Extended Abstract

Motivation Robotic manipulation policies are typically brittle to changes in robot embodiment: when the robot's body or dynamics change, a flat policy that jointly learns task planning and motor control must be retrained, often at significant cost. Hierarchical reinforcement learning (HRL) offers an appealing response, since decomposing a policy into a high-level task planner and a low-level motor controller raises the possibility that the high-level plan could remain fixed across embodiments while only the low-level controller adapts. This motivated our original goal of using object-centric hierarchy to improve sample-efficient embodiment transfer. As the project developed, however, we found that a more basic question came first: whether hierarchy provides any benefit at all on the manipulation tasks we study. We therefore focus on the behavior of the hierarchical methods themselves rather than on extensive transfer evaluation.

Method We study HIRO-style hierarchy, in which a high-level policy proposes subgoals for a goal-conditioned low-level policy and an off-policy correction relabels past subgoals as the worker improves. We additionally study an object-centric variant, in which subgoals are defined as relative vectors between task-relevant entities such as gripper-to-object and object-to-goal offsets rather than as raw, embodiment-specific robot states, and a latent object-centric variant that compresses these subgoals into a lower-dimensional learned space. The intuition behind the object-centric design is that subgoals expressed as object relationships describe what the task requires independently of which robot performs it, and so should remain valid when the embodiment changes. We compare all of these against a flat Soft Actor-Critic (SAC) baseline.

Implementation We evaluate on the ManiSkill PickCube task, where a Franka Emika Panda arm must grasp a cube and move it to a target, using a shared SAC learner across all levels and methods. To study transfer, we define a modified arm that share the same observation and action spaces as the source but differ in gripper dynamics, allowing a trained policy to be carried over without changing its architecture. We measure success-once rate and evaluation reward over environment interactions, and for transfer we measure how quickly and how well a transferred policy adapts to a new goal or embodiment.

Results Flat SAC reliably solves PickCube, reaching near-perfect success and an evaluation reward of about 0.8 within roughly 200K environment interactions. The hierarchical methods consistently underperform it. The original HIRO formulation stays near zero success, and a cascade of reward and subgoal-space designs intended to close this gap each exposes a new exploitable failure mode, with our best variant, latent object-centric subgoals with reward shaping, still falling well short of the baseline. A controlled experiment makes the cause clear: when we anneal away the worker's access to the environment reward and force it to rely on the manager's subgoals alone, the policy never learns the task, whereas an otherwise identical run that keeps the environment reward solves it. Transfer improved early adaptation in both settings, but gains appeared to come mainly from policy reuse, with no clear hierarchical advantage over SAC.

Discussion These results indicate that the high-level policy is not the main driver of success on this task: success depends on the worker's direct access to the environment reward rather than on the subgoals the manager proposes. We attribute hierarchy's poor fit to three factors. The task is short-horizon and effectively single-stage, leaving little for a manager to decompose; the manipulation state space is largely non-Euclidean, making HIRO's positional subgoals a poor fit; and the reward landscape makes dense worker rewards easy to exploit while sparse rewards weaken learning. Our transfer comparisons point the same way, since the gains we observed appear to come from general policy reuse rather than from a uniquely reusable hierarchical abstraction.

Conclusion On short-horizon, dense-reward manipulation where a flat baseline is already strong, we find that hierarchical structure offers little benefit and that the high-level policy contributes little to task success. We do not read this as evidence against HRL in general, since this is precisely the regime in which hierarchy is least likely to help. Longer-horizon, sparser-reward tasks with genuine multi-stage structure, where a flat baseline is weaker, remain a future direction for revealing potential benefits of hierarchy that PickCube cannot, alongside the broader embodiment-transfer study that originally motivated this work.

Failing to Grasp the Point: Hierarchical Reinforcement Learning for Grasping Tasks

Ashwin Mahendran
Department of Computer Science
Stanford University
mashwin@stanford.edu

Anjali Sreenivas
Department of Computer Science
Stanford University
anjalisr@stanford.edu

Arianna Cao
Department of Computer Science
Stanford University
arianna.cao@stanford.edu

Abstract

Robotic manipulation policies are brittle to changes in robot embodiment: retraining from scratch when the robot's body or dynamics changes is costly, which motivates hierarchical reinforcement learning (HRL) as a potential solution. By decomposing a policy into a high-level task planner and a low-level motor controller, HRL raises the possibility that the high-level manager can remain fixed across embodiments while only the low-level controller is retrained. We test this hypothesis by implementing HIRO-style hierarchy, along with object-centric and latent object-centric subgoal variants, and comparing them against a flat Soft Actor-Critic (SAC) baseline on the ManiSkill PickCube task. We find that hierarchy does not improve over flat SAC on this short-horizon, dense-reward task: flat SAC solves PickCube efficiently, while the hierarchical methods require substantially more environment interactions, do not match the baseline's final reward, and depend on extensive reward engineering in which closing one failure mode repeatedly exposes another. A controlled experiment further shows that task success depends entirely on the worker's direct access to the environment reward: when forced to rely solely on the manager's subgoals, the worker fails to learn the task at all. We attribute hierarchy's poor fit to the task being short-horizon and effectively single-stage, to manipulation states being largely non-Euclidean and ill-suited to positional subgoals, and to a reward landscape in which dense shaping is easily exploited while sparse shaping weakens learning. Our results do not argue against HRL in general, but rather identify single-stage, dense-reward manipulation tasks where a flat baseline is already strong as a setting in which hierarchical decomposition offers little benefit.

