RETURNING SKILL DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised skill discovery (USD) learns a repertoire of distinct behaviors without external reward, and across every major approach the learned skills are one-way: a skill leaves its starting state and stops wherever it ends, because nothing in the objective rewards coming back. We first show this is structural and then introduce Returning Skill Discovery (RSD), a method-agnostic add-on that makes skills return without changing the base objective. RSD adds three pieces: a phase input, without which a skill that retraces its path is provably inexpressible; a reward weight that flips sign at mid-episode, under which the optimal skill travels out for half the episode and home for the other half; and a dimensionless return constraint that catches everything the weighted reward cannot see. The weight is the same for every method, and a checkable property of the reward decides only what it multiplies. Paired with five USD methods (DIAYN, DADS, LSD, CSD, METRA) on locomotion and manipulation, with state and pixel inputs, RSD produces skills that return to their starting position and posture, chain zero-shot, and outperform one-way libraries as long-horizon options. Videos of returning skills are at `https://sites.google.com/view/rsd-usd`.

1 INTRODUCTION

The behavior of a reinforcement learning (RL) agent is driven by its reward function (Sutton et al., 1998; Vincent et al., 2018), and reward functions are notoriously hard to design (Ma et al., 2024). Unsupervised skill discovery (USD) (Eysenbach et al., 2018; Park et al., 2024) sidesteps this difficulty by learning a repertoire of diverse, distinguishable skills without any external reward; the learned skills can then be composed as temporally extended options (Sutton et al., 1999) to solve downstream tasks.

Yet across every major USD variant, the learned skills share a limitation that is easy to miss and expensive in practice: they are one-way. A skill departs from the initial state and simply stops wherever it happens to end. This matters most where resetting the agent is difficult or costly (Sharma et al., 2021b; Yang et al., 2024): a repertoire of one-way skills leaves the agent stranded after every execution, whereas skills that return to their starting state reset themselves, leaving the agent ready to execute the next skill from the same home state and making skill chaining (Konidaris & Barto, 2009) trivial by construction.

We show this one-wayness is structural rather than incidental: neither of the two frameworks that most USD methods fall into contains any objective that asks for return. Mutual-information methods such as DIAYN (Eysenbach et al., 2018) maximize the mutual information $I(S; Z)$ between visited states and the skill; the return is a functional of the visited states and is indifferent to whether the trajectory comes home. Distance-maximizing methods such as METRA (Park et al., 2024) reward displacement along a skill direction z in a learned latent space ϕ, and here the situation is worse than indifference: the return telescopes to the difference of endpoints, $(\phi(s_T) - \phi(s_0))^\top z$, so a trajectory that returns to its start earns exactly zero, and the return leg refunds, unit for unit, everything the outbound leg earned. In both frameworks the agent learns to travel as far as possible or to strike diverse postures, and in neither is coming back ever optimal.

We propose Returning Skill Discovery (RSD), a method-agnostic addition that plugs into existing USD methods, leaves the base objective untouched, and makes their skills return. RSD adds three pieces: a phase input, without which out-and-back behavior is provably inexpressible for a skill retracing its path and untimeable for any skill at all; a reward weight that flips sign at mid-episode, so

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

that on the inbound leg both weight and displacement are negative and returning is paid rather than refunded, with the optimal skill turning exactly at mid-episode; and a dimensionless return constraint, enforced by an annealed learned multiplier, which supplies returning pressure on everything the weighted reward cannot see. The weight is identical in every case; the only method-dependent choice is which increment it multiplies, the method's own reward when its return telescopes and an added observation-space excursion otherwise. This is a property one checks rather than designs, so a single recipe covers both frameworks.

Our contributions are three-fold:

- **Why skills never return.** We give a structural diagnosis: distance-maximizing returns telescope, so a closed trajectory earns exactly zero; mutual-information returns are functionals of the visited states and are indifferent to returning; and a skill that retraces its path is not even representable by the stationary policy class these methods share.
- **Returning Skill Discovery.** We introduce RSD, a method-agnostic add-on that makes skills return, consisting of a phase input, a sign-flipping reward reweighting, and an annealed return constraint. We characterize its optimum: the ideal skill turns at mid-episode and reaches half the excursion of the base objective, the price of the trip home.
- **Validation across five methods.** We pair RSD with five USD methods spanning both frameworks (DIAYN and DADS; LSD, CSD, and METRA), on state- and pixel-based observations, across locomotion and manipulation tasks. RSD yields returning skills throughout while retaining the base methods' diversity, and the returning skills chain zero-shot and pay off as long-horizon options.

2 PRELIMINARIES

We consider USD in a controlled Markov process $(\mathcal{S}, \mathcal{A}, f, \rho_0)$ with deterministic transition map $s_{t+1} = f(s_t, a_t)$ and initial distribution ρ_0. A skill is a latent code $z \in \mathcal{Z}$ drawn from a fixed prior $p(z)$; a skill-conditioned policy $\pi(a \mid s, z)$ is rolled out for a fixed horizon T. A base method supplies a per-step reward $r^{\text{base}}(s_t, s_{t+1}, z)$ together with any auxiliary learned components it requires: a representation ϕ, a discriminator q, or a skill-dynamics model $p(s' \mid s, z)$. We use $\tau_t \in [0, 1)$ for a normalized within-episode phase, defined in Equation (5).

USD methods largely fall into two frameworks, mutual-information based and distance-maximizing (Section 3 reviews both). We instantiate RSD on five representatives, DIAYN and DADS from the former and LSD, CSD, and METRA from the latter, and detail their objectives here. The choice is one of exposition: Section 4 shows that the only method-dependent choice RSD makes is determined by a single checkable property of the reward, so other members of either framework can be wrapped in the same way.

Mutual-information objectives. The first framework maximizes the statistical dependence between the skill and the trajectory it produces. DIAYN maximizes the marginal mutual information $I(S; Z)$ between visited states and the skill, so that skills are identifiable from the states they reach. Its full objective also carries a policy-entropy term inherited from maximum-entropy RL, which our setup leaves untouched; optimizing a variational lower bound on the mutual information with a learned discriminator $q(z \mid s)$ yields the per-step reward

$$r^{\text{base}}_{\text{DIAYN}}(s_t, s_{t+1}, z) \;=\; \log q(z \mid s_{t+1}) \;-\; \log p(z). \tag{1}$$

DADS (Sharma et al., 2019) is also mutual-information based, but maximizes the conditional mutual information $I(S'; Z \mid S)$: given the current state, the skill should predict the next state. It optimizes a tractable estimate of this quantity through a learned skill-dynamics model $p(s' \mid s, z)$, with per-step reward

$$r^{\text{base}}_{\text{DADS}}(s_t, s_{t+1}, z) \;=\; \log p(s_{t+1} \mid s_t, z) \;-\; \log \tfrac{1}{L} \textstyle\sum_{i=1}^{L} p(s_{t+1} \mid s_t, z_i), \tag{2}$$

where the second term is a sample estimate of the marginal transition probability $\mathbb{E}_{z' \sim p(z)}\, p(s_{t+1} \mid s_t, z')$ over L skills drawn from the prior, so the reward estimates the integrand of the conditional mutual information rather than bounding it. The two objectives differ in what must be identifiable: for DIAYN the states themselves, for DADS the transitions between them.

Distance-maximizing objectives. The second framework rewards displacement along the skill direction in a learned latent space. We say a reward admits a potential (Ng et al., 1999) if there exists a scalar function $F_z : \mathcal{S} \to \mathbb{R}$ with

$$r^{\text{base}}(s_t, s_{t+1}, z) \;=\; F_z(s_{t+1}) - F_z(s_t), \qquad \text{and we write} \qquad f_t \coloneqq F_z(s_t), \quad \Delta f_t \coloneqq f_{t+1} - f_t. \tag{3}$$

The distance-maximizing framework is exactly of this form with $F_z(s) = \phi(s)^\top z$; its canonical instance is

$$J(\pi, \phi) = \mathbb{E}\Big[\textstyle\sum_{t=0}^{T-1} \big(\phi(s_{t+1}) - \phi(s_t)\big)^\top z\Big] \quad \text{s.t.} \quad \|\phi(s) - \phi(s')\| \le 1 \;\; \forall (s, s') \text{ adjacent}, \tag{4}$$

where adjacency means s' is reachable from s in one transition. The framework's members share this objective and differ only in which metric bounds a latent step. LSD requires ϕ to be Lipschitz with respect to the state-space metric, $\|\phi(s) - \phi(s')\| \le \|s - s'\|$ for all state pairs, so displacement is measured in how far the state moves. CSD keeps the Lipschitz form but replaces the Euclidean metric with a controllability-aware distance derived from a learned transition density, so that rarely achievable transitions count as long and skills are steered toward harder-to-achieve state changes. METRA imposes the temporal-distance constraint of Equation (4), bounding the latent step per environment transition, so displacement is measured in how long a state change takes to reach.

