bar{x}_m, c)^{\alpha}}{q_{\theta,\beta}(A_m \mid \bar{x}_m, c)} T_m(\xi_m \mid \bar{s}_m, A_m, c)^{\alpha-1}.
\end{aligned}$$

Hence the full-target update requires evaluating the chunk-rollout density itself. Expanding the chunk density into low-level transitions and observations gives

$$T_m(\xi_m \mid \bar{s}_m, A_m, c)^{\alpha-1} = \prod_{k=0}^{K-1} P_1(s_{mK+k+1} \mid s_{mK+k}, a_{mK+k})^{\alpha-1} O(x_{mK+k+1} \mid s_{mK+k+1}, c)^{\alpha-1},$$

so every imagined rollout must carry not only sampled states and observations, but also their transition and observation densities.

Under the policy-only target from the main text, the unnormalized prefix target is instead

$$\widetilde{\gamma}_m(\Omega_{0:m-1} \mid c) := \prod_{j=0}^{m-1} q_\theta(A_j \mid \bar{x}_j, c)^{\alpha} T_j(\xi_j \mid \bar{s}_j, A_j, c),$$

and the same proposal gives

$$\begin{aligned}
\widetilde{\omega}_m(A_m, \xi_m) &:= \frac{\widetilde{\gamma}_{m+1}(\Omega_{0:m} \mid c)}{\widetilde{\gamma}_m(\Omega_{0:m-1} \mid c)\, M_m} \\
&= \frac{q_\theta(A_m \mid \bar{x}_m, c)^{\alpha} T_m(\xi_m \mid \bar{s}_m, A_m, c)}{q_{\theta,\beta}(A_m \mid \bar{x}_m, c)\, T_m(\xi_m \mid \bar{s}_m, A_m, c)} \\
&= \frac{q_\theta(A_m \mid \bar{x}_m, c)^{\alpha}}{q_{\theta,\beta}(A_m \mid \bar{x}_m, c)}.
\end{aligned}$$

The rollout term cancels exactly. This is the precise computational advantage of policy-only powering for chunk-level Power-SMC: the simulator can be used purely as a forward generator of imagined rollouts, while all reweighting depends only on base-policy and tempered-policy log probabilities.

A.3. Quantized Chunk-Head Instantiation

For the quantized chunk head used in the OpenVLA-OFT and RLinf-VLA setting considered here (Kim et al., 2025a; Zang et al., 2026), one chunk is represented by a finite tuple of quantized action-token values. Let $J := Kd_{\text{act}}$ be the number of action-token slots in one chunk, and let $N_{\text{bin}} :=$ `n_action_bins` be the number of valid bins per slot. At boundary observation $\bar{x}_n$, the action head emits masked logits

$$z_{n,j}(b \mid \bar{x}_n, c), \qquad j \in \{1, \dots, J\}, \;\; b \in \{1, \dots, N_{\text{bin}}\}.$$

The unit-temperature base chunk pmf is therefore

$$q_\theta(A_n = a \mid \bar{x}_n, c) = \prod_{j=1}^{J} \frac{\exp z_{n,j}(a_j \mid \bar{x}_n, c)}{\sum_{b=1}^{N_{\text{bin}}} \exp z_{n,j}(b \mid \bar{x}_n, c)},$$

and the tempered proposal family used for MH is

$$q_{\theta,\beta}(A_n = a \mid \bar{x}_n, c) = \prod_{j=1}^{J} \operatorname{softmax}\left(\frac{z_{n,j}(\cdot \mid \bar{x}_n, c)}{\beta}\right)(a_j).$$

If top-k or top-p truncation is enabled, each factor is replaced by the corresponding truncated and renormalized categorical pmf. The implementation stores both base-policy and proposal-policy log probabilities for each chunk in the sampled suffix, and if a proposed suffix terminates earlier than the current one it aligns suffix lengths before applying the chunkwise acceptance-ratio sum.

A.4. Locally Variance-Minimizing Chunk Proposal

Fix a replanning boundary $(\bar{s}_m, \bar{x}_m)$ and let $r_m(A \mid \bar{x}_m, c)$ be any proposal distribution satisfying

$$r_m(A \mid \bar{x}_m, c) > 0 \qquad \text{whenever} \qquad q_\theta(A \mid \bar{x}_m, c) > 0.$$

Under the policy-only target, the incremental importance weight is

$$\omega_m(A) = \frac{q_\theta(A \mid \bar{x}_m, c)^\alpha}{r_m(A \mid \bar{x}_m, c)}.$$

Therefore the conditional second moment is

$$\mathbb{E}_{A \sim r_m}\left[\omega_m(A)^2 \mid \bar{x}_m, c\right] = \int \frac{q_\theta(A \mid \bar{x}_m, c)^{2\alpha}}{r_m(A \mid \bar{x}_m, c)}\, dA.$$

Exactly as in the token-level theorem of Azizi et al. (2026), the unique minimizer is

$$r_m^\star(A \mid \bar{x}_m, c) = \frac{q_\theta(A \mid \bar{x}_m, c)^\alpha}{Z_m(\alpha \mid \bar{x}_m, c)}, \qquad Z_m(\alpha \mid \bar{x}_m, c) := \int q_\theta(U \mid \bar{x}_m, c)^\alpha \, dU.$$

Under $r_m^\star$, the incremental weight is deterministic:

$$\omega_m(A) \equiv Z_m(\alpha \mid \bar{x}_m, c).$$

Thus, within the class of prefix-only chunk proposals, the power-tempered chunk density is the locally variance-minimizing choice.

A.5. Temperature Form for the Current Binned Chunk Head

For the current OpenVLA-OFT action head, and before any top-k or top-p truncation, the chunk pmf factorizes across action-token slots:

$$q_\theta(A_m = a \mid \bar{x}_m, c) = \prod_{j=1}^{J} \frac{\exp z_{m,j}(a_j \mid \bar{x}_m, c)}{\sum_{b=1}^{N_{\text{bin}}} \exp z_{m,j}(b \mid \bar{x}_m, c)}.$$

Raising this pmf to the power α gives

$$q_\theta(A_m = a \mid \bar{x}_m, c)^\alpha \propto \prod_{j=1}^{J} \mathrm{softmax}\big(\alpha z_{m,j}(\cdot \mid \bar{x}_m, c)\big)(a_j).$$

Hence the locally optimal proposal retains the same slotwise factorization and is exactly a temperature-scaled categorical proposal with

$$\tau^\star = \frac{1}{\alpha}.$$

Equivalently,

$$r_m^\star(A_m = a \mid \bar{x}_m, c) = \prod_{j=1}^{J} \mathrm{softmax}\left(\frac{z_{m,j}(\cdot \mid \bar{x}_m, c)}{\tau^\star}\right)(a_j).$$

This is the direct chunk-level analogue of Corollary 1 in Azizi et al. (2026). The statement is exact for the untruncated finite-support chunk pmf implemented by the current quantized head. If top-k or top-p truncation is enabled, $\tau = 1/\alpha$ remains the untruncated optimum, but the actual proposal becomes the corresponding truncated and renormalized categorical pmf.

