From Sample to Subset Construction: Coverage-Aware Curation of Robot Demonstrations

Shuo Liu[1] **Minxin Lai**[1] **Wuyang Zhang**[1,2] **Yu Zhang**[1,†]
Zhuangqiu Huang[1] **Yao Li**[1,3,2] **Yanyong Zhang**[1]
[1]University of Science and Technology of China, Hefei, China
[2]Suzhou Artificial Intelligence Laboratory, Suzhou, China
[3]Suzhou Institute for Advanced Research, University of Science and Technology of China
[†]Corresponding author.

Abstract

Scaling robot demonstrations does not guarantee better imitation: redundant trajectories dilute learning, while rare, imperfect segments may encode critical transitions. Prior curation methods overlook such dependencies by scoring samples independently at the trajectory or segment level. We reframe robot demonstration curation as a coverage-aware sequential selection problem and propose PROSE, which selects data by maximizing marginal utility under a fixed budget. PROSE unifies three signals: (i) influence on closed-loop return, (ii) kinematic reliability, and (iii) coverage gain measuring novelty with respect to the retained set, allowing full trajectories and fine-grained segments jointly participate in scoring and filtering. Across three Robomimic simulation tasks and three real Franka tasks, PROSE achieves the best success rate on every task we evaluate against three state-of-the-art curators (CUPID, SCIZOR, Demo-SCORE), with average closed-loop gains of +15% in simulation and +35% in real-world. Ablations attribute the gain primarily to the coverage step.

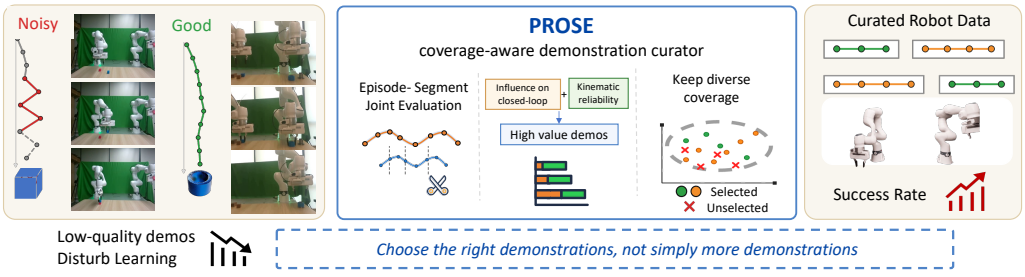


Figure 1: **Motivation of PROSE.** PROSE curates mixed-quality demonstrations into a compact, diverse, and high-value subset for better robot policy learning.

1 Introduction

Robot imitation learning is sensitive to the composition of its training data, not just its size. The same policy trained on more demonstrations can perform worse when those demonstrations are mixed-quality: local mistakes provide incorrect supervision and produce jitter or premature release; redundant trajectories reinforce dominant modes and let the policy overlook rare-but-critical contact states; uneven coverage leaves long-tail initial conditions and recovery actions effectively unsupervised [11–13]. These failure modes are widely documented across imitation-learning datasets [14–17].

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

Useful supervision is therefore not synonymous with successful demonstration: data curation should improve what the policy is trained on, not just shrink it.

Two camps of robot demonstration curators have emerged. Trajectory-level methods rank whole demonstrations by influence on validation return, rollout-success scoring, or distributional statistics [14, 16, 17]. They preserve global task semantics, but a trajectory-level score cannot remove a locally bad segment inside an otherwise successful trajectory: a poorly executed grasp at one frame contaminates supervision even when the rest is clean. Segment-level methods cut at the opposite extreme, filtering or deduplicating at segment scope [15]. They localize errors well, but a segment score cannot tell whether a smooth segment is repeating a frequent pattern or covering a rare one; phase context is lost the moment a trajectory is diced. Both camps share a deeper limitation: each demonstration receives a fixed score that is independent of every other selected demonstration [12, 13], so the curated subset is whatever the top-K of that score happens to be, regardless of whether those K demonstrations cover one behavioral mode or twelve. Demonstration value is therefore not a static property of the sample; it is a function of what is already retained.

Our key observation is that the policy is trained on the retained subset, not the full demonstration dataset. Existing curators either ignore coverage or compute it against the full dataset; the appropriate reference is the subset that actually trains the policy, which grows during selection. A demonstration's value is therefore its contribution to that subset, not a property of the demonstration itself. Two design choices follow. The score of a candidate must be computed against the current subset, because two units with identical isolation scores can have opposite marginal values: a smooth segment that would rank highly in a static distribution becomes redundant once a similar one is retained, while an unusual contact transition that would rank low in isolation becomes essential when the subset has no example of it. The unit of selection must also be flexible, because mixed-quality demonstrations exhibit errors at multiple scales, from frame-level jitter to mode-level redundancy across whole trajectories, and binding the curator to one granularity binds it to one scale of error.

We propose **PROSE**, which maintains a retained-subset state during selection and, at each step, adds the unit (episode or segment) whose combination of policy-target attribution and marginal coverage gain against that state is greatest. Empirically, this design choice dominates: subset-state-conditioned coverage accounts for the largest share of PROSE's improvement over current static-rank curators [14–17].Our main contributions are:

1. **A finding: coverage matters more than scoring.** Across three Robomimic tasks, three real Franka tasks, and three representative curators, removing PROSE's coverage module while leaving every scoring signal intact costs 12 absolute success points, over $4\times$ the drop from removing any single scoring component. Static ranking used by existing curators [14–17] ignores this dependency, exposing a structural blind spot that dominates performance.
2. **PROSE: a coverage-aware, multi-granularity curator.** PROSE selects demonstrations sequentially over a value-and-coverage objective at episode and segment granularities. Unit value fuses an influence-based contribution to closed-loop validation return with a kinematic reliability prior. The curator supports both small policies (ACT [8]) and large vision–language–action models ($\pi_{0.5}$ [10]) without retuning the scoring backbone.
3. **Closed-loop gains across simulation and real-world tasks.** PROSE consistently improves both ACT and $\pi_{0.5}$ over Full Data and three representative curators (CUPID [14], SCIZOR [15], Demo-SCORE [17]). On a bimanual packing task whose mixed teleoperation strategies leave Full Data at 0% success, PROSE reaches 38%.
4. **Closing the curation–acquisition loop.** PROSE's selection statistics enable targeted next-round demonstration acquisition: routing a small additional budget according to model–task preference improves average success by 7 absolute points over random addition.

2 Related Work

Robot demonstration curation. Mixed-quality demonstrations have made data curation a distinct problem in robot imitation learning [11–13], and four representative methods now define the solution space. CUPID [14] estimates each demonstration's influence on closed-loop validation return via inverse-Hessian-vector products and ranks trajectories accordingly. Demo-SCORE [17] trains a binary classifier on online rollout outcomes and ranks demonstrations by classifier confidence. DemInf [16]

computes each trajectory's contribution to the mutual information between states and actions using k-nearest-neighbor estimators over VAE embeddings. SCIZOR [15] drops to segment scope, filtering suboptimal segments via a self-supervised task-progress predictor and deduplicating redundant ones in joint state-action representation. The four methods occupy four different design corners (influence, classifier output, mutual information, similarity), but they share one computational shape: each demonstration receives a fixed score independent of every other selected demonstration, and the curated subset is whatever the top-K of that score happens to be.

Influence-based data attribution. Influence functions [18] estimate how upweighting a training point shifts the post-training validation objective via inverse-Hessian-vector products, providing a target-aware attribution signal at a cost that scales poorly. The scalability lineage substitutes cheaper approximations: LiSSA [19] replaces the inverse Hessian with a stochastic Neumann series; TracIn [20] traces test-loss change across saved training checkpoints, avoiding Hessian inversion entirely; representer-point selection [21] reformulates attribution through a last-layer decomposition; scaled-IF [22] adapts Arnoldi iteration to billion-parameter regimes; GraSS [23] exploits per-sample gradient sparsity to reach sub-linear space and time. CUPID [14] adapts this lineage to robot demonstrations, attributing closed-loop validation return. Influence remains a per-sample signal, however: scaling and accuracy improvements do not change the fact that the top-K of influence scores need not cover the long-tail behavioral modes a closed-loop policy encounters at deployment.

Subset selection, coverage, and active learning. Outside robotics, three lineages of subset selection inform our objective. Core-set methods [24] construct subsets that are representative under feature-space distance, ensuring geometric coverage of the data manifold. Training-dynamics methods score samples by training behavior, such as forgetting events [25], early-training gradient norms [26], or learnability against a holdout target [27]. Subset-objective methods optimize a downstream loss directly: GLISTER [28] solves a bilevel problem to maximize validation log-likelihood; CRAIG [29] selects a weighted subset whose gradient approximates the full-batch gradient; GradMatch [30] adds error-bounded matching; bilevel-coreset [31] unrolls the bilevel objective for continual learning. Active learning balances predictive uncertainty [33] with diversity, and BADGE [32] unifies both via the magnitude and disparity of hallucinated gradient embeddings. These methods inform our coverage formulation, but they were designed for image classification: their coverage features, including pre-softmax embeddings, gradient norms, learnability signals, do not directly encode the task phases, contact states, and recovery behaviors that determine closed-loop robot success.