1 Introduction

Robotic manipulation policies are brittle to changes in robot embodiment: when the robot's body or dynamics change, a flat policy that jointly learns task planning and motor control must be retrained from scratch, often at significant sample cost. Hierarchical reinforcement learning (HRL) offers a compelling alternative by decomposing a policy into a high-level task planner and a low-level motor controller trained to achieve the subgoals proposed by the manager. This structure raises the possibility that the high-level plan can remain fixed across embodiments while only the low-level worker is retrained, potentially reducing the cost of adapting to new robot dynamics. We test this

hypothesis by implementing HIRO-style hierarchy (Nachum et al., 2018), along with object-centric and latent object-centric subgoal variants, and comparing them against a flat Soft Actor-Critic (SAC) baseline (Haarnoja et al., 2018) on the ManiSkill PickCube task (Tao et al., 2024).

We find that hierarchy does not improve over flat SAC on this short-horizon, dense-reward task: flat SAC reliably solves PickCube within roughly 200K environment interactions, while the hierarchical methods require substantially more interaction and fail to match the baseline's final reward. This gap persisted despite extensive reward engineering, in which closing one failure mode consistently exposed another. To probe whether the high-level policy was contributing meaningful task guidance, we ran a controlled experiment in which we annealed the worker's environment-reward weight to zero, forcing it to rely solely on the manager's subgoals. The policy under these conditions failed to learn the task entirely, whereas an otherwise identical run that retained the environment reward solved it. This suggests that the high-level policy is not load-bearing: task success in our setting depends on the worker's direct access to the environment reward rather than on the subgoals proposed by the manager.

We attribute hierarchy's poor fit to three factors. First, PickCube is short-horizon and effectively single-stage, leaving little meaningful structure for a manager to decompose. Second, the manipulation state space is largely non-Euclidean, making HIRO's positional subgoal formulation a poor fit for contact-rich grasping. Third, the reward landscape is difficult to engineer under hierarchical decomposition: dense worker rewards are consistently exploitable, while sparser rewards slow learning to the point where the worker cannot discover grasping at all. Our transfer comparisons point the same way: any gains from hierarchical initialization appear to reflect general policy reuse rather than a uniquely reusable hierarchical abstraction that supports effective learning. Our results do not argue against HRL entirely, but they identify a regime—single-stage, dense-reward manipulation tasks—in which hierarchical decomposition offers little benefit over an already strong flat baseline.

Our contributions are:

- We implement and evaluate HIRO-style, object-centric, and latent object-centric hierarchical policies against a flat SAC baseline, exploring a series of subgoal-space and reward-shaping designs.
- We show that on short-horizon, dense-reward manipulation tasks like PickCube where a flat baseline is already strong, the high-level policy is not load-bearing: task success depends on the worker's direct access to the environment reward, not on the subgoals proposed by the manager.
- We identify task structure, subgoal geometry, and reward landscape as the three factors that make hierarchical decomposition a poor fit for this task regime.

2 Related Work

Hierarchical reinforcement learning (HRL) aims to improve learning efficiency by decomposing a policy into multiple levels of temporal abstraction. Nachum et al. (2018) propose HIRO, which learns a high-level policy that proposes subgoals in the state space for a goal-conditioned low-level policy, and uses an off-policy correction to relabel past subgoals as the low-level policy changes. HIRO was evaluated primarily on long-horizon locomotion and navigation tasks, such as ant maze navigation, where decomposing the task into a sequence of intermediate goals is naturally beneficial. We instead study HIRO-style hierarchy in the setting of short-horizon, contact-rich manipulation, where it is less clear that such decomposition helps. Our low-level controller (and baseline approach) is built on Soft Actor-Critic (Haarnoja et al., 2018), an off-policy maximum-entropy actor-critic algorithm. Prior work has shown that HRL can be effective for manipulation, but typically in sparse-reward or multi-stage settings: for example, Zhang et al. (2021) learn task-agnostic options in a self-supervised manner and report improved success rates on sparse-reward manipulation and navigation tasks where flat baselines struggle. These successes occur in a regime that is complementary to ours, since the task we study is short-horizon and densely rewarded, and is already solved efficiently by a flat policy.

Additional literature argues that manipulation policies benefit from structuring learning around objects rather than raw robot state. Sharma et al. (2020) learn to compose hierarchical object-centric controllers, where a policy selects among position, force, and rotation controllers defined relative to objects in the scene, and show that this structure improves sample efficiency and generalization. More

recently, Haramati et al. (2026) propose a two-level goal-conditioned framework in which subgoals are factored over entities in the scene, simplifying long-horizon tasks by changing only the relevant objects at each stage. Both works support the intuition that object relationships, rather than absolute robot configurations, are a useful structure for manipulation. We adopt this intuition at the level of the subgoal interface, defining high-level subgoals as object-centric quantities such as gripper-to-object and object-to-goal offsets, and we examine whether this representation allows the hierarchy to learn the task on our setting.

