Controllability-Aware Unsupervised Skill Discovery

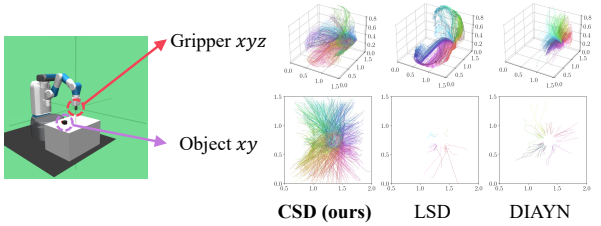
Seohong Park [1] **Kimin Lee** [2] **Youngwoon Lee** [1] **Pieter Abbeel** [1]

Abstract

One of the key capabilities of intelligent agents is the ability to discover useful skills without external supervision. However, the current unsupervised skill discovery methods are often limited to acquiring simple, easy-to-learn skills due to the lack of incentives to discover more complex, challenging behaviors. We introduce a novel unsupervised skill discovery method, **Controllability-aware Skill Discovery** (**CSD**), which actively seeks complex, hard-to-control skills without supervision. The key component of CSD is a controllability-aware distance function, which assigns larger values to state transitions that are harder to achieve with the current skills. Combined with distance-maximizing skill discovery, CSD progressively learns more challenging skills over the course of training as our jointly trained distance function reduces rewards for easy-to-achieve skills. Our experimental results in six robotic manipulation and locomotion environments demonstrate that CSD can discover diverse complex skills including object manipulation and locomotion skills with no supervision, significantly outperforming prior unsupervised skill discovery methods. Videos and code are available at https://seohong.me/projects/csd/

1. Introduction

Humans are capable of *autonomously* learning skills, ranging from basic muscle control to complex acrobatic behaviors, which can be later combined to achieve highly complex tasks. Can machines similarly discover useful skills without any external supervision? Recently, many unsupervised skill discovery methods have been proposed to discover diverse behaviors in the absence of extrinsic rewards (Gregor et al., 2016; Eysenbach et al., 2019; Sharma et al., 2020; Achiam et al., 2018; Campos Camúñez et al., 2020; Hansen et al., 2020; Kim et al., 2021; Liu & Abbeel, 2021a; Park et al., 2022; Laskin et al., 2022). These methods have also demonstrated efficient downstream reinforcement learning (RL) either by fine-tuning (Laskin et al., 2021; 2022) or sequentially combining (Eysenbach et al., 2019; Sharma et al., 2020; Park et al., 2022) the discovered skills.



(a) FetchPush (b) Skill trajectories

Figure 1. Object trajectories and gripper trajectories of 2-D continuous skills discovered by three unsupervised skill discovery methods, CSD (ours), LSD (Park et al., 2022), and DIAYN (Eysenbach et al., 2019), in the FetchPush environment. Trajectories with different colors represent different skills. While previous methods focus only on maneuvering the gripper, CSD discovers object manipulation skills in the absence of supervision.

However, in complex environments, current unsupervised skill discovery methods are often limited to discovering only simple, easy-to-learn skills. For example, as illustrated in Figure 1, previous approaches (LSD and DIAYN) only learn to gain control of the agent's own 'body' (*i.e.*, the gripper and joint angles), completely ignoring the object in the Fetch environment. This is because learning difficult skills, such as interacting with the object, has no incentive for them compared to learning easy skills. In other words, their objectives can be fully optimized with simple skills.

To mitigate this issue, prior approaches incorporate human supervision, such as limiting the agent's focus to specific dimensions of the state space of interest (Eysenbach et al., 2019; Sharma et al., 2020; Park et al., 2022; Adeniji et al., 2022). However, this not only requires manual feature engineering but also significantly limits the diversity of skills. On the other hand, we humans consistently challenge ourselves to learn more complex skills after mastering simple skills in an autonomous manner.

Inspired by this, we propose a novel unsupervised skill discovery method, **Controllability-aware Skill Discovery** (**CSD**), which explicitly seeks complex, hard-to-learn behaviors that are potentially more useful for solving down-

[1]University of California, Berkeley [2]Google Research. Correspondence to: Seohong Park <seohong@berkeley.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

stream tasks. Our key idea is to train a controllability-aware distance function based on the current skill repertoire and combine it with distance-maximizing skill discovery. Specifically, we train the controllability-aware distance function to assign larger values to harder-to-achieve state transitions and smaller values to easier-to-achieve transitions with the current skills. Since CSD aims to maximize this controllability-aware distance, it autonomously learns increasingly complex skills over the course of training. We highlight that, to the best of our knowledge, CSD is the first unsupervised skill discovery method that demonstrates diverse object manipulation skills in the Fetch environment without any external supervision or manual feature engineering (*e.g.*, limiting the focus only to the object).

To summarize, the main contribution of this work is to propose CSD, a novel unsupervised skill discovery method built upon the notion of controllability. We also formulate a general distance-maximizing skill discovery approach to incorporate our controllability-aware distance function with skill discovery. We empirically demonstrate that CSD discovers various complex behaviors, such as object manipulation skills, with no supervision, outperforming previous state-of-the-art skill discovery methods in diverse robotic manipulation and locomotion environments.

2. Preliminaries

Unsupervised skill discovery aims at finding a potentially useful set of skills without external rewards. Formally, we consider a reward-free Markov decision process (MDP) defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, p)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, $\mu : \mathcal{P}(\mathcal{S})$ is the initial state distribution, and $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition dynamics function. Each skill is defined as a skill latent vector $z \in \mathcal{Z}$ and a skill-conditioned policy $\pi(a|s, z)$ that is shared across the skills. The skill space $\mathcal{Z}$ can be either discrete skills ($\{1, 2, \dots, D\}$) or continuous skills ($\mathbb{R}^D$).

To collect a skill trajectory (behavior), we sample a skill z from a predefined skill prior distribution $p(z)$ at the beginning of an episode. We then roll out the skill policy $\pi(a|s, z)$ with the sampled z for the entire episode. For the skill prior, we use a standard normal distribution for continuous skills and a uniform distribution for discrete skills.

Throughout the paper, $I(\cdot;\cdot)$ denotes the mutual information and $H(\cdot)$ denotes either the Shannon entropy or differential entropy depending on the context. We use uppercase letters for random variables and lowercase letters for their values (*e.g.*, S denotes the random variable for states s).

3. Related Work

In this section, we mainly discuss closely related prior unsupervised skill discovery work based on mutual information maximization or Euclidean distance maximization. A more extensive literature survey on unsupervised skill discovery and unsupervised RL can be found in Appendix A.

3.1. Mutual Information-Based Skill Discovery

Mutual information-based unsupervised skill discovery maximizes the mutual information (MI) between states S and skills Z, $I(S; Z)$, which associates different states with different skill latent vectors so that the behaviors from different zs are diverse and distinguishable. Since computing exact MI is intractable, previous MI-based methods approximate MI in diverse ways, which can be categorized into reverse-MI and forward-MI (Campos Camúñez et al., 2020).

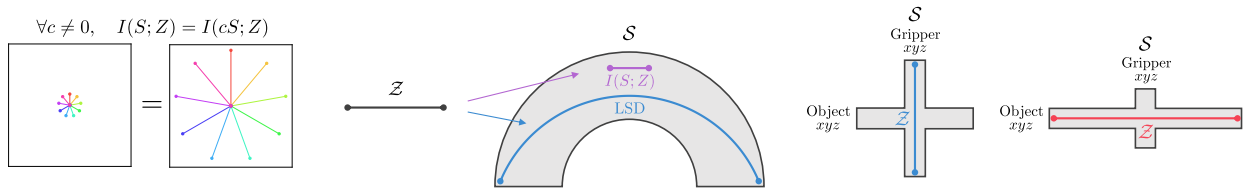
First, reverse-MI approaches (Gregor et al., 2016; Eysenbach et al., 2019; Achiam et al., 2018; Hansen et al., 2020) optimize MI in the form of $I(S; Z) = H(Z) - H(Z|S)$, where $H(Z)$ is a constant as we assume that the skill prior distribution $p(z)$ is fixed. Thus, maximizing $I(S; Z)$ corresponds to minimizing $H(Z|S)$, which can be approximated with a variational distribution $q_\theta(z|s)$. For instance, DIAYN (Eysenbach et al., 2019) maximizes the variational lower bound of MI as follows:

$$
\begin{aligned}
I(S; Z) &= -H(Z|S) + H(Z) && (1)\\
&= \mathbb{E}_{z,s}[\log p(z|s)] - \mathbb{E}_z[\log p(z)] && (2)\\
&\geq \mathbb{E}_{z,s}[\log q_\theta(z|s)] + (\text{const}), && (3)
\end{aligned}
$$

where $q_\theta(z|s)$ is a variational approximation of $p(z|s)$ (Barber & Agakov, 2003). Intuitively, $q_\theta(z|s)$ works as a 'skill discriminator' that tries to infer the original skill z from the state s, encouraging the skill policy to generate distinguishable skill trajectories for different zs (*i.e.*, diverse skills). Other reverse-MI methods optimize the MI objective similarly but computing MI on entire trajectories (Achiam et al., 2018) or only on final states (Gregor et al., 2016) rather than all intermediate states, or using von Mises-Fisher distributions (Hansen et al., 2020) for the skill prior distribution instead of Gaussian or uniform distributions.

On the other hand, forward-MI approaches (Sharma et al., 2020; Campos Camúñez et al., 2020; Liu & Abbeel, 2021a; Laskin et al., 2022) employ the other decomposition of MI: $I(S; Z) = H(S) - H(S|Z)$. This decomposition explicitly maximizes the state entropy $H(S)$, which helps diversify skill trajectories in practice (Laskin et al., 2022). Forward-MI methods minimize the $H(S|Z)$ term with a variational approximation (Sharma et al., 2020; Liu & Abbeel, 2021a; Campos Camúñez et al., 2020) or a contrastive estimator (Laskin et al., 2022). $H(S)$ can be estimated using a particle-based entropy estimator (Liu & Abbeel, 2021a; Laskin et al., 2022), a state marginal matching objective (Lee et al., 2019; Campos Camúñez et al., 2020), or sampling-based approximation (Sharma et al., 2020).

One major limitation of MI-based approaches is that optimizing the MI objective does not necessarily lead to cov-



(a) Two skill sets having the same MI

(b) Skill space mappings of LSD and the MI objective

(c) LSD (Euclidean distance)

(d) **CSD (ours)** (Controllability-aware distance)

Figure 2. Illustration of unsupervised skill discovery methods. (a) MI is invariant to traveled distances. (b) The MI objective simply seeks *any* mapping between $\mathcal{Z}$ and $\mathcal{S}$, while LSD finds the largest (longest) possible mapping. (c) LSD maximizes the *Euclidean* traveled distance, which can lead to simple or trivial behaviors. (d) Our CSD maximizes the traveled distance with respect to our learned *controllability-aware* distance function that assigns larger values to harder-to-achieve state transitions. This leads to more complex skills that can be useful for downstream tasks.

ering a larger region in the state space. This is because MI is invariant to traveled distances or any invertible transformation (Figure 2a), *i.e.*, $I(S;Z) = I(f(S);Z)$ for any invertible f (Kraskov et al., 2004). Since there is no incentive for the MI objective to further explore the state space, they often end up discovering 'static' skills with limited state coverage (Gu et al., 2021; Park et al., 2022; Laskin et al., 2022).