3 Preliminary

We consider offline robot imitation learning with a demonstration dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, where each trajectory $\tau_i = \{(s_i^t, a_i^t)\}_{t=1}^{T_i}$ consists of multimodal observations s_i^t, expert actions a_i^t, and length T_i. A parameterized policy $\pi_\theta(a \mid s)$ is trained by behavioral cloning [2, 3]:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(\tau_i;\theta), \qquad \ell(\tau_i;\theta) = \frac{1}{T_i}\sum_{t=1}^{T_i} \ell(s_i^t, a_i^t;\theta), \tag{1}$$

yielding $\theta^* = \arg\min_\theta \mathcal{L}(\theta)$. While this objective is defined over offline demonstrations, policy quality is ultimately determined by closed-loop execution. Following CUPID [14], we define the closed-loop validation objective as

$$\mathcal{J}(\theta) = \mathbb{E}_{e\sim\mathcal{E}_{\text{val}}}[R(\pi_\theta, e)]\,, \tag{2}$$

where $R(\pi_\theta, e) \in \{+1, -1\}$ is a binarized success/failure return in validation environment e. The sign convention is deliberate: it lets influence-based attribution (introduced below) assign positive value to units associated with successful rollouts and negative value to units associated with failures.

Curation as subset selection. Let $\mathcal{U}$ be a candidate set of training units. PROSE selects from two granularities: complete trajectories $\tau_i \in \mathcal{D}$ and contiguous fixed-length segments $w_{i,k}$ extracted from each trajectory, formally constructed in §4.1. Given a budget B and a minimum coverage threshold c, the ideal curation objective is

$$\max_{\mathcal{S}\subseteq\mathcal{U}} \mathcal{J}(\theta^*(\mathcal{S}))\,, \qquad \text{s.t.} \quad |\mathcal{S}| \le B, \quad \text{Cov}(\mathcal{S}) \ge c, \tag{3}$$

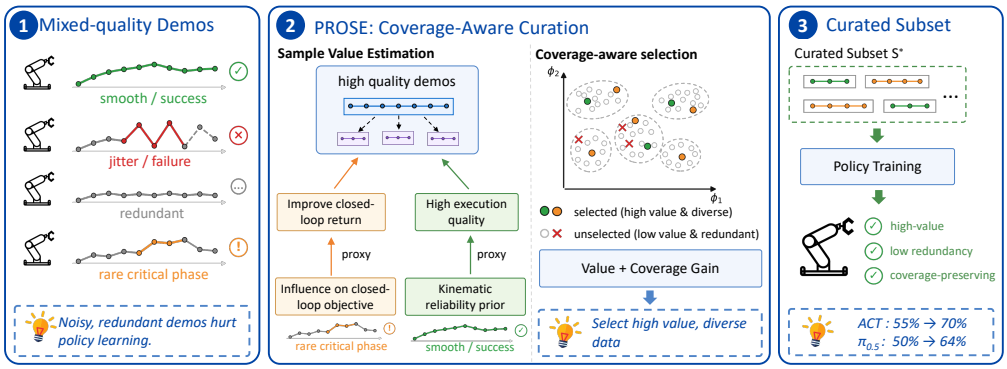


Figure 2: Overview of PROSE. The framework constructs multi-granularity episode–segment training units, evaluates their sample value using contribution scores and kinematic quality priors, and performs coverage-aware selection to obtain a high-value, coverage-preserving subset.

where $\theta^*(\mathcal{S})$ is the policy obtained by training on the retained subset $\mathcal{S}$, and $\mathrm{Cov}(\mathcal{S})$ measures behavioral coverage in a task-relevant state–action feature space, made precise in §4.3. The formulation makes explicit that curation should improve downstream closed-loop return, not minimize offline loss. Eq. (3) is intractable, however, because evaluating any candidate $\mathcal{S}$ would require retraining $\theta^*(\mathcal{S})$ and running closed-loop rollouts. PROSE replaces the per-subset retraining loop with two computable signals: an attribution-based estimate of each unit's contribution to $\mathcal{J}$, defined next, and a marginal coverage gain against the currently retained subset, defined in §4.3.

Influence functions for closed-loop attribution. We use influence functions [18] to estimate how each candidate unit would change $\mathcal{J}(\theta^*)$ without retraining. Upweighting a unit u with loss $\ell(u;\theta)$ by an infinitesimal ϵ shifts the optimum to

$$\theta_{\epsilon,u} = \arg\min_{\theta} \left[\mathcal{L}(\theta) + \epsilon\, \ell(u;\theta)\right], \tag{4}$$

and the first-order influence of u on the closed-loop objective is

$$I_{\mathcal{J}}(u) = \left.\frac{d\mathcal{J}(\theta_{\epsilon,u})}{d\epsilon}\right|_{\epsilon=0} = -\nabla_\theta \mathcal{J}(\theta^*)^\top H_{\theta^*}^{-1}\, \nabla_\theta \ell(u;\theta^*), \qquad H_{\theta^*} = \nabla_\theta^2 \mathcal{L}(\theta^*). \tag{5}$$

A positive $I_{\mathcal{J}}(u)$ indicates that upweighting u is expected to improve closed-loop return; a negative value indicates that u provides low-quality, redundant, or distribution-mismatched supervision. PROSE uses $I_{\mathcal{J}}(u)$ as the attribution component of its sample-value score (§4.2); the coverage component is defined separately in §4.3.

4 Method

The ideal curation objective (Eq. 3) is intractable, since evaluating any candidate subset would require retraining the policy and running closed-loop rollouts. PROSE approximates it by answering three questions in sequence (Figure 2): what is a unit of selection? what is each unit worth? and how do the chosen units fit together? Implementation details are in Appendix B.2.

❶ The first step constructs the candidate set $\mathcal{U}$. Errors in mixed-quality demonstrations span time scales from frame-level jitter to mode-level redundancy across whole trajectories, so no single granularity captures all of them. PROSE therefore represents each demonstration at two granularities: complete trajectories τ_i provide global reliability context, while contiguous fixed-length segments $w_{i,k}$ provide the units selected for training. The two granularities jointly inform scoring and filtering.

❷ Given $\mathcal{U}$, the second step asks what a unit is worth. The natural target is the closed-loop validation objective $\mathcal{J}$: we estimate each unit's effect on $\mathcal{J}(\theta^*)$ by first-order influence $I(u)$, avoiding retraining. Because influence is sensitive to validation-rollout noise and Hessian approximation, we calibrate it with a kinematic reliability prior $q(u)$ that scores execution quality independently of attribution. The fused score $s(u)$ combines the two (§4.2).

❸ A static ranking by $s(u)$ would already define a curator, but it is precisely the curator we set out to improve upon: two units with identical $s(u)$ can contribute very differently depending on what is already retained. The third step therefore selects greedily over $\mathcal{S}$: each step adds the unit maximizing a marginal gain $\Delta(u \mid \mathcal{S})$ combining $s(u)$ and coverage novelty $N(u, \mathcal{S})$ in $\phi(u)$ (§4.3). Selection stops when the marginal gain falls below an adaptive threshold or the budget is exhausted, yielding $\mathcal{S}^*$ for training π_θ. Influence backends and cost are detailed in Appendix A.

4.1 Stage 1: Episode and Segment Unit Construction

Each demonstration trajectory $\tau_i \in \mathcal{D}$ contributes two kinds of training unit to the candidate set $\mathcal{U}$. The episode-level unit is the trajectory itself. The segment-level units are obtained by sliding a fixed-length window over τ_i: given a window length L_s and a stride h_s,

$$w_{i,k} = \{(s_i^t, a_i^t)\}_{t=t_k}^{t_k+L_s-1}, \qquad t_k = 1 + (k-1)h_s, \qquad k = 1, \ldots, K_i, \tag{6}$$

where $K_i = \lfloor (T_i - L_s)/h_s \rfloor + 1$ is the number of segments produced by trajectory i. The candidate set on which subsequent stages operate is the union of all episode-level and segment-level units:

$$\mathcal{U} \;=\; \{\tau_i\}_{i=1}^{N} \;\cup\; \big\{w_{i,k} \,:\, i = 1, \ldots, N,\; k = 1, \ldots, K_i\big\}. \tag{7}$$

Subsequent stages treat $\mathcal{U}$ uniformly: a candidate $u \in \mathcal{U}$ is either a complete trajectory or a contiguous segment, and PROSE's scoring (§4.2) and selection (§4.3) operate on both unit types. Trajectories shorter than L_s contribute only their episode-level unit. When $T_i - L_s$ is not divisible by h_s, the final segment is right-aligned to T_i so that every retained frame belongs to at least one segment.

4.2 Stage 2: Sample Value Estimation

Each candidate $u \in \mathcal{U}$ receives a sample-value score that fuses two complementary signals: the influence-based contribution $\widetilde{I}(u)$ from §3 and a kinematic reliability prior $q(u)$ defined below. The prior calibrates attribution against execution noise: a unit with high influence but poor execution quality is more likely an attribution artifact than a useful demonstration. We define q separately at the two granularities, since the statistics that are reliable at each scale differ.

Episode-level prior. For a complete trajectory τ_i, q_i aggregates three components.

Terminal-state convergence. Let $\mathcal{N}_i = \mathrm{kNN}(\mathbf{s}_{i,T_i})$. Define $d_i = \frac{1}{k}\sum_{\mathbf{z}\in\mathcal{N}_i} \|\mathbf{s}_{i,T_i} - \mathbf{z}\|_2$ (average distance from the terminal state to its k nearest terminal states across the dataset), $g_i = \|\mathbf{s}_{i,T_i} - \mathbf{s}_{i,1}\|_2$ (overall displacement), and $e_i = g_i / \sum_t \|\mathbf{s}_{i,t+1} - \mathbf{s}_{i,t}\|_2$ (path efficiency). Trajectories that converge to common terminal regions, displace meaningfully, and travel directly score high.