Our work also draws on goal-conditioned reinforcement learning and reward shaping. Because our low-level policy is conditioned on subgoals proposed by the high-level policy, it can be viewed as a goal-reaching policy. Andrychowicz et al. (2017) introduce Hindsight Experience Replay, which relabels achieved states as goals so that failed trajectories still provide useful learning signal, an idea that is relevant for training subgoal-conditioned policies under sparse feedback. To provide denser guidance to the low-level controller without changing the optimal policy, we use potential-based reward shaping (Ng et al., 1999), which guarantees that adding a shaping term of the form $F(s, s') = \gamma\Phi(s') - \Phi(s)$ leaves the set of optimal policies unchanged. We use this property when designing intermediate rewards for grasping, so that shaping terms encourage progress without introducing reward that the worker can exploit.

3 Methods

3.1 SAC Baseline

As a non-hierarchical baseline, we train a Soft Actor-Critic (SAC) (Haarnoja et al., 2018) policy that jointly learns task planning and motor control in a single network, mapping observations directly to actions. SAC is an off-policy, maximum-entropy actor-critic algorithm that augments the return with a policy entropy term, optimizing

$$J(\theta) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[r(s_t,a_t) + \alpha\mathcal{H}\big(\pi_\theta(\cdot \mid s_t)\big)\right], \qquad \mathcal{H}\big(\pi_\theta(\cdot \mid s_t)\big) = \mathbb{E}_{a_t\sim\pi_\theta}[-\log \pi_\theta(a_t \mid s_t)],$$

where α is a temperature controlling the trade-off between reward and exploration. The critic $\hat{Q}_\phi$ is trained by minimizing the soft Bellman error against the target

$$y_t = r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1}\sim\pi_\theta}\Big[\hat{Q}_{\phi'}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\theta(a_{t+1} \mid s_{t+1})\Big],$$

where $\hat{Q}_{\phi'}$ is a slowly-updated target network. We choose SAC because its off-policy replay mechanics are directly comparable to the hierarchical methods we study, making it a fair reference for measuring whether hierarchical structure adds any benefit. Because this flat policy does not decompose the task into high-level and low-level components, it lets us isolate the effect of adding hierarchy. We train SAC from scratch for each task and embodiment.

3.2 HIRO

We first experiment with implementing the HIRO algorithm Nachum et al. (2018) under our robot and environment setup for the `PickCube-v1` task. HIRO (Hierarchical Reinforcement learning with Off-policy correction) structures the policy as two levels: a high-level policy μ^{hi} that operates at a timescale of c steps and outputs subgoals $g_t \in \mathcal{G}$, as well as a low-level policy μ^{lo} that executes primitive actions to achieve those subgoals. The high-level policy is rewarded by the environment signal r_t, while the low-level policy receives an intrinsic reward measuring progress toward the current subgoal:

$$r^{lo}(s_t, g_t, a_t, s_{t+1}) = -\|\text{proj}(s_t) + g_t - \text{proj}(s_{t+1})\|_2$$

where $\text{proj}(\cdot)$ extracts the task-relevant state dimensions (in our setting, the three-dimensional TCP and cube coordinates, the only Euclidean state space components). Subgoals are defined as state differences and are updated at each step via $g_{t+1} = \text{proj}(s_t) + g_t - \text{proj}(s_{t+1})$, ensuring that the low-level target always points to the same absolute position in space as the agent moves.

Both policy levels train with off-policy SAC. The core challenge is that transitions $(s_t, g_t, \ldots, s_{t+c})$ stored in the high-level replay buffer become stale: subgoals issued by an older μ^{hi} may be inconsistent with the current μ^{lo} (since the worker is improving over time). HIRO addresses this with

off-policy correction: when sampling a high-level transition, it relabels g_t with the candidate $\tilde{g}$ that maximizes the likelihood of the observed low-level action sequence $\{a_t, \ldots, a_{t+c-1}\}$ under the current μ^{lo}:

$$\tilde{g} = \arg\max_{g \in \mathcal{C}} \sum_{i=0}^{c-1} \log \mu^{lo}(a_{t+i} \mid s_{t+i},\, g')$$

where g' is propagated forward using the subgoal transition function and $\mathcal{C}$ is a small candidate set sampled around the original g_t. This correction stabilizes high-level learning by ensuring replayed goals remain consistent with what the current low-level policy can pursue.

3.3 Object-Centric HIRO

A limitation of the HIRO subgoal formulation above is that, although subgoals are expressed as state differences, they are still defined over raw robot state coordinates and therefore remain tied to the specific embodiment. For manipulation (as opposed to navigation tasks), this is a suboptimal fit: task progress depends on the relationship between the gripper and the object, not on where either one is in the robot's own coordinate frame. We therefore modify the subgoal representation to be object-centric: instead of proposing raw Cartesian targets, the high-level policy outputs relative vectors between task-relevant entities,

$$g_t = \big[\,\text{tcp\_to\_obj},\ \text{obj\_to\_goal}\,\big],$$

where tcp_to_obj is the vector from the tool center point to the cube and obj_to_goal is the vector from the cube to the target. By construction, these subgoals describe what the task requires in terms of object relationships rather than absolute robot configurations, which makes them invariant to changes in embodiment and, in principle, reusable across robots whose dynamics differ but whose task geometry is the same. The two-level structure, intrinsic reward, and off-policy correction are matching the above HIRO formulation.

