Learning Action-based Representations Using Invariance

Max Rudolph[*1], **Caleb Chuck**[*1], **Kevin Black**[*2], **Misha Lvovsky**[3], **Scott Niekum**[3], **Amy Zhang**[1]
The University of Texas at Austin[1]
University of California, Berkeley[2]
University of Massachusetts Amherst[3]

Abstract

Robust reinforcement learning agents using high-dimensional observations must be able to identify relevant state features amidst many exogeneous distractors. A representation that captures *controllability* identifies these state elements by determining what affects agent control. While methods such as inverse dynamics and mutual information capture controllability for a limited number of timesteps, capturing long-horizon elements remains a challenging problem. Myopic controllability can capture the moment right before an agent crashes into a wall, but not the control-relevance of the wall while the agent is still some distance away. To address this we introduce *action-bisimulation encoding*, a method inspired by the bisimulation invariance pseudometric, that extends single-step controllability with a recursive invariance constraint. By doing this, action-bisimulation learns a multi-step controllability metric that smoothly discounts distant state features that are relevant for control. We demonstrate that action-bisimulation pretraining on reward-free, uniformly random data improves sample efficiency in several environments, including a photorealistic 3D simulation domain, Habitat. Additionally, we provide theoretical analysis and qualitative results demonstrating the information captured by action-bisimulation. Code and video: https://maxrudolph1.github.io/action-bisimulation-site/

1 Introduction

Learning control for complex decision-making from high-dimensional observation spaces such as video and depth is vital for real-world applications of reinforcement learning (RL). To do this, a representation of the observation space allows agents to reason about the environment and take intelligent actions. However, learning these representations is often sample inefficient. One reason for this is that real-world scenarios often contain many irrelevant and distracting features embedded in a high-dimensional space. Correlating reward with relevant state elements, and not causally confusing distractors in this setting, is challenging—especially since reward signals are often sparse.

Representation learning has emerged as a promising approach to address this challenge by extracting a compressed and informative representation of the observation space that is useful for learning (Bengio et al., 2013). Representation learning removes irrelevant distractors from the state space used to learn the policy, which improves sample efficiency and performance. In RL, task-specific representation learning uses reward or expert behavioral similarity (Ferns et al., 2011; Agarwal et al., 2021) to discover the compressed representation, only describing task-specific elements. This has the advantage of capturing only information that is either useful for solving the task or relevant to the demonstrations while being limited by requiring either expert behavior or task-achieving policies, both of which

* Authors contributed equally, corresponding author mrudolph@cs.utexas.edu

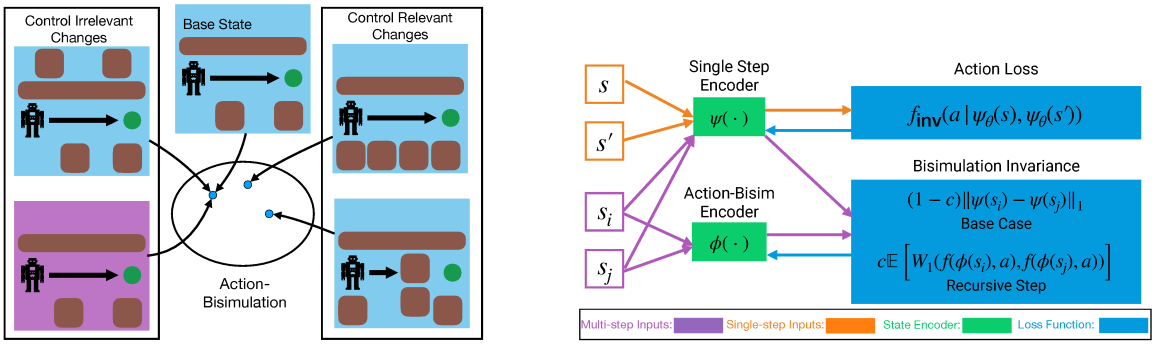


Figure 1: **Left: mapping equivalent** *controllability* together with action-bisimulation. Other methods can be too aggressive (single-step inverse dynamics would map together **(a)** and **(e)**, reward-based methods would map together **(a)** and **(d)**), or permissive (autoregressive methods would map **(a)** and **(c)** to the same value). **Right: data flow** of action-bisimulation training. The single-step encoder is trained with inverse dynamics (Section 4.1). The multi-step encoder is trained with bootstrapped single-step representation distance (Equation 6).

can be difficult to obtain prior to learning. On the other hand, task-free methods use unsupervised signals like reconstruction (Lange & Riedmiller, 2010) and contrastive objectives (Laskin et al., 2020) and can be pre-trained on any data, including random actions. However, these methods are trained without action information. As a result they can capture exogenous distractors that are not useful for improving RL policy performance.