A.6. Exact Exponent-Bridging (Alpha-Ramping)

Power-SMC also introduces an exact exponent-bridging schedule (Azizi et al., 2026). In the chunked VLA setting, let

$$1 = \alpha^{(0)} < \alpha^{(1)} < \cdots < \alpha^{(L)} = \alpha$$

and define the stage-ℓ unnormalized prefix target by

$$\widetilde{\gamma}_m^{(\ell)}(\Omega_{0:m-1} \mid c) := \prod_{j=0}^{m-1} q_\theta(A_j \mid \bar{x}_j, c)^{\alpha^{(\ell)}} T_j(\xi_j \mid \bar{s}_j, A_j, c).$$

Within stage ℓ, the locally optimal chunk proposal is

$$r_{m,\ell}^\star(A \mid \bar{x}_m, c) \propto q_\theta(A \mid \bar{x}_m, c)^{\alpha^{(\ell)}}.$$

If the exponent changes from $\alpha^{(\ell-1)}$ to $\alpha^{(\ell)}$ after prefix length m, the exact reweighting update is

$$\log \widetilde{W} \leftarrow \log \widetilde{W} + \big(\alpha^{(\ell)} - \alpha^{(\ell-1)}\big) \sum_{j=0}^{m-1} \log q_\theta(A_j \mid \bar{x}_j, c).$$

This update is exact because the policy-only target changes only the policy exponents; the rollout factors T_j are unchanged across stages. Consequently, the cumulative reweighting is identical to directly targeting the final exponent α, just as in the token-level alpha-ramping construction of Azizi et al. (2026). This exact chunk-level extension also applies to the alpha-ramped Power-SMC experiments considered in our setting.

A.7. Residual Path-Wise Weight Dispersion

Following the Renyi-entropy interpretation in Azizi et al. (2026), define

$$Z_m(\alpha \mid \bar{x}_m, c) := \int q_\theta(A \mid \bar{x}_m, c)^\alpha \, dA.$$

Under the locally optimal proposal $r_m^\star$, the accumulated pre-resampling log weight of particle i over a P-chunk plan is

$$\log \widetilde{W}_P^{(i)} = \sum_{m=0}^{P-1} \log Z_m(\alpha \mid \bar{x}_m^{(i)}, c).$$

Define the Renyi entropy of order α for a discrete chunk distribution q by

$$H_\alpha(q) := \frac{1}{1-\alpha} \log \sum_a q(a)^\alpha.$$

Then

$$\log Z_m(\alpha \mid \bar{x}_m, c) = (1-\alpha) H_\alpha\big(q_\theta(\cdot \mid \bar{x}_m, c)\big).$$

Hence particles that reach replanning boundaries where the chunk policy is more diffuse accumulate lower weights, while particles on more confident boundary paths accumulate higher weights. The important point is that, after the locally optimal within-boundary proposal removes chunk-choice variance, the remaining degeneracy is entirely path-wise: it comes from the sequence of boundary observations encountered by each particle.

B. Proposal Family and MH Ratio

Fix a current planning window of length L chunks starting at boundary $(\bar{s}_n, \bar{x}_n)$, and write the current imagined plan as

$$\Omega := (\bar{s}_n, \bar{x}_n, A_n, \xi_n, \ldots, A_{n+L-1}, \xi_{n+L-1}).$$

Choose an edit chunk r uniformly from $\{n, \ldots, n+L-1\}$, keep the prefix before r fixed, and regenerate the suffix from chunk r onward. The resulting suffix-regeneration proposal is

$$Q(\Omega' \mid \Omega) = \frac{1}{L} \prod_{j=r}^{n+L-1} q_{\theta,\beta}(A'_j \mid \bar{x}'_j, c)\, T_j(\xi'_j \mid \bar{s}'_j, A'_j, c).$$

Because the reverse move from Ω' back to Ω redraws the same suffix under the same edit index,

$$Q(\Omega \mid \Omega') = \frac{1}{L} \prod_{j=r}^{n+L-1} q_{\theta,\beta}(A_j \mid \bar{x}_j, c)\, T_j(\xi_j \mid \bar{s}_j, A_j, c),$$

and therefore

$$\begin{aligned}\log \frac{Q(\Omega \mid \Omega')}{Q(\Omega' \mid \Omega)} = &\sum_{j=r}^{n+L-1} \Big[\log q_{\theta,\beta}(A_j \mid \bar{x}_j, c) - \log q_{\theta,\beta}(A'_j \mid \bar{x}'_j, c)\Big] \\ &+ \sum_{j=r}^{n+L-1} \Big[\log T_j(\xi_j \mid \bar{s}_j, A_j, c) - \log T_j(\xi'_j \mid \bar{s}'_j, A'_j, c)\Big].\end{aligned}$$

Full trajectory-level power target. If the imagined plan is scored by the exact full power target, the corresponding unnormalized L-chunk target is

$$\gamma_L^{\text{full}}(\Omega \mid \bar{s}_n, \bar{x}_n, c) := \prod_{j=n}^{n+L-1} q_\theta(A_j \mid \bar{x}_j, c)^\alpha T_j(\xi_j \mid \bar{s}_j, A_j, c)^\alpha.$$

Since Ω and Ω' agree on chunks $n, \ldots, r-1$, their target ratio only involves the regenerated suffix:

$$\begin{aligned}\log \frac{\gamma_L^{\text{full}}(\Omega' \mid \bar{s}_n, \bar{x}_n, c)}{\gamma_L^{\text{full}}(\Omega \mid \bar{s}_n, \bar{x}_n, c)} = \alpha &\sum_{j=r}^{n+L-1} \Big[\log q_\theta(A'_j \mid \bar{x}'_j, c) - \log q_\theta(A_j \mid \bar{x}_j, c)\Big] \\ &+ \alpha \sum_{j=r}^{n+L-1} \Big[\log T_j(\xi'_j \mid \bar{s}'_j, A'_j, c) - \log T_j(\xi_j \mid \bar{s}_j, A_j, c)\Big].\end{aligned}$$