3.3.1 Latent Object-Centric HIRO

We additionally experiment with learning a lower- dimensional latent subgoal space, motivated by the idea that a more compact representation may make the high-level policy easier to learn by reducing the dimensionality of the space it must search over. We train an autoencoder that compresses the six-dimensional object-centric subgoal into a four-dimensional latent code, with an encoder E and a decoder D such that

$$z_t = E(g_t), \qquad \hat{g}_t = D(z_t), \qquad z_t \in \mathbb{R}^4,\ g_t \in \mathbb{R}^6,$$

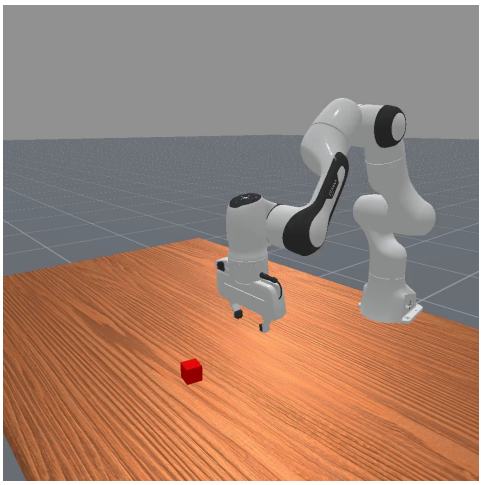
trained to minimize the reconstruction error $\|g_t - D(E(g_t))\|_2^2$. In this variant, the high-level policy proposes subgoals directly in the latent space z_t, which are decoded into object-centric subgoal vectors before being passed to the low-level policy.

3.4 Transfer Learning

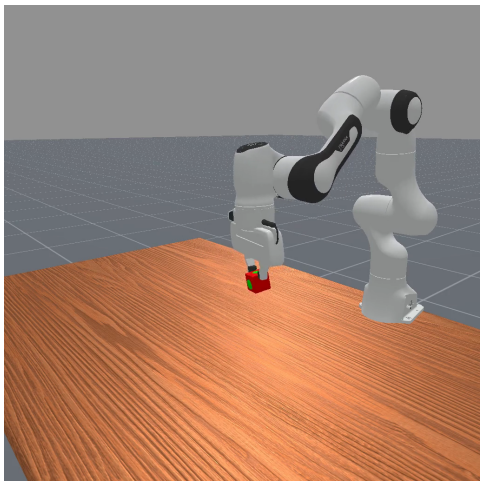
We study two transfer experiments. In the first, we evaluate *warm-start* initialization under a mild distribution shift (goal perturbation): we train both SAC and HRL on the standard `panda` embodiment and initialize from their respective source checkpoints (taken at 400K training steps), then continue training on a new goal configuration (initialized via a new randomly sampled goal seed). This tests whether a pretrained hierarchical policy provides a more useful initialization than a pretrained flat policy when the goal state shifts slightly out of distribution.

In the second experiment, we study cross-embodiment transfer to the `weakdampedpanda` embodiment. We initialize SAC from the `panda` checkpoint at 400K training steps and continue training on the target embodiment, comparing against SAC trained from scratch on `weakdampedpanda`. For the hierarchical method, we do a *manager-only* transfer from the `panda` HRL checkpoint at 400K steps, freeze it, and train a fresh worker on the target embodiment from scratch. This isolates whether the high-level policy and hierarchical structure offer any advantage in transfer learning when the robot dynamics change.

We assess transfer by reporting eval success rate and eval reward over environment interactions on the target embodiment, which together capture how quickly and how well each method adapts.



(a) Initial state: the cube rests on the table and the arm begins away from it.



(b) Goal state: the arm has grasped the cube and moved it to the target location.

Figure 1: The PickCube task in ManiSkill. The Panda arm must grasp the cube and bring it to a goal position

4 Experimental Setup

We run all experiments in the ManiSkill simulation environment (Tao et al., 2024), a GPU-parallelized benchmark for robotic manipulation.

4.1 Tasks

Our chosen task is PickCube (Figure 1), where the robot arm must grasp a cube and move it to a target position. We chose PickCube as our testbed under the assumption that its short horizon and well-defined reward structure would provide a controlled setting in which to first establish whether hierarchical decomposition offers any benefit at all, before studying embodiment transfer in more complex settings.

4.2 Robots

Our source embodiment is the standard Franka Emika Panda arm (`panda`). To study embodiment transfer, we define a modified arm that shares identical kinematics, observation space, and action space with the source, but differ in gripper dynamics. `weakdampedpanda` reduces the gripper finger drive force by $0.75\times$, giving it a weaker grasp and increases joint damping by $1.5\times$, resulting in slower gripper closure. Because the variant leave the observation and action dimensions unchanged, a policy trained on the source can be transferred to the perturbed embodiment without modifying its architecture, which lets us study adaptation to new dynamics in isolation.

4.3 Control Modes

We use `pd_joint_delta_pos` control, in which each action specifies an incremental change to the robot's joint positions, tracked by a PD controller. An alternative is end-effector (EE) space control, where actions instead specify changes to the position and orientation of the gripper and an inverse kinematics solver computes the joint commands needed to achieve them. We use joint-space control across all embodiments so that the action interface stays fixed when the robot dynamics change, which keeps the transfer setting consistent.

4.4 Evaluation

We evaluate policies using two measures: eval success rate, reported as a success-once rate (the fraction of evaluation episodes in which the task is completed at least once during the episode), and

Table 1: Peak evaluation success and reward within the first 500K steps (PickCube, Panda Arm).