Motion smoothness. For $x \in \{s, a\}$, $r_i^x = \mathbb{E}_t[\|\mathbf{x}_{i,t+1} - \mathbf{x}_{i,t}\|_2] / \mathbb{E}_t[\|\mathbf{x}_{i,t}\|_2]$ is the relative step-wise jitter at the state and action levels.

Temporal-structure regularity. Projecting each step-wise change onto the displacement direction $\mathbf{u}_i = (\mathbf{s}_{i,T_i} - \mathbf{s}_{i,1})/\|\mathbf{s}_{i,T_i} - \mathbf{s}_{i,1}\|_2$ yields the monotonic-progress, backtracking, and stagnation ratios $p_i^{\uparrow}, p_i^{\downarrow}, p_i^{0}$. The length-regularity term $r_i^l = \exp(-|T_i - L_{\mathrm{med}}|/(\alpha \cdot \mathrm{MAD}))$ penalizes trajectories abnormally short or long relative to the dataset median L_{med}.

After normalizing each statistic to $[0, 1]$ across the dataset, the three components combine into the trajectory-level prior:

$$c_{i,1} = H_1(d_i, g_i, e_i), \quad c_{i,2} = H_2(r_i^s, r_i^a), \quad c_{i,3} = H_3(p_i^{\uparrow}, p_i^{\downarrow}, p_i^{0}, r_i^l), \tag{8}$$

$$q_i = \sum_{m=1}^{3} \omega_m \, c_{i,m}, \qquad \omega_m \geq 0, \quad \sum_m \omega_m = 1, \tag{9}$$

where each H_m is a monotone weighted aggregator of normalized inputs: H_1 rewards convergence and progress, H_2 penalizes jitter, and H_3 rewards sustained progress while suppressing backtracking, stagnation, and length irregularity. The decomposition catches trajectories that are not explicit failures but still mislead policy learning—for example, smooth trajectories whose terminal states deviate from successful regions ($c_{i,1}$ low), or long trajectories with little effective task progress ($c_{i,3}$ low).

Segment-level prior. For short segments $w_{i,k}$, long-horizon progress and terminal-state cues are unreliable; therefore, q only retains the local smoothness term:

$$j^x_{i,k} = \mathbb{E}_{t \in w_{i,k}} \|\mathbf{x}_{i,t+1} - \mathbf{x}_{i,t}\|_2, \quad x \in \{s, a\}, \qquad q_{i,k} = H_s(j^s_{i,k}, j^a_{i,k}), \tag{10}$$

where H_s penalizes high-frequency jitter at either the state or action level. This design captures local jitter, control discontinuities, and short-horizon execution anomalies without making unstable terminal-state judgments on too little context.

Joint episode-segment scoring. The episode-level sample value combines contribution and prior:

$$s_i = \eta_1 \widetilde{I}(\tau_i) + \eta_2 \, q_i, \qquad \eta_1, \eta_2 \geq 0, \quad \eta_1 + \eta_2 = 1. \tag{11}$$

s_i has two roles: it identifies low-value or unreliable trajectories, and it injects global reliability into the segment score. We first drop the bottom ρ_e fraction of trajectories ranked by ascending s_i ($\eta_1 = \eta_2 = 0.5, \rho_e = 0.1$ by default), and for each surviving segment compute

$$s_{i,k} = \lambda_1 \widetilde{I}(w_{i,k}) + \lambda_2 \, q_{i,k} + \lambda_3 \, s_i, \qquad \lambda_m \geq 0, \quad \sum_m \lambda_m = 1. \tag{12}$$

With default $\lambda_1 = \lambda_2 = 0.4$ and $\lambda_3 = 0.2$, the segment score weights local contribution and local quality equally and adds a smaller share of parent-trajectory reliability, so a locally clean segment from a globally unreliable trajectory cannot dominate the budget on local appearance alone. The scores $\{s_{i,k}\}$ feed Stage 3 (§4.3).

4.3 Stage 3: Coverage-aware Selection

The third stage constructs the final curated subset $\mathcal{S}^*$ via a greedy procedure: starting from $\mathcal{S} = \emptyset$, each step adds the unit whose marginal gain—a weighted combination of sample-value score and coverage novelty against the current subset—is largest among remaining candidates.

Marginal gain. For a candidate unit u (an episode τ_i or a segment $w_{i,k}$) with sample-value score $s(u)$ and current subset $\mathcal{S}$, define

$$\Delta(u \mid \mathcal{S}) = \gamma_1 \, s(u) + \gamma_2 \, N(u, \mathcal{S}), \qquad \gamma_1, \gamma_2 \geq 0, \quad \gamma_1 + \gamma_2 = 1, \tag{13}$$

where $N(u, \mathcal{S})$ denotes the coverage novelty of segment u with respect to the already selected subset $\mathcal{S}$, as defined next.

Coverage novelty. Each unit u is associated with a feature vector $\phi(u)$ summarizing its task-relevant content, such as within-window state statistics, phase position, or end-effector pose. The default coverage representation uses low-dimensional proprioceptive states observable by the policy and does not require privileged environment information. For a segment of trajectory i starting at t_k and spanning L_s steps, let s^t_i denote the observable state at time t. We concatenate the mean state within the segment with its first and last states:

$$\phi(w_{i,k}) = \mathrm{concat}\left[\frac{1}{L_s} \sum_{t=t_k}^{t_k+L_s-1} s^t_i, \; s^{t_k}_i, \; s^{t_k+L_s-1}_i \right]. \tag{14}$$

Novelty is the (normalized) minimum feature distance from u to the current subset:

$$N(u, \mathcal{S}) = \begin{cases} 1, & \mathcal{S} = \emptyset, \\ \mathrm{clip}\left(\dfrac{\min_{u' \in \mathcal{S}} \|\phi(u) - \phi(u')\|_2}{D^{\max}}, \; 0, \; 1 \right), & \text{otherwise}, \end{cases} \tag{15}$$

where $D^{\max} = \max_{u,v \in \mathcal{U}} \|\phi(u) - \phi(v)\|_2$ normalizes distances to $[0, 1]$ and clip prevents outlier candidates with anomalously large feature distances from dominating selection.

Adaptive stopping. Greedy selection stops when the best available marginal gain saturates. With Δ_1 the gain of the first selected unit and $\Delta^\star$ the gain of the next candidate in the queue, the stopping condition is

$$\Delta^\star < \max\big(\delta_{\text{abs}},\ \delta_{\text{rel}}\,\Delta_1\big), \tag{16}$$

combining an absolute floor δ_{abs} and a relative threshold scaled by Δ_1. Before stopping, we additionally check that each trajectory's temporal coverage

$$\text{Cov}(\mathcal{S}_i) = \frac{1}{T_i}\sum_{t=1}^{T_i} \mathbb{I}[\exists\, w \in \mathcal{S}_i,\ t \in w] \tag{17}$$

meets the threshold c from §3, so no trajectory is left under-represented in $\mathcal{S}^*$.

Output. PROSE returns the curated subset $\mathcal{S}^*$ for training π_θ, together with per-step selection statistics (the sequence of Δ values, removed regions, and low-coverage trajectories) that we use as a diagnostic signal for targeted next-round demonstration acquisition.

5 Experiments

We evaluate PROSE around three questions. **Q1: Does PROSE improve closed-loop performance in simulation and reality? Q2: How do the key design choices affect final selection quality? Q3: Are data-value preferences policy-conditioned?**

5.1 Experimental Setup

Datasets. We evaluate PROSE on simulated and real-world robot tasks. For simulation, we use three mixed-quality `robomimic` tasks [11], each with 300 successful demonstrations collected by six RoboTurk operators. For real-world evaluation, we use three Franka tasks with 200 demonstrations each: `Stack_Cube`, a single-arm stacking task with mixed failures; `Insert_Ring`, a fine insertion task with occlusion and limited camera views; and `Pack_Box`, a bimanual packing task with synchronized and staggered-start strategies. Together, these tasks cover mixed data quality, partial observability, multi-strategy distributions, contact uncertainty, and real-world control delay.

Baselines. We evaluate PROSE with ACT [8] and the general Vision-Language-Action model $\pi_{0.5}$ [10], which differ in temporal modeling, action prediction, and training scale. For data selection, we compare against `Full Data`, CUPID [14], SCIZOR [15], and Demo-SCORE [17]. For rollout-dependent methods, we follow CUPID and collect 50 test samples per task. All results are averaged over three independent runs. Additional setup details are provided in Appendix B.

5.2 Q1: Does PROSE improve closed-loop performance in simulation and reality?

Simulation performance. Table 1 shows that PROSE consistently improves simulation performance over both `Full Data` and existing curation baselines. Across the three simulated tasks, PROSE achieves the best average success rate for both ACT and $\pi_{0.5}$. For ACT, PROSE improves the average success rate from 55% with `Full Data` to 70%, outperforming CUPID (51%), SCIZOR (50%), and Demo-SCORE (50%). For $\pi_{0.5}$, PROSE improves the average success rate from 50% to 64%, and achieves the best result on all three simulation tasks. These tasks cover complementary challenges, including short-horizon grasping and placing in `Can_mh`, contact-rich alignment in `Square_mh`, and longer-horizon object transfer in `Transport_mh`. The largest gain appears on ACT `Square_mh`, where PROSE raises success from 35% to 55%, indicating that filtering noisy or poorly aligned segments is especially beneficial for contact-sensitive tasks. Overall, the simulation results show that combining sample value with coverage gain yields a stronger training subset than methods based only on influence, segment filtering, or rollout scoring.