The resulting log Metropolis-Hastings ratio is therefore

$$\begin{aligned}\Delta_{\text{full}}(\Omega, \Omega') := \log \frac{\gamma_L^{\text{full}}(\Omega' \mid \bar{s}_n, \bar{x}_n, c)\, Q(\Omega \mid \Omega')}{\gamma_L^{\text{full}}(\Omega \mid \bar{s}_n, \bar{x}_n, c)\, Q(\Omega' \mid \Omega)} \\ = \alpha \sum_{j=r}^{n+L-1} \Big[\log q_\theta(A'_j \mid \bar{x}'_j, c) - \log q_\theta(A_j \mid \bar{x}_j, c)\Big] \\ + \sum_{j=r}^{n+L-1} \Big[\log q_{\theta,\beta}(A_j \mid \bar{x}_j, c) - \log q_{\theta,\beta}(A'_j \mid \bar{x}'_j, c)\Big] \\ + (\alpha - 1) \sum_{j=r}^{n+L-1} \Big[\log T_j(\xi'_j \mid \bar{s}'_j, A'_j, c) - \log T_j(\xi_j \mid \bar{s}_j, A_j, c)\Big].\end{aligned}$$

Expanding one chunk density shows the remaining computational burden explicitly:

$$\log T_j(\xi_j \mid \bar{s}_j, A_j, c) = \sum_{k=0}^{K-1} \Big[\log P_1(s_{jK+k+1} \mid s_{jK+k}, a_{jK+k}) + \log O(x_{jK+k+1} \mid s_{jK+k+1}, c)\Big].$$

Thus the full target requires transition and observation log densities for every regenerated low-level step.

Policy-only power target. In the main text we instead use the policy-only target

$$\widetilde{\gamma}_L(\Omega \mid \bar{s}_n, \bar{x}_n, c) := \prod_{j=n}^{n+L-1} q_\theta(A_j \mid \bar{x}_j, c)^\alpha T_j(\xi_j \mid \bar{s}_j, A_j, c).$$

Its target ratio is

$$\begin{aligned}\log \frac{\widetilde{\gamma}_L(\Omega' \mid \bar{s}_n, \bar{x}_n, c)}{\widetilde{\gamma}_L(\Omega \mid \bar{s}_n, \bar{x}_n, c)} = \alpha \sum_{j=r}^{n+L-1} \Big[\log q_\theta(A'_j \mid \bar{x}'_j, c) - \log q_\theta(A_j \mid \bar{x}_j, c)\Big] \\ + \sum_{j=r}^{n+L-1} \Big[\log T_j(\xi'_j \mid \bar{s}'_j, A'_j, c) - \log T_j(\xi_j \mid \bar{s}_j, A_j, c)\Big].\end{aligned}$$

Combining this with the proposal ratio gives the implemented log Metropolis-Hastings ratio

$$\begin{aligned}\Delta(\Omega, \Omega') := \log \frac{\widetilde{\gamma}_L(\Omega' \mid \bar{s}_n, \bar{x}_n, c)\, Q(\Omega \mid \Omega')}{\widetilde{\gamma}_L(\Omega \mid \bar{s}_n, \bar{x}_n, c)\, Q(\Omega' \mid \Omega)} \\ = \alpha \sum_{j=r}^{n+L-1} \Big[\log q_\theta(A'_j \mid \bar{x}'_j, c) - \log q_\theta(A_j \mid \bar{x}_j, c)\Big] \\ + \sum_{j=r}^{n+L-1} \Big[\log q_{\theta,\beta}(A_j \mid \bar{x}_j, c) - \log q_{\theta,\beta}(A'_j \mid \bar{x}'_j, c)\Big].\end{aligned}$$

Here the chunk-rollout terms cancel exactly because the policy-only target contributes one factor of T_j while the proposal ratio contributes its reverse. This is the MH analogue of Appendix A.2: once rollout stochasticity is left unpowered, correction factors depend only on policy probabilities. When $\beta = 1$ and no top-k or top-p truncation is used, $q_{\theta,\beta} = q_\theta$ and the implemented ratio simplifies to

$$\Delta(\Omega, \Omega') = (\alpha - 1) \sum_{j=r}^{n+L-1} \Big[\log q_\theta(A'_j \mid \bar{x}'_j, c) - \log q_\theta(A_j \mid \bar{x}_j, c)\Big].$$

C. Chunk-Level Power-SMC Procedure

Algorithm A1 Chunk-Level Power-SMC for Chunked VLA Planning

Input: boundary $(\bar{s}_0, \bar{x}_0)$, instruction c, horizon P, particles N, ESS threshold κ, resampling-check interval Δ, and ramping schedule $\{(\alpha^{(\ell)}, \beta^{(\ell)})\}_{\ell=0}^{L}$
Initialize N particles at $(\bar{s}_0, \bar{x}_0)$ with empty prefixes and equal weights
for $m = 0$ to $P - 1$ **do**
  Let $\ell \leftarrow \ell(m)$ denote the current ramping stage
  for $i = 1$ to N in parallel **do**
    Sample $A_m^{(i)} \sim q_{\theta,\beta^{(\ell)}}(\cdot \mid \bar{x}_m^{(i)}, c)$
    Roll one chunk to obtain $\xi_m^{(i)}$, $\bar{s}_{m+1}^{(i)}$, and $\bar{x}_{m+1}^{(i)}$
    Append $(A_m^{(i)}, \xi_m^{(i)})$ to particle i
    Update $\widetilde{W}_{m+1}^{(i)} \leftarrow \widetilde{W}_m^{(i)} \cdot \dfrac{q_\theta(A_m^{(i)} \mid \bar{x}_m^{(i)}, c)^{\alpha^{(\ell)}}}{q_{\theta,\beta^{(\ell)}}(A_m^{(i)} \mid \bar{x}_m^{(i)}, c)}$
  end for
  if the stage changes after prefix length $m + 1$ **then**
    Apply the exact alpha-ramping reweighting from Appendix A.6
  end if
  if $(m + 1) \bmod \Delta = 0$ **then**
    Normalize weights, compute ESS, and systematically resample if $\mathrm{ESS}_{m+1} < \kappa N$
  end if
end for
Sample and output one full P-chunk plan from the terminal weighted particles

D. MH Planning Procedure

Algorithm A2 MH Planning Procedure

Input: decision index n, boundary $(\bar{s}_n, \bar{x}_n)$, instruction c, power exponent α, proposal temperature β, planning size P, block size B, MH steps M
Initialize $\Omega \leftarrow (\bar{s}_n, \bar{x}_n)$
for $m = 0$ to $\lceil P/B \rceil - 1$ **do**
  Set $L \leftarrow \min((m + 1)B, P)$
  for $j = n + mB$ to $n + L - 1$ **do**
    Sample $A_j \sim q_{\theta,\beta}(\cdot \mid \bar{x}_j, c)$
    Roll the environment forward under P_1 and O to obtain ξ_j, $\bar{s}_{j+1}$, and $\bar{x}_{j+1}$
  end for
  Write the resulting plan as $\Omega = (\bar{s}_n, \bar{x}_n, A_n, \xi_n, \ldots, A_{n+L-1}, \xi_{n+L-1})$
  for $i = 1$ to M **do**
    Sample edit chunk $r \sim \mathrm{Unif}\{n, \ldots, n + L - 1\}$
    Set $\bar{s}'_r \leftarrow \bar{s}_r$ and $\bar{x}'_r \leftarrow \bar{x}_r$
    for $j = r$ to $n + L - 1$ **do**
      Sample $A'_j \sim q_{\theta,\beta}(\cdot \mid \bar{x}'_j, c)$
      Roll forward under P_1 and O to obtain ξ'_j, $\bar{s}'_{j+1}$, and $\bar{x}'_{j+1}$
    end for
    Construct Ω' by replacing the suffix of Ω from chunk r onward
    Compute $\Delta(\Omega, \Omega')$ as in Appendix B
    With probability $\min(1, \exp(\Delta))$, set $\Omega \leftarrow \Omega'$
  end for
end for
Output $(A_n, \ldots, A_{n+P-1})$

