SLAC: Simulation-Pretrained Latent Action Space for Whole-Body Real-World RL

Jiaheng Hu[1] **Peter Stone**[1,2] **Roberto Martín-Martín**[1,3]
[1]The University of Texas at Austin [2]Sony AI [3]Amazon

***Abstract*—Building capable household and industrial robots requires mastering the control of versatile, high-degree-of-freedom (DoF) systems such as humanoid robots. While reinforcement learning (RL) holds promise for autonomously acquiring robot control policies, scaling it to high-DoF embodiments remains challenging. Direct RL in the real world demands both safe exploration and high sample efficiency, which are difficult to achieve in practice. Sim-to-real RL, on the other hand, is often brittle due to the reality gap. This paper introduces SLAC, a method that renders real-world RL feasible for complex embodiments by leveraging a *low-fidelity simulator* to pretrain a *task-agnostic latent action space*. SLAC trains this latent action space via a customized unsupervised skill discovery method designed to promote temporal abstraction, disentanglement, and safety, thereby facilitating efficient downstream learning. Once a latent action space is learned, SLAC uses it as the action interface for a novel off-policy RL algorithm to autonomously learn downstream tasks through real-world interactions. We evaluate SLAC against existing methods on a suite of bimanual mobile manipulation tasks, where it achieves state-of-the-art performance. Notably, SLAC learns contact-rich whole-body tasks in under an hour of real-world interactions, without relying on any demonstrations or hand-crafted behavior priors. More information and robot videos at robo-rl.github.io**

I. Introduction

Future robots are expected to perform diverse tasks in unstructured environments. Achieving this requires controlling high-degree-of-freedom, multi-purpose systems like mobile manipulators and humanoids. These robots offer unique opportunities to handle complex and ambitious tasks: their many degrees of freedom allow them to pursue multiple control objectives simultaneously, including sophisticated contact-rich interactions (e.g., wiping a board while avoiding obstacles), and their combined locomotion and manipulation capabilities provide large workspaces for long-horizon operation. However, these opportunities come with significant challenges. The combination of long-horizon, multi-objective tasks and high-dimensional action spaces makes policy optimization particularly difficult. Moreover, performing contact-rich interactions with the whole body increases the risk of damaging the robot and its surroundings, making safety a critical concern.

As a result, prior policy learning methods that have shown success in simpler settings often struggle to scale to these high-DoF, multi-purpose robots. On the one hand, successes in Imitation Learning (IL) with simple fixed-base robot arm [1–3] are hard to replicate for high DoF robot systems due to the difficulty both in creating a teleoperation interface and in collecting a sufficient number of high-quality demonstrations. On the other hand, while Reinforcement Learning (RL) algorithms can learn without expert demonstrations, they require many environment steps before convergence, and has primarily been conducted in simulation [4–10], with policies transferred zero-shot to the real world. However, these approaches often face significant challenges in bridging the reality gap [11–15], which often widen as the complexity of robots and tasks increases, despite costly domain randomization or the tedious construction of high-fidelity digital twins [4, 16, 17].

Real-world RL offers a promising alternative: by enabling robots to learn directly through trial-and-error interactions with the physical world, we may bypass both the reality gap and the need for costly human demonstrations. Unfortunately, existing real-world RL approaches [18–21] remain largely limited to simple domains such as tabletop manipulation and quadruped locomotion. They do not scale effectively to more complex tasks and embodiments, due to fundamental challenges in ensuring **safe exploration** given larger workspaces and frequent physical contact with the environment, and in achieving **sample-efficient learning** in the presence of high-dimensional action spaces, long-horizon tasks, and intricate reward structures. These factors severely limit the applicability of current real-world RL techniques to complex robotics problems.

This paper introduces **SLAC**: **S**imulation-Pretrained **L**atent **AC**tion Space for Real-World RL, which utilizes *low-fidelity simulator* to make real-world downstream RL feasible for high DoF robots such as mobile manipulators. The SLAC framework introduces a two-step procedure to circumvent the main challenges of real-world RL, namely unsafe exploration and sample inefficiency. In the first step, SLAC learns a task-agnostic latent action space in a coarsely aligned, low-fidelity simulator via Unsupervised Skill Discovery (USD) [22–25]. SLAC utilizes a novel USD objective that shapes this latent action space to be (1) **temporally extended**, enabling more effective exploration than when directly using the low-level action space by reducing decision frequency; (2) **disentangled**, allowing each latent action dimension to independently affect the states, thereby facilitating joint optimization of multiple objectives without conflict; and (3) **safe**, avoiding dangerous behaviors that could damage the robot. In the second step, the learned SLAC latent action space is used by a novel off-policy RL algorithm to efficiently learn downstream tasks directly in the real world. Critically, this design offers robustness to the reality gap: even if latent actions exhibit slight behavioral mismatches between simulation and the real world, the downstream policy can still learn to solve the task by directly

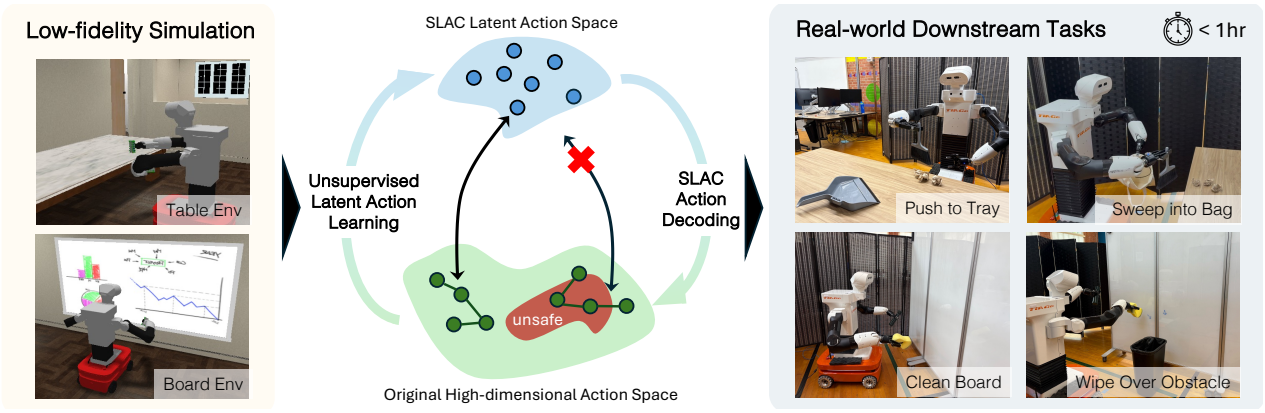


Fig. 1: SLAC uses a task-agnostic action space trained in low-fidelity simulation (*left*) to learn downstream tasks in the real world. This latent action space is safe, temporally extended, and disentangled, enabling a bimanual mobile manipulator to solve challenging contact-rich whole-body tasks (*right*) with less than an hour of autonomous real-world interactions.

selecting effective latent actions based on sparse reward.