Real-world evaluation. The real-world tasks provide a stronger transfer test with sensing noise, execution errors, contact uncertainty, camera-view limits, and control delay. As shown in Table 1, PROSE achieves the best success rate on all real-world evaluations and produces shorter successful rollouts. For ACT, the average success rate improves from 16% with `Full Data` to 45%: `Stack_Cube` rises from 28% to 50%, and `Insert_Ring` from 4% to 40%. For $\pi_{0.5}$, the average success rate improves

Table 1: Success rates (%), average completion lengths of successful test rollouts (Len., in steps), and training-frame drop ratios (%) on simulated and real-world robot tasks.

Model	Env.	Task	Full Data		CUPID [14]			SCIZOR [15]			Demo-SCORE [17]			**PROSE**		
			Succ. ↑	Len. ↓	Succ. ↑	Len. ↓	Drop	Succ. ↑	Len. ↓	Drop	Succ. ↑	Len. ↓	Drop	Succ. ↑	Len. ↓	Drop
ACT	Sim.	Can_mh	72	209	69	205	14.80	78	204	29.56	66	220	15.74	**82**	**196**	20.51
		Square_mh	35	252	35	275	16.51	31	266	29.51	26	255	32.83	**55**	**229**	30.90
		Transport_mh	58	626	48	660	18.94	42	640	29.30	59	620	29.25	**72**	**575**	43.01
	Real	Stack_Cube	28	682	20	705	20.18	10	678	29.41	30	676	36.87	**50**	**635**	23.91
		Insert_Ring	4	384	28	372	19.20	24	380	32.10	12	398	38.42	**40**	**356**	47.20
$\pi_{0.5}$	Sim.	Can_mh	83	232	83	232	14.80	87	226	29.56	81	238	16.31	**93**	**218**	19.36
		Square_mh	43	289	41	294	16.51	44	286	29.51	44	285	32.67	**55**	**266**	27.70
		Transport_mh	24	688	36	654	18.94	14	740	29.30	34	662	29.25	**44**	**628**	35.70
	Real	Stack_Cube	62	706	50	735	20.18	30	780	29.41	70	690	47.62	**90**	**642**	38.30
		Insert_Ring	12	402	32	380	19.20	48	394	32.10	28	402	39.20	**68**	**368**	49.05
		Pack_Box	0	N/A	16	1124	19.84	24	1089	29.51	N/A	N/A	N/A	**38**	**1042**	39.20

from 25% to 65%: `Stack_Cube` rises from 62% to 90%, `Insert_Ring` reaches 68% under limited field of view, and `Pack_Box` reaches 38% where `Full Data` obtains 0%. The `Pack_Box` result highlights the importance of coverage-aware selection: demonstrations contain different bimanual timing strategies, and directly mixing them can map similar observations to inconsistent phase structures, causing premature left-arm motion, asynchronous execution, or unstable transitions. By filtering low-value segments while preserving coverage over task phases and strategy variants, PROSE retains more learnable structure instead of collapsing to frequent strategies or mixing incompatible behaviors. Demo-SCORE is marked as `N/A` when its trajectory-level classifier cannot be trained stably in real-world settings. Further real-world protocol details are in Appendix B.5. Overall, these results show that a smaller, higher-value, and better-covered subset can transfer better than all available demonstrations.

5.3 Q2: How do the key design choices affect final selection quality?

We study how PROSE's design choices affect the final curated subset. The key question is whether the gain comes from simply removing data, or from combining sample-value estimation, multi-granularity modeling, and coverage-aware selection. Sample-value estimation combines downstream contribution with local kinematic quality, while coverage-aware selection prevents the retained subset from collapsing to high-scoring but redundant modes. We further test whether joint episode–segment modeling outperforms single-granularity filtering. Unless otherwise specified, all ablations are conducted on `robomimic: Can_mh` with ACT, using closed-loop success rate as the metric. Supplementary interpretation is in Appendix B.5.

Component and Granularity Ablations. Table 2 shows that each design choice is necessary for high-quality selection. Full PROSE achieves the best success rate of 82%, while all ablated variants trail it by 4–12 points. Using only contribution scores or only quality priors reduces success to 77% and 75%, respectively, showing that downstream usefulness and local execution quality provide complementary signals. The granularity ablations further show that `episode_only` misses harmful local segments inside otherwise useful trajectories, while `segment_only` lacks trajectory-level context and remains 4 points below full PROSE. The largest drop comes from removing coverage-aware selection, which reduces success to 70% despite dropping a similar fraction of data. This indicates that score-only selection tends to over-concentrate on a few high-scoring modes and weakens state–action coverage. Final selection quality depends not only on estimating valuable units, but also on preserving sufficient coverage across task phases and behavior modes.

Table 2: Component ablation results.

Variant	Succ. (%)	Drop (%)
PROSE	**82**	20.51
`episode_only`	75	18.52
`segment_only`	78	16.37
`influence_only`	77	19.87
`quality_only`	75	25.34
`w/o_coverage`	70	22.54

Hyperparameter Sensitivity. Table 3 shows that PROSE requires a moderate filtering strength and an appropriate segment time scale. For the drop ratio, increasing ρ from 0.05 to 0.10

improves success from 79% to 82%, suggesting that mild filtering removes low-value supervision. However, larger ratios reduce success to 71% at $\rho = 0.15$ and 75% at $\rho = 0.20$, indicating that aggressive filtering discards useful variation and damages retained-set coverage. Segment length shows a similar trade-off: $w_s = 25$ provides insufficient temporal context, while $w_s = 75$ smooths out local execution errors. The best result is achieved at $w_s = 50$, which balances local diagnostic ability with enough temporal context. PROSE is not effective because it arbitrarily removes data; it works when filtering is strong enough to suppress harmful segments but moderate to preserve coverage and structure.

Table 3: Hyperparameter ablation results.

Param.	Value	Success Rate (%)
ρ	0.05	79
	0.10	**82**
	0.15	71
	0.20	75
w_s	25	71
	50	**82**
	75	74

5.4 Q3: Are data-value preferences policy-conditioned?

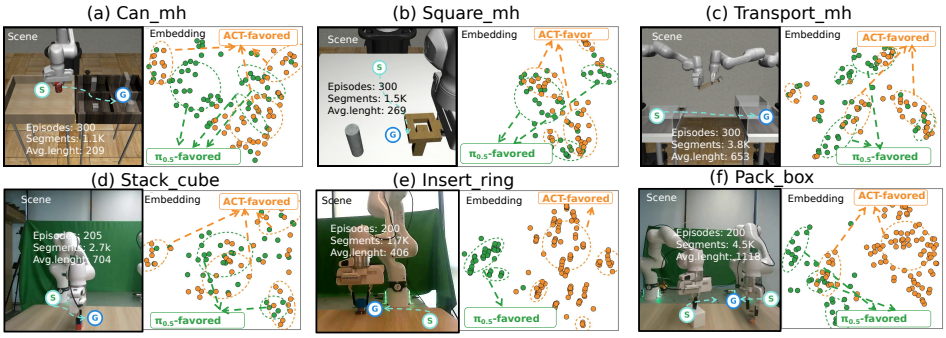


Figure 3: Policy-conditioned preference differences between ACT and $\pi_{0.5}$ on the same training sets, visualized in trajectory-statistic embedding spaces.

Figure 3 shows that data-value preferences are clearly policy-conditioned, but the preference pattern also depends on task structure. In `Can_mh`, where trajectories are relatively dispersed, ACT-only preferences concentrate on longer and smoother demonstrations, while $\pi_{0.5}$-only preferences appear more often in shorter regions with sharper state–action changes. In `Square_mh`, where trajectories form local clusters, ACT favors longer assembly processes with larger displacement, whereas $\pi_{0.5}$ covers more compact local-adjustment behaviors. In `Transport_mh`, which forms a longer temporal manifold, ACT tends to prefer medium-short trajectories with clearer progress, while $\pi_{0.5}$ assigns higher value to longer-horizon and smoother transport behaviors. These results show that there is no universal preference rule such as "longer and smoother is always better". Instead, the same trajectory property can play different roles across tasks. **Takeaway:** data curation should be conditioned on both the target policy and the task structure; high-value rankings should not be directly transferred across models or tasks.

Table 4 further tests whether policy-conditioned preferences can guide additional data acquisition. Using `ph` demonstrations as an offline proxy for next-round collection, preference-guided addition consistently outperforms random addition for $\pi_{0.5}$. Guided PH addition improves the average success rate from 64% with Curated MH to 71%, while random PH addition reaches only 63% on average. The improvement is especially clear on the harder tasks: guided addition improves `Square` from 55% to 64% and `Transport` from 44% to 52%, whereas random addition slightly hurts both tasks. This result indicates that the preference regions exposed by PROSE are not only descriptive, but also useful for allocating a limited demonstration budget. Policy-conditioned diagnostics can identify under-covered but valuable data regions, enabling targeted acquisition that is more effective than random data addition. Appendix C provides the experimental details.

Table 4: Preference-guided acquisition results. Success rates (%) are reported for $\pi_{0.5}$.