E. Detailed Benchmark Variants

We inherit the benchmark construction from Liu et al. (2026) and summarize only the details needed to interpret our results. We intentionally avoid reproducing the full prompt inventory or long near-verbatim task descriptions from the source paper.

E.1. Benchmark Summary

Training. The training distribution uses 16 objects and 16 table appearances. Each episode contains one target object, the default yellow-plate receptacle, and a short imperative instruction of the form "put O on R," with randomized placement inside the benchmark's standard training region.

Base checkpoint. We evaluate the public PPO-trained OpenVLA-OFT checkpoint released by RLinf-VLA and trained on the benchmark's training environment:

```
RLinf/RLinf-OpenVLAOFT-PPO-ManiSkill3-25ood
```

Vision variants. *Unseen Table* evaluates 5 held-out table appearances. *Dynamic Texture* and *Dynamic Noise* each sample from 16 textures; the weak and strong versions use mixing coefficients 0.3 and 0.5, respectively. Texture affects the manipulated entities for *Dynamic Texture* and the full image for *Dynamic Noise*.

Semantic variants. *Unseen Objects* draws from 9 held-out objects. *Unseen Receptacles*, *Distractive Receptacle*, and *Multi-Receptacle* draw from 16 held-out receptacles. *Unseen Instruction Phrasing* uses 16 held-out prompt variants with different wording, punctuation, and capitalization. *Multi-Object* adds a second object and is evaluated with either seen or unseen object pairs; *Multi-Receptacle* adds a second unseen receptacle while the instruction still names only the target pair.

Execution variants. *Unseen Position* expands the placement region beyond the training rectangle. *Unseen Robot Init Pose* randomizes articulation initialization. *Mid-Episode Object Reposition* moves the target object to a new table position at step 5.

E.2. Hardware Setup

Two evaluation regimes are relevant for interpreting the appendix. The main Power-SMC study, including the matched execution-axis sampler comparison in Appendix H, was run on two NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPUs, each with 96 GB of VRAM. By contrast, the supplementary MH proof-of-viability sweep in Appendix J was run on eight GeForce RTX 3090 GPUs, each with 24 GB of VRAM.

This split reflects available compute constraints during the project: to keep experimental turnaround fast, we ran different phases on the hardware that was available at the time rather than waiting to rerun every earlier exploratory sweep on the final workstation. We therefore separate these regimes because the matched execution comparison is intended to support the eventual accuracy-latency choice among baseline decoding, serial MH, and parallel Power-SMC, whereas the broader MH sweep is retained as supplementary proof-of-viability evidence rather than as a hardware-matched runtime comparison.

E.3. Environment and Inference Hyperparameters

Table A1. Environment and inference hyperparameters by appendix study.

Scope	Setting	Value
Shared evaluation settings		
Environment	Max episode steps	80
All samplers	Chunk length K (`num_action_chunks`)	8
All baseline comparisons	Baseline evaluation temperature	0.6
Main Power-SMC study		
Power-SMC	Planning size P	10
Power-SMC	Final power exponent α	2.5
Power-SMC	Particle count N	64
Power-SMC	ESS threshold κ	0.5
Power-SMC	Proposal temperature β	Dynamic, $\beta^{(\ell)} = 1/\alpha^{(\ell)}$
Power-SMC	Alpha-ramping blocks	7
Power-SMC	Resampling-check interval	Every 2 blocks
Matched execution-axis MH comparison		
MH-sampling	Planning size P	10
MH-sampling	Block size B	10
MH-sampling	MH steps M	64
MH-sampling	Power exponent α	2.5
MH-sampling	Proposal temperature β	0.6
Supplementary MH proof-of-viability sweep		
MH-sampling	Planning size P	10
MH-sampling	Block size B	10
MH-sampling	MH steps M	64
MH-sampling	Power exponent α	4.0
MH-sampling	Proposal temperature β	0.6

All appendix studies share the same episode horizon, chunk length, and baseline temperature so that success comparisons are not confounded by control discretization. Following the RLinf-VLA evaluation protocol of Liu et al. (2026), all baseline decoding results in this paper use evaluation temperature 0.6; the lower-temperature baselines in Appendix J are explicit deviations introduced only for the supplementary comparison against serial MH. For Power-SMC, we fix one configuration and reuse it across all 16 settings, with no per-shift retuning: a stagewise temperature schedule implied by the locally optimal proposal, final $\alpha = 2.5$, $N = 64$, $\kappa = 0.5$, 7 alpha-ramping blocks, and ESS monitoring every 2 blocks. The matched execution-axis MH comparison reuses the serial MH sampler family but sets $\alpha = 2.5$ to mirror Power-SMC, while the supplementary proof-of-viability MH sweep keeps the earlier $\alpha = 4.0$ configuration.

F. Detailed Experimental Results

This section reports the detailed results underlying the main paper. Success metrics are reported as rates in $[0, 1]$, Actions denotes the average number of executed actions until the first success event, and Latency denotes average per-episode wall-clock time normalized so that baseline decoding equals 1.0. We omit separate `wall_sec` totals. The available timing rows place Power-SMC at roughly $3\times$ baseline latency in the current implementation because each replanning step forwards a batch of particles instead of a single rollout. Appendix I explains why this still leaves Power-SMC in a much more favorable asymptotic latency regime than serial MH. For the execution axis, the next section additionally aligns baseline decoding, MH-sampling, and Power-SMC under common hardware and matched MH power exponent so that the eventual sampler choice can be read within the same grouped setting blocks.

Axis-level summary. Table A2 collects the training setting together with the three OOD axes.