3 Related Work

Unsupervised skill discovery. USD learns diverse, distinguishable skills without external reward (Laskin et al., 2021), and existing methods fall into the two frameworks of Section 2. Among **mutual-information objectives**, VIC (Gregor et al., 2016) maximizes the mutual information between the skill and the trajectory's final state; DIAYN (Eysenbach et al., 2018) extends identifiability to every visited state through a learned discriminator; and DADS (Sharma et al., 2019) makes transitions rather than states identifiable through a learned skill-dynamics model, yielding dynamics predictable enough for planning. Among **distance-maximizing objectives**, LSD (Park et al., 2022) rewards latent displacement under a Lipschitz constraint with respect to the state metric; CSD (Park et al., 2023) replaces that metric with a controllability-aware distance, steering skills toward hard-to-achieve state changes; and METRA (Park et al., 2024) bounds the latent step per transition, spreading skills over the most temporally distant reachable states and scaling to pixels. Recent members of this framework shape which skills are discovered rather than how far they reach: LGSD (Rho et al., 2025) learns semantically distinct skills through language descriptions of states, DoDont (Kim et al., 2024) learns skills matching the desirable behaviors shown in instruction videos, FoG (Yang et al., 2025) learns skills that foundation models judge desirable, and PSD (Park et al., 2026) learns periodic skills such as locomotion gaits. Each is a layer on METRA, LSD, or CSD that selects which behaviors are wanted while leaving the displacement objective in place, and all remain one-way. No objective in either framework rewards returning to the start, and Section 4 shows that distance-maximizing objectives actively refund it. We therefore pair RSD with the base objectives themselves, the simpler and more widely used starting point, rather than with these guided variants; RSD leaves the base objective untouched, and what it changes is determined by whether that objective's reward admits a potential in the sense of Equation (3).

Learning to return. A separate line of work teaches RL agents to reset themselves so that training can proceed without expensive external resets. LNT (Eysenbach et al., 2017) trains a dedicated reset policy alongside the task policy and uses its value function to abort unsafe actions; VaPRL (Sharma et al., 2021a) and MEDAL (Sharma et al., 2022) train the reset policy to return to states drawn from demonstrations; IBC (Kim et al., 2023) removes the need for demonstrations with a curriculum over both task and reset policies; R3L (Zhu et al., 2020) instead resets to diverse initial states, yielding task policies robust to where they start; MoReFree (Yang et al., 2024) trains a goal-conditioned reset policy in imagination with a world model; and LiSP (Lu et al., 2020) plans in a learned skill space to avoid irreversible states. All operate in the standard RL setting, with an external task reward and with returning delegated to a second, separately trained policy. Our setting differs on both counts: RSD is unsupervised, and return is a property of each skill itself, the same policy that travels out also comes home.

4 From One-Way to Returning Skills

4.1 Why skills never return

One might hope that skills fail to return because nothing asks them to, and that a small bonus for ending near the start would fix it. For the distance-maximizing framework the situation is worse: returning is not neglected but maximally punished.

Proposition 1 (Telescoping). *If r^{base} admits a potential in the sense of Equation* (3)*, then for any trajectory $\sum_{t=0}^{T-1} r_t^{\text{base}} = f_T - f_0$: the undiscounted return depends on the trajectory only through its endpoints. In particular, every closed trajectory ($s_T = s_0$) attains return exactly* 0*, and is strictly suboptimal whenever some reachable trajectory attains $f_T - f_0 > 0$.*

Proof. The sum telescopes, $\sum_t (f_{t+1} - f_t) = f_T - f_0$, and $s_T = s_0$ gives $f_T = f_0$. □

In words: the return leg refunds, step for step, everything the outbound leg earned. The argument covers the entire distance-maximizing framework, since Equation (3) is independent of how the constraint on ϕ is imposed. Mutual-information returns Equations (1) and (2) do not telescope and are merely indifferent to return. One framework punishes return and the other ignores it, and which case a reward falls in is checkable by inspection: this single bit is the only thing RSD ever asks of a base method.

4.2 Returning Skill Discovery

RSD adds three pieces to any base method: a clock (L1), a sign-flipping reward weight (L2), and a return constraint (L3). The clock makes out-and-back possible to express, the weight makes it the best thing to do, and the constraint catches everything the weight cannot see. Formal statements and proofs are in Appendix A; extended discussion and implementation details are in Appendix B.

4.2.1 L1: phase lift

A returning skill stands, halfway home, in a state it visited on the way out, and must now do the opposite of what it did then. A stationary policy $\pi(a \mid s, z)$ receives the same input at both visits and must repeat itself, so a skill that retraces its path is inexpressible regardless of the reward (Proposition 2); and even when the observation distinguishes the two passes, nothing in the state says when to turn so that the trajectory closes at $t = T$, and randomizing the turn cannot schedule it (Appendix A). We therefore give the policy a clock, $\pi(a \mid s, z, \tau)$, through the circular embedding

$$\tau_t = \frac{t + \frac{1}{2}}{T}, \qquad e(\tau) = \big(\cos 2\pi\tau,\ \sin 2\pi\tau\big) \in \mathbb{R}^2. \tag{5}$$

Its two components expose exactly what a returning skill must know, the distance to the turning point and the current leg; the half-step offset keeps the weight of L2 away from its sign changes; and since an extra input only enlarges the policy class, base methods that never needed a clock are unaffected. Time-awareness is known to matter under fixed horizons (Pardo et al., 2018); here it is what makes the return leg expressible at all.

4.2.2 L2: reweighting

The clock makes out-and-back expressible, but by Proposition 1 a telescoping return still charges for the trip home, so an agent equipped with a phase will simply ignore it: the objective itself must change its mind at mid-episode. For telescoping rewards we replace the base return with

$$J_{\text{RSD}}(\pi) = \mathbb{E}\Big[\textstyle\sum_{t=0}^{T-1} w(\tau_t)\, \Delta f_t\Big], \qquad w(\tau) = \sin 2\pi\tau, \tag{6}$$

subject to the base method's constraint on ϕ. The sign does all the work: outbound, weight and displacement are both positive; inbound, both are negative, and minus times minus is plus, so the trajectory that used to refund its reward is now paid on both legs. Appendix A derives the weight as the first Fourier harmonic of the single-excursion profile and characterizes the optimum (Proposition 3): the ideal skill travels outward for exactly half the episode and home for the other half, reaching

half the excursion of the base objective, because half the step budget buys the trip home; and up to $O(1/T)$ the objective equals the negated first cosine coefficient of the excursion profile, a trajectory shape where the base return is an endpoint. Because the sign flip offers no signal before the agent can travel at all, the weight is ramped in,

$$w_k(\tau) = (1 - \alpha_k) + \alpha_k \, \sin 2\pi\tau, \qquad \alpha_k = \min\big(1,\ k/K_w\big), \tag{7}$$

starting at the unmodulated base return and reaching Equation (6) at $\alpha = 1$ (evidence that the ramp is necessary is in Appendix B).

Rewards that do not telescope. When the base return has no potential, we supply one without touching the base method: the observation-space excursion $f_t^{\text{obs}} = \|s_t - s_0\|$ telescopes by construction, and the same weight plays the same role through the additive term

$$r_t^{\text{geom}} = \rho \; w_k(\tau_t) \; \widehat{\Delta f_t^{\text{obs}}}, \qquad \widehat{\Delta f_t^{\text{obs}}} = \frac{f_{t+1}^{\text{obs}} - f_t^{\text{obs}}}{\frac{1}{T}\sum_s \left| f_{s+1}^{\text{obs}} - f_s^{\text{obs}} \right| + \varepsilon}, \tag{8}$$

normalized by the episode's own mean step so that $\rho = 1$ transfers across environments. At $\alpha = 0$ this is a pure "get away from home" reward, exactly the exploration signal mutual-information objectives never supply, and the ramp then bends the travel into an out-and-back; the episode-level constraint alone cannot do this, because it says that the skill failed to return but never when it should have turned (Appendix B).

4.2.3 L3: RETURN CONSTRAINT

L2 weights one scalar, and a skill can serve the scalar while its state drifts: a potential is blind to everything it discards, and the excursion contains no z. Neither certifies that the world is restored, so we constrain that directly, on the observations themselves:

$$R(\xi) \;=\; \frac{\|s_T - s_0\|}{\max_{0 \le t \le T} \|s_t - s_0\| + \varepsilon}, \qquad \mathbb{E}_{z,\xi\sim\pi}\big[R(\xi)\big] \;\le\; \epsilon. \tag{9}$$

R asks how close the skill ended to home, relative to how far it went: $R(\xi) \in [0, 1)$, $\epsilon = 1$ is vacuous, and $\epsilon = 0$ demands exact return. It is dimensionless, presumes no structure on the state space, and is computed from the policy's own observations, which is what lets it wrap any base method, pixels included, unchanged. We enforce it with a learned multiplier $\lambda = \exp(\ell)$, clipped at $\lambda_{\max}$, following Tessler et al. (2018):

$$\max_{\pi} \min_{\lambda \ge 0} \; J_{\text{RSD}}(\pi) \;-\; \lambda \left(\mathbb{E}[R] - \epsilon\right), \qquad \mathcal{L}_\ell = \ell \cdot \big(\epsilon - \mathbb{E}[R]\big), \tag{10}$$

so that ℓ rises while the constraint is violated and decays once it is slack (three further enforcement details are in Appendix B).