Task	Curated MH	+ Random PH	+ Guided PH
Can	93	95	**98**
Square	55	52	**64**
Transport	44	42	**52**

5.5 Additional Validation

Coverage Baseline Comparison. We compare PROSE with a k-center baseline that incorporates influence and quality scores, using the same trajectory and segment representations and keeping all other settings identical. Across five policy–task settings, mean success increases from 60.6% for k-center to 71.4% for PROSE, a gain of 10.8 percentage points. For ACT, success increases from 81% to 82% on `Can_mh`, from 36% to 55% on `Square_mh`, and from 51% to 72% on `Transport_mh`. For $\pi_{0.5}$, success increases from 85% to 93% on `Can_mh` and from 50% to 55% on `Square_mh`. These results support jointly considering sample value and retained-subset composition rather than relying on a generic coverage strategy with sample scores alone (Appendix B.3, Table 8).

Coverage Representation Sensitivity. We isolate the effect of the coverage representation by varying only the input features of ϕ for ACT on `Can_mh`, with all other settings fixed. State, State + Image, and Image representations achieve success rates of 82%, 76%, and 74%, respectively. State-based coverage thus exceeds the two image-based alternatives by 6 and 8 percentage points, and adding image features does not improve performance under this protocol. This supports the default low-dimensional state representation in Equation 14 for the reported experiments. The result is limited to the evaluated task and policy: it does not establish that visual features are less useful for other tasks, policies, or representation encoders (Appendix B.4, Table 9).

Selection Overhead. We measure the final selection stage after candidate segments, sample scores, and coverage representations have been prepared. Coverage, k-center, and Static use the same retention ratios of 20%, 40%, 60%, and 80% on `Can_mh`, `Square_mh`, and `Transport_mh`; each configuration is measured 20 times. The reported Coverage times range from 42.0 to 101.4 ms, while the maximum observed time across configurations remains below 127 ms. Coverage takes 1.25–1.83 times as long as k-center, but allocates only 63.2–73.8 KiB, approximately 17–19% of the memory allocated by k-center. Static is faster, at 0.35–0.40 ms, so coverage-aware selection introduces a measurable time cost even though its absolute overhead remains limited at the evaluated scale. These measurements exclude influence estimation, representation construction, and distance-normalization preprocessing; they characterize final selection rather than the end-to-end curation pipeline (Appendix A.3, Table 5).

6 Conclusion and Limitations

Conclusion. We studies mixed-quality demonstration data in robot imitation learning and proposes PROSE, a coverage-aware multi-granularity curator. PROSE treats data curation as subset construction rather than static ranking, combining contribution and quality priors with episode–segment selection to retain useful supervision while avoiding redundant or harmful regions. Across simulated and real-world tasks, PROSE improves closed-loop success for both ACT and $\pi_{0.5}$ policies, and reveals stable model-specific preferences over the same demonstrations. These results support target-policy-conditioned curation as a practical alternative to generic demonstration cleaning. Reproducibility notes are provided in Appendix B.5.

Limitations. PROSE assumes access to a small set of validation rollouts, so it is most suitable for mixed-quality imitation-learning settings where such feedback is available. Its contribution estimates are also more expensive than purely offline heuristic filters, although this cost is confined to curation and can be reduced with random projection, blockwise approximation, and gradient caching. The current quality priors focus on terminal deviation, local smoothness, and trajectory duration; language-conditioned, contact-rich, or long-horizon tasks may require semantic signals such as language consistency, contact state, phase progress. The shared defaults reduce the need for tuning in the evaluated settings, but their suitability for larger and more diverse tasks remains to be validated.

Acknowledgments

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project under Grant 2025ZD0122604 and in part by the National Natural Science Foundation of China under Grant 62272434.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, et al. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, 1989.

[3] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[4] Todd Hester, Matej Vecerik, Olivier Pietquin, et al. Deep Q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

[5] Tianhe Yu, Chelsea Finn, Annie Xie, et al. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems*, 2018.

[6] Corey Lynch, Mohi Khansari, Ted Xiao, et al. Learning latent plans from play. In *Conference on Robot Learning*, 2020.

[7] Pete Florence, Corey Lynch, Andy Zeng, et al. Implicit behavioral cloning. In *Conference on Robot Learning*, 2021.

[8] Tony Z. Zhao, Vikash Kumar, Sergey Levine, et al. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

[9] Kevin Black, Noah Brown, Danny Driess, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv preprint arXiv:2410.24164, 2024.

[10] Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. arXiv preprint arXiv:2504.16054, 2025.

[11] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, et al. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Conference on Robot Learning*, 2021.

[12] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. In *Advances in Neural Information Processing Systems*, 2023.

[13] Mark Beliaev, Andy Shih, Stefano Ermon, et al. Imitation learning by estimating expertise of demonstrators. In *International Conference on Machine Learning*, 2022.

[14] Christopher Agia, Rohan Sinha, Jingyun Yang, et al. CUPID: Curating data your robot loves with influence functions. In *Conference on Robot Learning*, 2025.

[15] Yu Zhang, Yuqi Xie, Huihan Liu, et al. SCIZOR: Self-Supervised Data Curation for Large-Scale Imitation Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[16] Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, et al. Robot data curation with mutual information estimators. In *Robotics: Science and Systems*, 2025.

[17] Annie S. Chen, Alec M. Lessing, Yuejiang Liu, et al. Curating demonstrations using online experience. In *Robotics: Science and Systems*, 2025.

[18] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

[19] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017.

[20] Garima Pruthi, Frederick Liu, Satyen Kale, et al. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, 2020.

[21] Chih-Kuan Yeh, Joon Sik Kim, Ian E. H. Yen, et al. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[22] Andrea Schioppa, Polina Zablotskaia, David Vilar, et al. Scaling up influence functions. In *AAAI Conference on Artificial Intelligence*, 2022.

[23] Pingbang Hu, Joseph Melkonian, Weijing Tang, et al. GraSS: Scalable influence function with sparse gradient compression. In *Advances in Neural Information Processing Systems*, 2025.

[24] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[25] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, et al. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019.

[26] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, 2021.

[27] Sören Mindermann, Jan M. Brauner, Muhammed T. Razzak, et al. Prioritized training on points that are learnable, worth learning, or not yet learnt. In *International Conference on Machine Learning*, 2022.

[28] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, et al. GLISTER: Generalization based data subset selection for efficient and robust learning. In *AAAI Conference on Artificial Intelligence*, 2021.

[29] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, 2020.

[30] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, et al. GradMatch: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, 2021.

[31] Zalán Borsos, Mojmír Mutný, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In *Advances in Neural Information Processing Systems*, 2020.

[32] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, et al. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

[33] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, 2017.

A Influence estimation and efficiency analysis

The main text gives the basic definition of influence functions [18]. This appendix clarifies how PROSE turns influence into a scalable contribution signal for robot data curation. The main takeaway is that influence is used as one input to sample value, not as a stand-alone selection rule.

A.1 Influence estimation backend

Computation pipeline. PROSE estimates influence through a reusable attribution pipeline:

1. Given trained policy parameters θ^*, PROSE estimates the closed-loop objective gradient $\nabla_\theta \mathcal{J}(\theta^*)$ from validation rollouts.
2. For each episode or segment u in the training set, it computes or caches the training-loss gradient $\nabla_\theta \ell(u; \theta^*)$.
3. The ideal score requires the curvature-corrected quantity $H_{\theta^*}^{-1} \nabla_\theta \ell(u; \theta^*)$. Direct Hessian construction or inversion is impractical for both ACT and $\pi_{0.5}$, so PROSE uses inverse Hessian-vector products or low-dimensional approximations.
4. The corrected training-unit gradient is paired with the validation-objective gradient to obtain the contribution score of the training unit.

This pipeline produces a performance-aligned attribution score. The final retained subset is still determined only after quality calibration and coverage-aware selection.

Backend choices for ACT and $\pi_{0.5}$. The two policy families use different influence backends because their parameter structures create different computational bottlenecks. ACT has a smaller parameter scale, and its action-chunking structure allows training-sample gradients to be compressed in a unified space. We therefore use global random projection or low-dimensional gradient caching. This route is simple, fast for retrieval, and reusable for both episode-level and segment-level ranking after one caching pass, although it gives only a coarse view of local curvature differences because all parameters share the same compressed space.

$\pi_{0.5}$ has a larger model structure and stronger curvature differences across parameter blocks. We therefore use blockwise compression and local second-order approximation: gradients are represented by parameter module, inverse Hessian-vector products are approximated within each block, and block scores are aggregated into the training-unit contribution score. This route costs more computation but better preserves differences among visual encoding, state conditioning, and action-prediction modules. The backend choice is therefore architectural rather than methodological: both variants feed the same sample-value and coverage-aware selection logic.

Practical acceleration. PROSE keeps influence estimation tractable by caching training-sample gradient representations once, which avoids repeated backpropagation for each candidate subset. Curvature correction is approximated with random projection, blockwise low-rank approximation, or Neumann/LiSSA iterations [19, 23], so explicit Hessian construction is not required. Attribution results are also reused across episode and segment granularities. When fine-grained segment-level attribution is unavailable, segment contribution can be initialized from the trajectory-level contribution score and then calibrated by local quality priors. These accelerations make influence useful for ranking large candidate pools while keeping the expensive second-order computation off the critical selection path.