Table A2. Power-SMC summary over the training setting and the three OOD axes.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	Power-SMC
Training (IND)	Success@once ↑	0.9844	**0.9922**	0.9688	0.9844
	Success@end ↑	0.8438	**0.8594**	0.7969	0.8281
	Actions ↓	23.27	23.79	23.35	**22.16**
	Latency ↓	1.000	**0.937**	0.995	2.975
Vision (OOD)	Success@once ↑	0.9359	0.9266	0.9313	**0.9422**
	Success@end ↑	0.7250	0.7344	0.7438	**0.7656**
	Actions ↓	24.78	24.34	24.67	**22.90**
	Latency ↓	**1.000**	1.001	1.005	3.073
Semantic (OOD)	Success@once ↑	0.6685	0.6574	0.6417	**0.7031**
	Success@end ↑	0.4777	0.4732	0.4554	**0.5234**
	Actions ↓	37.75	37.86	37.25	**32.70**
	Latency ↓	1.000	**0.986**	0.988	3.106
Execution (OOD)	Success@once ↑	0.7422	0.7344	0.7188	**0.8073**
	Success@end ↑	0.5313	0.5339	0.5625	**0.6328**
	Actions ↓	38.71	39.16	38.94	**31.31**
	Latency ↓	1.000	**0.999**	1.004	3.145
OOD average	Success@once ↑	0.7822	0.7728	0.7639	**0.8175**
	Success@end ↑	0.5780	0.5805	0.5872	**0.6406**
	Actions ↓	33.75	33.79	33.62	**28.97**
	Latency ↓	1.000	**0.995**	0.999	3.108

Shift-specific breakdowns. Tables A3–A5 unpack the vision, semantic, and execution axes in the same metric order.

Table A3. Detailed vision-shift results.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	Power-SMC
Vision avg.	Success@once ↑	0.9359	0.9266	0.9313	**0.9422**
	Success@end ↑	0.7250	0.7344	0.7438	**0.7656**
	Actions ↓	24.78	24.34	24.67	**22.90**
	Latency ↓	**1.000**	1.001	1.005	3.073
Unseen table	Success@once ↑	0.9688	0.9766	0.9688	**0.9844**
	Success@end ↑	0.8125	**0.8438**	0.8281	0.8125
	Actions ↓	25.39	25.28	24.25	**23.56**
	Latency ↓	1.000	**0.998**	1.006	2.993
Dynamic texture (weak)	Success@once ↑	**0.9766**	0.9531	0.9531	0.9609
	Success@end ↑	0.7578	0.7734	**0.7891**	**0.7891**
	Actions ↓	24.82	24.43	24.74	**22.40**
	Latency ↓	1.000	**0.995**	1.001	3.053
Dynamic texture (strong)	Success@once ↑	0.8906	0.8828	0.9063	**0.9297**
	Success@end ↑	0.6406	0.6953	0.6875	**0.7812**
	Actions ↓	24.15	25.36	25.63	**23.61**
	Latency ↓	**1.000**	1.004	1.013	3.125
Dynamic noise (weak)	Success@once ↑	**0.9766**	0.9453	0.9531	0.9609
	Success@end ↑	0.7578	0.7422	0.7266	**0.7891**
	Actions ↓	24.82	22.64	24.18	**22.40**
	Latency ↓	**1.000**	1.007	1.001	3.079
Dynamic noise (strong)	Success@once ↑	0.8672	**0.8750**	**0.8750**	**0.8750**
	Success@end ↑	0.6562	0.6172	**0.6875**	0.6562
	Actions ↓	24.73	24.00	24.57	**22.52**
	Latency ↓	1.000	**0.999**	1.006	3.113

Table A4. Detailed semantic-shift results.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	Power-SMC
Semantic avg.	Success@once ↑	0.6685	0.6574	0.6417	**0.7031**
	Success@end ↑	0.4777	0.4732	0.4554	**0.5234**
	Actions ↓	37.75	37.86	37.25	**32.70**
	Latency ↓	1.000	**0.986**	0.988	3.106
Unseen objects	Success@once ↑	0.8672	0.9141	**0.9453**	**0.9453**
	Success@end ↑	0.7344	0.7344	0.7422	**0.8281**
	Actions ↓	25.51	27.63	28.06	**24.17**
	Latency ↓	**1.000**	1.002	1.012	3.045
Unseen receptacles	Success@once ↑	0.9062	0.8828	0.8672	**0.9141**
	Success@end ↑	**0.5391**	0.4609	0.4453	0.5078
	Actions ↓	30.06	27.81	27.90	**25.50**
	Latency ↓	1.000	**0.999**	1.000	3.074
Unseen instructions	Success@once ↑	0.6797	0.6875	**0.6953**	0.6719
	Success@end ↑	0.5703	**0.6016**	0.5313	0.5547
	Actions ↓	23.59	23.44	23.91	**22.37**
	Latency ↓	1.000	0.998	**0.996**	3.109
Multi-object (both seen)	Success@once ↑	0.6094	0.6406	0.5781	**0.6797**
	Success@end ↑	0.4453	0.5078	0.4688	**0.5625**
	Actions ↓	41.46	40.06	43.77	**35.85**
	Latency ↓	1.000	1.004	**0.995**	3.149
Multi-object (both unseen)	Success@once ↑	0.5938	0.5938	0.5625	**0.6562**
	Success@end ↑	0.4688	0.4453	0.4766	**0.5234**
	Actions ↓	41.32	42.80	39.88	**34.69**
	Latency ↓	1.000	**0.996**	0.998	3.136
Distractive receptacle	Success@once ↑	0.6406	0.5703	0.5547	**0.6562**
	Success@end ↑	0.4453	0.4609	0.3906	**0.4844**
	Actions ↓	50.85	52.77	50.23	**41.29**
	Latency ↓	1.000	**0.987**	0.989	3.157
Multi-receptacle (both unseen)	Success@once ↑	0.3828	0.3125	0.2891	**0.3984**
	Success@end ↑	0.1406	0.1016	0.1328	**0.2031**
	Actions ↓	51.45	50.50	47.03	**45.00**
	Latency ↓	1.000	**0.915**	0.927	3.074

Table A5. Detailed execution-shift results.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	Power-SMC
Execution avg.	Success@once ↑	0.7422	0.7344	0.7188	**0.8073**
	Success@end ↑	0.5313	0.5339	0.5625	**0.6328**
	Actions ↓	38.71	39.16	38.94	**31.31**
	Latency ↓	1.000	**0.999**	1.004	3.145
Unseen position (object & receptacle)	Success@once ↑	0.6094	0.5781	0.6094	**0.7500**
	Success@end ↑	0.4609	0.3594	0.4688	**0.5703**
	Actions ↓	40.77	39.93	42.19	**32.75**
	Latency ↓	1.000	**0.992**	1.005	3.206
Unseen robot init pose	Success@once ↑	0.7422	0.7266	0.7266	**0.8359**
	Success@end ↑	0.4922	0.5234	0.5781	**0.6328**
	Actions ↓	34.37	35.12	33.34	**27.48**
	Latency ↓	**1.000**	1.001	1.007	3.148
Mid-episode object reposition	Success@once ↑	0.8750	**0.8984**	0.8203	0.8359
	Success@end ↑	0.6406	**0.7188**	0.6406	0.6953
	Actions ↓	40.99	42.43	41.28	**33.70**
	Latency ↓	**1.000**	1.003	1.001	3.079

Same-setting different-seed replicate. Table A6 reports an additional axis-level replicate run with the same evaluation

setting and Power-SMC hyperparameters as the main study but a different random seed. Because this earlier run did not record standardized normalized latency, we report only `success_once`, `success_at_end`, and Actions. The aggregate OOD pattern matches the main study: Power-SMC improves both success metrics and lowers Actions on every OOD axis and on the axis-balanced OOD average.