4.3 AVOIDING THE DEGENERATE OPTIMUM

The constraint has a trivial feasible point, the policy that never leaves s_0, and early training makes it an attractor: before locomotion exists the constraint is violated at every step, so λ ratchets upward until the penalty dominates, and a collapsed behavior distribution takes the base method's learned components down with it (Appendix B). We therefore anneal the constraint from vacuous to tight,

$$\epsilon_k \;=\; 1 + \big(\epsilon^\star - 1\big) \min\big(1,\ k/K\big), \tag{11}$$

initializing λ at λ_0 and capping it at $\lambda_{\max}$ (Table 3): skills form first, and are asked to return second.

4.4 INSTANTIATION ACROSS BASE METHODS

RSD is a wrapper: the phase lift and the constraint are shared verbatim by every instantiation, and the single method-dependent choice is which increment the weight multiplies, determined by whether the base return telescopes (Table 2 in Appendix B). For distance-maximizing methods (METRA, LSD, CSD) the return telescopes, so the weight reshapes the method's own reward in place, Equation (6); the base constraint on ϕ is untouched, and its per-transition clipping is what supplies the box constraint of Proposition 3. For mutual-information methods (DIAYN, DADS) the base reward is left exactly as it is, since $\sin 2\pi\tau$ on Equation (1) would pay the agent to become unidentifiable on the way home; the weight instead acts on the excursion through the additive term Equation (8). DADS additionally phase-conditions its dynamics model to $p(s' \mid s, z, \tau)$, because an out-and-back skill traverses the same state with opposite successors and makes $p(s' \mid s, z)$ bimodal.

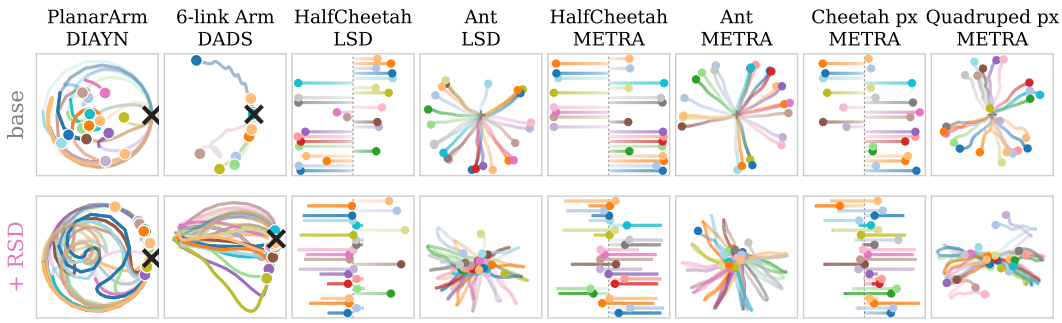


Figure 2: Skill trajectories, base (top) versus RSD (bottom), one column per objective and body. Endpoint dots mark final states: at the rim for one-way skills, at the start for returning ones. Arm columns trace the mechanism's end point; faint segments are outbound, solid segments the return leg.

5 Experiments

Setup. Base USD methods are not interchangeable across domains, so we pair each with the tasks its authors demonstrated it on (Figure 1): DIAYN on PlanarArm and DADS on the 6-link Arm; LSD on state-based HalfCheetah and Ant; CSD on the manipulation tasks FetchPush (Gupta et al., 2019) and Kitchen (Fu et al., 2020); and METRA on state-based (HalfCheetah, Ant) and pixel-based (Cheetah, Quadruped) locomotion (Tunyasuvunakool et al., 2020; Towers et al., 2026). All five run in one codebase with the authors' settings, which RSD leaves unchanged (Appendix C). Videos of learned skills can be found on our anonymous project webpage [1].

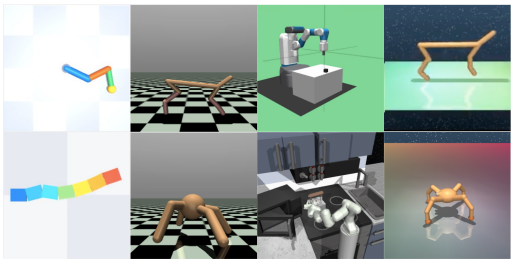


Figure 1: Six state-based (first three columns) and two pixel-based tasks.

5.1 Does RSD make skills return, across objectives and modalities?

Figure 2 shows discovered skills for four base objectives across state and pixel inputs, arms and locomotion. Locomotion bodies are traced in the plane (planar runners in per-skill lanes); arms are traced by their end point. The shared start is marked $\times$, the outbound leg faint and the return leg solid. Every column shows the same pattern: base skills are one-way, sweeping the arm to a pose and holding it or radiating outward to the rim, whereas RSD skills trace comparable excursions and come back, their endpoint dots collapsing onto the start.

Task	Objective	Turn back	Return	State Cov.
PlanarArm	DIAYN	0.77±0.01	0.53±0.10	n/a
6-link Arm	DADS	0.89±0.01	0.88±0.00	n/a
HalfCheetah	LSD	0.67±0.05	0.62±0.07	27±1%
Ant	LSD	0.50±0.09	0.20±0.04	21±5%
FetchPush	CSD	0.55±0.02	0.17±0.05	n/a
HalfCheetah	METRA	0.89±0.03	0.86±0.10	29±1%
Ant	METRA	0.84±0.02	0.71±0.08	15±2%
Cheetah px	METRA	0.61±0.05	0.36±0.01	32±1%
Quadruped px	METRA	0.48±0.03	0.13±0.07	43±8%

Table 1: Returning under RSD in the space of Figure 2. Turn back: mean fraction of the excursion retraced. Return: fraction of skills ending within 20% of their peak distance. State Cov.: unique planar grid squares visited by RSD skills, as a percentage of the base method's on the same seed.

Measuring returning. Under every base objective essentially no skill returns at any seed, so Table 1 reports RSD alone. RSD turns skills around under all four objectives and both input modalities, seed-stably on the state-based tasks (Table 7). Ant with LSD is the hardest

[1] https://sites.google.com/view/rsd-usd

state-based task; it is also the only task on which imposing the constraint from the first epoch fails at every multiplier cap we tried (Figure 6), and the one whose return gait is slowest relative to its outbound gait (Table 8): the difficulty of the task and the difficulty of coming home are the same difficulty. On pixels the effect persists with no change to the encoder or objective, but reliability splits by body: Cheetah returns consistently across seeds, whereas Quadruped finds returning in one seed of three, the honest limit of the current recipe.

Object manipulation. Manipulation asks a different question: reverting means restoring the object, and undoing a push is pushing the cube back from the opposite side, not retracing a path. Testing this with CSD exposes something locomotion cannot: the constraint changes not only how skills end but which behaviors are found at all. On FetchPush, every CSD skill drives the cube off the table (Figure 3(a)), an absorbing state no policy can undo; with RSD no cube leaves the table, because a cube on the floor can never satisfy the constraint. Irreversible regions become unattractive before the question of returning from them arises (Grinsztajn et al., 2021). On Kitchen, RSD is what makes objects get manipulated at all: nearly every skill engages the object, against a minority or none under CSD, whose skills mostly wave the arm in free space (Figure 3(b)). Whether those skills also revert depends on the mechanism as much as the method, so a freeze-at-half control rolls the policy to $\tau = 0.5$ and then applies zero actions: on the light switch and microwave door the reversal is the arm's doing, since freezing mostly fails to reproduce it, whereas the two cabinet doors swing shut unaided. The kettle, the only free body, is an honest negative: skills move it, none brings it back, and doubling the horizon does not change this, so reacquiring a released object is a skill RSD does not learn here (Appendix C.6).

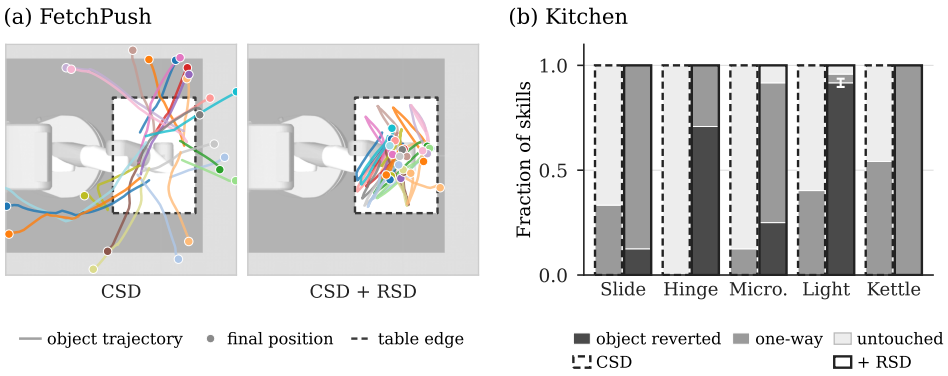


Figure 3: **Reverting object state.** (a) FetchPush: object trajectories for 24 skills on a top-down view, one line per skill with a dot at its final position and the tabletop dashed. CSD drives every cube off the table; with RSD every cube stays on the table and skills push it out and back. (b) Kitchen: one CSD-oracle pair per object, dashed outline CSD and solid outline + RSD. Each of the 24 skills is classified by whether it moved the object and, if so, whether it brought it back (return index < 0.5). A freeze control (text) attributes the reversion to the arm on the light switch and microwave, and to the mechanism on the two cabinets.