One promising direction of task-agnostic methods utilizes *controllability* to learn a behavior-relevant representation that is not task-specific (Lamb et al., 2022). These representations can avoid capturing task-irrelevant information while not requiring expert or reward-achieving behavior. Recent work in action-based representation learning for RL has shown promising results (Zhang et al., 2022) by utilizing inverse dynamics models to extract representations (Islam et al., 2022). These representations rely on a window of information by predicting the first action between two states separated by k-steps. If k is small this representation is myopic, but when k is large the prediction problem is underspecified. This underspecification restricts large k to offline datasets with correlated action data—such as expert trajectories.

We investigate utilizing a novel invariant metric to learn a multi-step control-based representation instead of directly applying k-step prediction. Our *action-bisimulation metric* offers a novel framework for controllability metrics that takes a myopic dynamics encoding and extends it to multi-step representations. This formulation is inspired by reward bisimulation (Zhang et al., 2020b), which utilizes single-step reward information to learn multi-step return-capturing representations. Action-bisimulation applies bootstrapping on the myopic $k = 1$ controllability representations to enforce multi-step invariance in an *action-bisimulation encoding*. Since the base case uses single-step prediction, the encoding can be trained with any offline data, even fully random. At the same time, boostrapping extends the action-bisimulation encoding to capture long-term controllability. Figure 1 captures how action-bisimulation maps control-irrelevant states together, while not doing the same for control-relevant states.

This work offers an empirical analysis and theoretical formulation of the novel control-based invariant metric for representation learning. We demonstrate empirically that in scenarios where complex, long-horizon, sparse-reward decision-making is required, the metric improves sample efficiency compared to RL agents trained directly from pixels, or pre-trained with existing representation learning methods in multiple domains. Next, we provide qualitative results demonstrating the robustness of the learned representation to uncontrollable distractors, as well as sensitivity to control-relevant state features.

2 Related Works

Representation Learning in RL. Learned representations have been widely applied to RL, formalized (Li et al., 2006) through hierarchical symbolic representations (Konidaris et al., 2014; Andre & Russell, 2002), skill abstractions (Dietterich, 2000), policy optimality (Auer et al., 2008; Jong & Stone, 2005; Abel et al., 2016), selective attention (Jones & Canas, 2010) and contingency awareness (Bellemare et al., 2012). One effective strategy is to use the representation to learn a model that is effective for planning (Hafner et al., 2019; Koul et al., 2023). These methods learn world models (Ha & Schmidhuber, 2018) and other representations that can be used for prediction (Singh et al., 2012), data generation and planning. Alternatively, other methods apply representation learning for filtering (Krishnan et al., 2015; Karl et al., 2016) or reduced complexity (Higgins et al., 2016; Oord et al., 2018; Laskin et al., 2020) representations. Action-bisimulation is a novel encoder that learns controllability-based representations to improve RL performance. Unlike other representation learning methods, action-bisimulation uses a soft invariance pseudometric to capture action information through time.

Action-based Representations. RL methods have directly leveraged action-relevant representations in several ways. This includes contingency awareness (Bellemare et al., 2012; Choi et al., 2018; Chuck et al., 2020; 2023), which is closely related to action controllability (Zhong et al., 2020) and control information measures like empowerment (channel capacity between actions and state) (Jung et al., 2011; Mohamed & Jimenez Rezende, 2015; Levy et al., 2023) or affordances (Cruz et al., 2016; Khetarpal et al., 2020; Nagarajan et al., 2020). Multi-step inverse models are most similar to action-bisimulation, but common multi-step inverse methods (Lamb et al., 2022; Islam et al., 2022; Koul et al., 2023) require selecting a specific k for the multi-step horizon, potentially leaving critical control information on the table. Further, it has been shown that multi-step inverse models can be insufficient when the dynamics are periodic (Levine et al., 2024). Action-bisimulation uses a soft invariance metric to extend single-step models, which better preserves long-term controllability.

Bisimulation methods. Bisimulation describes future invariant state representations, originally applied to stationary representations (Larsen & Skou, 1989; Dean et al., 1997; Ferns et al., 2004), before being extended to continuous state MDPs (Ferns et al., 2011). Reward-based bisimulation methods have gained popularity through learned deep representations (Zhang et al., 2020b). This has been extended to non-optimal policies (Castro et al., 2021), with generalized value function bounds (Kemertas & Aumentado-Armstrong, 2021) and augmented with state discretization (Kemertas & Jepson, 2022) and clustering (Liu et al., 2023). Bisimulation-based methods have also been applied in different contexts: expert policy similarity (Agarwal et al., 2021; Bertran et al., 2022; Mazoure et al., 2021), goal-conditioned RL (Hansen-Estruch et al., 2022) and reward-action policy equivalence (Liao et al., 2023; Castro, 2020). While this work draws on reward-bisimulation, action-bisimulation is fundamentally offline and task-agnostic because it takes an expectation over actions, removing its dependence on any policy.