Method	Max Eval Success	Max Eval Reward
SAC Baseline (flat)	1.00	0.82
HIRO Original	0.00	0.14
HIRO + Reward Shaping	0.00	0.12
Object-Centric	0.00	0.13
Latent Object-Centric (Baseline)	0.31	0.50
Latent Object-Centric + Reward Shaping	1.00	0.71

eval reward, the average return over evaluation episodes, which provides a finer-grained signal of partial progress than the binary success measure. We report both as learning curves over environment interactions to reflect sample efficiency rather than only final performance. To obtain stable estimates, we evaluate over 16 episodes, since we found small-sample estimates to be unreliable.

For embodiment transfer, we similarly assess performance by reporting eval success rate and eval reward over environment interactions on the target embodiment, capturing how quickly and how well each method adapts.

5 Results

5.1 Quantitative Evaluation

Table 1 summarizes the final performance of method we explored. Flat SAC reached near-perfect evaluation success (≈ 1.0) and an evaluation reward of ≈ 0.8 within roughly 200K environment interactions. The original HIRO formulation plateaued at an evaluation reward of ≈ 0.14 with a success rate near zero throughout training. The grasp-gated and object-centric variants showed modest improvement but similarly failed to achieve meaningful success. The latent object-centric variant with reward shaping was the strongest hierarchical method, reaching a comparable success rate to SAC but only after roughly 500K interactions and plateauing at a lower evaluation reward of ≈ 0.7. Across all hierarchical variants, final reward remained below the SAC baseline, and no variant matched SAC's sample efficiency, even with the cascade of subgoal-space and reward-shaping improvements we worked through (5.2.1).

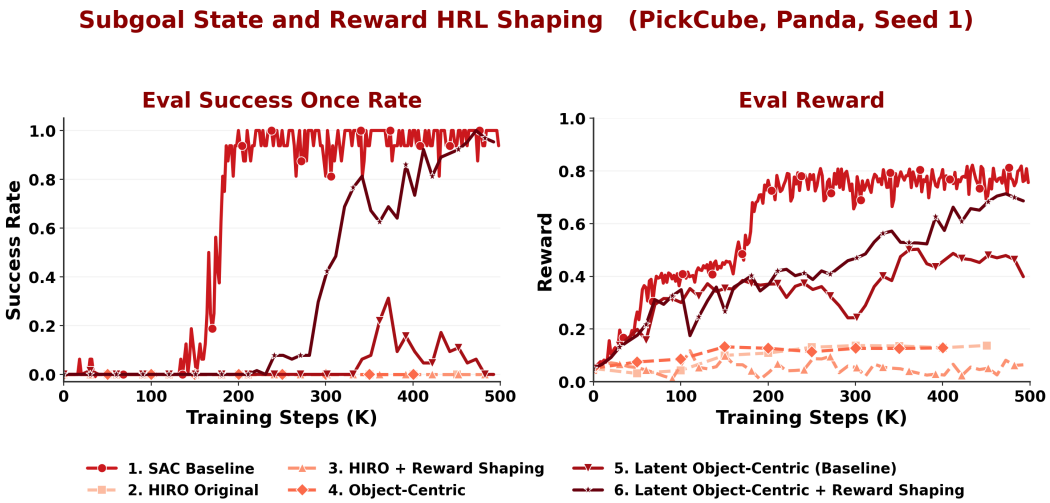


Figure 2: SAC consistently outperformed HIRO-inspired HRL policies.

5.2 Qualitative Analysis

5.2.1 Reward Shaping for HIRO-Inspired HRL Policies

Adapting HIRO for PickCube required substantial reward engineering. Unlike navigation, where Euclidean subgoal residuals translate naturally into worker rewards, we realized that grasping introduces contact dynamics and uncontrollable object states that make it difficult to engineer a meaningful worker reward signal. Our design evolved through a cascade of fixes in which resolving one failure consistently exposed the next.

Since the ManiSkill PickCube state space has 42 dimensions and prior HIRO work computes worker reward as a Euclidean distance to the subgoal, we projected onto the 6 Euclidean dimensions (TCP x, y, z and cube x, y, z) as our initial HIRO goal subspace.

Iteration 0: Subgoal residual. Following HIRO (Nachum et al., 2018), the worker minimizes the distance between its achieved state and the manager's subgoal:

$$r_{\text{worker}} = -\|\text{proj}(s_{t+1}) - (\text{proj}(s_t) + g_t)\|.$$

Because the subgoal includes cube position, the worker was penalized for not moving the cube even before contact was possible. This corrupted reward signal prevented meaningful learning entirely (Figure 2, line 2).

Iteration 1: Grasp gate. We then suppressed the cube residual until the robot was actually holding the object:

$$r_{\text{worker}} = -\|\Delta_{\text{TCP}}\| - \mathbf{1}[\texttt{is\_grasped}] \cdot \|\Delta_{\text{cube}}\|.$$

This removed the unfair pre-grasp penalty, but introduced a new failure: since the cube penalty only activates after grasping, the optimal strategy to minimize this penalty became to never grasp at all. The robot learned to approach the cube but never close its gripper.