What returning costs. Reversion is meaningless alone, since a skill that never leaves trivially ends at home, so every reversion number appears beside excursion and coverage. On the arms RSD reaches farther than the base (end-point excursion $114 \pm 4\%$ of DIAYN's on PlanarArm, $195 \pm 19\%$ of DADS's on the 6-link Arm): the phase lift adds structure to objectives that otherwise settle into small held poses. On the distance-maximizing bodies the exchange runs the other way, RSD retaining 15 to 43% of the base's coverage (Table 1). Two controls in Section 5.3 rule out the objection that RSD buys returning by moving less: freezing the phase halves the excursion while returning nothing, and removing the constraint leaves skills traveling as far as the base while returning nothing.

Why a quarter of the coverage, rather than the half a symmetric out-and-back would suggest? Table 8 gives a two-factor answer: RSD skills turn at the midpoint, so the outbound leg has half the time budget, and they travel outward at roughly half the base's speed because the return leg binds (running backwards is slow for HalfCheetah; Ant must turn around). Half the time at half the speed gives a quarter of the reach. Directional diversity survives on the one-dimensional HalfCheetah and falls

somewhat on the planar Ant. The exchange is set by the reverse gait rather than the objective, which is also where it can improve: a return gait as fast as the outbound one would approach the 50% ceiling that Proposition 3 predicts.

5.2 Do returning skills help downstream?

Standard goal reaching rewards how far skills travel; we present two settings that probe how skills compose, which is where returning pays. Both use the six locomotion libraries of Figure 2 unchanged (Appendix C.7).

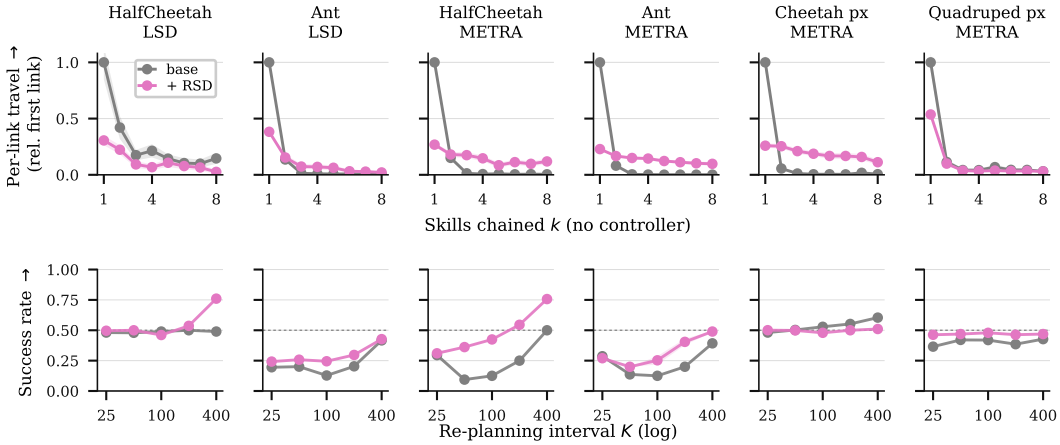


Figure 4: **Returning skills compose and pay off downstream.** Top: eight random skills executed back to back with no controller; travel of the k-th skill relative to the base's first invocation (mean$\pm$ste over 32 chains). Bottom: final success rate of a SAC meta-controller that re-selects a frozen skill every K steps, with goals alternating between a far target and the start. A library that cannot return caps at 0.5 (dashed). The ordering holds on the state-based bodies and pixel Quadruped, and reverses on pixel Cheetah.

Zero-shot chaining. Eight random skills are executed back to back with no learning or planning downstream. Every skill was trained from the initial state distribution, so the travel of the k-th skill measures whether the state its predecessor left is one it can start from. The result is the starkest in the paper (Figure 4, top): under every base objective the library is dead after one hand-off, later skills barely moving and the chain drifting far from home, whereas returning skills hand off in distribution and the chain ends essentially where it began. On the two locomotion tasks where RSD returns least, later links degrade for both libraries: a skill composes to the extent that it returns.

Goal reaching with fetch goals. The second setting keeps the standard hierarchy, a SAC controller picking a frozen skill every K steps, but alternates the goal between a far target and the origin, as in fetch-and-carry. Return legs then carry half the reward, so a library that can only depart is capped at 0.5, where the HalfCheetah LSD baseline sits for every K. What K controls is who supplies the return: a controller can synthesize one from one-way skills by chaining an outbound skill with one pointing back, which is cheap when K is short, but as K grows the return must live inside a single option, which RSD trains and one-way libraries lack. The state-based bodies follow this prediction (Figure 4, bottom): the libraries are indistinguishable at the shortest K, the gap opens as K grows, and at the longest K the base recovers only to its outbound level while RSD stays ahead on every state-based task and, on HalfCheetah, passes the 0.5 cap that no one-way state-based library reaches. The gain is largest where skills serve as long, temporally extended options, the regime skill discovery is intended for. The pixel tasks qualify this: Quadruped keeps the ordering at every K but not the widening gap, and Cheetah reverses it, the base eventually passing 0.5. The cap assumes a one-way skill cannot serve a return goal, and here it can, since the goal range is small relative to how far a long pixel-Cheetah skill travels. We therefore report the K trend as a state-based result.

5.3 Ablation

Figure 5 removes every subset of the three components on two tasks with different base objectives. The result is a cliff rather than a gradient: full RSD returns most of its skills, and all seven removals return at most one skill in 48, the base method's own rate. No pair substitutes for the third component, and none carries the method alone. The excursion markers rule out the obvious objection: the ablated arms travel as far as the full method or farther and simply never come home, so reduced excursion is a consequence of returning, not a substitute for it.

The three single-component removals fail in different ways. Without the reweighting, the agent has a clock and a multiplier pinned at its ceiling, yet travels almost as far as the base: it absorbs the penalty and ignores the constraint, so the reweighting is what converts constraint pressure into a turn. Without the phase, the reward is weighted correctly in time but the policy cannot tell where in the episode it is, and so cannot act on a signal that changes sign at the midpoint; this is Proposition 2 appearing in training. Without the constraint, the phase and weight shape a shorter trajectory but nothing anchors its endpoint: shaping the profile is not requiring the return. A third body class agrees: on the Kitchen light switch, the object RSD reverts most reliably, removing the constraint drops reversion to 0.00 for the rest of training.



Figure 5: **All three components are necessary; no pair substitutes for the third.** Left: components each arm keeps (filled = present). Right: fraction of skills that return, beside how far they travel relative to that body's own base run. Only the arm keeping all three returns anything; every removal returns at most one skill in 48 while still traveling roughly as far as the base, so they fail by not coming back rather than by not moving.

6 Conclusion and Limitations

In RL, skills learned with unsupervised skill discovery (USD) objectives are one-way for structural reasons: distance-maximizing returns telescope, so a closed trajectory earns exactly zero; mutual-information returns are indifferent to where a skill ends and vanish near home; and neither family's stationary policies can express a retracing skill. We proposed Returning Skill Discovery (RSD), which adds three components to a base method without changing its objective: a phase input, making out-and-back representable; a sign-flipping reward weight, making it optimal; and a dimensionless return constraint on the observations the weighted reward cannot see. Only the weight consults the base method, and only to choose which increment it multiplies. Paired with five USD methods across locomotion and manipulation with both state and pixel inputs, RSD raises the fraction of returning skills from zero to between 0.13 and 0.88, and those skills chain zero-shot and beat one-way libraries as long-horizon options.

Although RSD works well in these tasks, there are still two main limitations remain. Returning costs reach: half the budget buys the trip home, and a slower return gait costs more still, leaving roughly a quarter of the base coverage on locomotion at the tightest setting, though the target can be loosened when the downstream use does not compose skills. And asking for return is not obtaining it: the constraint expresses what we want without the agent always finding it, as the free kettle and pixel Quadruped show. Making returning emerge as reliably as traveling is the natural next step. We hope RSD encourages USD to ask not only how far a skill can go, but whether it can come back.

AI USE STATEMENT

Generative AI was used in two ways. First, for code. Part of the implementation, the analysis scripts, the table and figure generators were written with AI assistance. Second, for polishing the writing. Drafts were polished with AI assistance and edited by the authors. We have reviewed all AI-assisted work and take responsibility for the final content, including text and claims.

ETHICS STATEMENT

This work involves no human subjects and no personal data.

REPRODUCIBILITY STATEMENT

All environments, base-method configurations, and RSD hyperparameters are listed in Appendix C (Tables 3 to 5); the evaluation protocol, per-seed values, and downstream protocols are in Appendices C.4 and C.7; and all proofs are in Appendix A. The codebase can be found in the supplementary materials.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. Pmlr, 2017.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Matthieu Geist, et al. There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. *Advances in Neural Information Processing Systems*, 34:1898–1911, 2021.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Hyunseung Kim, Byungkun Lee, Hojoon Lee, Dongyoon Hwang, Donghu Kim, and Jaegul Choo. Do's and don'ts: Learning desirable skills with instruction videos. *Advances in Neural Information Processing Systems*, 37:47741–47766, 2024.