Why influence is not used alone. Influence is closer to closed-loop policy performance than purely offline quality heuristics, but it is still affected by rollout noise, Hessian approximation error, and non-convex training dynamics. PROSE therefore uses influence only to estimate whether a training unit is expected to help the target policy. The quality prior then calibrates whether that attribution is kinematically reliable, and coverage-aware selection prevents high-value retained samples from collapsing into a small number of repeated modes. The practical takeaway is that influence identifies candidate utility, while quality and coverage determine whether that utility should be trusted and retained in the final subset.

A.2 Efficiency analysis

The two influence implementations are included to show that PROSE is not tied to one approximation scheme. ACT favors a unified global projection for fast retrieval, while $\pi_{0.5}$ favors blockwise compression and local preconditioning for higher fidelity to parameter-block structure.

Shared base cost. Both implementations first extract reusable gradient representations for all training samples. This shared pass has base cost $\mathcal{O}(N_{\text{train}}P)$, where N_{train} is the number of training samples and P is the model parameter dimension.

ACT global-compression cost. For ACT, let K be the unified projection dimension. The resulting complexity is time $= \mathcal{O}(N_{\text{train}}P + N_{\text{train}}K)$ and space $= \mathcal{O}(N_{\text{train}}K + P)$. The advantage is structural simplicity: additional overhead depends only linearly on K, making this backend suitable for large-scale influence retrieval. The limitation is that the same projection compresses all parameter dimensions, so local curvature and block-level heterogeneity are represented coarsely.

$\pi_{0.5}$ blockwise cost. For $\pi_{0.5}$, let K_i be the compression dimension of parameter block i, let B be the number of blocks, and define $K_{\text{eff}} = \sum_{i=1}^{B} K_i$. The time complexity is

$$\text{time} = \mathcal{O}\left(N_{\text{train}}P + \sum_{i=1}^{B}(N_{\text{train}}K_i^2 + K_i^3) + N_{\text{train}}K_{\text{eff}} \right).$$

The corresponding space complexity is $\mathcal{O}\Big(N_{\text{train}}K_{\text{eff}} + \sum_{i=1}^{B} K_i^2\Big)$. The advantage is a finer approximation of block-level second-order information. The limitation is the added blockwise computation and storage overhead, especially when the model has many blocks or some K_i values are large.

Takeaway. ACT corresponds to an efficient global-compression paradigm for fast retrieval. $\pi_{0.5}$ corresponds to a finer blockwise second-order paradigm for architectures where local structure matters more. PROSE uses either backend only to estimate contribution; the data-curation logic remains the same across policy architectures.

A.3 Final selection-stage overhead

This section reports the absolute overhead, relative comparisons, and incremental complexity of the final selection stage. The measurements assume that scores, representations, and candidate segments are already available and therefore exclude the influence-estimation backend costs discussed in this appendix. Exact computation of $D^{\max}$ over candidate pairs may also incur quadratic-time normalization preprocessing; this cost should be accounted for separately from incremental distance updates. Influence estimation, representation construction, normalization preprocessing, and final selection together constitute the end-to-end curation pipeline, so the selection-stage measurements alone do not represent its total cost. The correspondence between the measurement hardware and the reported experimental setup, as well as other protocol details, remains to be verified.

Experimental setup. We measure final selection-stage overhead after preparing candidate segments, sample scores, and coverage representations. Coverage, k-center, and Static use the same retention ratios across `Can_mh`, `Square_mh`, and `Transport_mh`; each configuration is measured 20 times. Table 5 reports absolute overhead, with memory allocation in KiB.

Table 5: Absolute selection overhead (ms, KiB).

Retained	Coverage		k-center		Static	
	Time	Mem.	Time	Mem.	Time	Mem.
20%	42.0	63.2	22.9	367.1	0.35	52.2
40%	66.8	69.8	41.7	373.5	0.40	52.6
60%	92.1	73.8	60.7	379.9	0.35	51.2
80%	101.4	72.7	81.0	386.3	0.40	52.2

Efficiency and complexity. Coverage takes 42.0–101.4 ms, or 1.25–1.83 times as long as k-center, while allocating only 17–19% as much memory. It takes 121–265 times as long as Static, but the maximum observed time across all configurations remains below 127 ms. Thus, at the evaluated data scale, Coverage reduces memory allocation relative to k-center with a limited absolute time overhead. Let M be the number of candidate segments, d the representation dimension, and B_{sel} the selection budget. By caching and incrementally updating each candidate's distance to its nearest selected center, the selection stage has time complexity $\mathcal{O}(B_{\text{sel}}Md)$ and space complexity $\mathcal{O}(Md + M)$.

B Supplementary experimental details

This appendix is organized around three questions. B.1 describes what is evaluated, including datasets, evaluation protocols, and spatial generalization settings. B.2 summarizes how policies and baselines are configured, including policy training, PROSE implementation, and baseline migration rules. B.5 explains how to interpret the supplementary results, including ablations, model-preference analysis, real-world robot evaluation details, and reproducibility scope. Related appendix sections provide supporting details: the influence backend and complexity analysis are in A.1 and A.2, while the targeted data acquisition use of PROSE diagnostic signals is in C.1 and C.2.

Additional coverage-baseline and feature-sensitivity experiments are detailed in Sections B.3 and B.4.

B.1 Experimental protocols and evaluation scope

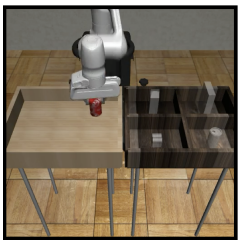
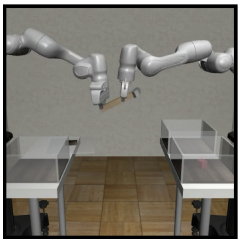
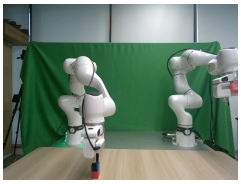
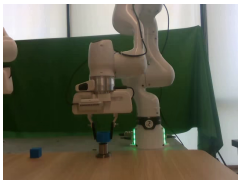
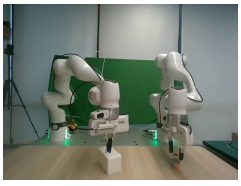
Robomimic simulation tasks. The simulation experiments are mainly conducted on mixed-quality tasks from `robomimic` [11], including `can_mh`, `square_mh`, and `transport_mh`. These tasks involve long temporal dependencies and multi-stage contact processes. Their mixed-quality demonstrations expose globally low-quality episodes that should be removed before policy training, locally corrupted segments that may appear inside otherwise useful trajectories, and repeated high-scoring modes that can cause coverage collapse after selection. The takeaway is that the simulation suite tests whether PROSE can reason jointly about global episode value, local segment value, and retained-set diversity.

Real-world tasks. The real-world robot experiments cover `Stack_Cube`, `Insert_Ring`, and `Pack_Box`. Compared with simulation, these settings are more affected by perception noise, control latency, gripper contact errors, and workspace boundary effects, and therefore test whether the gains from data curation transfer to real deployment. Each of the three real-world robot datasets contains 200 demonstration trajectories.

- `Stack_Cube`**.** This standard stacking task has relatively stable camera and object states. The dataset still includes failed demonstrations, so it tests whether curation removes obvious low-value trajectories without discarding useful standard behavior.
- `Insert_Ring`**.** This fine insertion task is conducted under limited visibility. Occlusion between the bearing and the ring can prevent the external camera from initially seeing the ring, and part of the insertion region may lie outside the external camera's field of view. The task therefore stresses contact precision and recovery under partial observability.
- `Pack_Box`**.** This bimanual packing task contains mixed strategies. Some demonstrations start both arms simultaneously, while others require the right arm to grasp the lid first and the left arm to enter after the lid is stabilized. Both strategies can work under human teleoperation, but they induce different bimanual start timings and phase-transition conditions under similar observations.

The main failure mode in `Pack_Box` is policy averaging across incompatible strategies: the learned policy may start the left arm too early, enter the next phase before the right arm stably grasps the lid, or produce inconsistent bimanual coordination rhythm. The 0% `Full Data` success rate in the main text should therefore be read as evidence of mode conflict in mixed-strategy imitation data, not as evidence that all demonstrations are unusable. If classifier-dependent baselines such as Demo-SCORE cannot stably train an effective classifier, the main tables mark the corresponding result as `N/A`.

Table 6: Descriptions of the simulated and real-world robot task scenarios used in the experiments.

Task	Example Image	Task Description
`Can_mh`		The robot arm must grasp a can from a larger container and place it into the target container. This task mainly tests grasping, transport, and target-region placement. Mixed-quality demonstrations often contain local issues such as grasp offsets, redundant approach paths, or unstable placement.
`Square_mh`		The robot arm must grasp a square nut and insert it onto the target peg. Compared with `Can_mh`, this task requires more precise end-effector pose, insertion angle, and contact stability; small local deviations can lead to alignment failure.
`Transport_mh`		The robot arm must transfer an object between two work regions and place the target object at the specified location. This task includes longer-horizon cross-region motion, phase transitions, and contact constraints, exposing local stagnation, path backtracking, and insufficient cross-phase coordination.
`Stack_Cube`		The real robot must grasp a cube and stack it. The camera and target states are relatively stable in this setting, but the data include failed demonstrations and inefficient trajectories, making it useful for evaluating curation on a standard real-world manipulation task.
`Insert_Ring`		The real robot must complete fine insertion under limited visibility. Occlusion between the bearing and the ring affects the initial observation, and part of the insertion region may fall outside the external camera's field of view; the task therefore emphasizes contact precision and recovery under partial observability.
`Pack_Box`		The real bimanual robot must complete a packing task. The data contain both synchronous bimanual execution and a sequential strategy in which the right arm first grasps the lid and the left arm enters later. Multi-strategy timing differences under similar observations can cause policy averaging, making this a key setting for validating coverage-aware curation.