Table A6. Axis-level same-setting replicate with a different random seed. Latency is omitted because this earlier run did not log the standardized normalized timing metric.

Split	Metric	Baseline (0.6)	Power-SMC
Training (IND)	Success@once ↑	**1.0000**	0.9875
	Success@end ↑	0.7750	**0.8375**
	Actions ↓	23.05	**22.19**
Vision (OOD)	Success@once ↑	0.9175	**0.9575**
	Success@end ↑	0.7450	**0.7700**
	Actions ↓	24.54	**23.07**
Semantic (OOD)	Success@once ↑	0.6464	**0.6964**
	Success@end ↑	0.4607	**0.5125**
	Actions ↓	37.98	**33.25**
Execution (OOD)	Success@once ↑	0.7167	**0.7958**
	Success@end ↑	0.5250	**0.6625**
	Actions ↓	42.15	**31.18**
OOD average	Success@once ↑	0.7602	**0.8166**
	Success@end ↑	0.5769	**0.6483**
	Actions ↓	34.89	**29.17**

G. Qualitative Analysis of Gain Patterns

Trajectory inspection suggests three consistent patterns behind Tables A3–A5. First, when the base policy is already strong on a setting, binary success metrics leave little headroom, so gains in `success_once` or `success_at_end` are naturally modest. This is visible in the training setting and in several easier vision shifts. Even there, however, Power-SMC still improves efficiency: across every setting in Tables A3–A5, it lowers Actions relative to baseline. The most consistent benefit is therefore not always higher success rates, but reaching the same successful state with fewer corrective motions and fewer repeated attempts.

Second, Power-SMC helps less when the dominant failure lies outside the base policy's effective behavioral support. On *Unseen instructions*, a frequent failure mode is that the robot never seriously initiates the grasp: it rises away from the object or wanders without committing to pickup. On *Multi-receptacle (both unseen)*, another common failure is that the policy does not identify which receptacle the object should target. In these cases, future-aware reweighting has little useful material to amplify, because the sampler can only redistribute probability mass among continuations that the base policy already proposes with non-negligible probability. This helps explain why the gains on these hardest semantic failures are small or inconsistent.

Third, when Power-SMC does help, the improvement usually comes from better handling of fragile intermediate states rather than from discovering qualitatively new long-horizon strategies. The clearest pattern is grasp acquisition: trajectories that previously required three or four pickup attempts are often reduced to one or two attempts. We also observe better retention after the first correct placement event. Once the object reaches the target receptacle, Power-SMC more often keeps the episode in a successful state instead of immediately losing the placement. Together, these effects explain why Actions improves most consistently and why gains in `success_at_end` are often larger than gains in `success_once`: the sampler is especially helpful at turning near-miss rollouts into earlier grasps and more stable completions.

H. Matched Execution-Axis Sampler Comparison

This section directly aligns baseline decoding, MH-sampling, and Power-SMC on the execution axis under matched experimental conditions. All three samplers were run on the same two-GPU Blackwell workstation used for the main Power-SMC study, and the MH sampler uses power exponent $\alpha = 2.5$ to match Power-SMC rather than the $\alpha = 4.0$ setting used in the supplementary MH proof-of-viability sweep. Together with the engine-independent analysis in Appendix I, this

is the appropriate appendix block for the eventual accuracy-latency tradeoff discussion.

Table A7. Matched execution-shift comparison across baseline decoding, MH-sampling, and Power-SMC. Latency is average per-episode wall-clock time normalized so that baseline decoding equals 1.0; we omit separate `wall_sec` totals.

Setting	Metric	Baseline	MH-sampling	Power-SMC
Execution avg.	Success@once ↑	0.7422	0.7891	**0.8073**
	Success@end ↑	0.5313	**0.6615**	0.6328
	Actions ↓	38.71	35.03	**31.31**
	Latency ↓	**1.000**	39.316	3.043
Unseen position (object & receptacle)	Success@once ↑	0.6094	0.7109	**0.7500**
	Success@end ↑	0.4609	**0.5781**	0.5703
	Actions ↓	40.77	37.12	**32.75**
	Latency ↓	**1.000**	39.669	3.122
Unseen robot init pose	Success@once ↑	0.7422	0.7969	**0.8359**
	Success@end ↑	0.4922	**0.6719**	0.6328
	Actions ↓	34.37	31.65	**27.48**
	Latency ↓	**1.000**	38.960	3.023
Mid-episode object reposition	Success@once ↑	**0.8750**	0.8594	0.8359
	Success@end ↑	0.6406	**0.7344**	0.6953
	Actions ↓	40.99	36.34	**33.70**
	Latency ↓	**1.000**	39.318	2.985

I. Theoretical Planning-Overhead Analysis

We adapt the engine-independent compute model of Azizi et al. (2026) to chunked VLA replanning. This section isolates *planner-side* overhead for one local replanning problem rather than full closed-loop episode time: after planning, the selected chunk still has to be executed in the actual environment, and episode-level wall-clock also depends on how many chunks each method ultimately executes before success or timeout. Those execution-time effects are outcome-dependent and shared only imperfectly across samplers, so we omit them here to keep the theory focused on the planning mechanisms themselves. Accordingly, every Time quantity below should be read as planner-side time for one local P-chunk planning window.

Because one OpenVLA-OFT forward pass emits an entire action chunk, the natural cost unit is one *chunk propagation*: sample and score one chunk from one replanning-boundary observation, then roll the simulator for K low-level steps to the next boundary. Let c_{f} be the single-particle policy-forward cost and c_{r} be the single-particle chunk-rollout cost. For batch size b, let $s_{\mathrm{f}}(b) \geq 1$ and $s_{\mathrm{r}}(b) \geq 1$ denote the corresponding throughput multipliers relative to batch 1. Then one batch-b chunk propagation takes time proportional to

$$\tau(b) := c_{\mathrm{f}} \frac{b}{s_{\mathrm{f}}(b)} + c_{\mathrm{r}} \frac{b}{s_{\mathrm{r}}(b)}.$$

Throughout this section, P is the local planning horizon in chunks, B is the MH block size in chunks, $G := P/B$ is the number of MH planning blocks when $B \mid P$, M is the number of MH refinement moves per block, and N is the number of Power-SMC particles.