3.2. Euclidean Distance-Maximizing Skill Discovery

To resolve the limitation of MI-based skill discovery, Park et al. (2022) recently proposed Lipschitz-constrained Skill Discovery (LSD), which aims to not only establish a mapping between Z and S but also maximize the Euclidean traveled distance in the state space for each skill. Specifically, LSD maximizes the state change along the direction specified by the skill z with the following objective:

$$\mathcal{J}^{\text{LSD}} := \mathbb{E}_{z,s,s'}[(\phi(s') - \phi(s))^\top z] \tag{4}$$

$$\text{s.t.} \quad \forall x, y \in \mathcal{S}, \quad \|\phi(x) - \phi(y)\| \leq \|x - y\|, \tag{5}$$

where s' denotes the next state and $\phi : \mathcal{S} \to \mathbb{R}^D$ denotes a mapping function. LSD maximizes Equation (4) with respect to both the policy and ϕ. Intuitively, this objective aims to align the directions of z and $(\phi(s') - \phi(s))$ while maximizing the length $\|\phi(s') - \phi(s)\|$, which leads to an increase in the state difference $\|s' - s\|$ due to the Lipschitz constraint. As illustrated in Figure 2b, LSD finds the largest possible mapping in the state space by maximizing Euclidean traveled distances in the state space in diverse directions, which leads to more 'dynamic' skills. On the other hand, the MI objective finds *any* mapping between the skill space and the state space, being agnostic to the area of the mapped region, which often results in 'static' skills with limited state coverage.

While promising, LSD is still limited in that it maximizes *Euclidean* traveled distances in the state space, which often does not match the behaviors of our interests because the Euclidean distance treats all state dimensions equally. For example, in the Fetch environment in Figure 1, simply diversifying the position and joint angles of the robot arm is sufficient to achieve large Euclidean traveled distances because both the coordinates of the object and the gripper lie in the same Euclidean space (Figure 2c). As such, LSD and any previous MI-based approaches mostly end up learning skills that only diversify the agent's own internal states, ignoring the external states (*e.g.*, object pose).

Instead of maximizing the Euclidean distance, we propose to maximize traveled distances with respect to a learned *controllability-aware distance function* that 'stretches' the axes along hard-to-control states (*e.g.*, objects) and 'contracts' the axes along easy-to-control states (*e.g.*, joint angles), so that maximizing traveled distances results in the discovery of more complex, useful behaviors (Figure 2d).

3.3. Unsupervised Goal-Conditioned RL

Another line of unsupervised RL focuses on discovering a wide range of *goals* and learning corresponding goal-reaching policies, which leads to diverse learned behaviors (Warde-Farley et al., 2019; Pong et al., 2020; Pitis et al., 2020; Mendonca et al., 2021). On the other hand, unsupervised skill discovery, including our approach, (1) focuses on more general behaviors (*e.g.*, running, flipping) not limited to goal-reaching skills, whose behaviors tend to be 'static' (Mendonca et al., 2021; Jiang et al., 2022), and (2) aims to learn a *compact* set of distinguishable skills embedded in a low-dimensional, possibly discrete skill space, rather than finding all possible states, making it more amenable to hierarchical RL by providing a low-dimensional high-level action space (*i.e.*, skill space). While these two lines of approaches are not directly comparable, we provide empirical comparisons and further discussion in Appendix C.

4. Controllability-Aware Skill Discovery

To discover complex, useful skills without extrinsic reward and domain knowledge, we introduce the notion of *controllability*[1] to skill discovery – once an agent discovers

[1]The term *controllability* in this paper describes whether an agent can manipulate hard-to-control states (*e.g.*, external objects) or not, different from the one used in control theory (Ogata et al., 2010).

easy-to-achieve skills, it continuously moves its focus to hard-to-control states and learns more diverse and complex skills. We implement this idea in our Controllability-aware Skill Discovery (CSD) by combining a distance-maximizing skill discovery approach (Section 4.1) with a *jointly* trained controllability-aware distance function (Section 4.2), which enables the agent to find increasingly complex skills over the course of training (Section 4.3).

4.1. General Distance-Maximizing Skill Discovery

As explained in Section 3.2, Euclidean distance-maximizing skill discovery does not necessarily maximize distances along hard-to-control states (*i.e.*, hard-to-achieve skills). To discover more challenging skills, we propose to learn a skill policy with respect to a jointly learned controllability-aware distance function.

To this end, we first present a general **Distance-maximizing Skill Discovery** approach (**DSD**) that can be combined with any arbitrary distance function $d(\cdot,\cdot) : \mathcal{S}\times\mathcal{S} \to \mathbb{R}_0^+$. Specifically, we generalize the Euclidean distance-maximizing skill discovery (Park et al., 2022) by replacing $\|x-y\|$ in Equation (5) with $d(x,y)$ as follows:

$$\mathcal{J}^{\text{DSD}} := \mathbb{E}_{z,s,s'}[(\phi(s') - \phi(s))^\top z] \tag{6}$$
$$\text{s.t.} \quad \forall x, y \in \mathcal{S}, \quad \|\phi(x) - \phi(y)\| \le d(x,y), \tag{7}$$

where $\phi(\cdot) : \mathcal{S} \to \mathbb{R}^D$ is a function that maps states into a D-dimensional space (which has the same dimensionality as the skill space). DSD can discover skills that maximize the traveled distance under the given distance function d in diverse directions by (1) aligning the directions of z and $(\phi(s') - \phi(s))$ and (2) maximizing its length $\|\phi(s') - \phi(s)\|$, which also increases $d(s,s')$ due to the constraint in Equation (7). Here, LSD can be viewed as a special case of DSD with $d(x,y) = \|x-y\|$.

When dealing with a *learned* distance function d, it is generally not straightforward to ensure that d is a *valid* distance (pseudo-)metric, which must satisfy symmetry and the triangle inequality. However, DSD has the nice property that d in Equation (7) does not have to be a valid metric. This is because DSD implicitly converts the original constraint (Equation (7)) into the one with a valid pseudometric $\tilde{d}$. As a result, we can use any arbitrary non-negative function d for DSD, with the semantics being implicitly defined by its *induced pseudometric* $\tilde{d}$. We summarize our theoretical results as follows and the proofs are in Appendix B.1.

Theorem 4.1. *Given any non-negative function $d : \mathcal{S}\times\mathcal{S} \to \mathbb{R}_0^+$, there exists a valid pseudometric $\tilde{d} : \mathcal{S}\times\mathcal{S} \to \mathbb{R}_0^+$ that satisfies the following properties:*

1. *Imposing Equation (7) with d is equivalent to imposing Equation (7) with $\tilde{d}$,* i.e.*,*

$$\forall x, y \in \mathcal{S}, \quad \|\phi(x) - \phi(y)\| \le d(x,y) \tag{8}$$
$$\iff \forall x, y \in \mathcal{S}, \quad \|\phi(x) - \phi(y)\| \le \tilde{d}(x,y). \tag{9}$$

2. *$\tilde{d}$ is a valid pseudometric.*
3. *$\tilde{d}$ is a lower bound of d,* i.e.*,*

$$\forall x, y \in \mathcal{S}, \quad 0 \le \tilde{d}(x,y) \le d(x,y). \tag{10}$$

Training of DSD. While LSD implements the Lipshitz constraint in Equation (5) using Spectral Normalization (Miyato et al., 2018), similarly imposing DSD's constraint in Equation (7) is not straightforward because it is no longer a Euclidean Lipschitz constraint. Hence, we optimize our objective with dual gradient descent (Boyd et al., 2004): *i.e.*, with a Lagrange multiplier $\lambda \ge 0$, we use the following dual objectives to train DSD:

$$r^{\text{DSD}} := (\phi(s') - \phi(s))^\top z, \tag{11}$$
$$\mathcal{J}^{\text{DSD},\phi} := \mathbb{E}[(\phi(s') - \phi(s))^\top z + \lambda \cdot \min(\epsilon, d(x,y) - \|\phi(x) - \phi(y)\|)], \tag{12}$$
$$\mathcal{J}^{\text{DSD},\lambda} := -\lambda \cdot \mathbb{E}[\min(\epsilon, d(x,y) - \|\phi(x) - \phi(y)\|)], \tag{13}$$

where r^{DSD} is the intrinsic reward for the policy, and $\mathcal{J}^{\text{DSD},\phi}$ and $\mathcal{J}^{\text{DSD},\lambda}$ are the objectives for ϕ and λ, respectively. x and y are sampled from some state pair distribution $p^{\text{cst}}(x,y)$ that imposes the constraint in Equation (7). $\epsilon > 0$ is a slack variable to avoid the gradient of λ always being non-negative. With these objectives, we can train DSD by optimizing the policy with Equation (11) as an intrinsic reward while updating the other components with Equations (12) and (13).

4.2. Controllability-Aware Distance Function

To guide distance-maximizing skill discovery to focus on more challenging skills, a distance function d is required to assign larger values to state transitions that are hard-to-achieve with the current skills and smaller values to easy-to-achieve transitions. d also needs to be adaptable to the current skill policy so that the agent continuously acquires new skills and finds increasingly difficult state transitions over the course of training.

Among many potential distance functions, we choose a negative log-likelihood of a transition from the current skill policy, $-\log p(s'|s)$, as a *controllability-aware distance function* in this paper. Accordingly, we define the degree to which a transition is "hard-to-achieve" as $-\log p(s'|s)$ with respect to the current skill policy's transition distribution. This suits our desiderata since (1) it assigns high values for rare transitions (*i.e.*, low $p(s'|s)$) while assigns small

values for frequently visited transitions (*i.e.*, high $p(s'|s)$); (2) $p(s'|s)$ can be approximated by training a density model $q_\theta(s'|s)$ from policy rollouts; and (3) the density model $q_\theta(s'|s)$ continuously adjusts to the current skill policy by jointly training it with the skill policy. Here, while it is also possible to employ multi-step transitions $p(s_{t+k}|s_t)$ for the distance function, we stick to the single-step version for simplicity. We note that even though we employ single-step log-likelihoods, DSD maximizes the sum of rewards, $\sum_{t=0}^{T-1}(\phi(s_{t+1}) - \phi(s_t))^\top z = (\phi(s_T) - \phi(s_0))^\top z$ for the trajectory $(s_0, a_0, s_1, \ldots, s_T)$, which maximizes the traveled distance of the *whole* trajectory while maintaining the directional alignment with z.