Train/validation split and rollout budget. All data-curation methods use the same original training set and the same validation-rollout budget. Offline demonstrations are used for policy training and data curation, while validation rollouts are used to construct the performance-aligned contribution signal and to support final evaluation. Baselines that depend on online rollouts receive the same rollout count as PROSE, so performance cannot improve merely because a method receives more test opportunities. This makes the selected subset, rather than the available evaluation budget, the main experimental variable.

Evaluation metric. All main-text tables report task success rate (%). Simulation experiments report the average success rate over three independent training runs by default. The main real-world robot table also reports success rates to keep the format consistent. If a baseline cannot produce valid evaluation results because its classifier cannot be trained stably, the result is marked as `N/A`; we do not replace the missing experiment with extrapolated numbers.

Spatial generalization protocol. To more comprehensively evaluate policy robustness to changes in initial state, we use a unified spatial sampling protocol for object initial positions at test time. The test points cover grid-center points in the workspace, boundary points that probe workspace limits, and additional randomly sampled points that reduce dependence on a fixed pose set. The takeaway is that evaluation is not concentrated on a small number of standard initial poses; it directly probes spatial distribution shifts after data curation.

B.2 Policy, PROSE, and baseline configurations

Table 7: Training-configuration summary for the two policy models.

Policy	Input / output form	Core training setup
ACT	Multi-view image observations + robot state, outputting action chunks	Fixed ACT backbone, optimizer, and checkpoint-selection rule; simulation uses batch size = 32 and training steps = 100k; real-world robot experiments use training steps = 300k
$\pi_{0.5}$	Visual observations + task-relevant state conditioning, outputting autoregressive action sequences / action blocks	All experiments use LoRA fine-tuning; simulation uses batch size = 32 and training steps = 10k; real-world robot experiments use batch size = 16 and training steps = 30k

Fair comparison principle. Across all comparisons, the policy backbone, input modalities, training budget, optimizer type, checkpoint-selection rule, validation-rollout budget, and final evaluation protocol are fixed. The only intended difference between methods is the curated subset used for training and the data-selection logic that produces it. Performance differences in the main results should therefore be interpreted as data-curation effects rather than backbone or training-script effects.

Hardware setting. All experiments are conducted on a single NVIDIA A100 GPU. The host CPU is an Intel(R) Xeon(R) Gold 6338N CPU @ 2.20GHz with x86_64 architecture, little-endian byte order, and 128 online logical CPUs. The CPU supports both 32-bit and 64-bit modes, with 46-bit physical and 57-bit virtual address sizes.

ACT setting. ACT experiments use a fixed ACT backbone [8]. On the same task, different data-curation methods share the same input modalities, action representation, and training budget. Simulation experiments use batch size 32 and train for 100k steps, while real-world robot experiments train for 300k steps. No selected subset receives a different backbone width, depth, or extra training steps. The takeaway is that ACT results measure the effect of the retained data composition, not retuning for smaller subsets.

$\pi_{0.5}$ setting. $\pi_{0.5}$ experiments also follow a unified backbone and evaluation protocol [9, 10]. Because $\pi_{0.5}$ action prediction has stronger long-horizon conditioning dependencies, we keep observation encoding, action horizon, inference parameters, and checkpoint-selection criterion fixed across methods. LoRA fine-tuning is used in all $\pi_{0.5}$ experiments. Simulation uses batch size 32 and 10k

training steps, while real-world robot experiments use batch size 16 and 30k training steps. The $\pi_{0.5}$ results therefore isolate the effect of curated subsets on closed-loop policy performance.

Training stability. All policy training uses the same number of random seeds and the same checkpoint-evaluation logic. If numerical instability, abnormal interruption, or an obviously failed run occurs during training, the run is restarted or removed according to a unified rule, and the same handling standard is applied to all methods in the final statistics. This rule is especially important for small-sample real-world robot evaluation, where a single abnormal run can substantially perturb the final mean.

Algorithm 1 PROSE: Coverage-aware robot demonstration curation

Require: Offline demonstration dataset $\mathcal{D}$, validation rollouts, target policy model π_θ, segment length L_s, stride h_s, coverage budget.
Output: Retained segment set $\mathcal{S}$ and selection diagnostic statistics.
1: Stage 1: Episode–segment unit construction.
2: Represent each trajectory τ_i as an episode-level unit and split it into segment-level units $\{w_{i,k}\}_{k=1}^{K_i}$.
3: Stage 2: Sample value estimation.
4: Estimate $\widetilde{I}(\tau_i)$ and $\widetilde{I}(w_{i,k})$ to characterize the marginal contributions of trajectories and segments to validation return.
5: Compute trajectory quality scores q_i and segment quality scores $q_{i,k}$, and fuse them into sample-value scores s_i and $s_{i,k}$.
6: Stage 3: Coverage-aware selection.
7: Greedily select retained segments using $s_{i,k}$ and coverage gain to obtain $\mathcal{S}$.
8: Train the policy with $\mathcal{S}$ and record selection statistics for distribution-gap analysis.

PROSE implementation details.

Stage 1: episode–segment construction. For each trajectory τ_i, we keep both the complete episode representation and a fixed-length segment representation. The construction is controlled by segment length and sliding stride. The hyperparameter ablation in the main text denotes the segment length as w_s and compares $w_s \in \{25, 50, 75\}$; the main results use $w_s = 50$. The stride h_s is fixed within the same task and policy model, ensuring that different baselines and PROSE process the same candidate training units. This stage preserves global task semantics through episodes and local correction ability through segments.

Stage 2: sample value estimation. PROSE's episode-level and segment-level scores both come from two complementary signals: *influence-based contribution* and *kinematic quality prior*. The contribution score measures the marginal contribution of a training unit to closed-loop validation return, while the quality prior calibrates the instability of pure attribution signals under noise, approximation error, and insufficient validation coverage. In implementation, the quality prior is not a hard gate. It is fused with the contribution score through weighted fusion, reducing the risk of incorrectly removing marginal but still valuable samples. The fused sample-value score is used to determine whether a candidate episode or segment should enter the curated subset.

Quality components. The trajectory-level quality score consists of three components: **terminal-state consistency and path progress**, which judge whether a trajectory converges toward task completion; **local motion smoothness**, which identifies high-frequency jitter and control discontinuities; and **temporal-structure consistency**, which detects abnormal backtracking, stagnation, and overly long trailing behavior. The local segment quality score retains only smoothness terms directly related to short-horizon segment stability, avoiding unreliable terminal-state judgments on short segments.

Stage 3: coverage-aware selection. Stage 3 further adds coverage gain on top of sample-value ranking. The purpose of this step is not to encourage retaining outlier noise, but to prevent the selected result from collapsing into a few repeated high-scoring modes. In implementation, PROSE first removes the bottom ρ_e fraction of low-value trajectories at the episode level and then performs segment-level coverage-aware greedy selection inside the retained trajectories. The hyperparameter ablation in the main text compares $\rho_e \in \{0.05, 0.10, 0.15, 0.20\}$, and the main results use $\rho_e = 0.10$.

Key weights and thresholds include λ_1, λ_2, λ_3, γ_1, γ_2, δ_{abs}, and δ_{rel}. They are fixed within the same task and policy model and kept unchanged outside the ablation experiments. The takeaway is that coverage is applied after low-value filtering, so the method preserves diversity among plausible training units rather than retaining arbitrary outliers.

Shared defaults and tuning. The main experiments use shared defaults of $\eta_1 = \eta_2 = 0.5$, $\rho_e = 0.10$, $(\lambda_1, \lambda_2, \lambda_3) = (0.4, 0.4, 0.2)$, and $L_s = w_s = 50$, without separately searching these parameters for each dataset. The ACT/`Can_mh` ablation evaluates $\rho_e \in \{0.05, 0.10, 0.15, 0.20\}$ and $w_s \in \{25, 50, 75\}$, yielding success rates of 79/82/71/75% and 71/82/74%, respectively. In practice, one can first normalize the score components and use the shared default weights, then adjust them based on selection diagnostics; the segment length should match the temporal scale of the actions or manipulation phases.

Influence estimation in practice. PROSE does not depend on a single influence approximation route. ACT uses a unified low-dimensional projection to build reusable training-sample gradient representations and reduce the cost of large-scale retrieval. $\pi_{0.5}$ uses blockwise compression and local second-order approximation to preserve block-level structural information at an acceptable cost. The full backend and complexity analysis are provided in A.1 and A.2.

Baseline migration details. The migrated baselines are organized by the curation signal they represent:

- `Full Data`**.** The policy is trained on the complete training set without filtering, providing the reference point for measuring whether curation improves over using all demonstrations.
- **CUPID.** CUPID [14] migrates performance-aligned attribution into the current policy-training pipeline and uses a unified rollout budget to estimate trajectory-level influence scores. The main text reports the best stable CUPID configuration for each task, and this baseline isolates attribution-based data selection because it lacks PROSE's joint episode–segment selection and coverage-aware selection.
- **SCIZOR.** SCIZOR [15] preserves the original idea of local segment filtering by identifying low-value local units and filtering at the segment level. We do not provide SCIZOR with a stronger episode-level prior or additional rollouts, so it isolates pure segment-level curation.
- **Demo-SCORE.** Demo-SCORE [17] relies on rollout-driven trajectory-level scoring and uses the same online evaluation budget, training-set split, and final policy-evaluation protocol as PROSE, CUPID, and the other methods.