3 Preliminaries

A Markov decision process is defined by the tuple $\mathcal{M} \coloneqq (\mathcal{S}, \mathcal{A}, p, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space and $s \in \mathcal{S}, a \in \mathcal{A}$ are states and actions respectively. $p(s'|s,a)$ is the transition function that gives the probability of the next state s' given the current state and action (s,a). The reward function $R(s,a)$ maps state and action to a scalar reward. A policy $\pi(a|s)$ is the probability of an action given the current state.

This work utilizes the following two-phase paradigm: in the first phase, the agent first takes actions without access to the reward function $R(s,a)$ to generate a dataset of ordered state action tuples $\mathcal{D} \coloneqq \{(s^{(0)}, a^{(0)}), \dots (s^{(|\mathcal{D}|-1)}, a^{(|\mathcal{D}|-1)})\}$. Then, a representation $\phi : \mathcal{S} \to \mathcal{Z}$ is learned from $\mathcal{S}$. In the second phase, the agent learns from extrinsic reward utilizing the learned representation.

The action-bisimulation representation method is inspired by reward bisimulation (Dean et al., 1997). In RL, bisimulation is a state abstraction that groups reward-equivalent states:

Definition 3.1 (Bisimulation Relations (Givan et al., 2003))**.** In MDP $\mathcal{M}$, an equivalence relation B between states is a bisimulation relation if: $\forall s_i, s_j \in \mathcal{S}$ where the states are equivalent under B ($s_i \equiv_B s_j$), the following conditions hold:

$$R(s_i, a) = R(s_j, a) \quad \forall a \in \mathcal{A} \tag{1}$$
$$P(\mathcal{G}|s_i, a) = P(\mathcal{G}|s_j, a) \quad \forall a \in \mathcal{A}, \forall \mathcal{G} \in \mathcal{S}_B \tag{2}$$

where $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups $\mathcal{G}$ of equivalent states), and $P(\mathcal{G}|s, a) = \sum_{s' \in \mathcal{G}} p(s'|s, a)$

Bisimulation Metrics (Ferns et al., 2011; Castro, 2020) soften the notion of state partitions with a pseudometric space $(\mathcal{S}, d)$, where distance function $d : \mathcal{S} \times \mathcal{S} \to R_{\geq 0}$ measures the similarity between two states.[1] The on-policy bisimulation metric (Kemertas & Aumentado-Armstrong, 2021) is:

$$d_{\text{r-bisim}}(s_i, s_j) = \max_a \underbrace{(1-c) \cdot |R(s_i, a) - R(s_j, a)|}_{\text{base case}} + \underbrace{cW_1(d)(p(\cdot|s_i, a), p(\cdot|s_j, a))}_{\text{recursive step}}, \tag{3}$$

where $W_1(d)$ is the 1-Wasserstein distance and c is a scalar hyperparameter that weights the multi-step sensitivity of the distance. The 1-Wasserstein metric measures the distance between next-state distributions in the latent bisimulation space. We propose a novel controllability-based relation, which replaces reward equivalence with single-step control equivalence. By replacing rewards in the equivalence, the relation is task-agnostic.

Definition 3.2 (Action-Bisimulation Relations)**.** Let $\psi : \mathcal{S} \to \mathcal{Z}_{ss}$ be a single step controllability encoder such that $p(a|\psi(s), \psi(s')) = p(a|s, s')$ for all s, a, s'. In MDP $\mathcal{M}$, an equivalence relation AB between states is an action-bisimulation relation according to ψ if: $\forall s_i, s_j \in \mathcal{S}$ where the states are equivalent under AB ($s_i \equiv_{AB} s_j$), the following conditions hold:

$$\psi(s_i) = \psi(s_j) \tag{4}$$
$$P(\mathcal{G}|s_i, a) = P(\mathcal{G}|s_j, a) \quad \forall a \in \mathcal{A}, \forall \mathcal{G} \in \mathcal{S}_{AB} \tag{5}$$

where $\mathcal{S}_{AB}$ is the partition of $\mathcal{S}$ under the relation AB (the set of all groups $\mathcal{G}$ of equivalent states), and $P(\mathcal{G}|s, a) = \sum_{s' \in \mathcal{G}} p(s'|s, a)$