4.3. Controllability-Aware Skill Discovery

Now, we introduce **Controllability-aware Skill Discovery** (**CSD**), a distance-maximizing skill discovery method with our controllability-aware distance function. With the distance function in Section 4.2 we can rewrite the constraint of DSD in Equation (7) as follows:

$$\forall s, s' \in \mathcal{S}, \quad \|\phi(s) - \phi(s')\| \leq d^{\text{CSD}}(s, s'), \tag{14}$$

$$d^{\text{CSD}}(s, s') \triangleq (s' - \mu_\theta(s))^\top \Sigma_\theta^{-1}(s)(s' - \mu_\theta(s)) \tag{15}$$

$$\propto -\log q_\theta(s'|s) + (\text{const}), \tag{16}$$

where the density model is parameterized as $q_\theta(s'|s) = \mathcal{N}(\mu_\theta(s), \Sigma_\theta(s))$, which is jointly trained using (s, s') tuples collected by the skill policy. We also use the same $p(s, s')$ distribution from the skill policy for the dual constraint distribution $p^{\text{cst}}(x, y)$ introduced in Section 4.1 as well. Here, we note that $d^{\text{CSD}}(\cdot, \cdot)$ is not necessarily a valid distance metric; however, we can still use it for the constraint in Equation (7) according to Theorem 4.1, because it automatically transforms d^{CSD} into its induced valid pseudometric $\tilde{d}^{\text{CSD}}$. Further discussion about its implications and limitations can be found in Appendix B.2.

CSD has several main advantages. First, the agent actively seeks rare state transitions and thus acquires increasingly complex skills over the course of training, which makes the skills discovered more useful for downstream tasks. In contrast, LSD or previous MI-based approaches only maximize Euclidean distances or are even agnostic to traveled distances, which often leads to simple or static behaviors. Second, unlike LSD, the optimal behaviors of CSD are agnostic to the semantics and scales of each dimension of the state space; thus, CSD does not require domain knowledge about the state space. Instead, the objective of CSD only depends on the difficulty or sparsity of state transitions. Finally, unlike curiosity- or disagreement-based exploration methods that only seek unseen transitions (Pathak et al., 2017; 2019; Mendonca et al., 2021), CSD finds a balance between covering unseen transitions and learning maximally different skills across zs via directional alignments, which leads to diverse yet consistent skills.

Algorithm 1 Controllability-aware Skill Discovery (CSD)

Initialize skill policy $\pi(a|s, z)$, function $\phi(s)$, conditional density model $q_\theta(s'|s)$, Lagrange multiplier λ
for $i \leftarrow 1$ to (# epochs) **do**
  for $j \leftarrow 1$ to (# episodes per epoch) **do**
    Sample skill $z \sim p(z)$
    Sample trajectory τ with $\pi(a|s, z)$
  end for
  Fit conditional density model $q_\theta(s'|s)$ using current trajectory samples
  Update $\phi(s)$ with gradient ascent on $\mathcal{J}^{\text{DSD},\phi}$
  Update λ with gradient ascent on $\mathcal{J}^{\text{DSD},\lambda}$
  Update $\pi(a|s, z)$ using SAC with intrinsic reward r^{DSD}
end for

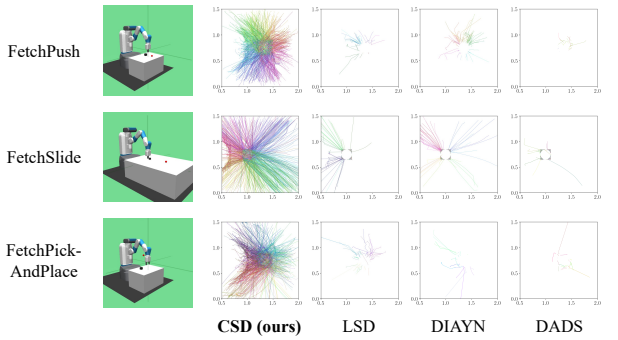
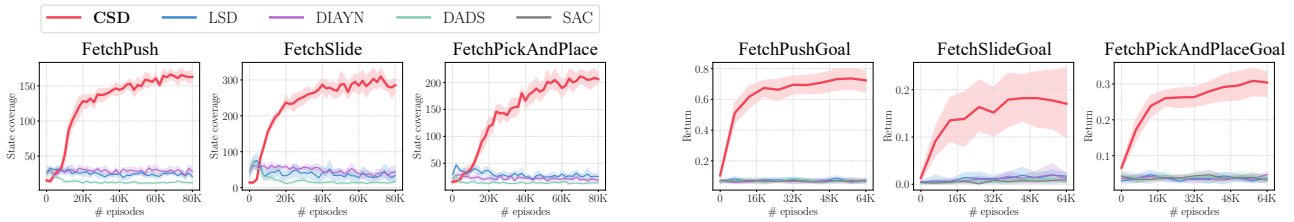


Figure 3. The object trajectories in the xy plane of randomly sampled 1000 continuous skills learned by CSD, LSD, DIAYN, and DADS in three Fetch manipulation environments without any supervision. Trajectories with different colors represent different skills. Only CSD learns to manipulate the object across all three tasks without supervision while other methods focus only on moving the robot arm. We refer to Appendix D for the complete qualitative results from all random seeds.

Training of CSD. We train the skill policy $\pi(a|s, z)$ with Soft Actor-Critic (SAC) (Haarnoja et al., 2018b) with Equation (11) as an intrinsic reward. We train the other components with stochastic gradient descent. We summarize the training procedure of CSD in Algorithm 1 and provide the full implementation details in Appendix E.

5. Experiments

The goal of our experiments is to verify whether our controllability-aware skill discovery method can learn complex, useful skills without supervision in a variety of environments. We test CSD on six environments across three different domains: three Fetch manipulation environments (FetchPush, FetchSlide, and FetckPickAndPlace) (Plappert et al., 2018), Kitchen (Gupta et al., 2019), and two MuJoCo locomotion environments (Ant and HalfCheetah) (Todorov et al., 2012; Brockman et al., 2016). We mainly compare CSD with three state-of-the-art unsupervised skill discovery methods: LSD (Park et al., 2022), DIAYN (Eysenbach et al.,



(a) Object state coverage (b) Downstream task performance

Figure 4. Comparison of the object state coverage and downstream task performances of skill discovery methods in three Fetch manipulation environments. Only CSD learns to manipulate the object without external supervision, while the other methods mainly focus on controlling the internal states (Figure 16) because there is little incentive for them to discover more 'challenging' skills.

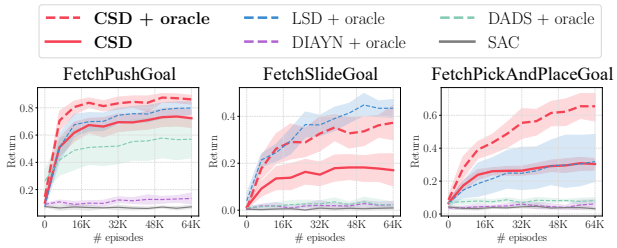


Figure 5. Comparison of the downstream task performances of skill discovery methods with the oracle prior, which restricts the input to the skill discriminators to the object xyz coordinates.

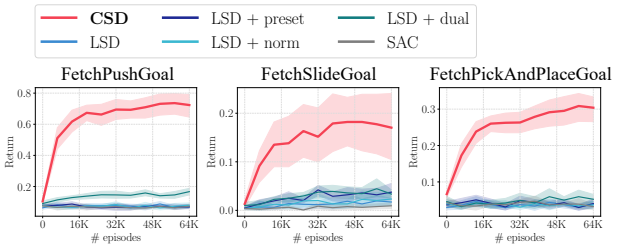


Figure 6. Ablation study of distance-maximizing skill discovery in three Fetch environments. This suggests that CSD's performance cannot be achieved by just applying simple tricks to the previous Euclidean distance-maximizing skill discovery method.

2019), and DADS (Sharma et al., 2020). They respectively fall into the categories of Euclidean distance-maximizing skill discovery, reverse-MI, and forward-MI (Section 3). We also compare with disagreement-based exploration used in unsupervised goal-conditioned RL, such as LEXA (Mendonca et al., 2021), in Appendix C. We evaluate state coverage and performance on downstream tasks to assess the diversity and usefulness of the skills learned by each method. For our quantitative experiments, we use 8 random seeds and present 95% confidence intervals using error bars or shaded areas. We refer to our project page for videos.

5.1. Fetch Manipulation

We first show (1) whether CSD can acquire object manipulation skills without any supervision, (2) how useful the learned skills are for the downstream tasks, and (3) which component allows CSD to learn complex skills in the Fetch manipulation environments (Plappert et al., 2018). Each Fetch environment consists of a robot arm and an object but has a unique configuration; *e.g.*, FetchSlide has a slippery table and FetchPickAndPlace has a two-fingered gripper.

We train CSD, LSD, DIAYN, and DADS on the three Fetch environments for 80K episodes with 2-D continuous skills (FetchPush, FetchSlide) or 3-D continuous skills (FetchPickAndPlace). Note that we do not leverage human prior knowledge on the state space (*e.g.*, object pose); thus, all methods are trained on the *full state* in this experiment.[2]

[2]We note that the Fetch experiments in the LSD paper (Park et al., 2022) are using the 'oracle' prior, which enforces an agent to only focus on the state change of the object.

Figure 3 illustrates the object trajectories of continuous skills learned by skill discovery methods in the absence of any supervision. CSD successfully learns to move the object in diverse directions without external supervision. On the other hand, all of the previous methods fail to learn such skills and instead focus on diversifying the joint angles of the robot arm itself. This is because there is no incentive for the previous methods to focus on *challenging* skills such as object manipulation, while CSD explicitly finds hard-to-achieve state transitions.

Following the setup in Park et al. (2022), we evaluate two quantitative metrics: the object state coverage and goal-reaching downstream task performance. Figure 4a compares the four skill discovery methods in terms of the object state coverage, which is measured by the number of 0.1×0.1 square bins occupied by the object at least once, in the three Fetch environments. Figure 4b shows the comparison of the goal-reaching downstream task performances, where we train a hierarchical controller $\pi^h(z|s, g)$ that sequentially combines skills z for the frozen skill policy $\pi(a|s, z)$ to move the object to a goal position g. We additionally train the vanilla SAC baseline to verify the effectiveness of leveraging autonomously discovered skills. We refer to Appendix E.2 for further details. On both quantitative metrics, CSD outperforms the prior methods by large margins, successfully discovering diverse manipulation skills that are useful for solving downstream tasks.