Iteration 2: Potential-based grasp incentive. We needed to reward grasping without creating a signal the robot could exploit by repeatedly grasping and dropping. Potential-based reward shaping (PBRS) (Ng et al., 1999) provides this guarantee: for any potential function $\Phi(s)$ that assigns a scalar value to each state, adding a shaping term of the form $F(s, s') = \gamma\Phi(s') - \Phi(s)$ to the reward telescopes over a full trajectory and cancels out, leaving the optimal policy unchanged. We defined Φ to be high when the robot is holding the cube and zero otherwise, which translates to a one-time bonus on grasp and a penalty on drop. Because the grasping bonus is non-repeatable by construction, it cannot be farmed. Eval videos showed some improvement, but grasp discovery remained slow: without a continuous signal pulling the gripper toward the cube, the policy still rarely made contact.

Further iterations. The reward shaping difficulty was what ultimately motivated us to also explore the object-centric and latent object-centric subgoal spaces, since we struggled to see signs of effective learning of the task structure within the vanilla HIRO state space, even after the above fixes (Figure 2, line 3). The same exploit pattern ended up persisting across all three subspaces, but reward shaping in the latent object-centric subspace ultimately yielded the strongest HRL results—still well below SAC, but substantially ahead of all the other HRL variants we explored (line 6). Examples of additional reward shaping iterations we tried were:

- **Dense reach signal:** We added a continuous reward for reducing the distance between the gripper and the cube, which sped up cube approaching. The robot quickly learned to hover just above the cube to collect this reward without ever closing its gripper.
- **Post-grasp lift reward:** A bonus conditioned on `is_grasped` rewarded lifting the cube off the table. But the policy learned to lift slightly and then stop, treating "barely off the ground" as success.
- **Cube-centric reward:** To prevent hovering, we removed all arm-to-cube distance terms for the manager's reward (since we are in a dense environment setup) and rewarded only raw cube displacement. But this made the reward signal too sparse, and the robot struggled to discover grasping in the first place.

- **Deadband near goal:** We tried adding a stillness bonus for stopping near the target, in attempt to suppress late-stage shaking. Instead, the robot learned to stop just inside the threshold rather than placing the cube accurately.

Across all variants, every fix we experimented with exposed a new exploitable boundary elsewhere. The reward landscape for contact-rich manipulation proved to be too complex for hierarchical decomposition to help: flat SAC, which jointly learns approach, grasping, and placement without subgoal alignment, consistently outperformed every HRL variant we tried.

5.2.2 Reward Annealing: Removing Environment Signal Causes Low-Level Worker Collapse

To isolate the contribution of the high-level policy to task success, we ran a controlled experiment comparing two otherwise identical runs that differ only in the worker's environment-reward weight w (Figure 3). In the first run, we keep the environment reward fixed at $w = 0.5$. Here, the policy solves the task, reaching an eval success rate of 1.0. In the second run, we anneal the environment reward from $w = 0.5$ to $w = 0$ over training, forcing the worker to rely solely on the subgoals proposed by the high-level policy. Under this annealing, the policy never solves the task, with success remaining near zero throughout training. Since the only difference between the two runs is whether the worker can observe the environment reward, this result suggests that task success depends on the worker's direct access to the environment reward rather than on the subgoals from the manager. If the high-level policy were providing useful task guidance, the annealed worker should still have been able to learn from its subgoals alone—instead, it fails entirely. We take this as evidence that the high-level policy is not load-bearing in this setting: the hierarchy is not the source of task success, and the low-level worker is effectively learning the task on its own when given the environment reward.

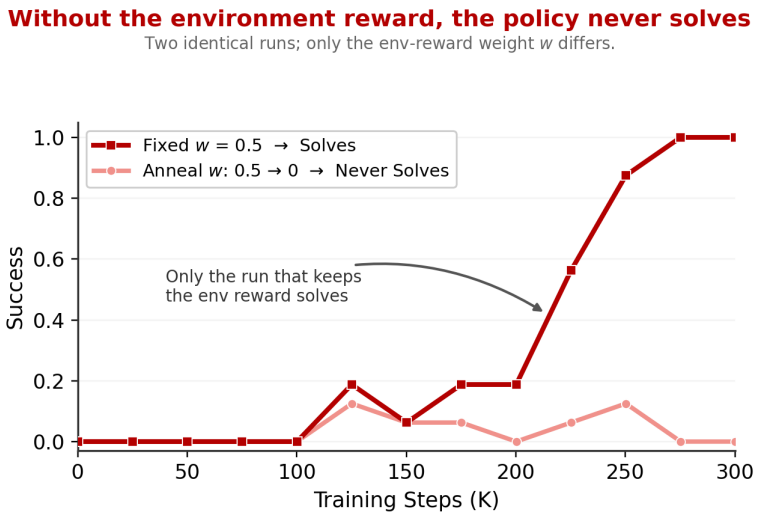


Figure 3: Two identical runs differing only in the environment-reward weight w. Keeping the reward ($w = 0.5$) solves the task; annealing it to zero never does, showing the worker depends on the environment reward, not the manager's subgoals.

5.2.3 Transfer Comparison

Having established that hierarchy does not improve over flat SAC on the source embodiment, we next ask whether hierarchical structure provides any advantage at all. First, we evaluate both SAC and HRL under a warm-start transfer setting, initializing each method from a checkpoint trained on the standard Panda after 300k environment steps and reuse the weights to see if hierarchical learning can learn the task with fewer interactions than flat SAC (Figure 4) on a new target.