Jigang Kim, Daesol Cho, and H Jin Kim. free autonomous reinforcement learning via implicit and bidirectional curriculum. In *International Conference on Machine Learning*, pp. 16441–16457. PMLR, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22, 2009.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, et al. Eureka: Human-level reward design via coding large language models. In *International conference on learning Representations*, volume 2024, pp. 26516–26560, 2024.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*, pp. 4045–4054. PMLR, 2018.

Jonghae Park, Daesol Cho, Jusuk Lee, Dongseok Shim, Inkyu Jang, and H Jin Kim. Periodic skill discovery. *Advances in Neural Information Processing Systems*, 38:105404–105438, 2026.

Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2022.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. *arXiv preprint arXiv:2302.05103*, 2023.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. In *International Conference on Learning Representations*, volume 2024, pp. 18579–18603, 2024.

Seungeun Rho, Laura Smith, Tianyu Li, Sergey Levine, Xue Bin Peng, and Sehoon Ha. Language guided skill discovery. In *International Conference on Learning Representations*, volume 2025, pp. 87731–87752, 2025.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021a.

Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. *arXiv preprint arXiv:2112.09605*, 2021b.

Archit Sharma, Rehaan Ahmad, and Chelsea Finn. A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*, 2022.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. *Advances in Neural Information Processing Systems*, 38, 2026.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Francois-Lavet Vincent, Henderson Peter, Islam Riashat, G Bellemare Marc, and Pineau Joelle. An introduction to deep reinforcement learning. *Foundations and trends® in machine learning*, 11 (3-4):219–354, 2018.

Zhao Yang, Thomas M Moerland, Mike Preuss, Aske Plaat, and Edward S Hu. Reset-free reinforcement learning with world models. *arXiv preprint arXiv:2408.09807*, 2024.

Zhao Yang, Thomas M Moerland, Mike Preuss, Aske Plaat, Vincent François-Lavet, and Edward S Hu. Guiding skill discovery with foundation models. *arXiv preprint arXiv:2510.23167*, 2025.

Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

Appendix

Table of Contents

A Formal Statements, Proofs, and Derivations **13**
- A.1 Why the phase is necessary . . . 13
- A.2 Deriving the weight and the optimal skill . . . 14
- A.3 The Fourier view . . . 14

B Extended Discussion and Implementation Details **14**
- B.1 Getting training off the ground . . . 15
- B.2 Why the constraint needs help . . . 16
- B.3 Enforcement details . . . 16
- B.4 Implementation . . . 17

C Experimental details **18**
- C.1 Environments . . . 18
- C.2 Baseline implementations . . . 19
- C.3 Training setup and compute . . . 19
- C.4 Evaluation protocol . . . 20
- C.5 Where the coverage goes . . . 21
- C.6 Kitchen, per object . . . 21
- C.7 Downstream protocols . . . 22

A Formal Statements, Proofs, and Derivations

This appendix collects the formal counterparts of the claims made in Section 4: the representability obstruction behind the phase input (L1), and the derivation and characterization of the reweighting (L2). The constraint (L3) needs no separate treatment, since its properties are proved inline in the main text.

A.1 Why the phase is necessary

The main text claims that a skill retracing its path cannot be expressed by a stationary policy. The formal statement is the following.

Proposition 2 (Out-and-back is not representable)**.** *Let the transition map be deterministic and let ξ be a trajectory that visits some state s at two times $t_1 \neq t_2$ with distinct continuations, $s_{t_1+1} \neq s_{t_2+1}$. Then no deterministic stationary policy $\pi(s, z)$ generates ξ.*

Proof. Determinism gives $s_{t+1} = f(s_t, a_t)$, so distinct continuations force $a_{t_1} \neq a_{t_2}$, while a stationary policy outputs $a_{t_1} = \pi(s, z) = a_{t_2}$. □

One might hope that stochasticity rescues the stationary class. It does not. In a deterministic CMP, the next-state distribution induced by a stationary policy at a state s is the pushforward of $\pi(\cdot \mid s, z)$ through $f(s, \cdot)$, and this distribution is the same at every visit to s. A stochastic stationary policy can therefore randomize between the outbound and inbound continuations at a revisited state, but it cannot make the choice depend on which visit it is: it can randomize the turn, not schedule it. A reliable out-and-back remains outside the class, and the phase input supplies exactly the missing bit of information.

A.2 Deriving the weight and the optimal skill

The weight in Equation (6) follows from the profile we want the optimal skill to trace. That profile is a single excursion: symmetric about the midpoint, $f(\tau) = f(1-\tau)$, with $f(0) = 0$. Evenness about $\tau = \frac{1}{2}$ excludes the $\sin 2\pi k\tau$ terms of a Fourier expansion, and $f(0) = 0$ fixes the constant against the $\cos 2\pi\tau$ term; truncating to the first harmonic gives $f(\tau) = \frac{A}{2}(1 - \cos 2\pi\tau)$, and since the per-step reward weights $\Delta f_t \approx f'(\tau)\,\Delta\tau$, the weight must be proportional to $f'(\tau) = \pi A \sin 2\pi\tau$. Choosing $\cos 2\pi\tau$ instead would target $f \propto \sin 2\pi\tau$, which is odd about the midpoint: a symmetric oscillation through s_0 rather than one excursion with one turning point.

With the weight fixed, the reweighted objective admits a complete characterization.

Proposition 3 (Optimal skill)**.** *Let T be even and suppose the base method's constraint implies $|\Delta f_t| \le v$ for all t, as the per-transition clipping of Equation* (4) *does. Then, over increment sequences satisfying the box constraint, Equation* (6) *is uniquely maximized by $\Delta f_t^\star = v \operatorname{sign} w(\tau_t)$, and the maximizer (a) moves outward for $t < T/2$ and inward thereafter, (b) attains zero net displacement $f_T = f_0$, (c) has a unique turning point at $t = T/2$ with peak excursion $vT/2$, and (d) is time-reversal symmetric, $f_t = f_{T-t}$.*

Proof. The objective is linear in the increments, which interact only through the box constraint, so the maximizer saturates it with the sign of the weight; for even T no τ_t hits a zero of w, so it is unique. With $\tau_t = (t + \frac{1}{2})/T$, $\sin 2\pi\tau_t > 0$ iff $t \le T/2 - 1$, giving (a) with exactly $T/2$ positive and $T/2$ negative signs, hence (b); the partial sums rise linearly to $vT/2$ at $t = T/2$ and descend symmetrically, giving (c) and (d). □

The proposition also clarifies what the derivation does and does not pin down: under the box constraint, any weight sharing the sign pattern of $\sin 2\pi\tau$ attains the same bang-bang optimum, so what optimality determines is the sign pattern alone. The first harmonic is the canonical smooth representative of that pattern, and it is the one the Fourier view below singles out.

A.3 The Fourier view

The change of objective from the base return to Equation (6) can be stated exactly. Abel summation rewrites $\sum_t w(\tau_t)\Delta f_t$ as boundary terms minus $\sum_t f_t\,(w_t - w_{t-1})$, and with the half-step offset of Equation (5) the difference of weights has the closed form $w_t - w_{t-1} = 2\sin(\pi/T)\cos(2\pi t/T)$; the offset is what makes this identity exact. The boundary terms are $w_{T-1}f_T - w_0 f_0$ with $w_0 = -w_{T-1} = \sin(\pi/T) = O(1/T)$. Up to these $O(1/T)$ terms, J_{RSD} is therefore proportional to the negated fundamental cosine Fourier coefficient of the profile f; a single positive excursion has a negative cosine coefficient, so the objective rewards it. The unweighted return, by contrast, is the profile's endpoint functional: RSD optimizes a trajectory shape rather than an endpoint. The same calculation is a recipe: a base return that computes some other functional of the profile would call for a weight derived by the same transport.

Finally, a caveat about the geometric term. Nothing clips the observation-space increments of Equation (8), so Proposition 3 does not literally govern that term: without a box constraint the linear objective has no interior bang-bang optimum. The term shapes the trajectory, paying the outbound leg and the way home in the same sign pattern, rather than pinning an optimum; the constraint Equation (9) supplies the actual guarantee of return.

B Extended Discussion and Implementation Details

This appendix explains the design decisions that the main text states without argument: the two schedules that let training get off the ground, the help the episode-level constraint needs, the details of its enforcement, and the plumbing that carries the phase and the constraint through a standard off-policy pipeline. Table 2 summarizes the instantiation across base methods.

Base method	Per-step reward r^{base}	Reweighted increment	Scorer change
METRA / LSD / CSD	$(\phi(s') - \phi(s))^\top z$	$\Delta(\phi^\top z)$	none
DIAYN	$\log q(z \mid s') - \log p(z)$	$\Delta\|s - s_0\|$	none
DADS	$\log p(s' \mid s, z) - \log \frac{1}{L} \sum_i p(s' \mid s, z_i)$	$\Delta\|s - s_0\|$	$p(s' \mid s, z, \tau)$

Table 2: RSD across base objectives. The phase lift Equation (5) and the constraint Equation (9) are identical in all cases; the weight $w = \sin 2\pi\tau$ is also the same, and only the increment it multiplies differs. When the base return telescopes, the weight reshapes the base reward in place, Equation (6); when it does not, the weight acts on the observation-space excursion as an additive term, Equation (8), leaving the base reward untouched.