Skill discovery with the oracle prior on the state space. While our experiments show that our approach can discover

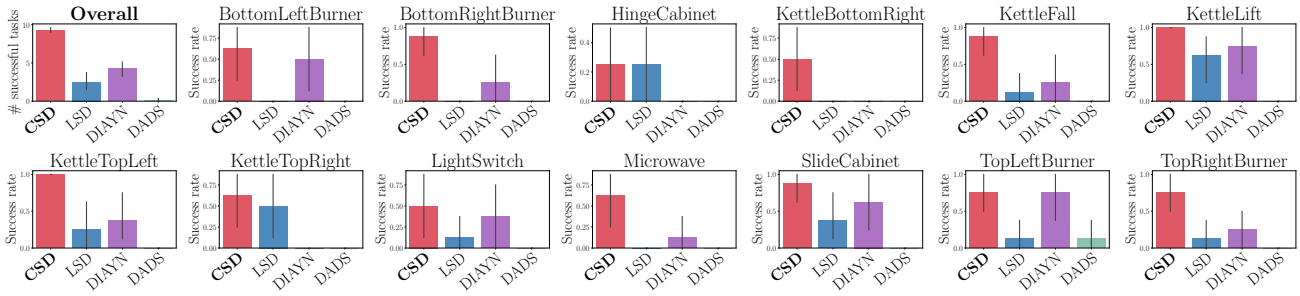
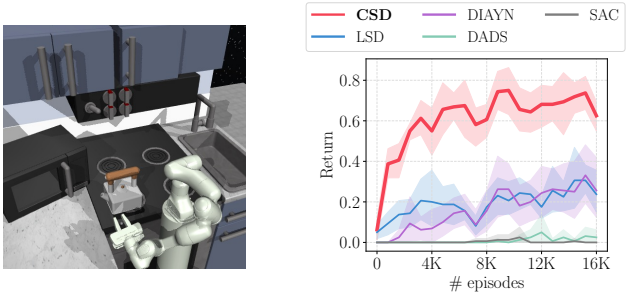
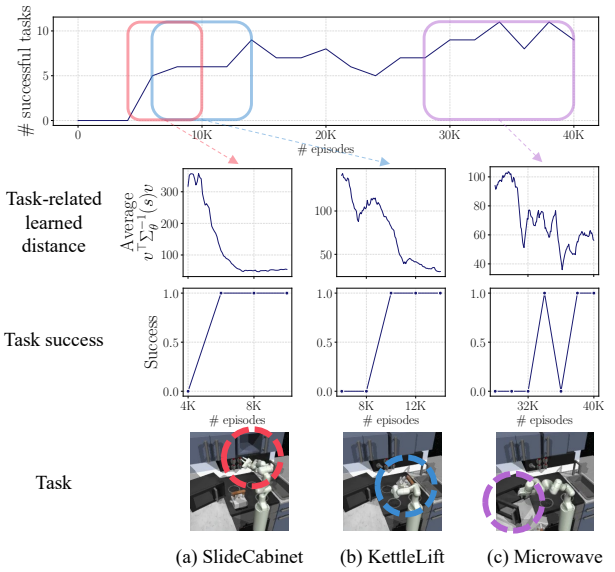


Figure 7. Task success rates of 16 discrete skills discovered by CSD, LSD, DIAYN, and DADS in the Kitchen environment. CSD learns to manipulate diverse objects in the kitchen without any supervision. We refer to Appendix D for the results with 2-D continuous skills.



(a) Kitchen environment (b) Downstream task performance

Figure 8. Comparison of the downstream task performances of skill discovery methods in the Kitchen environment.



(a) SlideCabinet (b) KettleLift (c) Microwave

Figure 9. Evolution of task-related distances and corresponding task success rates. Our learned task-related distances decrease once the agent gains control of the corresponding objects, which makes the agent focus on other new objects consistently over the course of training. Distance plots are smoothed over a window of size 10 for better visualization.

useful manipulation skills without any human prior on the state space, previous unsupervised skill discovery methods (Eysenbach et al., 2019; Sharma et al., 2020; Park et al., 2022) mostly do not work without the *oracle state prior*, which restricts the skill discriminator module's input to only the xyz coordinates of the object. To investigate how CSD and the prior methods perform in the presence of this supervision, we train them with the oracle state prior. Figure 5 demonstrates that even without the oracle state prior, our CSD is mostly comparable to the previous best method with the oracle prior. This result demonstrates the potential of our approach in scalability to more complex environments, where human prior is no longer available. Moreover, with the oracle state prior, CSD further improves its performance. We refer to Figure 17 for the full qualitative results of CSD and LSD with the oracle prior in FetchPickAndPlace.

Ablation study. To understand the importance of our controllability-aware distance function in CSD, we examine whether similar results can be achieved without some components of CSD or by just applying simple tricks to LSD, a previous Euclidean distance-maximizing skill discovery method. Specifically, we consider the following three variants: (1) LSD + preset: LSD with a normalized state space using the precomputed standard deviation of each state dimension from randomly generated trajectories, (2) LSD + norm: LSD with a normalized state space using the moving average of the standard deviation of state differences ($s' - s$), and (3) LSD + dual: LSD trained with dual gradient descent instead of spectral normalization (*i.e.*, CSD without our learned distance function). Figure 6 compares the performances of these variants with CSD, LSD, and SAC in three downstream tasks. The results show that only CSD learns to manipulate objects, which suggests that our controllability-aware distance function is indeed necessary to discover such complex skills without supervision.

5.2. Kitchen Manipulation

To verify the scalability of unsupervised skill discovery in a complex environment with diverse objects, we evaluate our method on the Kitchen manipulation environment (Gupta et al., 2019), which includes 13 downstream tasks in total, such as opening a microwave, turning a light switch, moving a kettle, and opening slide/hinge cabinet doors (Figure 8a). We train CSD, LSD, DIAYN, and DADS with both 2-D continuous skills and 16 discrete skills for 40K episodes

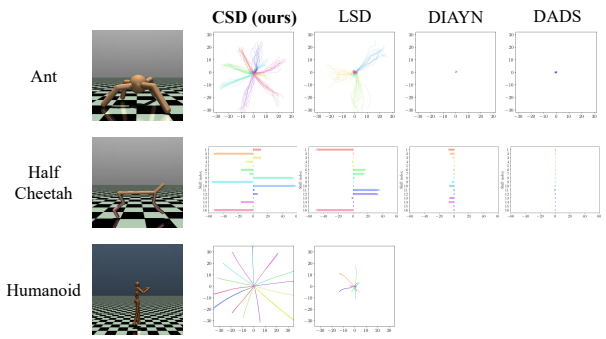
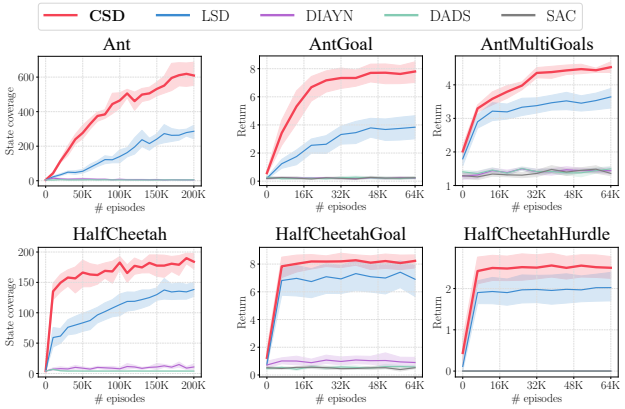


Figure 10. The agent's xy (Ant and Humanoid) or x (HalfCheetah) trajectories of skills discovered by CSD, LSD, DIAYN, and DADS in MuJoCo locomotion environments. Trajectories with different colors represent different skills. We refer to Appendix D for the complete qualitative results from all random seeds.



(a) State coverage (b) Downstream task performance

Figure 11. Comparison of the state coverage and downstream task performance of skills discovery methods in Ant and HalfCheetah.

without any supervision. We refer to Appendix E for further experimental details regarding the Kitchen environment.

We first measure the task success rates of the skills learned by the four methods. After the unsupervised skill training, we roll out the skill policy to collect 50 trajectories with 50 randomly sampled zs and measure whether each of the 13 tasks has at least one successful trajectory. The results with 16 discrete skills in Figure 7 suggest that CSD learns on average 10 out of 13 skills, while the prior methods fail to discover such skills (2 for LSD, 4 for DIAYN, 0 for DADS) because they mainly focus on diversifying the robot state. Continuous skills in Figure 14 also show similar results.

We then evaluate the downstream task performance by training a high-level controller $\pi^h(z|s,g)$ with the learned 2-D continuous skills $\pi(a|s,z)$ as behavioral primitives to achieve a task specified by a 13-D one-hot vector g. The high-level controller chooses a skill z every 10 steps until the episode ends. The results in Figure 8b show that CSD significantly outperforms the previous methods.

Qualitative analysis. Figure 9 illustrates how our controllability-aware distance evolves over time and how this leads to the discovery of diverse, complex skills, *e.g.*, SlideCabinet, KettleLift, and Microwave. Over training, we measure the task-related controllability-aware distance $v^\top \Sigma_\theta^{-1}(s) v$ for each task v using skill trajectories, where v is the one-hot task vector corresponding to each of the three tasks. At around 4K episodes (Figure 9a), our controllability-aware distance encourages the agent to control the sliding cabinet with a large distance value (*i.e.*, high reward). Once the agent learns to manipulate the sliding cabinet door, our controllability-aware distance for that skill decreases, letting the agent move its focus to other harder-to-achieve skills, *e.g.*, lifting kettle (Figure 9b) or opening a microwave (Figure 9c). As a result, the number of successful tasks gradually increases over the course of training.

5.3. MuJoCo Locomotion

To assess whether the idea of controllability-aware skill discovery works on domains other than manipulation, we evaluate CSD mainly on two MuJoCo locomotion environments (Todorov et al., 2012; Brockman et al., 2016): Ant and HalfCheetah. We additionally employ 17-DoF Humanoid, the most complex environment in the benchmark, for a qualitative comparison between CSD and LSD. In these environments, we train skill discovery methods for 200K episodes (100K for Humanoid) with 16 discrete skills.

Figure 10 shows examples of skills discovered by each method, which suggests that CSD leads to the largest state coverage thanks to our controllability-aware distance function. For quantitative evaluation, we first measure the state space coverage by counting the number of 1×1 bins occupied by the agent's xy coordinates (xz coordinates for 2-D HalfCheetah) at least once. Figure 11a demonstrates that CSD covers the largest area among the four methods. This is because CSD's controllability objective makes the agent mainly focus on diversifying the global position of the agent, which corresponds to the 'challenging' state transitions in these locomotion environments. We emphasize that CSD not just learns to navigate in diverse directions but also learns a variety of behaviors, such as rotating and flipping in both environments (videos). We also note that MI-based methods (DIAYN and DADS) completely fail to diversify the agent's location and only discover posing skills, because the MI objective is agnostic to the distance metric, not providing incentives to maximize traveled distances in the state space.

We also evaluate the downstream learning performance on four tasks: AntGoal, AntMultiGoals, HalfCheetahGoal, and HalfCheetahHurdle, following previous works (Eysenbach et al., 2019; Sharma et al., 2020; Kim et al., 2021; Park et al., 2022). In AntGoal and HalfCheetahGoal, the agent should reach a randomly sampled goal position, and in

AntMultiGoals, the agent should follow multiple randomly sampled goals in sequence. In HalfCheetahHurdle (Qureshi et al., 2020), the agent should jump over as many hurdles as possible. With downstream task rewards, we train a high-level policy that sequentially combines the learned skills. In Figure 11b, CSD consistently demonstrates the best performance among the four methods, which suggests that the skills discovered by CSD are effective not just on locomotion tasks but also on a wide variety of tasks, such as hurdle jumping.

6. Conclusion

In this paper, we present Controllability-aware Skill Discovery (CSD), a novel unsupervised skill discovery method that explicitly looks for hard-to-achieve skills. Specifically, we first formulate a distance-maximizing skill discovery approach (DSD), which can be combined with any arbitrary distance function. We then propose a jointly trained controllability-aware distance function, which consistently encourages the agent to discover more complex, hard-to-achieve skills. We empirically show that the idea of controllability-awareness enables the agent to acquire diverse complex skills in the absence of supervision in a variety of robotic manipulation and locomotion environments.