We evaluate our method on a complex, high-DoF, bimanual mobile manipulator, where SLAC can learn contact-rich whole-body tasks in less than an hour of real-world interactions, using only onboard sensor signals. To the best of our knowledge, SLAC is the first algorithm that enables a high-DoF mobile manipulator to learn with RL in the real world without relying on any demonstrations/mocap data [26, 27] or hand-crafted behavior priors [26, 28, 29].

II. Related Work

Robots can be trained to perform tasks using four main approaches: (1) sim-to-real reinforcement learning, (2) real-world reinforcement learning, (3) classical motion planning and control, and (4) learning from demonstrations. In this section, we review the first two approaches, which are most relevant to SLAC. We defer discussion of the latter two [1–3, 30–48] to the appendix.

Sim-to-Real Reinforcement Learning: While Reinforcement Learning (RL) provides a way for agents to learn sophisticated behaviors from trial and error, popular algorithms like PPO [49] are quite sample-inefficient and can require billions of samples before they converge. Many works have therefore resorted to performing the RL training completely in simulation [4–11, 13, 14], and zero-shot transfer the learned policy into the real-world. Such a procedure requires the simulation to have very high fidelity, and can pose significant challenges for simulated object creation, especially for tasks that are contact-rich/non-rigid [12]. Unlike these works, SLAC relies on simulated interactions only to provide a suitable action space for downstream real-world RL, which reduces or eliminates the reliance on high-fidelity simulation.

RL in the Real World: Directly doing RL in the real world offers a promising direction to avoid the requirement of high-fidelity simulation [18–21, 50–52]. However, these methods often target simple domains such as fixed-base manipulator, and fall short when applied to more complex embodiments such as whole-body mobile manipulation due to the high requirements for sample efficiency and safe exploration. In the rare exceptions where a mobile manipulator does learn through trial-and-error in the real world [26–29], domain knowledge is often injected to simultaneously facilitate safety and efficient exploration, in the form of ad-hoc hand-crafted motion priors [26, 28, 29] and/or demonstrations [26, 27]. By comparison, SLAC enables high-degree-of-freedom mobile manipulators to learn downstream tasks in the real world without relying on any demonstrations or hand-crafted behavior priors.

III. SLAC: Simulation-Pretrained Latent Action Space for Real-World RL

SLAC aims to enable sample-efficient and safe real-world reinforcement learning (RL) for high DoF robots such as mobile manipulators. We formulate the real-world RL problem as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R_{task}, \gamma)$, where $\mathcal{S}$ is the set of underlying environment states, $\mathcal{A}$ is the high-dimensional native action space (e.g., joint velocities or torques), $\mathcal{O}$ is the observation space (e.g. camera images), $P(s'|s,a)$ is the state transition function, $R_{task}(s,a) = \sum_{i=1}^{m} R_i(s,a)$ is a composite reward function with $m \geq 1$ term(s)[1], and $\gamma \in (0,1]$ is the discount factor. The objective is to learn a policy $\pi(a|o)$ that maximizes the expected return:

$$\pi^*(a|o) = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{task}(s_t, a_t)\right] \quad (1)$$

Due to the high dimensionality of $\mathcal{A}$ and the complexity of real-world tasks, directly optimizing $\pi(a|o)$ in the real world is prohibitively sample-inefficient and unsafe. To address these issues, we propose to replace the native control space $\mathcal{A}$ with an N-dimensional multi-discrete [2] *latent action space* $\mathcal{Z} = \mathcal{Z}^1 \times \cdots \times \mathcal{Z}^N$ learned in a **low-fidelity simulation**, which

[1]This formulation is general, as any reward function can be expressed as a sum of component functions.

[2]While our method is compatible with both continuous and discrete latent actions, we focus on the discrete case in our experiments. We discuss this choice in the appendix.

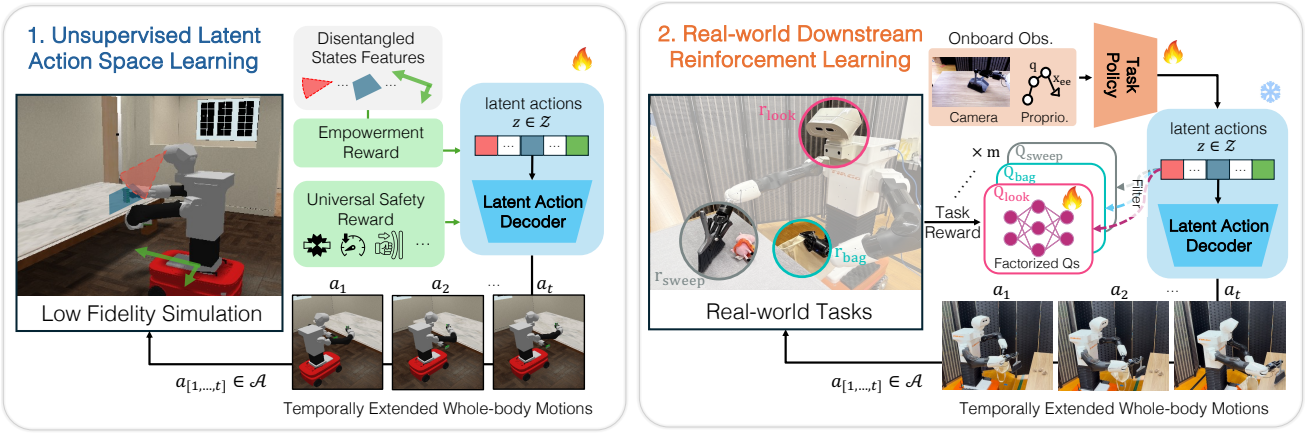


Fig. 2: The two-step SLAC procedure to enable real-world policy learning. (*Left*) In the first step, SLAC learns a **Latent Action Decoder** that maps each latent action, $z \in \mathcal{Z}$, to a sequence of low-level robot actions, $(a_0, \dots, a_T), a_t \in \mathcal{A}$. This decoder is learned in low-fidelity simulation via unsupervised skill discovery with novel objectives that encourage the robot to independently control different state features (e.g., camera directions, contacts with table, base locations) while being safe. (*Right*) In the second step, once the decoder is trained, the robot learns downstream tasks with RL in the real world using the SLAC latent action space. The task policy directly takes in the onboard sensor observations of the robot (i.e., images, proprioception) and outputs latent actions z that are decoded into safe robot actions. SLAC applies Factorized Latent-Action SAC to optimize the policy for downstream tasks with multi-term reward (e.g., look at the objects, keep a bag close, sweep the trash) directly in the real world with very few samples, converging in less than an hour, by taking advantage of high-frequency off-policy updates and factorized Q decomposition.

does not accurately replicate the visual or physical properties of the real world and does not implement the task reward R_{task}, but approximately retains key physical affordances and shares the same robot action space $\mathcal{A}$.