We also apply cross-embodiment transfer by training both SAC and HRL on standard Panda up to 400k environment steps then using these weights to retrain on the weakpanda embodiment (Figure 5).

In both these experiments, we see that SAC shows a large short-term drop in reward and success but quickly recovers, eventually plateauing near 0.8 reward. HRL improves more smoothly but plateaus early at roughly 0.7 reward. The large initial SAC drop suggests that the source policy is not immediately robust to embodiment shift but easy to repair under dense reward and shared observation spaces. HRL shows smaller adaptation shock and a smoother improvement curve, suggesting that the transferred structure is initially more stable under the new embodiment. However, it plateaus earlier and below SAC, which indicates that this stability does not translate into better asymptotic performance in this environment. The transfer gains appear to come from policy reuse rather than a uniquely reusable hierarchical abstraction.

The smoother HRL curve suggests that the transferred hierarchical structure is initially more stable under the new embodiment; however, it plateaus earlier and below SAC. Importantly, Figure 5 shows that the HRL manager-only transfer, which isolates the manager's contribution by training a fresh worker from scatch, performs the worst than all

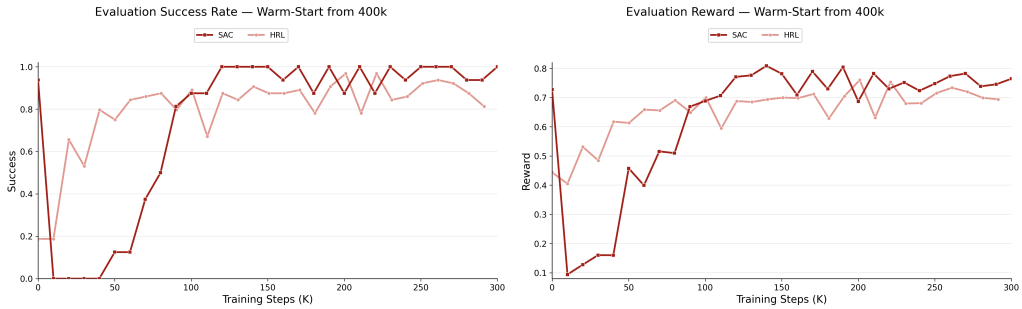


Figure 4: Warm start experiment demonstrates quick recovery in SAC and stability of HRL.

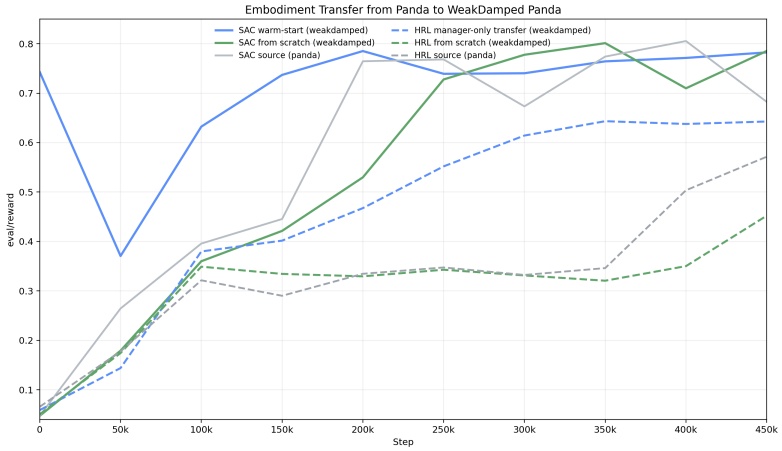


Figure 5: Embodiment transfer from Panda to WeakDamped Panda, with both SAC and HRL initialized from a Panda policy pretrained for 400k interactions.

6 Discussion

6.1 Why HIRO Underperforms on PickCube: Tasks That Do Not Benefit From Hierarchy

Our results suggest that hierarchical decomposition is poorly matched to the PickCube task along three dimensions: task structure, subgoal geometry, and reward landscape.

Task structure: short-horizon and effectively single-stage. HIRO was designed for long-horizon tasks like ant maze navigation, where decomposing a trajectory into intermediate subgoals is naturally beneficial. PickCube lacks this structure: the robot must simply reach the cube, grasp it, and lift it to

a target, a sequence that flat SAC learns within roughly 200K training steps (as shown in Figure 2), leaving minimal gap for hierarchy to close. As shown in Figure 3, the worker struggles to learn from manager-posed subgoals alone and only makes progress when given direct access to the environment reward, suggesting that the high-level policy does not provide useful enough guidance to justify the added complexity when the task itself is this straightforward.

Subgoal geometry: manipulation is very different from navigation. HIRO proposes subgoals as Euclidean positional targets, which suits navigation but is a poor fit for manipulation. Of the 42 ManiSkill state dimensions for PickCube, only 6 are Euclidean (TCP x, y, z and cube x, y, z); the rest encode quaternion orientations and joint velocities for which Euclidean distance is not meaningful. Projecting onto these 6 dimensions was necessary to apply HIRO's subgoal formulation, but discards most of the state information the worker needs for precise manipulation, including any signal about gripper state or contact. Switching to object-centric subgoals, expressing goals as relative vectors between the gripper, cube, and target, partially addressed this mismatch and yielded our best hierarchical result (Figure 2, line 6), though it still fell well short of flat SAC.