Limitations and future directions. Although the general idea of controllability-aware skill discovery is still applicable to pixel domains, *e.g.*, in combination with representation learning techniques (Hafner et al., 2020; Srinivas et al., 2020; Seo et al., 2022), where they will reveal both the object and agent representations and CSD will focus on the object representation, we did not verify the scalability of our controllability-aware distance function to pixel-based environments. We leave it as future work. Another limitation is that CSD in its current form might not discover 'slowly moving' skills because underlying DSD prefers skills with large state variations. We believe acquiring skills with diverse moving speeds is another interesting future direction.

Acknowledgement

We would like to thank Amber Xie, Younggyo Seo, and Jaekyeom Kim for their insightful feedback and discussion. This work was funded in part by Darpa RACER, Komatsu, a Berkeley Graduate Fellowship, and the BAIR Industrial Consortium. Seohong Park was partly supported by Korea Foundation for Advanced Studies (KFAS).

References

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *ArXiv*, abs/1807.10299, 2018.

Adeniji, A., Xie, A., and Abbeel, P. Skill-based reinforcement learning with intrinsic reward matching. *ArXiv*, abs/2210.07426, 2022.

Barber, D. and Agakov, F. The IM algorithm: a variational approach to information maximization. In *Neural Information Processing Systems (NeurIPS)*, 2003.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *ArXiv*, abs/1606.01540, 2016.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

Campos Camúñez, V., Trott, A., Xiong, C., Socher, R., Giró Nieto, X., and Torres Viñals, J. Explore, discover and learn: unsupervised discovery of state-covering skills. In *International Conference on Machine Learning (ICML)*, 2020.

Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning (ICML)*, 2018.

Du, Y., Abbeel, P., and Grover, A. It takes four to tango: Multiagent selfplay for automatic curriculum generation. In *International Conference on Learning Representations (ICLR)*, 2022.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *ArXiv*, abs/1611.07507, 2016.

Gu, S. S., Diaz, M., Freeman, D. C., Furuta, H., Ghasemipour, S. K. S., Raichuk, A., David, B., Frey, E., Coumans, E., and Bachem, O. Braxlines: Fast and interactive toolkit for rl-driven behavior engineering beyond reward maximization. *ArXiv*, abs/2110.04686, 2021.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018b.

Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

Hansen, S., Dabney, W., Barreto, A., Wiele, T., Warde-Farley, D., and Mnih, V. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations (ICLR)*, 2020.

Jiang, Z., Gao, J., and Chen, J. Unsupervised skill discovery via recurrent skill training. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Kamienny, P.-A., Tarbouriech, J., Lazaric, A., and Denoyer, L. Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. In *International Conference on Learning Representations (ICLR)*, 2022.

Kim, J., Park, S., and Kim, G. Unsupervised skill discovery with bottleneck option learning. In *International Conference on Machine Learning (ICML)*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kraskov, A., Stögbauer, H., and Grassberger, P. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Laskin, M., Liu, H., Peng, X. B., Yarats, D., Rajeswaran, A., and Abbeel, P. Unsupervised reinforcement learning with contrastive intrinsic control. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *ArXiv*, abs/1906.05274, 2019.

Liu, H. and Abbeel, P. APS: Active pretraining with successor features. In *International Conference on Machine Learning (ICML)*, 2021a.

Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. In *Neural Information Processing Systems (NeurIPS)*, 2021b.

Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., and Pathak, D. Discovering and achieving goals via world models. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Ogata, K. et al. *Modern control engineering*. Prentice hall Upper Saddle River, NJ, 2010.

OpenAI, O., Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., Welinder, P., D'Sa, R., Petron, A., de Oliveira Pinto, H. P., Paino, A., Noh, H., Weng, L., Yuan, Q., Chu, C., and Zaremba, W. Asymmetric self-play for automatic goal discovery in robotic manipulation. *ArXiv*, abs/2101.04882, 2021.

Park, S., Choi, J., Kim, J., Lee, H., and Kim, G. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations (ICLR)*, 2022.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Pathak, D., Gandhi, D., and Gupta, A. K. Self-supervised exploration via disagreement. In *International Conference on Machine Learning (ICML)*, 2019.

Pitis, S., Chan, H., Zhao, S., Stadie, B. C., and Ba, J. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *ArXiv*, abs/1802.09464, 2018.

Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. Skew-Fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Qureshi, A. H., Johnson, J. J., Qin, Y., Henderson, T., Boots, B., and Yip, M. C. Composing task-agnostic policies with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Rajeswar, S., Mazzaglia, P., Verbelen, T., Pich'e, A., Dhoedt, B., Courville, A. C., and Lacoste, A. Unsupervised model-based pre-training for data-efficient control from pixels. *ArXiv*, abs/2209.12016, 2022.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *International Conference on Machine Learning (ICML)*, 2020.

Seo, Y., Hafner, D., Liu, H., Liu, F., James, S., Lee, K., and Abbeel, P. Masked world models for visual control. In *Conference on Robot Learning (CoRL)*, 2022.

Shafiullah, N. M. M. and Pinto, L. One after another: Learning incremental skills for a changing world. In *International Conference on Learning Representations (ICLR)*, 2022.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Strouse, D., Baumli, K., Warde-Farley, D., Mnih, V., and Hansen, S. S. Learning more skills through optimistic exploration. In *International Conference on Learning Representations (ICLR)*, 2022.

Sukhbaatar, S., Kostrikov, I., Szlam, A. D., and Fergus, R. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations (ICLR)*, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Touati, A. and Ollivier, Y. Learning one representation to optimize all rewards. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Touati, A., Rapin, J., and Ollivier, Y. Does zero-shot reinforcement learning exist? *ArXiv*, abs/2209.14935, 2022.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations (ICLR)*, 2019.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning (ICML)*, 2021.

Zhao, A., Lin, M., Li, Y., Liu, Y., and Huang, G. A mixture of surprises for unsupervised reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Zhao, R., Gao, Y., Abbeel, P., Tresp, V., and Xu, W. Mutual information state intrinsic control. In *International Conference on Learning Representations (ICLR)*, 2021.

A. Extended Related Work on Unsupervised RL

The goal of unsupervised RL is to learn useful knowledge, such as dynamics models, state representations, and behavioral primitives, without predefined tasks so that we can later utilize them to efficiently solve downstream tasks. One line of research focuses on gathering knowledge of the environment with pure exploration (Pathak et al., 2017; Burda et al., 2019; Pathak et al., 2019; Sekar et al., 2020; Liu & Abbeel, 2021b; Yarats et al., 2021; Rajeswar et al., 2022). Unsupervised skill discovery methods (Gregor et al., 2016; Co-Reyes et al., 2018; Eysenbach et al., 2019; Sharma et al., 2020; Kim et al., 2021; Kamienny et al., 2022; Strouse et al., 2022; Park et al., 2022; Shafiullah & Pinto, 2022; Jiang et al., 2022; Zhao et al., 2022) aim to learn a set of temporally extended useful behaviors, and our CSD falls into this category. Another line of work focuses on discovering *goals* and corresponding goal-conditioned policies via pure exploration (Warde-Farley et al., 2019; Pong et al., 2020; Pitis et al., 2020; Mendonca et al., 2021) or asymmetric/curriculum self-play (Sukhbaatar et al., 2018; OpenAI et al., 2021; Du et al., 2022). Lastly, Touati & Ollivier (2021); Touati et al. (2022) aim to learn a set of policies that can be instantly adapted to task reward functions given an unsupervised exploration method or an offline dataset.

B. Theoretical Results

B.1. Proof of Theorem 4.1

We assume that we are given an arbitrary non-negative function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$. We first introduce some additional notations. For $x, y \in \mathcal{S}$, define $d_s(x, y) \triangleq \min(d(x, y), d(y, x))$. For $x, y \in \mathcal{S}$, let $P(x, y)$ be the set of all finite state paths from x to y. For a state path $p = (s_0, s_1, \dots, s_t)$, define $D_s(p) \triangleq \sum_{i=0}^{t-1} d_s(s_i, s_{i+1})$.

Now, for $x, y \in \mathcal{S}$, we define the *induced pseudometric* $\tilde{d} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$ as follows:

$$\tilde{d}(x, y) \triangleq \begin{cases} \inf_{p \in P(x,y)} D_s(p) & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}. \tag{17}$$

Then, the following theorems hold.

Lemma B.1. *$\tilde{d}$ is a lower bound of d,* i.e.*,*

$$\forall x, y \in \mathcal{S}, \quad 0 \leq \tilde{d}(x, y) \leq d(x, y). \tag{18}$$

Proof. If $x = y$, then $\tilde{d}(x, y) = 0$ by definition and thus $0 \leq \tilde{d}(x, y) \leq d(x, y)$ always holds. Otherwise, $0 \leq \tilde{d}(x, y) \leq D_s((x, y)) = d_s(x, y) \leq d(x, y)$ holds and this completes the proof. □

Theorem B.2. *For $\phi : \mathcal{S} \to \mathbb{R}^D$, imposing Equation (7) with d is equivalent to imposing Equation (7) with $\tilde{d}$,* i.e.*,*

$$\forall x, y \in \mathcal{S}, \quad \|\phi(x) - \phi(y)\| \leq d(x, y) \iff \forall x, y \in \mathcal{S}, \quad \|\phi(x) - \phi(y)\| \leq \tilde{d}(x, y). \tag{19}$$

Proof. From Lemma B.1, we know that $\|\phi(x) - \phi(y)\| \leq \tilde{d}(x, y)$ implies $\|\phi(x) - \phi(y)\| \leq d(x, y)$. Now, we assume that $\|\phi(x) - \phi(y)\| \leq d(x, y)$ holds for any $x, y \in \mathcal{S}$. First, if $x = y$, then $\|\phi(x) - \phi(y)\|$ becomes 0 and thus $\|\phi(x) - \phi(y)\| \leq d(x, y)$ always holds. For $x \neq y$, let us consider any state path $p = (s_0 = x, s_1, s_2, \dots, s_{t-1}, s_t = y) \in P(x, y)$. For any $i \in \{0, 1, \dots, t-1\}$, we have

$$\|\phi(s_i) - \phi(s_{i+1})\| \leq d(s_i, s_{i+1}), \tag{20}$$
$$\|\phi(s_{i+1}) - \phi(s_i)\| \leq d(s_{i+1}, s_i), \tag{21}$$

and thus we get $\|\phi(s_i) - \phi(s_{i+1})\| = \|\phi(s_{i+1}) - \phi(s_i)\| \leq \min(d(s_i, s_{i+1}), d(s_{i+1}, s_i)) = d_s(s_i, s_{i+1})$. Now, we have the following inequalities:

$$\|\phi(s_0) - \phi(s_1)\| \leq d_s(s_0, s_1), \tag{22}$$
$$\|\phi(s_1) - \phi(s_2)\| \leq d_s(s_1, s_2), \tag{23}$$
$$\dots, \tag{24}$$
$$\|\phi(s_{t-1}) - \phi(s_t)\| \leq d_s(s_{t-1}, s_t). \tag{25}$$

From these, we obtain $\|\phi(x)-\phi(y)\| = \|\phi(s_0)-\phi(s_t)\| \le \sum_{i=0}^{t-1} \|\phi(s_i)-\phi(s_{i+1})\| \le \sum_{i=0}^{t-1} d_s(s_i, s_{i+1}) = D_s(p)$. Then, by taking the infimum of the right-hand side over all possible $p \in P(x, y)$, we get $\|\phi(x) - \phi(y)\| \le \inf_{p\in P(x,y)} D_s(p) = \tilde{d}(x, y)$ and this completes the proof. □

Theorem B.3. *$\tilde{d}$ is a valid pseudometric,* i.e.*,*

(a) $\forall x \in \mathcal{S}$, $\tilde{d}(x, x) = 0$.