Power-SMC cost. Chunk-level Power-SMC advances all N particles through all P imagined chunks, so the total number of particle-chunk propagations is

$$C_{\mathrm{SMC}} = NP.$$

Because those N propagations at each chunk index are executed in one batch, the wall-clock time satisfies

$$\mathrm{Time}_{\mathrm{SMC}} \propto P\,\tau(N) = P\left(c_{\mathrm{f}} \frac{N}{s_{\mathrm{f}}(N)} + c_{\mathrm{r}} \frac{N}{s_{\mathrm{r}}(N)}\right).$$

Weight updates and resampling are only $O(N)$ per chunk and do not change the leading term.

Baseline planning cost. A single-trajectory decoder that rolls out one P-chunk plan under the same local planning window uses

$$C_{\text{base}} = P$$

chunk propagations, so

$$\text{Time}_{\text{base}} \propto P(c_{\text{f}} + c_{\text{r}}).$$

If one additionally wants a fixed-boundary end-to-end control-cycle model, one may add a common post-planning execution cost c_{exec} for applying the selected chunk in the real environment, i.e.,

$$\text{Time}^{\text{cycle}}_{\text{method}} = \text{Time}_{\text{method}} + c_{\text{exec}}.$$

This additive term shrinks planner-side ratios toward 1 but does not change their ordering.

MH-sampling cost. For the blockwise suffix-regeneration MH procedure in Appendix D, extending the current imagined plan by one new block costs B chunk propagations. If $L_{g,m}$ denotes the regenerated suffix length of MH move $m \in \{1, \ldots, M\}$ in planning block $g \in \{1, \ldots, G\}$, then

$$C_{\text{MH}} = P + \sum_{g=1}^{G} \sum_{m=1}^{M} L_{g,m}.$$

Our current implementation samples the edit chunk uniformly from the entire currently planned prefix. At planning block g, that prefix has length $t_g = gB$, so

$$\mathbb{E}[L_g] \approx \frac{t_g}{2} = \frac{gB}{2}.$$

Therefore

$$\mathbb{E}[C_{\text{MH}}] \approx P\left(1 + \frac{M(G+1)}{4}\right).$$

Every suffix regeneration is serial, so the corresponding wall-clock time is

$$\text{Time}_{\text{MH}} \propto (c_{\text{f}} + c_{\text{r}})\, \mathbb{E}[C_{\text{MH}}].$$

If edits were restricted to the newest block only, the more optimistic bound $\mathbb{E}[C_{\text{MH}}] \approx P(1 + M/2)$ would be recovered; our current sampler is less favorable because its edit location ranges over the full planned prefix.

Planner-side ratios. Define the effective throughput multiplier of batch-N chunk propagation by

$$s_{\text{eff}}(N) := \frac{c_{\text{f}} + c_{\text{r}}}{c_{\text{f}}/s_{\text{f}}(N) + c_{\text{r}}/s_{\text{r}}(N)}.$$

Then the baseline-normalized planner-side latencies are

$$\frac{\text{Time}_{\text{MH}}}{\text{Time}_{\text{base}}} \approx 1 + \frac{M(G+1)}{4}, \qquad \frac{\text{Time}_{\text{SMC}}}{\text{Time}_{\text{base}}} \approx \frac{N}{s_{\text{eff}}(N)}.$$

Equivalently, the expected MH-to-SMC wall-clock ratio becomes

$$\frac{\text{Time}_{\text{MH}}}{\text{Time}_{\text{SMC}}} \approx \left(1 + \frac{M(G+1)}{4}\right) \frac{s_{\text{eff}}(N)}{N}.$$

This makes the structural difference explicit: MH pays an $\Omega(M)$ serial suffix-regeneration overhead, while Power-SMC pays an N-particle factor that can be amortized by batching. Raw compute can therefore favor MH even when wall-clock time favors Power-SMC.

Implication for the present OpenVLA-OFT setting. Using the matched MH settings $P = B = 10$ and $M = 64$ from Table A1, together with the main Power-SMC particle count $N = 64$, a like-for-like 10-chunk planning comparison gives

$$C_{\text{base}} = 10, \qquad \mathbb{E}[C_{\text{MH}}] \approx 10\left(1 + \frac{64}{2}\right) = 330, \qquad C_{\text{SMC}} = 64 \times 10 = 640.$$

Therefore the idealized baseline-normalized planner-side MH latency is

$$\frac{\text{Time}_{\text{MH}}}{\text{Time}_{\text{base}}} \approx \frac{330}{10} = 33,$$

which is the relevant local-window planning-overhead prediction. The matched execution-axis table reports a somewhat larger baseline-normalized MH latency of about 39, which is reasonable because the chunk-propagation model omits serial bookkeeping overheads such as repeated proposal sampling, acceptance-ratio evaluation, host-device synchronization, and imperfect hardware utilization; the empirical episode-level metric also includes actual chunk execution and outcome-dependent episode length. Likewise,

$$\frac{\text{Time}_{\text{SMC}}}{\text{Time}_{\text{base}}} \approx \frac{64}{s_{\text{eff}}(64)}.$$

Thus Power-SMC uses about $1.94\times$ more raw chunk propagations than MH over the same 10-chunk plan, but it is already wall-clock favorable once

$$s_{\text{eff}}(64) > \frac{64}{33} \approx 1.94.$$

This threshold is mild for modern batched GPU inference. For example, if Power-SMC runs at about $3\times$ baseline in the current implementation, then $s_{\text{eff}}(64) \approx 64/3 \approx 21.3$, which in turn predicts

$$s_{\text{eff}}(64) \approx 21.3 \gg 1.94,$$

so the theory predicts that Power-SMC should remain substantially faster than serial MH.

J. Supplementary MH Proof-of-Viability Results

This section retains the earlier suffix-regeneration MH results used as proof-of-viability evidence for VLA power sampling in a serial setting. To separate serial trajectory regeneration from ordinary temperature sharpening, we additionally compare the default decoding temperature 0.6 against lower-temperature decoding baselines at 0.4 and 0.25. Unlike the matched execution-axis comparison in Appendix H, these experiments were run on eight GeForce RTX 3090 GPUs and use MH power exponent $\alpha = 4.0$. They are therefore best interpreted as a supplementary sweep demonstrating that serial MH can help beyond direct low-temperature decoding, rather than as a hardware-matched runtime comparison to the main Power-SMC study.

Axis-level and shift-wise MH results. This supplementary sweep also uses 128 trajectories per environment, matching the main Power-SMC study and the matched Appendix H comparison. Across all tables, we report `success_once`, `success_at_end`, and Actions. We omit latency here because these early MH proof-of-viability runs predate the standardized normalized latency protocol used elsewhere in the paper, so their timing values are not directly comparable; the separate `wall_sec` totals are likewise omitted. Table A8 reports the aggregate axis summary, while Tables A9–A11 give the shift-wise four-way breakdowns.