The takeaway is that each baseline removes one part of PROSE's design: multi-granularity modeling, coverage-aware selection, or the joint use of contribution and quality.

Why PROSE is not just a stronger baseline ensemble. PROSE should not be understood as a simple combination of multiple baseline modules. Its selection process has three coupled decisions: sample-value estimation asks whether a candidate training unit is worth retaining in itself, joint episode–segment modeling preserves both global task semantics and local correction ability, and coverage-aware selection controls the distributional shape after selection. PROSE's performance improvement therefore comes from the interaction between sample value and coverage constraints within one selection process, not from simply adding more scoring signals.

B.3 Coverage-baseline comparison

Using the same trajectory and segment representations, we compare PROSE with a k-center baseline that incorporates influence and quality scores; all other settings are identical. Across five policy–task settings (Table 8), PROSE improves mean success from 60.6% to 71.4%, a gain of 10.8 percentage points. Even with sample scores, the general-purpose coverage strategy falls short of the full method, supporting the joint consideration of sample value and retained-subset composition during selection.

Table 8: Success rates (%): k-center vs. PROSE.

Model	Task	k-center	**PROSE**
ACT	Can_mh	81	**82**
	Square_mh	36	**55**
	Transport_mh	51	**72**
$\pi_{0.5}$	Can_mh	85	**93**
	Square_mh	50	**55**

B.4 Coverage representation sensitivity

Beyond the component ablation in Table 2, we assess the choice of coverage representation by varying only the input features of ϕ for ACT on `Can_mh`, with all other settings fixed. Table 9 reports success rates of 82%, 76%, and 74% for State, State + Image, and Image, respectively. The low-dimensional state representation performs best, indicating sensitivity to the input features. Equation 14 defines the default segment representation.

Table 9: Coverage feature ablation.

ϕ input	Success (%)
State	**82**
State + Image	76
Image	74

B.5 Supplementary interpretation and reproducibility

Component ablations. The component ablations in the main text support three takeaways. The gap between `influence_only_two_stage` and `quality_only_two_stage` shows that neither pure attribution nor pure quality is sufficient as a data-value criterion on its own. The results of `episode_only` and `segment_only` show that global semantics and local correction provide complementary signals. The degradation of `w/o_coverage` further shows that retaining only high-sample-value data introduces coverage collapse and ultimately harms closed-loop generalization.

Policy preference analysis. The purpose of the model-preference analysis is not to prove that one policy is intrinsically better, but to show that different policy families produce stable yet not fully identical data-value judgments over the same demonstrations. Thus, RQ3 in the main text should be understood as evidence for target-policy-oriented data curation, rather than as an absolute comparison between ACT and $\pi_{0.5}$.

Real-world evaluation details.

Stack-cube protocol. In the `Stack_Cube` real-world robot experiment, all methods share the same camera setup, robot-arm control frequency, gripper-control interface, test script, and predefined workspace for sampling initial object poses. Each test records the result according to a unified success criterion. Test samples are not subjectively filtered between methods, avoiding manual selection of favorable scenes.

Success criterion. Real-world success is determined by the task terminal state: the manipulated object must reach the target state of the corresponding scene within the step limit, without manual correction or additional intervention at the end. A trial is recorded as a failure if it contains obvious slipping, dropping, incorrect grasping, assembly failure, incorrect placement, or failure to reach the target state within the time limit. If finer-grained statistics are needed in later work, three submetrics can be added according to task type: grasp success, insert/place success, and final stabilization.

Reproducibility notes.

Implementation checklist. To facilitate reproducing PROSE, the final open-source release should at least provide data-splitting and segment-construction scripts, influence-score computation scripts

and cache formats, implementations of sample-value scoring and coverage-aware selection, training configuration files for ACT and $\pi_{0.5}$, and simulation and real-world robot evaluation scripts.

Reproducibility scope. These materials cover six stages: data splitting, candidate training-unit construction, sample-value estimation, coverage-aware selection, policy training, and final closed-loop evaluation. In our experiments, different data-curation methods share the training budget, validation-rollout budget, and final evaluation protocol. The performance differences in the main tables mainly come from the composition of the curated subset rather than additional training resources or evaluation opportunities.

C Guided data acquisition from PROSE diagnostics

This appendix explains how PROSE selection statistics can be converted into guidance signals for the next round of targeted data acquisition. The method section in the main text focuses on data curation itself, namely sample-value estimation, joint episode–segment selection, and coverage-aware selection. The acquisition guidance described here is not an additional module of the main algorithm, but a downstream diagnostic use of the selection results. The practical takeaway is that the same statistics used to select a subset can also reveal where the next demonstrations should be collected.

C.1 Diagnostic signals and collection use

PROSE guidance for targeted data acquisition is naturally derived from the sample-value scores and coverage statistics produced by Stage 2 and Stage 3. The diagnostic pipeline first maps removed, low-coverage, or low-scoring episodes and segments onto task-relevant aggregate units R. Each aggregate unit is then described by removal concentration, coverage deficiency, average contribution, average quality score, and local motion instability. These metrics are monotonically aggregated into a replenishment priority:

$$P(R) = G(\mathrm{drop}(R),\, 1-\mathrm{cov}(R),\, 1-\bar{s}(R),\, 1-\bar{q}(R),\, 1-\mathrm{stab}(R))\,. \tag{18}$$

Here $G(\cdot)$ is a monotone aggregation function. A larger $P(R)$ indicates that the aggregate unit should receive higher priority in the next data-collection round.

How to interpret $P(R)$. $P(R)$ is a region-level acquisition priority rather than a sample-level copying instruction. High-priority units usually indicate weak state–action regions, unstable trajectory patterns, or long-tail but task-critical recovery actions. Low-priority units usually correspond to regions that are already well covered or heavily retained by the current subset. Saturated regions with many similar retained samples should receive less collection budget unless they correspond to a known deployment risk. The goal is not to increase data scale itself, but to focus limited demonstration budget on the distributional gaps exposed by the curated subset.

How to use the priority in collection. In real-world robot data collection, one can sort aggregate units by $P(R)$ to generate a replenishment list. Each high-priority aggregate unit is then translated into concrete collection instructions, including workspace regions, initial object poses or robot configurations, and trajectory-characteristic requirements such as recovery behavior, contact stabilization, or phase-transition timing. This turns one-round offline data curation into a longer-cycle data-iteration process: curate the current dataset, diagnose the remaining gaps, collect targeted demonstrations, and repeat.

C.2 Preference-guided acquisition experiment

In the model-preference analysis in the main text, ACT and $\pi_{0.5}$ do not fully agree on which samples are high value within the same demonstrations. Here we further test whether this preference difference can guide the next round of demonstration acquisition. Specifically, using $\pi_{0.5}$ as the target policy, we use `ph` demonstrations as an offline proxy for newly collected demonstrations and compare three training-data construction strategies.

- **Curated MH** uses only the PROSE-curated `mh` data.
- **Random PH addition** randomly adds 50 `ph` demonstrations to the `mh` data.

- **Guided PH addition** directionally adds 50 `ph` demonstrations according to PROSE-diagnosed model–task preferences, prioritizing demonstrations that represent regions where the current policy is weak or where data are insufficient.

This experiment asks whether the distribution of newly added data matches the regions most needed by the target model.

Table 4 in the main text shows that randomly adding high-quality `ph` demonstrations does not consistently outperform the original `mh` training set. For Can, random `ph` addition improves success from 93% to 95%. On Square and Transport, however, random addition reduces performance to 52% and 42%, respectively. Guided PH addition achieves the best result on all three tasks, improving average success from 64% with Curated MH to 71% and outperforming random addition by about 8 points. On Square and Transport, it improves over random addition by about 12 and 10 points, respectively. The takeaway is that model–task preferences exposed by PROSE can guide limited additional collection budget toward more effective data regions, rather than merely increasing the number of demonstrations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main problem setting, contributions, experimental scope, and empirical claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a discussion of limitations, including assumptions, computational cost, and the scope of settings where the approach is most suitable.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present new theoretical results requiring formal theorem statements or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper and supplemental material describe the experimental protocols, data construction, training and evaluation settings, and implementation details needed to reproduce the reported main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The supplemental material provides access instructions and reproduction materials for the data, code, configurations, and scripts used for the main experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper and supplemental material specify the datasets, splits, training budgets, evaluation protocols, hyperparameters, and implementation details used in the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports results over repeated runs and describes the evaluation protocol used to summarize experimental performance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The supplemental material reports the compute hardware and discusses the computational requirements and scaling properties of the experimental pipeline.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses benchmark and robot-learning experiments and is conducted in accordance with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper focuses on a research method and experimental evaluation, without introducing a deployed user-facing system or direct application involving personal data.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not release high-risk generative models, scraped web datasets, or other assets requiring special responsible-release safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing datasets, models, and code assets used in the work are credited in the paper or supplemental material, with licensing and usage information provided alongside the released materials where applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Newly released code and data are documented in the supplemental material and are accompanied by instructions for use and reproduction.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing experiments or human-subject studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The work does not involve human-subject research requiring institutional review board approval or equivalent review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development and experimental pipeline do not use large language models as an important, original, or non-standard research component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.