(b) (Symmetry) $\forall x, y \in \mathcal{S}$, $\tilde{d}(x, y) = \tilde{d}(y, x)$.

(c) (Triangle inequality) $\forall x, y, z \in \mathcal{S}$, $\tilde{d}(x, y) \le \tilde{d}(x, z) + \tilde{d}(z, y)$.

Proof. (*a*) By definition, $\tilde{d}(x, x) = 0$ always holds for all $x \in \mathcal{S}$.

(*b*) If $x = y$, then $\tilde{d}(x, y) = \tilde{d}(y, x) = 0$. Otherwise, with $p = (s_0 = x, s_1, s_2, \dots, s_{t-1}, s_t = y) \in P(x, y)$, we can prove the symmetry of $\tilde{d}$ as follows:

$$\begin{aligned}
\tilde{d}(x, y) &= \inf_{p\in P(x,y)} D_s(p) && (26)\\
&= \inf_{p\in P(x,y)} \sum_{i=0}^{t-1} d_s(s_i, s_{i+1}) && (27)\\
&= \inf_{p\in P(x,y)} \sum_{i=0}^{t-1} d_s(s_{i+1}, s_i) && (28)\\
&= \inf_{p\in P(y,x)} D_s(p) && (29)\\
&= \tilde{d}(y, x). && (30)
\end{aligned}$$

(*c*) If $x = y$, $y = z$, or $z = x$, then it can be easily seen that $\tilde{d}(x, y) \le \tilde{d}(x, z) + \tilde{d}(z, y)$ always holds. Hence, we assume that they are mutually different from each other. Then, the following inequality holds:

$$\begin{aligned}
\tilde{d}(x, y) &= \inf_{p\in P(x,y)} D_s(p) && (31)\\
&\le \inf_{p_1\in P(x,z), p_2\in P(z,y)} D_s(p_1) + D_s(p_2) && (32)\\
&= \inf_{p_1\in P(x,z)} D_s(p_1) + \inf_{p_2\in P(z,y)} D_s(p_2) && (33)\\
&= \tilde{d}(x, z) + \tilde{d}(z, y), && (34)
\end{aligned}$$

which completes the proof. □

B.2. Implications of Theorem 4.1

Theorem 4.1 suggests that the constraint in Equation (7) implicitly transforms an arbitrary distance function d into a tighter valid pseudometric $\tilde{d}$. Intuitively, this $\tilde{d}(x, y)$ corresponds to the minimum possible (symmetrized) path distance from x to y. Hence, if we train DSD with Equation (7), it will find long-distance transitions that cannot be equivalently achieved by taking multiple short-distance transitions. Intuitively, in the context of CSD (Section 4.2), this implies that the agent will find rare state transitions that cannot be bypassed by taking 'easy' intermediate steps, which is a desirable property.

However, there are some limitations regarding the use of our distance function d^{CSD} (Equation (16)). First, while the DSD constraint in Equation (7) implicitly symmetrizes the distance function by taking the minimum between $d(x, y)$ and $d(y, x)$, this may not be ideal in highly asymmetric environments involving many irreversible transitions. In practice, this may be resolved by only imposing one-sided constraints of our interest. Second, in our implementation, we only consider a single-step transition (s, s') and a single-step density model $q_\theta(s'|s)$ as we found this simple design choice to be sufficient for our experiments. However, in order to fully leverage the aforementioned property of the induced pseudometric, the constraint may be imposed on any state pairs with a multi-step density model, which we leave for future work.

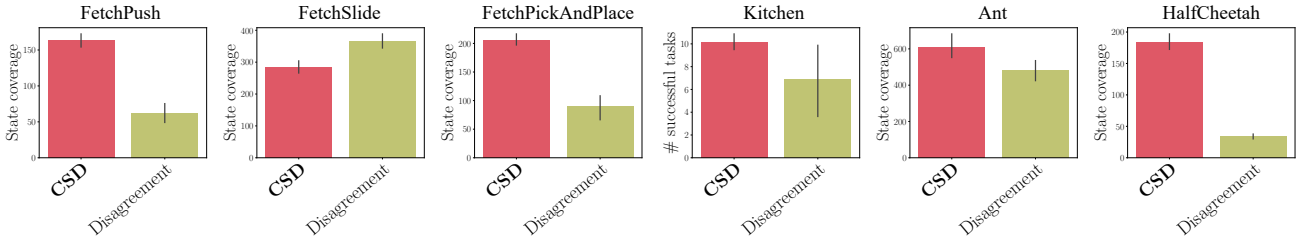


Figure 12. Comparison of unsupervised state coverage metrics between CSD and ensemble disagreement-based exploration (Pathak et al., 2019) in all six environments. CSD mostly outperforms disagreement-based exploration in our state coverage metrics mainly because it actively diversifies hard-to-control states such as the object position or the agent location.

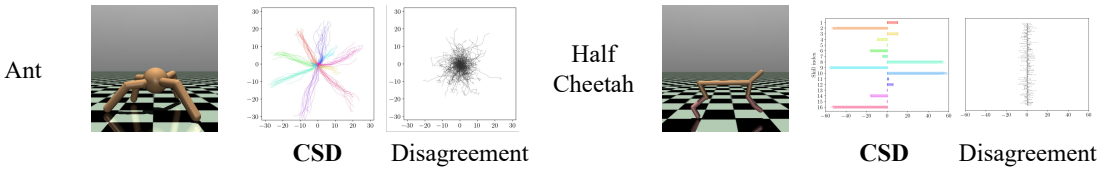


Figure 13. The agent's xy (Ant) or x (HalfCheetah) trajectories of CSD and disagreement-based exploration. While CSD seeks very consistent, directed behaviors, disagreement-based exploration only focuses on diversifying states with chaotic, random behaviors. We provide videos illustrating this difference on our project page.

C. Comparison with Unsupervised Disagreement-Based Exploration

In this section, we discuss the difference between CSD and unsupervised goal-conditioned RL and present an empirical comparison between them. Unsupervised goal-conditioned RL approaches, such as DISCERN (Warde-Farley et al., 2019), Skew-Fit (Pong et al., 2020), MEGA (Pitis et al., 2020), and LEXA (Mendonca et al., 2021), learn diverse behaviors typically by (1) running an exploration method that collects diverse 'goal' states g and (2) learning a goal-conditioned policy $\pi(a|s,g)$ to reach the states discovered. Hence, treating g as a $|\mathcal{S}|$-dimensional skill latent vector, these approaches may be viewed as a special type of unsupervised skill discovery.

However, the main focuses of unsupervised skill discovery are different from that of unsupervised goal-conditioned RL. First, unsupervised skill discovery aims to discover more general skills not restricted to goal-reaching behaviors, which tend to be *static* as the agent is encouraged to stay still at the goal state (Mendonca et al., 2021; Jiang et al., 2022). For instance, our approach maximizes traveled distances, which leads to more 'dynamic' behaviors like consistently running in a specific direction (Figure 10). Second, unsupervised skill discovery aims to build a *compact* set of skills, which could also be discrete, rather than finding all the possible states in the given environment. For example, if we train CSD with three discrete skills, these behaviors will be as 'distant' as possible from one another, being maximally distinguishable. As such, we can have useful behaviors with a much low-dimensional skill space, making it more amenable to hierarchical RL.

Despite the difference in goals, to better illustrate the difference between them, we make an empirical comparison between CSD and ensemble disagreement-based exploration (Pathak et al., 2019), which some previous unsupervised goal-conditioned RL methods like LEXA (Mendonca et al., 2021) use as the exploration method. Disagreement-based exploration learns an ensemble of E forward dynamics models $\{\hat{p}_i(s'|s,a)\}_{i\in\{1,2,\dots,E\}}$, and uses its variance $\sum_k^{|\mathcal{S}|}\mathbb{V}[\hat{p}_i(\cdot_k|s,a)]$ as an intrinsic reward, in order to seek unexplored transitions with high epistemic uncertainty. While unsupervised goal-condition RL approaches additionally learn a goal-conditioned policy, we do not separately learn it since the state coverage metrics of the exploration policy can serve as an approximate upper bound of the corresponding optimal goal-conditioned policy's performance.

Figure 12 presents the comparisons of unsupervised state coverage metrics between CSD and disagreement-based exploration in all of our six environments. The results suggest that CSD mostly outperforms disagreement-based exploration in our state coverage metrics, mainly because CSD actively diversifies hard-to-control states such as the object position or the agent location, while the pure exploration method only focuses on finding unseen transitions. This difference is especially prominent in Ant and HalfCheetah (Figure 13), in which CSD seeks very consistent, directed behaviors, such as moving in

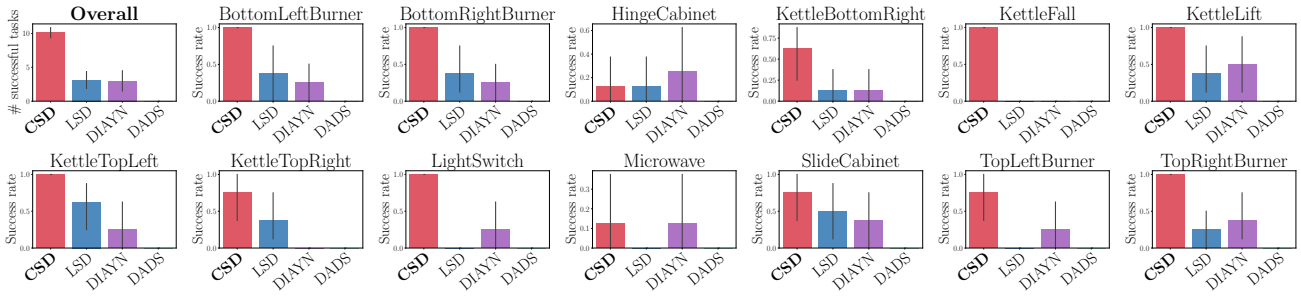


Figure 14. Task success rates of 2-D continuous skills discovered by four methods in the Kitchen environment.