B.1 Getting training off the ground

Both schedules exist because RSD's pressures are self-defeating when applied to an agent that cannot yet move. The modulated weight changes sign at the midpoint, so before the agent can travel at all, the rewards of the outbound and inbound halves cancel and the objective points nowhere: the ramp Equation (7) exists for discovery, not optimization. Ant with LSD never locomotes under the unramped weight, reaching a peak excursion of 1.9 against 58.7 for the unmodified base method, whereas HalfCheetah, which can travel by falling forward, is unaffected. Proposition 3 characterizes the $\alpha = 1$ objective that training ultimately reaches, and K_w is listed per run in Table 3.

The anneal exists for the same reason at the level of the constraint, where the failure mode is sharper because two pathologies compound. A Lagrangian self-limits only against a feasible constraint, and early in training the agent cannot yet locomote, so Equation (9) is violated at every step and ℓ ratchets monotonically toward its ceiling. Worse, the base method's learned components (ϕ, q, or the dynamics model) are trained on the policy's own rollouts, so an early collapse of the behavior distribution collapses them too, and this is not recovered when λ later decays. The anneal Equation (11) removes both: where it is active, $\epsilon_0 = 1$ is satisfied by construction, λ stays near zero during the first phase, and skills form before they are required to return. Several cells use the anneal (Table 3); Ant with LSD is the one cell that fails outright without it.

Figure 6 shows this directly on that cell, comparing two Ant/LSD runs that differ in a single flag. It also shows something we did not anticipate and report rather than smooth over: the multiplier saturates at $\lambda_{\max}$ in both runs and does not come back down, so what the anneal buys is not a gentler equilibrium but roughly 2500 additional epochs of unconstrained exploration before the penalty switches on.

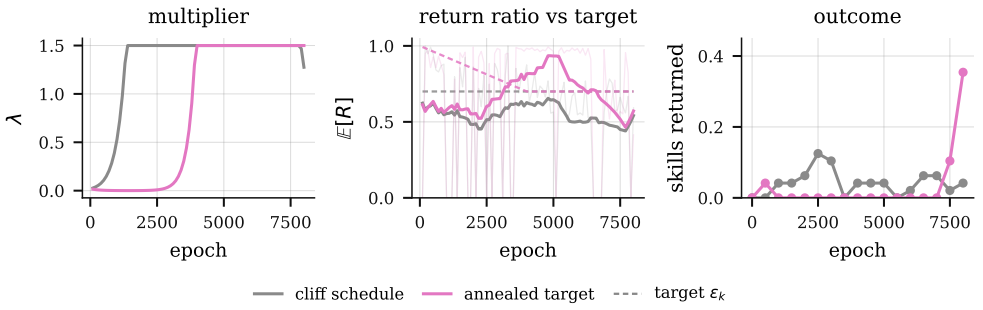


Figure 6: **The constraint has to be annealed in, and the multiplier is closer to a switch than to a dial.** Two Ant/LSD runs differing in one flag: both target $\epsilon^\star = 0.7$ with cap $\lambda_{\max} = 1.5$, one imposing the target from the first epoch and one annealing it over 4000 epochs, Equation (11). Left: under the cliff schedule the constraint is infeasible from the start and the multiplier reaches its ceiling within 1500 epochs; under the anneal it stays near zero until the target has tightened. It saturates in both cases. Middle: achieved ratio (rolling mean, raw values faint) against the target each run is held to (dashed). Right: the consequence for returning. Committing the multiplier before the agent can locomote does not buy returning; it removes the exploration that returning is built from.

B.2 WHY THE CONSTRAINT NEEDS HELP

The constraint is a single scalar per episode, and that creates two signal problems. The first is temporal: R carries no credit assignment, so the agent learns that it failed to return but never when it should have turned around. In our early runs without the geometric term Equation (8), λ saturated at its ceiling while the return ratio did not move. The additive term supplies both missing pieces, the exploration signal (at $\alpha = 0$ it purely rewards getting away from home) and the timing (the sign flip says when to come back). The mutual-information family remains the sharper test of the wrapper, since a constraint-satisfying skill ends where every other skill ends and the excursion itself must carry all the discriminative signal; but the test is of L1 and L3 together with a method-agnostic L2, not of L1 and L3 alone.

The second problem is one of reach. Even the pressure the constraint does exert is anchored at the episode's end, and a terminal-only penalty would reach the episode start only through T steps of γ-discounted bootstrapping, retaining γ^T of its magnitude. This is why line 4 of Algorithm 1 applies the penalty at every transition: R is an episode-level constant, so broadcasting it densely is exact rather than approximate, and it puts the full-strength signal at every point of the value backup (Achiam et al., 2017).

Algorithm 1 Returning Skill Discovery (one epoch, any base method)

1: sample $z \sim p(z)$; roll out $\pi(a \mid s, z, e(\tau_t))$ for T steps
2: compute $R(\xi)$ and, where Equation (8) is active, $\widehat{\Delta f_t^{\text{obs}}}$ from the policy's observations; store per transition
3: $r_t \leftarrow w_k(\tau_t) \cdot r^{\text{base}}(s_t, s_{t+1}, z)$ ▷ if the base return telescopes; else $r_t \leftarrow r_t^{\text{base}} + \rho\, w_k(\tau_t)\, \widehat{\Delta f_t^{\text{obs}}}$
4: $r_t \leftarrow r_t - \lambda\, \bar{r}\, R(\xi)$ ▷ episode-level penalty, applied densely; $\bar{r}$ tracks $\mathbb{E}|r^{\text{base}}|$
5: update the base method's learned components with its own losses on $\{r_t\}$
6: update π with SAC (Haarnoja et al., 2018) on $\{r_t\}$
7: $\ell \leftarrow \ell - \eta_\ell \left(\epsilon_k - \widehat{\mathbb{E}}[R]\right)$ ▷ descent on $\mathcal{L}_\ell$; clip $\lambda = e^\ell$ at $\lambda_{\max}$

B.3 ENFORCEMENT DETAILS

Three further choices govern how the penalty is computed. The first concerns scale. On the cells marked in Table 3, the penalty is scaled by the running magnitude of the base reward, $\lambda\, \bar{r}\, R$ with $\bar{r}$ an exponential moving average of $\mathbb{E}|r^{\text{base}}|$ (decay 0.995). The ratio R is dimensionless but the base return is not: its scale differs across methods and drifts during training, so without this scaling a single $\lambda_{\max}$ does not transfer, which would give back exactly the per-environment constant that motivated a dimensionless R in the first place.

The second concerns what R measures. On some environments, R is evaluated on a fixed slice of the observation rather than on all of it. Over a full MuJoCo observation, $\|s_t - s_0\|$ is dominated by whichever block varies most within an episode, and on a body whose translation is small that is the velocity block (14 of Ant's 29 dimensions); the constraint then measures speed, and asks the agent to accelerate and decelerate rather than to leave a configuration and return to it. Where used, the slice is the configuration block, given per environment in Table 3. This is the one place where RSD consults the layout of the observation, so we state its extent precisely: it is not used for the LSD and METRA locomotion results, which constrain the full observation, and it is unavailable under a pixel encoder, where the flattened frame is the only honest quantity and the full observation is used. It is used on the two mutual-information cells, where it also defines f^{obs} in Equation (8), and on the two manipulation environments.

The third is specific to Kitchen. There we replace the maximum in Equation (9) by twice the mean excursion, $R = \|s_T - s_0\| \,/\, \left(2\, \overline{\|s_t - s_0\|}\right)$, and treat an episode whose peak excursion never exceeds a small floor as neutral, reporting R equal to the current target ϵ_k so that it neither violates nor credits the constraint. Both changes close degeneracies specific to that environment: a knob driven past its stop snaps back when released, so a brief excursion earns $R \approx 0$ for free under the maximum, and a near-stationary episode scores tiny over tiny, which the dual reads as a satisfied constraint and so

exerts no pressure on the skills that do move objects. The mean normalization is the same statistic we use for evaluation, under which a one-way trip scores ≈ 1 and an out-and-back ≈ 0.

One property of our implementation deserves stating plainly, because it qualifies the description of Equation (10) as a self-regulating dual. We clamp $\lambda = \exp(\ell)$ at $\lambda_{\max}$ when the penalty is applied, but ℓ itself is unbounded unless a floor $\lambda_{\min}$ is set, which we do only on HalfCheetah with METRA. Where it is unbounded, ℓ integrates past $\log \lambda_{\max}$ while the constraint is violated and cannot return quickly once it is slack: on the annealed Ant/LSD run ℓ reaches 10.15, some nine nats above $\log 1.5$, so the effective multiplier stays pinned at its ceiling for the remainder of training even though the achieved ratio ends at 0.686 against a target of 0.773. In these runs the multiplier therefore behaves less like a dial that finds an equilibrium and more like a delayed switch, and the schedule that decides when it turns on matters more than the value it converges to. Clamping ℓ in log space rather than clamping its exponential is a one-line change that restores the intended behavior, and we recommend it for anyone building on this.