Specifically, we aim to learn a **latent action decoder** $\pi_{dec}(a|o_{dec}, z)$, which converts a latent action $z \in \mathcal{Z}$ into low-level actions $a \in \mathcal{A}$ based on a low-dimensional decoder observation o_{dec} that is shared across simulation and the real world (e.g., proprioceptive states, furniture poses). We discuss in Sec. III-A how we learn this latent action decoder through unsupervised skill discovery.

Once the latent action decoder is learned, SLAC trains a perception-to-latent **task policy** $\pi_{task}(z|o)$ in the real world given a downstream task reward. $\pi_{task}(z|o)$ selects latent actions based on (history of) high-dimensional real-world observations $o \in \mathcal{O}$ (e.g. camera images), and is **trained entirely in the real world** using a novel sample-efficient off-policy RL method explained in Sec. III-B.

Together, the task policy and the latent action decoder define a hierarchical visuomotor policy over low-level robot actions, which can be run directly on a real robot:

$$\pi(a|o) = \int_z \pi_{dec}(a|o_{dec}, z)\, \pi_{task}(z|o)\, dz \tag{2}$$

We show the full pipeline of our two-step method in Fig. 2.

A. Learning a Latent Action Space in Simulation

The first step of SLAC seeks to learn a task-agnostic latent action space capable of supporting a wide range of real-world task variations. Unsupervised Skill Discovery (USD) [22–25], which learns diverse task-agnostic behaviors without relying on explicit task rewards, offers a promising approach for acquiring such an action space. This process yields a latent skill decoder $\pi_{dec}(a|o_{dec}, z)$, where each latent skill z induces a distinct behavior. These learned skills can then be composed by a task policy $\pi_{task}(z|o)$ to efficiently solve downstream tasks, where the learned skill space serves as a temporally extended action space of the task policy. [3]

However, despite its potential, USD has seen limited adoption in robotics due to its high sample complexity, which renders it impractical for direct deployment on real robots. Instead, SLAC addresses this limitation by conducting USD entirely in simulation, where data collection is fast and inexpensive. Our key insight is that even **low-fidelity simulation** can serve as an effective substrate for behavior pretraining: as long as the learned skills span a sufficiently diverse range of behaviors, they can be composed downstream to solve real-world tasks. To this end, SLAC employs simulation environments that do not directly replicate a real-world counterpart (e.g. not visually realistic, no hard-to-simulate objects like marker traces), but still preserve key geometric affordances that are potentially useful for downstream tasks (e.g. a whiteboard that the robot can touch, an obstacle that the robot may collide with). We show some of these environments in Fig. 1. Importantly, because we do not need the simulation environment to exactly match the real world, prototyping a new environment and training a corresponding latent action

[3]For the rest of this paper, we will use "skills" and "latent actions" interchangeably.

space can be done quickly.

Given this low-fidelity simulation, SLAC leverages the Disentangled Unsupervised Skill Discovery (DUSDi) framework [25] for learning a disentangled latent action space, which has been shown to facilitate sample-efficient downstream learning. The DUSDi framework optimizes the following mutual-information-based objective:

$$\mathcal{J}(\theta) = \sum_{i=1}^{N} I(\mathcal{S}^i; \mathcal{Z}^i) - \lambda I(\mathcal{S}^{\neg i}; \mathcal{Z}^i), \tag{3}$$

where $\{\mathbf{S}^i\}_{i=1}^N$ is a set of state entities (e.g. whiteboard, table, body parts) in the environment that the robot can interact with; $\mathcal{Z} = \mathcal{Z}^1 \times \cdots \times \mathcal{Z}^N$ is the latent action space, factorized by design into N dimensions; and $\lambda < 1$ is a weighting factor for the disentanglement objective. Intuitively, this objective encourages each latent action dimension $\mathcal{Z}^i$ to control only its corresponding state entity $\mathcal{S}^i$, thereby creating a **disentangled** and **temporarily extended** action space that allows the robot to independently and simultaneously control different entities in the environment – an ability critical for downstream learning. To optimize this objective tractably, we can approximate Eq. 3 through variational inference [54], resulting in the following reward function:

$$r_{skill}(s, a) \triangleq \sum_{i=1}^{N} q_\phi^i(z^i|s^i) - \lambda q_\psi^i(z^i|s^{\neg i}) \tag{4}$$

where q_ϕ^i and q_ψ^i are variational distributions that can either be learned through self-supervised learning or manually constructed, in which case the objective reduces to a form of goal-conditioned reinforcement learning [55]. In SLAC, we opt for the latter to constrain the learned behavior.

However, naively optimizing the objective above provides the robot with no notion of **safety**, which can result in irreversible damage when deployed on real hardware. To address this issue, SLAC incorporates *universal safety constraints* in the form of a safety reward function r_{safe} that discourages unsafe behaviors. In principle, r_{safe} can take any form. In practice, for our robot experiments, we found that the same safety reward function can be used universally across all environments and tasks. Specifically, our safety reward r_{safe} consists of the following components: (1) Penalizing large absolute actions. (2) Penalizing large relative changes in action. (3) Penalizing collisions. (4) Penalizing excessive force on the robot. We provide the detailed formulation for r_{safe} in the appendix. The final objective for learning the latent action space combines task-agnostic exploration with these safety considerations, as shown below:

$$r_{latent} = r_{skill} + r_{safe} \tag{5}$$

We directly optimize Eq. 5 via RL in simulation. In appendix, we show the pseudo-code for latent action learning, and discuss the properties of the learned action space.