Table A8. Axis-level summary for the supplementary MH proof-of-viability sweep with lower-temperature decoding baselines.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	MH-sampling
Training (IND)	Success@once ↑	0.9922	0.9922	0.9844	**1.0000**
	Success@end ↑	0.7891	0.8125	0.7734	**0.8750**
	Actions ↓	23.43	23.37	23.40	**22.81**
Vision (OOD)	Success@once ↑	0.9266	0.9359	0.9250	**0.9438**
	Success@end ↑	0.7359	0.7313	0.7406	**0.7969**
	Actions ↓	25.12	25.11	25.05	**24.01**
Semantic (OOD)	Success@once ↑	0.6518	0.6239	0.6172	**0.6853**
	Success@end ↑	0.4587	0.4531	0.4408	**0.5123**
	Actions ↓	38.31	37.68	37.81	**35.34**
Execution (OOD)	Success@once ↑	0.7474	0.7734	0.7389	**0.7943**
	Success@end ↑	0.5469	0.5677	0.5911	**0.6302**
	Actions ↓	41.05	40.62	40.05	**34.65**
OOD average	Success@once ↑	0.7752	0.7778	0.7603	**0.8078**
	Success@end ↑	0.5805	0.5840	0.5909	**0.6465**
	Actions ↓	34.83	34.47	34.30	**31.33**

Table A8 shows the same qualitative emphasis as the main Power-SMC study. On the OOD average, MH-sampling improves success_once/success_at_end from $0.775/0.581$ to $0.808/0.647$ and lowers Actions from 34.83 to 31.33. As in the main study, the strongest aggregate gains appear in success_at_end and Actions, and the largest OOD gains occur on the semantic and execution axes. Compared with direct low-temperature decoding, the shift-wise tables still show exceptions on individual settings, but the overall pattern suggests that serial trajectory regeneration improves stability and efficiency beyond simple temperature sharpening.

Table A9. Detailed vision-shift results for the supplementary MH proof-of-viability sweep with lower-temperature decoding baselines.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	MH-sampling
Vision avg.	Success@once ↑	0.9266	0.9359	0.9250	**0.9438**
	Success@end ↑	0.7359	0.7313	0.7406	**0.7969**
	Actions ↓	25.12	25.11	25.05	**24.01**
Unseen table	Success@once ↑	**0.9844**	0.9766	0.9766	0.9609
	Success@end ↑	0.7813	0.7344	**0.8438**	0.8281
	Actions ↓	24.68	25.78	25.19	**23.63**
Dynamic texture (weak)	Success@once ↑	**0.9688**	**0.9688**	0.9453	0.9609
	Success@end ↑	**0.8281**	0.7813	0.7656	**0.8281**
	Actions ↓	24.63	24.41	23.52	**23.35**
Dynamic texture (strong)	Success@once ↑	0.9063	0.9141	0.8672	**0.9219**
	Success@end ↑	0.7188	0.7188	0.6563	**0.7891**
	Actions ↓	25.95	25.60	25.14	**25.08**
Dynamic noise (weak)	Success@once ↑	0.9375	0.9531	**0.9766**	**0.9766**
	Success@end ↑	0.7422	0.7734	**0.8281**	0.8047
	Actions ↓	23.98	23.91	24.50	**23.03**
Dynamic noise (strong)	Success@once ↑	0.8359	0.8672	0.8594	**0.8984**
	Success@end ↑	0.6094	0.6484	0.6094	**0.7344**
	Actions ↓	26.38	25.86	26.88	**24.96**

Table A10. Detailed semantic-shift results for the supplementary MH proof-of-viability sweep with lower-temperature decoding baselines.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	MH-sampling
Semantic avg.	Success@once ↑	0.6518	0.6239	0.6172	**0.6853**
	Success@end ↑	0.4587	0.4531	0.4408	**0.5123**
	Actions ↓	38.31	37.68	37.81	**35.34**
Unseen objects	Success@once ↑	0.8828	0.8594	0.8672	**0.8984**
	Success@end ↑	0.6719	0.6719	0.6953	**0.7109**
	Actions ↓	29.19	29.35	29.46	**24.70**
Unseen receptacles	Success@once ↑	**0.8906**	0.8672	0.8594	0.8672
	Success@end ↑	0.5234	0.5078	0.4453	**0.5547**
	Actions ↓	29.29	28.11	28.45	**25.05**
Unseen instructions	Success@once ↑	0.6875	0.6875	**0.6953**	**0.6953**
	Success@end ↑	0.5313	**0.5938**	0.5703	0.5391
	Actions ↓	23.83	23.01	24.88	**22.61**
Multi-object (both seen)	Success@once ↑	0.6172	0.5859	0.6406	**0.7109**
	Success@end ↑	0.4453	0.4766	0.4922	**0.5781**
	Actions ↓	44.78	44.49	46.60	**39.68**
Multi-object (both unseen)	Success@once ↑	0.6563	0.5859	0.5469	**0.6797**
	Success@end ↑	0.5078	0.3984	0.4375	**0.5547**
	Actions ↓	43.23	43.83	43.70	**38.95**
Distractive receptacle	Success@once ↑	0.5391	0.4688	0.4688	**0.6172**
	Success@end ↑	0.3906	0.3438	0.3359	**0.4688**
	Actions ↓	49.42	48.03	47.70	**44.94**
Multi-receptacle (both unseen)	Success@once ↑	0.2891	0.3125	0.2422	**0.3281**
	Success@end ↑	0.1406	**0.1797**	0.1094	**0.1797**
	Actions ↓	48.41	46.98	**43.90**	51.43

Table A11. Detailed execution-shift results for the supplementary MH proof-of-viability sweep with lower-temperature decoding baselines.

Split	Metric	Baseline (0.6)	Low temp (0.4)	Low temp (0.25)	MH-sampling
Execution avg.	Success@once ↑	0.7474	0.7734	0.7389	**0.7943**
	Success@end ↑	0.5469	0.5677	0.5911	**0.6302**
	Actions ↓	41.05	40.62	40.05	**34.65**
Unseen position (object & receptacle)	Success@once ↑	0.7031	0.6875	0.7422	**0.7734**
	Success@end ↑	0.5391	0.4766	0.5391	**0.6406**
	Actions ↓	44.26	43.01	44.51	**38.02**
Unseen robot init pose	Success@once ↑	0.7344	0.8047	**0.8359**	0.7969
	Success@end ↑	0.4766	0.5156	**0.6563**	0.6406
	Actions ↓	34.01	34.96	34.56	**30.22**
Mid-episode object reposition	Success@once ↑	0.8047	**0.8281**	0.7734	0.8125
	Success@end ↑	0.6250	**0.7109**	0.5781	0.6094
	Actions ↓	44.89	43.80	41.09	**35.71**