Reward landscape: dense shaping encourages hacking, sparse shaping weakens learning. As discussed in Section 5.2.1, the core tension we discovered is that dense worker rewards are consistently exploitable in this manipulation task setting. For example, hovering near the cube earned the manager farmable approach reward, and avoiding contact prevented the cube residual penalty from activating on the worker. Closing each loophole introduced a new one; making the reward sparser eliminated loopholes but it also slowed learning until the worker could no longer discover grasping. Flat SAC avoids this entirely by optimizing the environment reward directly, with no intermediate subgoal objective that can be gamed independently of task progress.

6.2 Manager-Worker Misalignment: Workers Ignore Subgoals When Environment Reward Is Visible

A consistent theme across our experiments is that the high-level policy does not appear to be load-bearing: task success is driven by the worker's access to the environment reward rather than by the manager's subgoals. The reward annealing experiment (Figure 3) shows this most directly. When the environment reward is removed and the worker must rely on the manager's subgoals alone, the policy never learns the task, whereas the otherwise identical run that keeps the reward solves it. This suggests that the worker does not extract enough useful guidance from the subgoals to solve the task on their basis alone.

Our transfer results point in the same direction. If the high-level policy had learned genuinely reusable task structure, transferring it to a new embodiment should have produced a distinct advantage over simply initializing from a pretrained flat policy. Instead, both SAC and HRL benefited similarly from warm-start initialization, and HRL did not outperform SAC after adaptation. This suggests that the transfer gains reflect general policy reuse (starting from a good initialization) rather than anything specific to the hierarchical abstraction. Combined with the reward annealing result, this points to a consistent conclusion: on this task, the worker learns through the environment reward, and the high-level policy contributes little to that process.

7 Conclusion

We set out to study whether hierarchical reinforcement learning could improve learning and embodiment transfer for robotic manipulation, using HIRO-style hierarchy and an object-centric variant on the ManiSkill PickCube task. Across our experiments, we found that hierarchy did not improve over a flat SAC baseline on this class of tasks. Flat SAC solved PickCube efficiently, while the hierarchical methods required substantially more interaction, did not match the baseline's final reward, and depended heavily on extensive reward engineering that repeatedly exposed new exploitable failure modes.

Our analysis points to a single underlying explanation: on this task, the high-level policy is not load-bearing. The reward annealing experiment showed that the worker only learns the task when it has direct access to the environment reward, and not when it must rely on the manager's subgoals alone, while our transfer results suggested that any gains came from general policy reuse rather than

from a uniquely reusable hierarchical abstraction. We also identified why hierarchy is poorly matched to this setting, namely that PickCube is short-horizon and effectively single-stage, that its state space is largely non-Euclidean and a poor fit for HIRO's positional subgoals, and that its reward landscape makes dense worker rewards consistently exploitable while sparser rewards weaken learning.

These conclusions come with clear limitations. We studied a single short-horizon, dense-reward task on which a flat baseline is already strong, which is precisely the regime where hierarchy is least likely to help, so our results should not be read as evidence against hierarchical RL in general. It is possible that longer-horizon, sparser-reward tasks, where a flat baseline is weaker and the task has genuine multi-stage structure for a manager to decompose, would reveal benefits from hierarchy that PickCube cannot. Promising directions for future work include evaluating these methods on such tasks, and extending the embodiment-transfer study that motivated this project to a wider range of robots and perturbations.

8 Team Contributions

- **Ashwin:** Co-built the HIRO implementation and also researched / built the object-centric version. Diagnosed manager-worker misalignment and helped with the reward modeling experiments and analysis. Planned and helped set up training on Maniskill environment. Co-wrote paper.
- **Anjali:** Designed and conducted experiments on reward shaping / reward hacking experiments. Co-built the HIRO implementation. Helped design experiments, debug, and analyze results. Researched / debugged action space configuration. Co-wrote paper.
- **Arianna:** Led the transfer experiments across goal state and robot. Built the latent object-centric HIRO implementation, helped with reward modeling and analyzing results. Planned and helped set up training on Maniskill environment. Co-wrote paper.

Changes from Proposal Our proposal centered on object-centric hierarchical RL for embodiment transfer, with transfer as the main focus. As the project developed, we found that implementing the different HIRO variants and exploring reward modeling raised deeper questions about whether hierarchy was helping at all, so we shifted our focus toward understanding the behavior of the hierarchical methods themselves. We see this as a natural pivot, since establishing whether hierarchy helps on the base task is a prerequisite for studying whether it transfers.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning (ICML)*.

Dan Haramati, Carl Qi, Tal Daniel, Amy Zhang, Aviv Tamar, and George Konidaris. 2026. Hierarchical Entity-Centric Reinforcement Learning with Factored Subgoal Diffusion. In *International Conference on Learning Representations (ICLR)*.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *International Conference on Machine Learning (ICML)*.

Mohit Sharma, Jacky Liang, Jialiang Zhao, Alex LaGrassa, and Oliver Kroemer. 2020. Learning to Compose Hierarchical Object-Centric Controllers for Robotic Manipulation. In *Conference on Robot Learning (CoRL)*.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. 2024. ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI. *arXiv preprint arXiv:2410.00425* (2024).

Jesse Zhang, Haonan Yu, and Wei Xu. 2021. Hierarchical Reinforcement Learning By Discovering Intrinsic Options. In *International Conference on Learning Representations (ICLR)*.