This equivalence can be similarly relaxed into a pseudometric. However, in the off-policy setting, we are not interested in a particular policy, but all policies. Thus, action-bisimulation uses the expectation over uniform actions to encode all possible policies.

$$d_{\text{a-bisim}}(s_i, s_j, \psi) = \underbrace{(1-c) \cdot \|\psi(s_i) - \psi(s_j)\|_1}_{\text{base case}} + \underbrace{c \cdot \mathbb{E}_{a \sim U(\mathcal{A})} \left[W_1(p(\cdot|s_i, a), p(\cdot|s_j, a))\right]}_{\text{recursive step}} \tag{6}$$

In the next section, we describe how $\psi(\cdot)$ can be learned from data, and how to use $d_{\text{a-bisim}}$ to learn an action-bisimulation encoder.

4 Methods

This section describes the algorithm for training an action-bisimulation encoder. First, the single-step encoder is learned, then the distance in single step space is used as the "base case" for the recursive step. The training flow and inputs are visualized in Figure 1b.

4.1 Single-Step Controllability

Inverse dynamics describes the probability of an action given two sequential states (s, s'): $P(a|s, s')$. To get a single-step encoding of the action-relevant state features we define the single step state

[1] This is a pseudometric, meaning that two different states can have 0 distance.

encoder $\psi_\theta(s) : \mathcal{S} \to \mathcal{Z}_{ss}$, where $\mathcal{Z}_{ss}$ is the embedded single-step space (Lamb et al., 2022), and ψ_θ is parameterized by θ. Then, for dataset $\mathcal{D}$ of (s, a, s') tuples, the regularized single-step representation is learned by optimizing the single-step (ss) inverse dynamics loss:

$$L_{ss}(\mathcal{D}, \theta, \nu) = - \sum_{(s,a,s') \sim \mathcal{D}} \log f_{\nu,\text{inverse}}(a|\psi_\theta(s), \psi_\theta(s')) + \beta \left(\|\psi_\theta(s)\|_1 + \|\psi_\theta(s')\|_1 \right), \tag{7}$$

where $f_{\nu,\text{inverse}}$ is a learned inverse dynamics model parameterized by ν. The regularization ensures that the learned representation includes the minimum information necessary to capture the action-dependent inverse dynamics.

This inverse model is optimized to predict a distribution over actions $P(\cdot|\psi_\theta(s), \psi_\theta(s'))$ using the single-step embeddings as inputs. In this work, we represent the parameters of the distribution as a function of $[\psi_\theta(s), \psi_\theta(s')]$. Intuitively, $\psi(\cdot)$ embeds control-relevant features by embedding action-relevant components of the state. We use a relatively weak inverse model under the intuition that the simpler the model used to capture inverse dynamics, the more information is forced into the embedding rather than the inverse dynamics model.

4.2 Action-Bisimulation Metric

This section describes how the action-bisimulation metric (Equation 6) is used to learn an encoder $\phi_\eta(s) : \mathcal{S} \to \mathcal{Z}$, where $\mathcal{Z}$ is the representation space and ϕ_η is parameterized by η. This definition uses the single-step representation space $\mathcal{Z}_{ss}$ to define the multi-step representation space $\mathcal{Z}$.

The recursive step $\mathbb{E}[cW_1(d)(p(\cdot|s_i, a), p(\cdot|s_j, a))]$ requires computing $p(\cdot|s_i, a)$ and $p(\cdot|s_j, a)$. This can be done by learning a forward model parameterized by υ: $f_\upsilon(\phi_\eta(s_i), a) : \mathcal{Z} \times \mathcal{A} \to P(\cdot|\phi_\eta(s_i), a)$ that takes in the state embedding and action and outputs a probability distribution over the next embedded state. We model this by outputting the parameters of a conditional Gaussian model $\mathcal{N}(\mu, \Sigma)$ following the practice of Zhang et al. (2020b). Using the notation $f(\phi_\eta(s_i), a)[s']$ to denote the probability of state s' under the distribution $f_\upsilon(\phi_\eta(s_i), a)$, we train the forward model by minimizing the negative log-likelihood of the observed data in $\mathcal{D}$:

$$L_{\text{forward}}(D) = - \sum_{s,a,s' \sim D} \log f_\upsilon(\phi_\eta(s), a)[\phi_\eta(s')]. \tag{8}$$

In deterministic dynamics, the 1-Wasserstein distance equals the l_1 distance of the mean. $f_\upsilon(\cdot)$ is a function of the encoded state $\phi(s)$ rather than the observation s because forward dynamics over the observations is more costly due to the inherent reconstruction objectives they minimize; this reconstruction could bring in uncontrollable elements and does not inherently include control centric components.