B.4 Implementation

The embedding $e(\tau_t)$ is appended to the skill code, giving the per-timestep input $[\, z, \cos 2\pi\tau_t, \sin 2\pi\tau_t \,]$ and adding two dimensions to the policy and critic; under a pixel encoder it is concatenated after the convolutional trunk, so no architectural change is needed. The base method's learned components (ϕ, q) are deliberately not given the phase, so they cannot read the clock; the single exception is the DADS dynamics model, where phase conditioning is itself the fix. $R(\xi)$ and the normalized increments $\widehat{\Delta f_t^{\mathrm{obs}}}$ are computed once per episode at collection time, from the policy's own observations and never from privileged simulator state, and stored as per-transition scalars, which keeps them well defined under i.i.d. replay sampling and avoids storing s_0 alongside every transition, prohibitive for image observations.

One consequence of reading the weight off the phase input is worth recording for anyone reproducing the ablations. Because $w(\tau_t)$ is computed from the phase channels the policy receives, freezing the policy's phase also flattens the weight to a constant, so the naive "remove L1" arm silently removes L2 as well. Isolating the two requires carrying the true τ alongside the frozen one and weighting the reward by the former, which is how the single-component removals in the ablation are produced.

Cell	Base	T	epochs	K_w	λ_0	$\lambda_{\max}$	$\bar{r}$ scale	$\epsilon^\star$	K	R computed on
PlanarArm	DIAYN	50	3000	300	1.00	10	no	0.20	none	obs[0:3]
6-link Arm	DADS	50	3001	300	1.00	10	no	0.20	none	obs[0:6]
HalfCheetah	LSD	200	8001	4000	0.02	1.5	yes	0.30	none	full obs
Ant	LSD	200	8001	4000	0.02	1.5	yes	0.70	4000	full obs
HalfCheetah	METRA	200	8000	0	0.02	3.0	no	0.20	1500	full obs
Ant	METRA	200	9000	4000	1.00	1.0	no	0.20	none	full obs
Cheetah px	METRA	200	5001	300	0.02	3.0	no	0.20	none	full frame
Quadruped px	METRA	200	5001	300	0.02	1.0	no	0.75	2000	full frame
FetchPush	CSD	50	8001	500	0.02	10	yes	0.50	4000	object, obs[3:6]
Kitchen	CSD	100	6001	500	0.02	30	yes	0.50	3000	object slice†

Table 3: RSD hyperparameters per reported cell. K_w is the modulation ramp Equation (7) in epochs (0 means the full weight from the start), λ_0 the initial multiplier, $\epsilon^\star$ the final constraint target and K its anneal horizon Equation (11) ("none" means the target is $\epsilon^\star$ throughout). The multiplier is learned in every cell, with Adam at 10^{-4} except HalfCheetah METRA at 5×10^{-4}, which also carries a floor $\lambda_{\min} = 0.01$. $\rho = 1$ wherever Equation (8) is active, which is the two mutual-information cells, and 0 elsewhere. $\gamma = 0.99$ everywhere. †Kitchen additionally uses the mean-normalized ratio with a floor of 0.05; the slice is the object block of the CSD oracle representation, chosen per object.

C Experimental details

C.1 Environments

All eight tasks are simulated in MuJoCo. Table 4 gives the interface each one presents to the agent. Observations are the raw vectors the policy receives; nothing in the observation is privileged, and ground-truth simulator state is used only for evaluation, exactly as METRA does.

Environment	Observation	dim $\mathcal{S}$	dim $\mathcal{A}$	T	dim z	Plotted coordinates
PlanarArm	state	6	3	50	16 (discrete)	fingertip x-y
6-link Arm	state	12	6	50	16 (discrete)	free-end x-y
HalfCheetah	state	18	6	200	2 (continuous)	root x-y
Ant	state	29	8	200	2 (continuous)	root x-y
FetchPush	state	25	4	50	2 (continuous)	cube x-y
Kitchen	state	30	7	100	2 (continuous)	object block
Cheetah px	pixels	$3\times64\times64$	6	200	2 (continuous)	root x-y
Quadruped px	pixels	$3\times64\times64$	12	200	4 (continuous)	root x-y

Table 4: Task interfaces. T is the episode horizon in agent steps. Plotted coordinates are used for figures and for the coverage metric only, never in a reward. Pixel observations are 64×64 RGB frames with a stack of 3.

PlanarArm. A three-link arm in a zero-gravity plane, actuated by three joint torques. The observation is the three joint angles followed by their velocities. Gravity is removed so that the reset pose is an equilibrium: on a body where the start state drifts under its own weight, s_0 is unreachable and the return question is unanswerable regardless of the policy. Posture is three numbers and the scene renders top-down with one color per link, so an out-and-back is visible in a single frame rather than only in a statistic.

6-link Arm. Seven rigid panels joined by six hinges in a chain, viewed from above, with the six crease angles leading the observation. Gravity is again zero and panel collisions are disabled, so the flat configuration is an equilibrium and stays reachable from any fold; allowing collisions would let tight folds trap the chain in configurations it cannot leave, which is a dynamics failure rather than a skill failure. This is an articulated approximation of a folding strip, not a deformable sheet.

HalfCheetah and Ant. The standard MuJoCo locomotion bodies as released with METRA, with all of `qpos` exposed, giving 18 and 29 dimensional observations of which 9 and 14 are velocities.

FetchPush. Gym Robotics `FetchPush-v1` with its goal removed and its time limit replaced by a 50 step horizon, following the setup CSD uses. The observation is the flat 25 dimensional vector (gripper pose, object pose and relative pose, rotations, and velocities); no goal is appended, since skill discovery has none. The tabletop is open, so a cube pushed past the edge falls to the floor and cannot be recovered, which is what makes the task a test of avoiding irreversibility rather than only of undoing motion.

Kitchen. The Franka kitchen used by CSD, in end-effector control mode (a 7 dimensional action: three translation deltas, three rotation deltas, and the gripper) with a 100 step horizon. The 30 dimensional observation is the arm configuration in dimensions 0 to 9 and the scene in 9 to 30: burners 9:17, light switch 17:19, slide cabinet 19:20, hinge cabinet 20:22, microwave 22:23, kettle 23:30. The per-object experiments give the CSD oracle one of these blocks at a time, and the constraint and the evaluation are restricted to the same block, so each run asks about exactly one mechanism.

Cheetah and Quadruped (pixels). DeepMind Control bodies rendered to 64×64 RGB and stacked three deep. The agent sees nothing else: the reward, the representation and the return constraint are all computed from the frames, and ground-truth state is used only to report x-y coverage after training.

C.2 BASELINE IMPLEMENTATIONS

All five base methods are run inside a single codebase, METRA's released implementation[2], so that differences between them are differences of objective rather than of infrastructure, and so that RSD is added to each in exactly one way. Every base method is trained with the authors' own settings for its objective, and RSD changes none of them. Table 5 lists the flags that select each method.

Method	`algo`	`inner`	`dual_reg`	`dual_dist`	`spec. norm`	`discrete`	Representation constrained by
DIAYN	`metra`	0	0	`one`	0	1	nothing (discriminator only)
DADS	`dads`	0	0	`one`	0	1	nothing (dynamics model)
LSD	`metra`	1	0	`one`	1	0	spectral normalization
METRA	`metra`	1	1	`one`	0	0	dual, one-step slack 10^{-3}
CSD	`metra`	1	1	`s2_from_s`	0	0	dual, slack 10^{-6}, λ_0=3000

Table 5: How each base method is instantiated. `inner` selects the inner-product reward $(\phi(s') - \phi(s))^\top z$ over the discriminator reward; `dual_reg` and `dual_dist` select the constraint imposed on ϕ; `discrete` selects a one-hot skill space. All other hyperparameters are shared (Appendix C.3).

DIAYN. The reward is $\log q(z \mid s')$ from a learned discriminator over 16 one-hot skills, with no constraint on the representation. In this codebase it is the `inner=0` branch: the same trajectory encoder network is trained as a classifier rather than as a metric-aware ϕ.

DADS. A learned skill-conditioned dynamics model $p(s' \mid s, z)$ gives the reward $\log p(s' \mid s, z) - \log \frac{1}{L} \sum_i p(s' \mid s, z_i)$ with $L = 100$ alternative skills sampled per transition, over 16 one-hot skills. The model is a Gaussian with constant standard deviation and batch normalization, as released (Ioffe & Szegedy, 2015). This is the one base method whose scorer RSD modifies, by conditioning the dynamics model on the phase (Section 4.4); every other component is unchanged.

LSD. The inner-product reward with the Lipschitz constraint on ϕ imposed by spectral normalization (Miyato et al., 2018) of the encoder rather than by a dual variable, and continuous skills sampled on the unit sphere.

METRA. The inner-product reward with the one-step distance constraint imposed by a learned dual multiplier (slack 10^{-3}), continuous skills on the unit sphere. This is the released configuration and is also what supplies the box constraint $|\Delta f_t| \leq v$ that Proposition 3 assumes.

CSD. METRA's machinery with the controllability-aware distance (`dual_dist=s2_from_s`, slack 10^{-6}, multiplier initialized at 3000), which is the configuration CSD uses for manipulation. On FetchPush and Kitchen we additionally restrict the oracle representation to the object block of the observation, which is CSD's own oracle setting, and the RSD constraint is restricted to the same block so that the two speak about the same object.