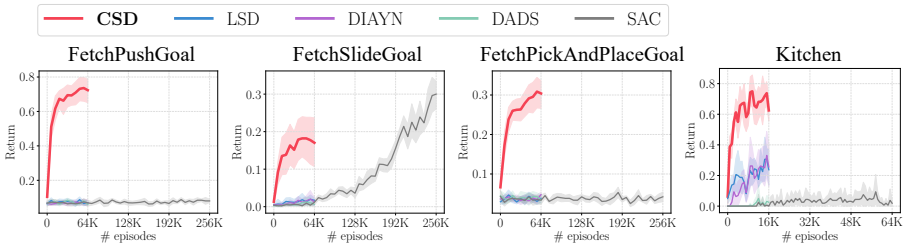


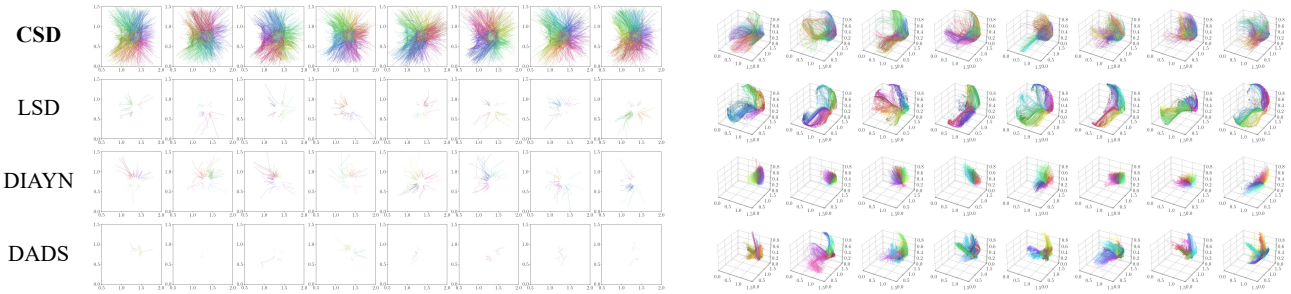
Figure 15. Extended learning curves of the SAC baseline in Fetch and Kitchen downstream tasks.

one direction, while disagreement-based exploration only focuses on diversifying states with chaotic, random behaviors. We provide videos illustrating this difference at https://seohong.me/projects/csd/.

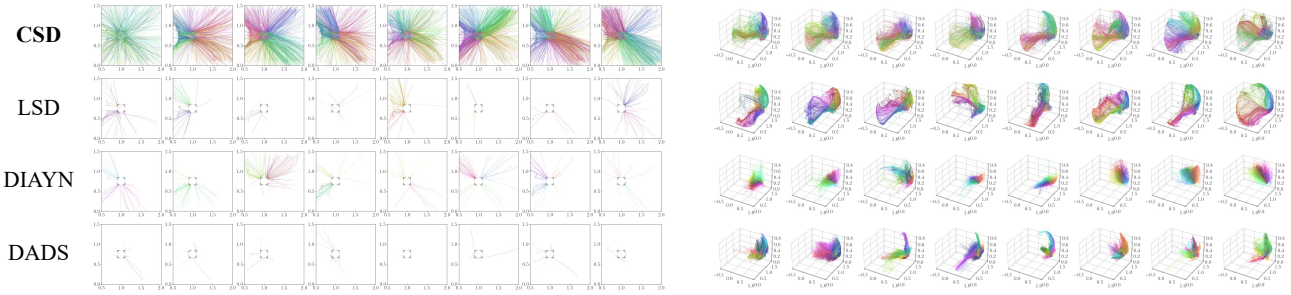
D. Additional Results

Additional quantitative results. Figure 14 shows the task success rates of the 2-D continuous skills learned by CSD, LSD, DIAYN, and DADS. As in the discrete case, CSD outperforms the other methods by a significant margin. Figure 15 demonstrates extended learning curves in Fetch and Kitchen downstream tasks, where we train SAC for four times as long as skill discovery methods. The results suggest that, while SAC alone can solve the FetchSlideGoal task with a lot more samples, it fails at learning FetchPushGoal, FetchPickAndPlaceGoal, and Kitchen mainly because they are challenging sparse-reward tasks. In contrast, agents can quickly learn all these tasks with temporally extended skills from CSD.

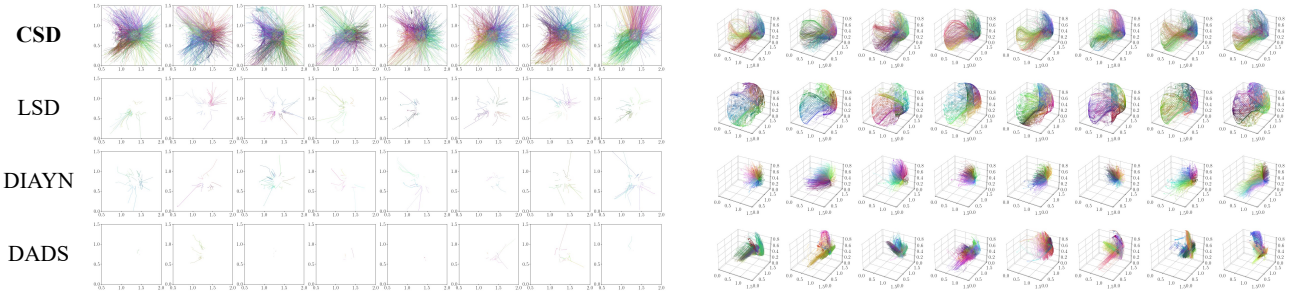
Additional qualitative results. Figures 16 and 19 illustrate the skill trajectories of all runs we use for our experiments in Fetch manipulation and two MuJoCo locomotion environments (eight random seeds for each method in each environment). In the Fetch environments, CSD is the only method that learns object manipulation skills without supervision (Figure 16). In Ant and HalfCheetah, CSD not only learns locomotion skills but also discovers a variety of diverse skills, such as rotating and flipping in both environments (Figure 19, videos). We provide the complete qualitative results in Humanoid in Figure 18. Figure 17 shows the full results of CSD and LSD equipped with the oracle prior in FetchPickAndPlace (eight seeds each). While CSD always learns to pick up the object, LSD discovers such skills in only three out of eight runs (Figure 17). This is because our controllability-aware distance function consistently encourages the agent to learn more challenging picking-up behaviors. As a result, CSD significantly outperforms LSD in downstream tasks (Figure 5).



(a) FetchPush



(b) FetchSlide



(c) FetchPickAndPlace

Figure 16. Complete qualitative results in three Fetch environments (eight runs for each method in each environment). We plot the skill trajectories of the object and the gripper with different colors. CSD is the only unsupervised skill discovery method that discovers object manipulation skills without supervision.

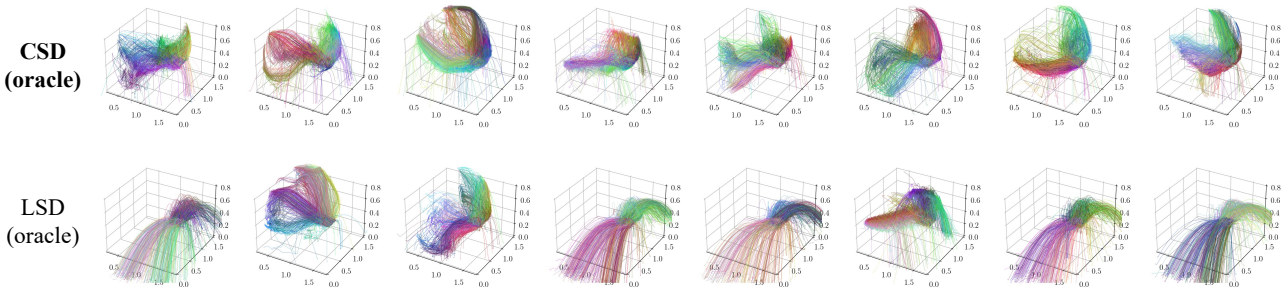


Figure 17. Complete qualitative results of CSD and LSD trained with the oracle prior in FetchPickAndPlace (eight runs for each method). We plot the skill trajectories of the object and the gripper with different colors. Note that while LSD mostly just throws the object away, CSD always learns to pick up the object in all eight runs.

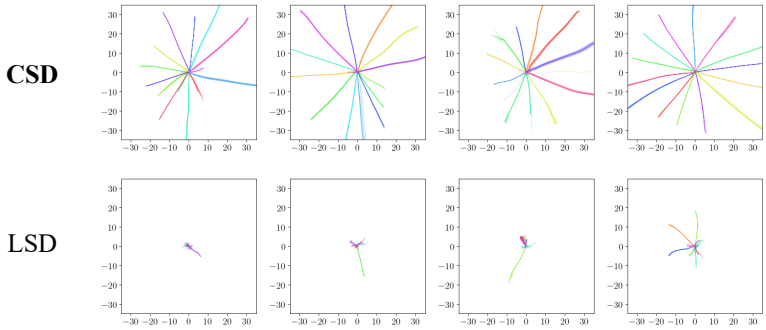
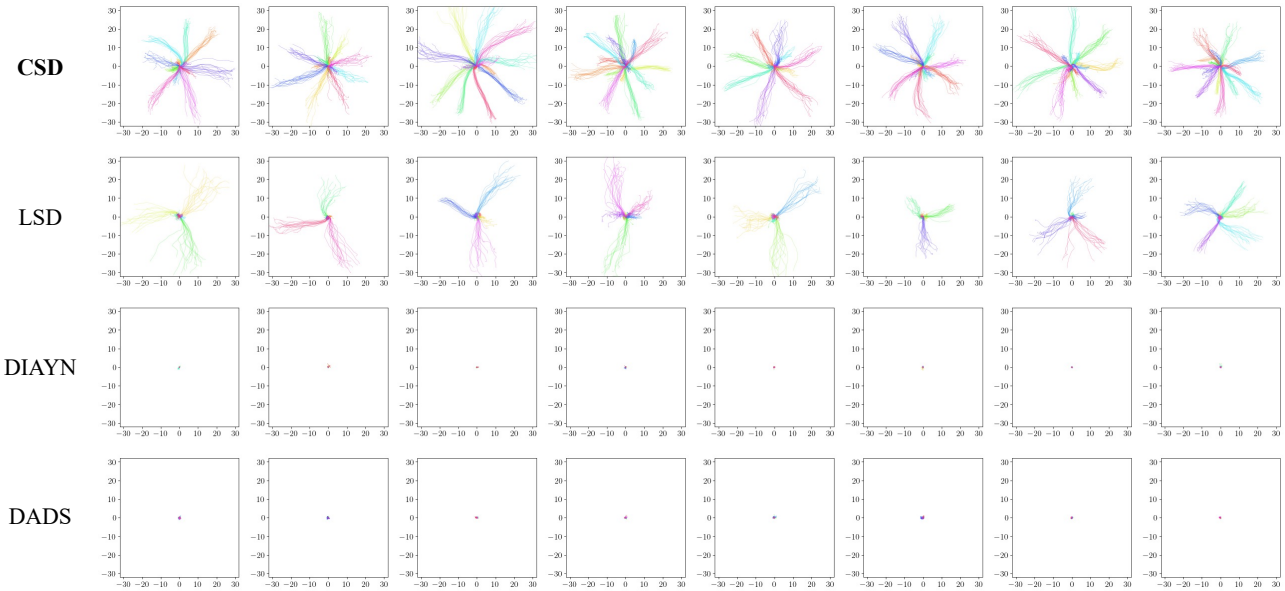
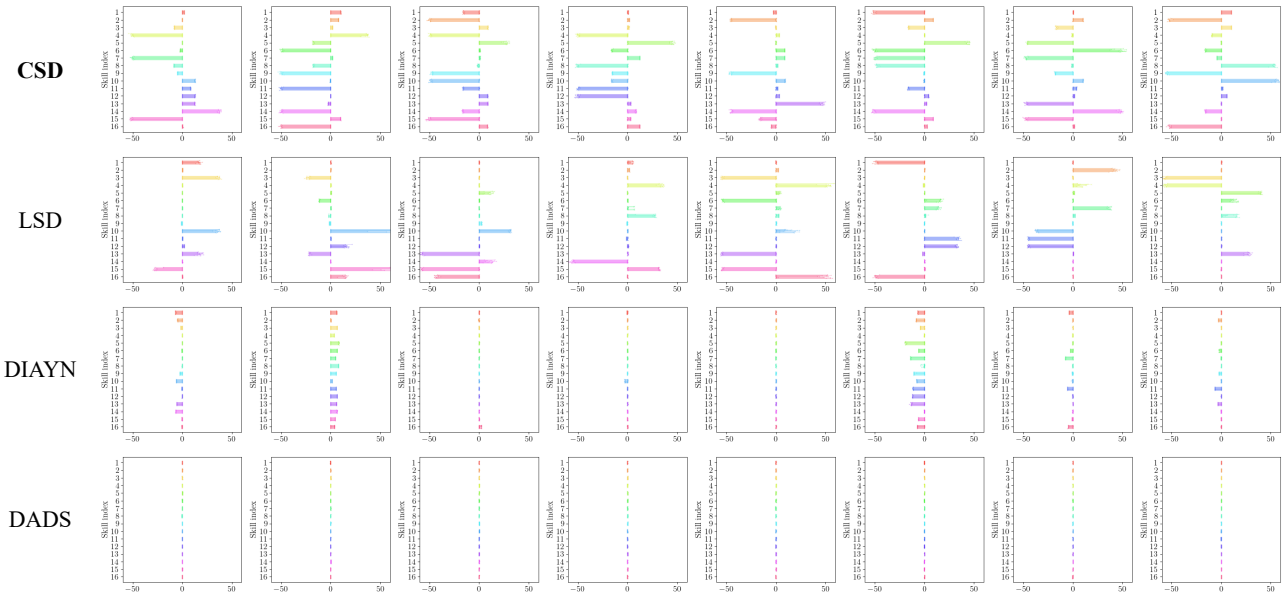