B. Sample-Efficient Learning of Downstream Tasks in the Real World

Given the SLAC latent action space learned in simulation (Sec. III-A), the second step of SLAC derives a sample-efficient off-policy Reinforcement Learning Algorithm, which we named Factorized Latent-Action SAC (FLA-SAC), to directly learn downstream tasks in the real world. FLA-SAC is built on top of Soft Actor-Critic [56], with three important algorithmic innovations to boost the performance. We show the pseudo-code for FLA-SAC in the appendix.

Efficient Use of Experiences: Due to the high cost of collecting real-world trajectories, our goal is to develop algorithms that efficiently learn from a few steps of environment interactions. One critical strategy to achieve this efficiency is giving an off-policy algorithm a high update-to-data (UTD) ratio, where the number of actor-critic updates is significantly higher than the number of environment steps, by repeatedly sampling from a replay buffer that stores all previous environment steps. FLA-SAC leverages such a high UTD ratio to maximize data efficiency.

Since a high UTD ratio can increase the risk of overfitting, recent work [19, 20, 53] proposed various techniques, such as layer normalization and critic ensembling, to regularize the policy update. However, we empirically found these methods to be ineffective in our setting. Instead, we observed that simply reducing the batch size during updates acts as powerful regularization that significantly improves performance. This adjustment helps the model escape poor local optima by introducing higher gradient variance, which promotes more effective exploration of the parameter space.

Large Discrete Action Space: Since we opt for a discrete latent action space in SLAC, we want our downstream learning algorithm to support discrete actions. Unfortunately, vanilla SAC only works for continuous actions due to the need to backpropagate through the action vector during policy update. While there exist off-policy algorithms that support discrete action spaces (e.g. DQN [57], Discrete-SAC [58]), they typically require enumerating the Q function for all possible actions and do not work for combinatorially large discrete action spaces (e.g. the latent action space of SLAC).

Instead, FLA-SAC extends SAC to large discrete action spaces by using the gumbel-softmax trick [59], which allows us to compute gradients through discrete random variables via reparametrization. Specifically, we used the non-straight-through Gumbel-softmax estimation shown in Eq. 6 for sampling actions, with a fixed temperature τ of 1.0, which we empirically found to give good performance even when the size of the action space is as large as $\mathcal{O}(10^6)$.

$$\hat{z}(s) = softmax\left(\frac{\log \pi_\theta(z \mid s) + g_z}{\tau}\right), \quad g_z \sim Gumbel(0, 1) \tag{6}$$

Factored Q-Function Decomposition: Challenging robotics tasks often come with a naturally *composite* reward function, where the eventual reward is the sum of a set

TABLE I: We compare the Success rates (↑) over 10 rollouts and the total safety violation counts during training (↓) of SLAC against baseline methods across four tasks. In all four tasks, SLAC achieves the highest success rate while also inducing the least number of safety violations.

Method / Task	Board		Board-Obstacle		Table-Tray		Table-Bag	
	Success	Unsafe #	Success	Unsafe #	Success	Unsafe #	Success	Unsafe #
SLAC (ours)	**0.9**	**1**	**0.8**	**4**	**0.9**	**0**	**0.7**	**0**
SERL [20]	0.0	8	0.0	22	0.0	6	0.0	9
Sim2Real [15]	0.2	-	0.2	-	0.4	-	0.0	-
RLPD [53]	0.4	34	0.2	37	0.3	26	0.0	33

of reward terms corresponding to a set of sub-objectives, e.g. a whole-body mobile manipulation task may require: (1) navigating to a location, (2) without colliding with obstacles, and (3) while holding an object at the right orientation. Directly optimizing this complex reward function with vanilla RL can be quite challenging, often requiring many steps of environment interactions, posing a significant challenge to real-world learning. Our key realization is that we can take advantage of the **disentangled** nature of the learned latent action space $\mathcal{Z}$, by finding and exploiting the strong dependencies between the latent action dimensions and the reward terms. Specifically, given a composite reward function with m terms[4]: $r_{task} = \sum_{i=1}^{m} r_i$, it can be shown (proof in the appendix) that $Q_\pi(s,z) = \sum_{i=1}^{m} Q_\pi^i(s,z)$, where each factored value function Q_π^i represents the expected return for a specific reward term r_i. Now, since each dimension z_i of our latent action z is trained to control and only control one environment entity, each reward term r_i typically only depends on a *small subset* of the latent action dimensions (for example, a reward for navigation is only associated with the latent action dimension that controls the robot base location). This property allows us to dissociate each Q_π^i with unrelated latent action dimensions, resulting in $Q_\pi^i(s, z_{\mathcal{I}_i})$, where $\mathcal{I}_i \subseteq \{1, \ldots, \dim(z)\}$ is the index set corresponding to the dimensions of z that reward r_i depends on.

Such a factorization brings us two significant advantages: First, it effectively prevents Q_π^i from learning spurious correlations with actions, thereby providing a better optimization landscape for accurate prediction of the Q value. Second, since we calculate the policy update by backpropagating from the Q functions, each action dimension will only be updated with reward terms that it can affect, resulting in a more accurate policy gradient estimation than the non-factorized version. In other words, our technique effectively decomposes a hard learning problem into multiple simpler problems that can be solved in parallel, leading to improved performance and sample efficiency. In practice, we implement the Q-decomposition by masking out irrelevant action dimensions for each Q_π^i using a binary adjacency matrix $\mathcal{B}$ that encodes action-reward dependencies. This implementation enables parallel computation of all Q-functions, significantly accelerating training. The matrix $\mathcal{B}$ can either be learned automatically from a small number of random interaction trajectories [10], or manually specified when the mapping is known *a priori* [13]. In SLAC, we assume ground-truth access to this mapping, leveraging the fact that each latent action dimension is intentionally constructed to control a specific environment entity (see Sec. III-A).

[4]Note that SLAC is still valid with a non-composite reward (i.e. $m = 1$) and without Q decomposition.