C.3 TRAINING SETUP AND COMPUTE

Every run uses SAC for the policy with entropy coefficient 0.01, target smoothing 5×10^{-3}, discount 0.99, and Adam (Kingma & Ba, 2014) at 10^{-4} for all components. Policies, critics and the trajectory encoder are two-layer MLPs of width 1024; the pixel runs prepend the convolutional encoder from METRA and concatenate the skill and phase after the trunk. Each epoch collects 8 trajectories and performs 50 gradient steps on minibatches of 256 transitions (100 steps for the pixel runs), from a replay buffer of 10^6 transitions (3×10^5 for pixels). Skills are resampled every episode.

Runs were executed on a single NVIDIA A100 (40 GB); no run uses more than one GPU, and rollout collection is single-process. The GPU is not the bottleneck: on Ant, 71% of wall-clock is environment stepping against 29% for gradient work (53% and 47% on HalfCheetah), so a 9000-epoch run is dominated by its 14.4M sequential MuJoCo steps rather than by its network updates. Table 6 gives

[2] `https://github.com/seohongpark/METRA`

wall-clock per run. A base and an RSD run of the same cell cost within 5% of each other on the state bodies, since RSD adds two input dimensions and one scalar per episode and nothing else. One clean replication of the main results, ten cells $\times$ two arms $\times$ three seeds, is approximately 460 GPU-hours. The total compute consumed by this project, including every exploratory, failed and abandoned run, is 2387 GPU-hours over 605 training runs.

Cell	Base method	Epochs	Env steps	Base (h)	+RSD (h)
PlanarArm	DIAYN	3000	1.2 M	1.1	1.1
6-link Arm	DADS	3001	1.2 M	1.5	1.5
HalfCheetah	LSD	8001	12.8 M	6.1	6.2
Ant	LSD	8001	12.8 M	7.4	7.6
FetchPush	CSD	8001	3.2 M	5.5	5.1
Kitchen	CSD	6001	4.8 M	10.0	10.2
HalfCheetah	METRA	8000	12.8 M	4.7	4.9
Ant	METRA	9000	14.4 M	7.4	7.3
Cheetah px	METRA	5001	8.0 M	13.6	18.9
Quadruped px	METRA	5001	8.0 M	14.7	18.4

Table 6: Training cost per seed, on one A100. Env steps is epochs $\times$ 8 trajectories $\times$ T.

C.4 Evaluation protocol

All reported numbers come from the final checkpoint, re-rolled rather than read off the training curve, with the policy acting at its mode so that the measured behavior is the skill rather than a sample around it. Each cell is evaluated with 24 skills, drawn from a fixed seed so that a base run and its RSD counterpart are asked about the same skill codes: sampled on the unit sphere for continuous skill spaces, uniformly among the 16 one-hot codes for discrete ones.

Turn back is the fraction of the peak excursion retraced by the end of the episode, and Return is the fraction of skills ending within 20% of their peak distance from the start, both computed in the plotted coordinates of Table 4. Skills whose peak excursion falls below 5% of the same seed's base median are excluded from both, since a skill that never leaves trivially satisfies a reversion criterion; the floor is computed per seed from that seed's own base run, so it never favors RSD.

For Kitchen, where "did the object come back" cannot be read off a planar path, skills are classified by the return index $\|s_T - s_0\| / \overline{\|s_t - s_0\|}$ computed on the object block, with each coordinate standardized by its own within-episode spread. The index is ≈ 0 for a return, ≈ 1 for a trajectory with no return structure, and ≈ 2 for one-way travel, and a skill counts as reverted below 0.5. We use the index rather than end-over-peak because end-over-peak scores a transient that snaps back as a perfect return: on the kettle it reports $24/24$ skills reverted where the index reports 0.

Table 7 gives the individual seed values behind the aggregate reversion numbers. We report the spread rather than only the summary because a standard error over three points is a weak description of two of the cells: pixel Quadruped returns on one seed of three and essentially not on the other two, and FetchPush has one weak seed among three.

Environment	Objective	seed 0	seed 1	seed 2	mean$\pm$ste	base
PlanarArm	DIAYN	0.29	0.58	0.71	0.53$\pm$0.10	0.00
6-link Arm	DADS	0.88	0.88	0.88	0.88$\pm$0.00	0.07$\pm$0.01
HalfCheetah	LSD	0.50	0.58	0.79	0.62$\pm$0.07	0.00
Ant	LSD	0.25	0.11	0.25	0.20$\pm$0.04	0.00
FetchPush	CSD	0.21	0.04	0.25	0.17$\pm$0.05	0.00
HalfCheetah	METRA	0.62	0.96	1.00	0.86$\pm$0.10	0.00
Ant	METRA	0.71	0.88	0.54	0.71$\pm$0.08	0.00
Cheetah px	METRA	0.33	0.38	0.38	0.36$\pm$0.01	0.00
Quadruped px	METRA	0.29	0.04	0.04	0.13$\pm$0.07	0.00

Table 7: Per-seed return fractions, the individual values behind Table 1. Every base run is 0.00 except the 6-link Arm with DADS, where one to two skills of 24 end near their start by chance.

C.5 WHERE THE COVERAGE GOES

Table 8 decomposes the coverage exchange of Section 5.1 on the four state-based locomotion cells.

	Turn at τ	**Out speed**	**Homing speed**	**Headings**	**Reach**
HalfCheetah / LSD	0.52±0.02	52±3%	33±2%	1.03±0.02	26±1%
Ant / LSD	0.50±0.02	62±8%	29±8%	0.74±0.08	29±3%
HalfCheetah / METRA	0.49±0.01	61±3%	54±2%	1.04±0.03	30±1%
Ant / METRA	0.48±0.01	49±1%	37±3%	0.87±0.09	23±1%

Table 8: Where the coverage goes (24 skills per cell). Turn at: fraction of the horizon at which skills reach peak distance (base skills peak at 0.99). Out speed: RSD outbound speed as a fraction of the base's. Homing speed: rate at which distance to the start shrinks, as a fraction of the base's forward speed. Headings: ratio of the effective number of distinct skill directions (RSD/base). Reach: RSD peak distance as a fraction of the base's.

C.6 KITCHEN, PER OBJECT

Table 9 gives the counts behind the Kitchen panel, and Table 10 the control that decides how much of the reversion is the arm's doing. The control matters because an end-state criterion cannot distinguish a skill that undoes its own work from a mechanism that resets itself once released, and two of the five objects are of the latter kind.

	CSD		+ RSD			
Object	engaged	reverted	engaged	reverted	median index	oracle slice
Slide cabinet	8/24	0	24/24	3	1.21	obs[19:20]
Hinge cabinet	0/24	0	24/24	17	0.01	obs[20:22]
Microwave door	3/24	0	22/24	6	2.18	obs[22:23]
Light switch	10/24	0	23/24	23	0.00	obs[17:19]
Kettle	13/24	0	24/24	0	0.88	obs[23:30]

Table 9: Kitchen, per object (seed 0; the light switch replicates at three seeds, 0.96 ± 0.02 reverted). Engaged means the object block moved by at least 0.1; reverted means return index < 0.5. The oracle slice is the block of the 30-dimensional observation given to CSD's representation, and the RSD constraint is restricted to the same block.

	+ RSD		CSD	
Object	reverted	self-resets	reverted	self-resets
Light switch	23/23	9/23	0/10	1/10
Hinge cabinet	17/24	15/24	0/0	0/0
Microwave door	6/22	0/9	0/3	0/3
Slide cabinet	3/24	13/13	0/8	0/8
Kettle	0/24	0/24	0/13	0/13

Table 10: Freeze-at-half control. Each skill is rolled to $\tau = 0.5$ and then given zero actions for the remainder of the episode, so the outbound half is unchanged and any remaining reversion is the mechanism resetting itself. Denominators differ because only skills that engaged the object are counted, and freezing changes how many do. The microwave is entirely arm-driven ($6/22 \to 0/9$) and the light switch mostly so ($23 \to 9$, median index $0.00 \to 1.11$); the two cabinet doors are largely the mechanism, the hinge being unaffected by freezing and the slide drifting shut whenever released; the kettle never self-resets, which is what makes its $0/24$ an honest negative rather than an artifact. Only the light switch and the microwave are claimed as reversion, and both are quoted with this control. The base arms self-reset essentially never, which is the control for the control: the artifact needs the policy to drive the object to the point where the mechanism takes over.

C.7 DOWNSTREAM PROTOCOLS

Zero-shot chaining. Eight skills are drawn at random and executed back to back for the full training horizon each, with no controller, no planner and no learning at the downstream stage, so nothing can correct the state between one skill and the next. We report the travel of the k-th skill and the drift of the chain from its start, averaged over 32 chains.

Goal reaching with fetch goals. A SAC meta-controller selects a frozen skill every K steps and holds it, with $K \in \{25, 50, 100, 200, 400\}$ and 800 environment steps per episode, so the controller makes $800/K$ decisions. Goals alternate between a target drawn at radius up to 10 (state bodies; 3 for pixel Cheetah and 4 for pixel Quadruped, matching the shorter reach of those skill sets) and the origin, four goals per episode, with the sparse reward of METRA's released downstream task and 200 steps allowed per goal. The controller is trained for 2000 epochs with the skill policy frozen. Because the maximum achievable return is the same at every K, success rates are comparable across the sweep. The goal set is the only modification to METRA's protocol: alternating the target with the origin is what makes half the achievable reward depend on returning, and is why a library that can only depart is capped at 0.5.