Figure 18. Complete qualitative results in Humanoid (four runs for each method in each environment). We plot the skill xy trajectories of the agent with different colors. We note that we train CSD and LSD for 100K episodes (which is a tenth of the number of episodes used in the LSD paper (Park et al., 2022)).



(a) Ant xy trajectories



(b) HalfCheetah x trajectories

Figure 19. Complete qualitative results in Ant and HalfCheetah (eight runs for each method in each environment). We plot the skill trajectories of the agent with different colors. We note that in both environments, CSD not only learns locomotion skills but also discovers a variety of diverse skills, such as rotating and flipping.

E. Implementation Details

For manipulation environments, we implement CSD on top of the publicly available codebase of MUSIC (Zhao et al., 2021). For MuJoCo environments, we implement CSD based on the publicly available codebase of LSD (Park et al., 2022). We mostly follow the hyperparameters used in the original implementations. Our implementation can be found in the following repositories: https://github.com/seohongpark/CSD-manipulation (manipulation environments) and https://github.com/seohongpark/CSD-locomotion (locomotion environments). We run our experiments on an internal cluster with NVIDIA Tesla V100 and NVIDIA GeForce RTX 2080 Ti GPUs. Each run mostly takes a day or less.

E.1. Environments

We adopt the same environment settings used in LSD (Park et al., 2022) for Fetch manipulation environments (FetchPush, FetchSlide, FetchPickAndPlace) (Plappert et al., 2018) and MuJoCo locomotion environments (Ant, HalfCheetah) (Todorov et al., 2012; Brockman et al., 2016). In Fetch environments, unlike LSD, we do not use any supervision, such as limiting the discriminator's input only to the object. For the Kitchen environment, we use a 7-DoF end-effector controller (Mendonca et al., 2021) with state-based observations. We use an episode length of 200 for locomotion environments and an episode length of 50 for manipulation environments. In locomotion environments, to ensure fair comparisons, we use preset normalizers for all skill discovery methods as done in Park et al. (2022), but we find that CSD can still discover diverse behaviors including locomotion skills without a normalizer.

E.2. Downstream Tasks

Fetch environments. We use the same downstream tasks in Park et al. (2022) for Fetch environments. In FetchPushGoal, FetchSlideGoal, and FetchPickAndPlaceGoal, a goal position is randomly sampled at the beginning of each episode. If the agent successfully places the object to the target position, a reward of 1 is given to the agent and the episode ends. We follow the original goal sampling range and reach criterion from Plappert et al. (2018).

Kitchen environment. We consider the following 13 downstream tasks for the Kitchen environment: BottomLeftBurner, BottomRightBurner, HingeCabinet, KettleBottomRight, KettleFall, KettleLift, KettleTopLeft, KettleTopRight, LightSwitch, Microwave, SlideCabinet, TopLeftBurner, TopRightBurner. For the success criteria of the tasks, we mostly follow Gupta et al. (2019); Mendonca et al. (2021) and refer to our implementation for detailed definitions. As in the Fetch tasks, the agent gets a reward of 1 when it satisfies the success criterion of each task.

MuJoCo locomotion environments. In AntGoal, a goal's xy position is randomly sampled from Unif$([-20, 20]^2)$, and if the agent reaches the goal, it gets a reward of 10 and the episode ends. In AntMultiGoals, the agent should follow four goals within 50 steps each, where goal positions are randomly sampled from Unif$([-7.5, 7.5]^2)$ centered at the current coordinates. The agent gets a reward of 2.5 every time it reaches a goal. In HalfCheetahGoal, a goal's x coordinate is randomly sampled from Unif$([-60, 60])$, and if the agent reaches the goal, it gets a reward of 10 and the episode ends. For these three environments, we consider the agent to have reached the goal if it enters within a radius of 3 from the goal. In HalfCheetahHurdle, the agent gets a reward of 1 if it jumps over a hurdle, where we use the same hurdle positions from Qureshi et al. (2020).

E.3. Training.

Skill policy. At the beginning of each episode, we sample a skill z from either a standard Gaussian distribution (for continuous skills) or a uniform distribution (for discrete skills), and fix the skill throughout the episode. For discrete skills, we use standard one-hot vectors for DIAYN and DADS, and zero-centered one-hot vectors for CSD and LSD, following Park et al. (2022). For DADS, we follow the original implementation choices, such as the use of batch normalization and fixing the output variance of the skill dynamics model. For CSD in manipulation environments, we start training the skill policy from epoch 4000, after the initial conditional density model has stabilized. When modeling $\Sigma_\theta(s)$ of the conditional density model, we use a diagonal covariance matrix as we found it to be practically sufficient for our experiments. Also, we normalize the diagonal elements with their geometric mean at each state for further stability.

We present the full list of the hyperparameters used in our experiments in Tables 1 and 2, where we indicate the values considered for our hyperparameter search with curly brackets. For the intrinsic reward coefficient, we use 50 (DADS), 500 (CSD and LSD), 1500 (DIAYN), 200 (Disagreement Fetch), or 50 (Disagreement Kitchen). For the learning rate, we use

Table 1. Hyperparameters for manipulation environments.

Hyperparameter	Value
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	10^{-3}
# training epochs	40000 (Fetch), 20000 (Kitchen)
# episodes per epoch	2
# gradient steps per episode	10
Episode length	50
Minibatch size	256
Discount factor γ	0.98
Replay buffer size	10^5
# hidden layers	2
# hidden units per layer	256
Nonlinearity	ReLU
Target network smoothing coefficient τ	0.995
Random action probability	0.3
Action noise scale	0.2
Entropy coefficient	0.02
Intrinsic reward coefficient	$\{5, 15, 50, 150, 500, 1500, 5000\}$
CSD ϵ	10^{-6}
CSD initial λ	3000
Disagreement ensemble size	5

0.0001 for all experiments except for CSD Humanoid, for which we find 0.0003 to work better (we also test 0.0003 for LSD Humanoid for a fair comparison, but we find the default value of 0.0001 to work better for LSD). For the reward scale, we use 1 (LSD, DIAYN, and DADS) or 10 (CSD). For the SAC α, we use 0.003 (LSD Ant and LSD HalfCheetah), 0.03 (CSD Ant and LSD Humanoid), 0.1 (CSD HalfCheetah), 0.3 (CSD Humanoid), or auto-adjust (DIAYN and DADS).

High-level controller. After unsupervised skill discovery, we train a high-level controller $\pi^h(z|s, g)$ that selects skills in a sequential manner for solving downstream tasks. We use SAC (Haarnoja et al., 2018a) for continuous skills and PPO (Schulman et al., 2017) for discrete skills. The high-level policy selects a new skill every R steps. We mostly follow the hyperparameters for low-level skill policies and present the specific hyperparameters used for high-level controllers in Tables 3 and 4.

[3]The original LSD implementation updates the target network every epoch, not every gradient step, but we find the latter to be about $10\times$ sample efficient in terms of the number of environment steps.

Table 2. Hyperparameters for locomotion environments.

Hyperparameter	Value
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	$\{0.0001, 0.0003\}$
# training epochs	20000
# episodes per epoch	5 (Humanoid), 10 (others)
# gradient steps per epoch	64 (policy), 32 (others)
Episode length	200
Minibatch size	1024
Discount factor γ	0.99
Replay buffer size	1000000 (Humanoid), 2000 (others)
# hidden layers	2
# hidden units per layer	1024 (Humanoid), 512 (others)
Nonlinearity	ReLU
Target network smoothing coefficient τ	0.995
Target network update frequency	every gradient step[3]
SAC α	$\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3,$ auto-adjust (Haarnoja et al., 2018b)$\}$
Reward scale	$\{1, 10\}$
CSD ϵ	10^{-6}
CSD initial λ	3000
Disagreement ensemble size	5

Table 3. Hyperparameters for SAC downstream policies in manipulation environments.

Hyperparameter	Value
# training epochs	4000 (Fetch), 8000 (Kitchen)
# episodes per epoch	16 (Fetch), 2 (Kitchen)
# gradient steps per epoch	4 (Fetch), 10 (Kitchen)
Replay buffer size	10^6
Skill sample frequency R	10
Skill range	$[-1.5, 1.5]^D$

Table 4. Hyperparameters for PPO downstream policies in locomotion environments.

Hyperparameter	Value
Learning rate	3×10^{-4}
# training epochs	1000
# episodes per epoch	64
# gradient steps per episode	10
Minibatch size	256
Entropy coefficient	0.01
Skill sample frequency R	25