IV. Experimental Results

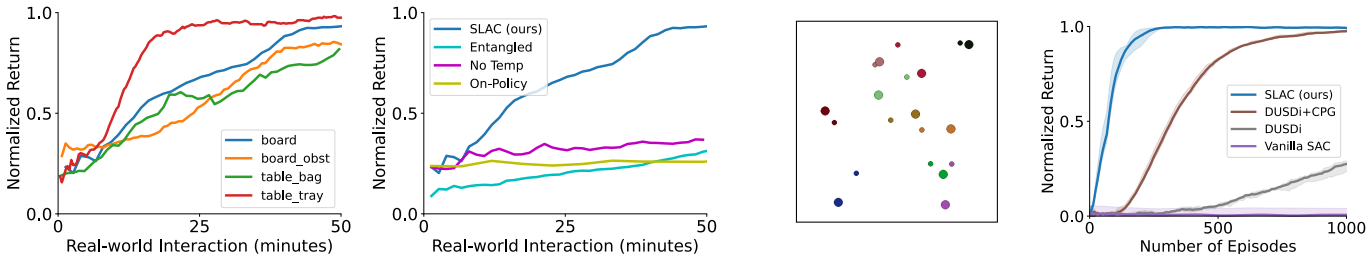
Our primary experiments evaluate SLAC on a bi-manual mobile manipulator – a domain that is ideally suited to our approach. This domain is not only highly challenging due to the complexity of the embodiment and task space, but also practically important for the development of capable household robots. Specifically, we evaluate our methods in two different environments: a table environment, where the robot faces a table with objects on it, and a whiteboard environment where the robot can interact with a whiteboard. Each simulation environment takes less than 20 minutes to create in iGibson [60], using off-the-shelf object models without any real2sim. In each environment (shown in Fig. 1, described in detail in the appendix), we evaluate multiple different **visuomotor contact-rich** tasks that require **whole-body** motion to solve, and present the results (Sec. IV-A) and ablations (Sec. IV-B). Finally, we present additional results in Sec. IV-C on a multi-agent domain, demonstrating the broad applicability of our method.

A. Training Setup & Results

Observations & Network: In all tasks, the observation of the robot consists **only** of an RGBD camera observation and robot proprioception. The pointcloud is first processed by a PointNet [61] and then passed into an MLP network along with the proprioceptive data. Both networks are randomly initialized and trained from scratch. We show the detailed hyperparameters in the appendix.

Baselines: We compare the performance of our method against state-of-the-art methods in realworld RL, sim2real RL, and realworld finetuning. Specifically, we compare against:

- **SERL** [20], a state-of-the-art real-world Reinforcement Learning framework that directly train a policy from scratch in the low-level action space using regularized SAC.
- **Zero-shot Sim2Real** [15], a task policy trained in sim is directly applied to the real world.



(a) SLAC training curves (b) Ablations for SLAC (c) SLAC on the Multi-Particle Domain

Fig. 3: Training curves for SLAC. SLAC can learn contact-rich whole-body manipulation tasks within an hour of real-world interactions (Fig. 3a), and can be applied to non-robotics domains as well (Fig. 3c). Ablation (Fig. 3b) shows that all the techniques in SLAC are critical to its success.

- **RLPD** [53], a state-of-the-art method for learning from both online and prior data, which we use for finetuning in the real world with prior data from simulation.

Notice that both Zero-shot Sim2Real and RLPD have an unfair advantage over our method, as they require implementing downstream task reward and objects (e.g. marker trace) in simulation. For contact-rich tasks, these objects are often quite hard to create in simulation. By comparison, our method does not require implementing the downstream tasks in simulation, since we are only learning a task-agnostic action space.

Metric and Results: For each method on each downstream task, we compare the success rate of the final policy across 10 rollouts with different initial states. For the three methods that require training in simulation (SLAC, Sim2Real, RLPD), we train for 10M steps in simulation for each task. For the three methods that require real-world interactions (SLAC, SERL, RLPD), we train each of them for 30k steps of real-world low-level robot actions, corresponding to 50 minutes of real-world interactions, and additionally report the number of times they have violated the safety constraints during training. The full result is shown in Table I. In all four tasks, SLAC learns to solve the task in less than an hour of real-world interactions (curves in Fig. 3a), while maintaining safety during real-world exploration, significantly outperforming the baselines.

B. Ablations

To study the effectiveness of each component of our method, we conduct ablation studies comparing against the following variations of our method:

- **No Disentanglement**: where we remove the disentangled constraint during latent action space learning. As a result, our latent action space is no longer factored, and we can no longer apply Q-Function Decomposition during downstream learning since now all reward terms depend on the entire latent action vector.
- **On-Policy**: where we replace our proposed FLA-SAC with PPO[49], a state-of-the-art on-policy RL algorithm that has achieved many successes in Sim2Real RL.
- **Not Temporally Extended**: where the task policy makes decisions at the same frequency as the latent action decoder (i.e. 10hz).

We report the training curve for each of these variances of our method on the Board task in Fig. 3b. We can see that removing any single component of SLAC results in a significant decrease in the learning efficiency. Note that for some of these variants (e.g., the on-policy version), it is likely that they would eventually achieve good performance with enough training steps. However, given the high cost of real-world interactions, it is not practical to evaluate this likelihood.

C. Applications to Non-Robotics Domain

Since SLAC is an embodiment-agnostic framework that does not require domain knowledge, we can in principle apply SLAC to any robots and even beyond robotics. We briefly illustrate this point on the Multi-Particle domain [62]. We follow the task setup of *food-poison-hard* [25], where a centralized controller needs to simultaneously control 10 agents to interact with different landmarks, and report the results in Fig. 3c. On this challenging task where learning from scratch completely fails, SLAC successfully learns policies that match the final performance of the previous state of the art [25], while being an order of magnitude more sample efficient. This result illustrates the broad applicability of SLAC.

V. Conclusion

This paper introduced SLAC, a framework for enabling high DoF robots to learn policy directly in the real world, by leveraging a latent action space trained in a low-fidelity simulation. SLAC learns this latent action space through unsupervised skill discovery, and employs a novel sample-efficient RL algorithm to learn task policy in the SLAC latent action space. Evaluated on a set of contact-rich whole-body manipulation tasks, SLAC is able to solve the tasks in under an hour of real-world interaction, where baseline methods failed. SLAC opens up new opportunities for advancing both latent action space learning (e.g., through improved skill discovery methods) and downstream policy learning (e.g., by leveraging or learning a world model). We believe SLAC provides a strong foundation for scaling real-world robot learning to increasingly complex and diverse tasks and embodiments, and discuss the limitations and future directions in the appendix.

References

[1] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[2] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[3] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[4] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani, "Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning," *arXiv preprint arXiv:2409.16578*, 2024.

[5] D. Honerkamp, T. Welschehold, and A. Valada, "n^2m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *arXiv preprint arXiv:2206.08737*, 2022.

[6] R. Yang, Y. Kim, A. Kembhavi, X. Wang, and K. Ehsani, "Harmonic mobile manipulation," *arXiv preprint arXiv:2312.06639*, 2023.

[7] N. Yokoyama, A. W. Clegg, E. Undersander, S. Ha, D. Batra, and A. Rai, "Adaptive skill coordination for robotic mobile manipulation," *arXiv preprint arXiv:2304.00410*, 2023.

[8] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 149–12 155.

[9] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *IEEE Robotics and Automation Letters*, 2022.

[10] J. Hu, P. Stone, and R. Martín-Martín, "Causal Policy Gradient for Whole-Body Mobile Manipulation," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[11] T. Li, J. Truong, J. Yang, A. Clegg, A. Rai, S. Ha, and X. Puig, "Robotmover: Learning to move large objects by imitating the dynamic chain," *arXiv preprint arXiv:2502.05271*, 2025.

[12] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, 2024.

[13] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*. PMLR, 2023, pp. 138–149.

[14] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan *et al.*, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," *arXiv preprint arXiv:2502.01143*, 2025.

[15] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[16] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 80–93.

[17] Y. Liu, H. Xu, D. Liu, and L. Wang, "A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102365, 2022.

[18] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6664–6671.

[19] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," *arXiv preprint arXiv:2208.07860*, 2022.

[20] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine, "Serl: A software suite for sample-efficient robotic reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 16 961–16 969.

[21] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn, "Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4804–4811.

[22] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," *arXiv preprint arXiv:1802.06070*, 2018.

[23] Z. Wang, J. Hu, C. Chuck, S. Chen, R. Martín-Martín, A. Zhang, S. Niekum, and P. Stone, "Skild: Unsupervised skill discovery guided by factor interactions," *arXiv preprint arXiv:2410.18416*, 2024.

[24] S. Park, K. Lee, Y. Lee, and P. Abbeel, "Controllability-aware unsupervised skill discovery," 2023. [Online]. Available: https://arxiv.org/abs/2302.05103

[25] J. Hu, Z. Wang, P. Stone, and R. Martín-Martín, "Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 76 529–76 552, 2024.

[26] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak, "Adaptive mobile manipulation for articulated objects in the open world," 2024.

[27] A. Herzog, K. Rao, K. Hausman, Y. Lu, P. Wohlhart, M. Yan, J. Lin, M. G. Arenas, T. Xiao, D. Kappler, D. Ho, J. Rettinghouse, Y. Chebotar, K.-H. Lee, K. Gopalakrishnan, R. Julian, A. Li, C. K. Fu, B. Wei, S. Ramesh, K. Holden, K. Kleiven, D. Rendleman, S. Kirmani, J. Bingham, J. Weisz, Y. Xu, W. Lu, M. Bennice, C. Fong, D. Do, J. Lam, Y. Bai, B. Holson, M. Quinlan, N. Brown, M. Kalakrishnan, J. Ibarz, P. Pastor, and S. Levine, "Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators," 2023.

[28] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully autonomous real-world reinforcement learning with applications to mobile manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 308–319.

[29] R. Mendonca, E. Panov, B. Bucher, J. Wang, and D. Pathak, "Continuously improving mobile manipulation with autonomous real-world rl," *arXiv preprint arXiv:2409.20568*, 2024.

[30] S. Dass, W. Ai, Y. Jiang, S. Singh, J. Hu, R. Zhang, P. Stone, B. Abbatematteo, and R. Martín-Martín, "Telemoma: A modular and versatile teleoperation system for mobile manipulation," *arXiv preprint arXiv:2403.07869*, 2024.

[31] Y. Jiang, R. Zhang, J. Wong, C. Wang, Y. Ze, H. Yin, C. Gokmen, S. Song, J. Wu, and L. Fei-Fei, "Behavior robot suite: Streamlining real-world whole-body manipulation for everyday household activities," *arXiv preprint arXiv: 2503.05652*, 2025.

[32] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[33] J. Li, Y. Zhu, Y. Xie, Z. Jiang, M. Seo, G. Pavlakos, and Y. Zhu, "Okami: Teaching humanoid robots manipulation skills through single video imitation," in *8th Annual Conference on Robot Learning*, 2024.

[34] H. Seraji, "A unified approach to motion control of mobile manipulators," *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 107–118, 1998.

[35] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," in *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*. IEEE, 1992, pp. 2643–2648.

[36] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *ICRA*, 2006, pp. 2641–2648.

[37] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 20–33, 2012.

[38] E. Papadopoulos and J. Poulakakis, "Planning and model-based control for mobile manipulators," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 1810–1815.

[39] J. Haviland, N. Sünderhauf, and P. Corke, "A holistic approach to reactive mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3122–3129, 2022.

[40] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE RAL*, vol. 5, no. 4, pp. 6177–6184, 2020.

[41] Q. Huang, K. Tanie, and S. Sugano, "Coordinated motion planning for a mobile manipulator considering stability and manipulation," *The International Journal of Robotics Research*, vol. 19, no. 8, pp. 732–742, 2000.

[42] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[43] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[44] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[45] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[46] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1656–1662.

[47] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *Twentieth international conference on automated planning and scheduling (ICAPS)*, 2010.

[48] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[50] R. C. Julian, E. Heiden, Z. He, H. Zhang, S. Schaal, J. J. Lim, G. S. Sukhatme, and K. Hausman, "Scaling simulation-to-real transfer by learning a latent space of robot skills," *The International Journal of Robotics*

Research, vol. 39, no. 10-11, pp. 1259–1278, 2020.

[51] J. Zhang, M. Heo, Z. Liu, E. Biyik, J. J. Lim, Y. Liu, and R. Fakoor, "Extract: Efficient policy learning by extracting transferable robot skills from offline data," 2024. [Online]. Available: https://arxiv.org/abs/2406.17768

[52] P. Yin, T. Westenbroek, S. Bagaria, K. Huang, C. an Cheng, A. Kobolov, and A. Gupta, "Rapidly adapting policies to the real world via simulation-guided fine-tuning," 2025. [Online]. Available: https://arxiv.org/abs/2502.02705

[53] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning*. PMLR, 2023.

[54] D. Barber and F. Agakov, "Information maximization in noisy channels: A variational approach," *Advances in Neural Information Processing Systems*, vol. 16, 2003.

[55] J. Choi, A. Sharma, H. Lee, S. Levine, and S. S. Gu, "Variational empowerment as representation learning for goal-conditioned reinforcement learning," in *International conference on machine learning*. PMLR, 2021, pp. 1953–1963.

[56] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv:1812.05905*, 2018.

[57] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[58] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[59] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[60] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain *et al.*, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Conference on Robot Learning*. PMLR, 2022, pp. 455–465.

[61] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1612.00593

[62] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente *et al.*, "Pettingzoo: Gym for multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 032–15 043, 2021.

[63] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI conference on artificial intelligence*, 2017.

[64] M. FISCHLER AND, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

APPENDIX

A. Policy and Training Visualizations

We encourage the reader to visit our project website (https://robo-rl.github.io/) for full videos of the SLAC training process in the real world and the learned policies.

B. Limitations and Future Works

Despite its strong empirical performance, SLAC is not without limitations. First, SLAC introduces an implicit tradeoff related to the granularity of the latent action space. For example, an identity mapping between the robot's raw action space and latent action space would make the downstream task policy capable of learning any task within the original capability of the robot, but would significantly reduce the sample efficiency (as shown in our results in Sec. IV-A). Similarly, the temporal length of each latent action entails another tradeoff, where shorter latent actions give more control to the task policy at the expense of longer task horizons, which may hamper learning (as shown in our ablation studies in Sec. IV-B). It is likely that the optimality of the latent action space will be strongly task-dependent. Second, in the current SLAC framework, the latent action decoder is kept fixed during downstream learning. However, for more fine-grained tasks, we might benefit from finetuning the latent action decoder while training the task policy. While this is conceptually doable via the option framework [63], we leave the empirical study of how to implement it in a stable manner to future work. Finally, while this paper primarily focuses on the algorithmic side of real-world learning, we expect future engineering efforts in the automation of task reset and downstream reward generation (e.g., via VLM/LLM or a learned reward function) to further boost the downstream learning efficiency.

C. Q Decomposition Proof

We provide proof the equation $Q_\pi(s,z) = \sum_{i=1}^{m} Q_\pi^i(s,z)$ discussed in Sec. III-B, using the linearity of expectations:

Proof.

$$\begin{aligned} Q_\pi(s,z) &= \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t] \\ &= \mathbb{E}_\pi[\sum_{i=1}^{m}\sum_{t=0}^{\infty} \gamma^t r_t^i] \\ &= \sum_{i=1}^{m} \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t^i] \\ &= \sum_{i=1}^{m} Q_\pi^i(s,z) \end{aligned}$$

□

D. Properties of the Learned Latent Action Space

Here, we discuss in detail the properties of the learned SLAC latent action space. In short, the latent action space of SLAC is **environment-aware but task-agnostic**. It is task-agnostic because it is trained without a task reward, and is only encouraged to induce diversity in behavior following the USD objective. Therefore, **the same latent action space can tackle different tasks within a particular environment** (e.g., the "push to tray" and "sweep to bag" tasks in our experiment utilize the same latent action space). On the other hand, the latent action space is environment-aware because it is trained to induce diverse behavior *in a particular scene* in simulation. Note that our latent action space is robust to small variance in the environment: for example, the action space learned in the board environment can be used to learn policies that wipe marks and avoid the trash can, even though there is neither a trash can nor wipeable marks in the simulated board environment.

E. Universal Safety Reward

In SLAC, we employ a universal safety reward for ensuring that the learned latent action space is safe. This reward is the same across all our environments, and is defined as follows:

$$r_{safe} = -\lambda_1\|a\|^2 - \lambda_2\|a - a_{\text{prev}}\|^2 - \lambda_3 \cdot \mathbb{I}_{\text{collision}} - \lambda_4 \cdot \mathbb{I}_{F>70} \quad (7)$$

In our experiments, we set $\lambda_1 = 0.01$, $\lambda_2 = 0.1$, $\lambda_3 = 0.2$, and $\lambda_4 = 0.05$. Additionally, we incorporate a shaping reward that encourages the robot to stay close to the board / table and find it to speed up training.

F. SLAC Unsupervised Latent Action Space Learning Pseudocode

Here, we omit standard SAC steps, such as target network creation and update, for simplicity.

G. SLAC Real-World Downstream RL Pseudocode

Again, we omit standard SAC steps, such as target network creation and update, for simplicity.

H. Hyperparameter

Here, we present the hyperparameter for both the latent action decoder training and the downstream task learning. The same hyperparamters are shared across all tasks. We use a low-level step size $steps\_per\_skill = 50$ for all our experiments.

I. Discrete vs Continuous Latent Actions

The SLAC framework supports both discrete and continuous latent action. In our experiments, we made a deliberate choice to use a discrete latent action space since a discrete latent action space encodes a **compact** set of distinguishable behaviors, making it more amenable to hierarchical downstream RL. This is also a standard design choice of previous unsupervised skill discovery methods [22, 24, 25].

Algorithm 1: SLAC Unsupervised Latent Action Space Learning

Initialize sim environment, skill prior distribution $p(z)$, replay buffer $\mathcal{D}_{sk}$;
Initialize latent action decoder π_{dec}, discriminators q_ϕ, q_ψ, and value function Q_{dec};
for $k \leftarrow 1$ **to** $skill\_learning\_epochs$ **do**
  Sample skill $z \sim p(z)$;
  for $j \leftarrow 1$ **to** $steps\_per\_skill$ **do**
    $(o_{dec}, a, {o'}_{dec}) \leftarrow \texttt{sim.step}(\pi_{dec}(a \mid o_{dec}, z))$;
    Store transition $(o_{dec}, a, z, {o'}_{dec})$ into replay buffer $\mathcal{D}_{sk}$;
    for $i \leftarrow 1$ **to** $n\_updates$ **do**
      Sample mini-batch $\{(o_{dec}, a, z, {o'}_{dec}\}$ from $\mathcal{D}_{sk}$;
      [Optional] Update q_ϕ and q_ψ;
      Calculate intrinsic reward r based on Eq. 5;
      Update Q_{dec} with r using SAC critic update;
      Update π_{dec} with Q_{dec} using SAC policy update;
return π_{dec}

Algorithm 2: FLA-SAC for Real-World Downstream Task Learning

Initialize replay buffer $\mathcal{D}$, task policy $\pi_{task}(z \mid o)$, factored Q-functions $\{Q^i\}_{i=1}^m$;
Load pre-trained latent action decoder π_{dec}, binary dependency matrix $\mathcal{B}$;
for $k \leftarrow 1$ **to** $task\_learning\_steps$ **do**
  $z \leftarrow \pi_{task}(z \mid o)$;
  $r_{\text{sum}} \leftarrow [0]^m$;
  for $t \leftarrow 1$ **to** $steps\_per\_skill$ **do**
    $(o_{\text{dec}}, r = [r^i]_{i=1}^m, o') \leftarrow \texttt{robot.real\_world\_step}(\pi_{dec}(a \mid o_{\text{dec}}, z))$;
    $r_{\text{sum}} = r_{\text{sum}} + r$;
  Store $(o, z, r_{\text{sum}}, o')$ into replay buffer $\mathcal{D}$;
  for $j \leftarrow 1$ **to** utd_ratio **do**
    Sample mini-batch $\{(o, z, r, o')\}$ from $\mathcal{D}$ with **small batch size**;
    Update $Q^i(o, \mathcal{B}_i \odot z)$ with r_i **in parallel** for all $i = 1, \ldots, m$;
    Update $\pi_{task}(\hat{z} \mid o)$ with $Q = \sum_i Q^i$ using SAC loss, with $\hat{z}$ sampled via **Gumbel-Softmax** (Eq. 6) for differentiability;
return π_{task}

TABLE II: Hyperparameters of Latent Action Decoder Learning.

	Name	Value
SAC	optimizer	Adam
	activation functions	ReLu
	learning rate	1×10^{-4}
	batch size	1024
	critic target τ	0.01
	MLP size	[1024, 1024]
	n updates	2
	# of environments	16
	entropy coefficient α	0.0
	log std bounds	[-10, 2]
	warmup samples	24000
	latent action dimension	4^5

TABLE III: Hyperparameters of Downstream Learning.

	Name	Value
FLA-SAC	optimizer	Adam
	activation functions	ReLu
	learning rate	4×10^{-4}
	batch size	64
	critic target τ	0.05
	MLP size	[256, 256]
	utd ratio	10
	# of environments	1
	entropy coefficient α	0.1
	log std bounds	[-10, 2]
	warmup samples	60
	gumbel temperature	1.0

J. Environment Description

In this section, we describe the two mobile manipulation environments that we tested SLAC on. In each environment, we apply our method to solve two different downstream tasks. We visualize the environments and the downstream tasks in Fig. 1. In both environments, the robot has a 17-dimensional action space, corresponding to base velocity (3d), head camera joint position (2d), right end-effector delta pose (6d) and left end-effector delta pose (6d). The observation space of the task policy consists of a 320×240 RGBD image that is segmented and down-sampled to 50 points, and a 7-dimensional vector corresponding to the proprioceptive data. For all downstream tasks, we employ a relatively **sparse** reward that is only given at the end of a high-level policy step.

1) Board Environment: **Simulation** In the board environment, the robot is initialized in front of an interactable whiteboard. The decoder observation o_{dec} consists of the proprioceptive data of the robot, the robot's previous action, the robot's distance and relative orientation with the board (which we estimate in the real world via a simple RANSAC line detector [64]), and the end effector's contact history with the board. The latent action space is trained to maximize empowerment for the following state entities: board contact history, robot base position, robot view, and board contact force.

Downstream Task 1: Clean Whiteboard The *Clean Whiteboard* task requires the robot to identify the location of the board that needs to be wiped, and then use a sponge to clean up the identified region. Specifically, we define a composite task reward function with the following terms: 1) Encourage the robot to look at the target marker to wipe. 2) Encourage the successful removal of the target marker. 3) Encourage the robot to move towards the target marker. 4) Penalize large contact forces and any collision. The task is considered successful if all four conditions are successfully achieved.

Downstream Task 2: Wipe Board over Obstacles The *Wipe over Obstacles* task is conceptually similar to the *Clean Whiteboard* task, except that now there is an obstacle between the robot and the board. Thus, the robot needs to additionally learn to keep a reasonable distance from the obstacle and still be able to wipe the mark. The reward function is the same as *Clean Whiteboard*.

2) Table Environment: **Simulation** In the table environment, the robot is initialized in front of a table. The table is not interactive, but would incur a penalty if the robot collides with it. The decoder observation o_{dec} consists of the proprioceptive data of the robot, the robot's previous action, and the robot's distance and relative orientation with the table (which we again estimate in the real world via RANSAC [64]). The latent action space is trained to maximize empowerment for the following state entities: robot left and right eef position relative to the table, robot base position, and robot view.

Downstream Task 3: Push Garbage on the Table to the Tray The *Push to Tray* task requires the robot to push some garbage on the table in a tray that is also placed on the table. The reward function consists of the following terms: 1) Encourage the robot to look at the location of the garbage. 2) Encourage successfully pushing the garbage into the tray. 3) Encourage the robot to move towards the garbage. 4) Penalize large contact forces and any collision.

Downstream Task 4: Sweep Garbage from the Table to the Bag For the *Sweep to Bag* task, the robot is initialized with a bag in its left gripper. The goal of the task is to sweep the garbage into the bag, which requires the coordinative control of both robot arms and the base. The reward function consists of the following terms: 1) Encourage the robot to look at the location of the garbage. 2) Encourage successfully pushing the garbage off the table. 3) Encourage the robot to move its base towards the garbage. 4) Encourage the bag to be close to the garbage. 5) Penalize large contact forces and any collision.

K. Discussion of Broader Areas Related to SLAC

In this section, we discuss SLAC's relation to two additional areas: planning & control, and learning from demonstrations.

Classical Motion Planning and Control: A traditional way to enable robots to perform tasks is through motion planning and control. In practice, however, uncertainty and inaccuracy in localization frequently impede the accurate execution of planned trajectories [44–48]. Moreover, when the robot needs to consider multiple objectives, creating a motion planner is even harder, as it requires solving complex multi-objective optimization problems [41–43]. On the side of control, existing methods [34–40] resort to sophisticated prioritized solutions that require extensive tuning and pre-determined task priorities. Moreover, these methods assume accurate models of the robot and the environment, which often break in unstructured environments and with high-dimensional sensor signals (e.g. images). By comparison, SLAC can autonomously learn closed-loop policies only based on onboard sensors, and does not require prior domain knowledge.

Learning from Demonstrations: Recently, learning from demonstration has gained popularity as a powerful paradigm for learning robot behaviors, particularly for tabletop manipulation [1–3]. As robot systems get more and more complex, however, collecting high-quality data can quickly become challenging due to high-degree-of-freedom embodiments that require coordinated control. Even with carefully designed systems that only work for very specific embodiments [30–33], getting enough data for imitation learning remains hard and costly, especially for dynamic and contact-rich tasks. SLAC does not require **any** demonstrations, and potentially can be applied to learn a wide range of tasks through autonomous interactions with the environment.