In the off-policy setting, we propose using one of two expectations for the recursive step: over the uniform distribution of actions $E_{a \sim U(\mathcal{A})}$ or over the behavior distribution: $E_{a \sim \pi_b(s_i)}$. The use of the behavioral distribution applies to settings where random actions might restrict the distribution of observed states. In practice, these are computed using the empirical mean. Then, the action-controllability bisimulation metric using learned models is:

$$d_{\text{a-bisim}}(s_i, s_j, \psi_\theta, \phi_\eta) = (1 - c) \cdot \|\psi_\theta(s_i) - \psi_\theta(s_j)\|_1 + c \cdot \mathbb{E}_{a \sim U(\mathcal{A})} \left[W_1(f(\phi_\eta(s_i), a), f(\phi_\eta(s_j), a)) \right] \tag{9}$$

To train the encoder, we match the l_1 distance between the embedded representations $\phi(s_i), \phi(s_j)$ to the metric distance:

$$L(\mathcal{D}) = \frac{1}{N} \sum_{s_i, s_j \sim \mathcal{D}} \left| \|\phi_\eta(s_i) - \phi_\eta(s_j)\|_1 - d_{\text{a-bisim}}(s_i, s_j, \psi, \phi) \right|. \tag{10}$$

In practice the parameters of ϕ used to calculate $d_{\text{a-bisim}}$ are trailed behind ϕ_η with the exponential moving average: $\phi = \tau\phi_\eta + (1 - \tau)\phi$.

Algorithm 1: Action-bisimulation Encoder Learning

Input: Dataset without reward $(s, a, s') \sim \mathcal{D}$, initial encoder $\phi^{\bar{\theta}}(s)$
Single-step Training Train ψ with $\mathcal{D}$ and Equation 7.
repeat
  Forward Model Update: Update the forward model $f_\upsilon(\cdot)$ according to the current multi-step encoder ϕ using Equation 8.
  Multi-step Update: Sample $s_i, s_j \sim \mathcal{D}$ pairs and minimize the loss as defined by the metric in Equation 9 and loss (Equation 10) to update the encoder parameters θ.
  Momentum Update: Update the parameters: $\bar{\theta} = \tau\theta + (1-\tau)\bar{\theta}$
until $\bar{\theta}$ converge

(a) Nav2D (b) Pointmaze (c) Distractor Pointmaze (d) Habitat environments

Figure 2: Visual representation of the RL environments.

5 Experiments

In this section, we aim to answer the following questions: **1)** Does pre-training with the action-bisimulation objective learn representations useful for arbitrary downstream tasks? **2)** How does this pretraining compare with existing methods, especially single-step action controllability? **(3)** Are the learned representations robust to background distractors? **(4)** How well does the action-bisimulation procedure capture multi-step relationships between state elements?

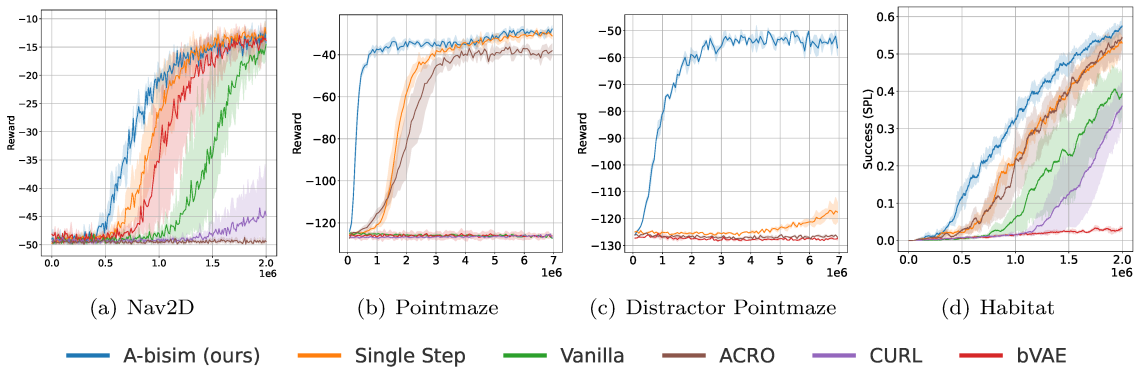
We evaluate experiments in three domains illustrated in Figure 3. **Nav2D** is a 15x15 grid environment where the agent navigates using cardinal directions to the center of the grid, avoiding randomly generated 2x2 obstacles. **Pointmaze** (Fu et al., 2020) is a 2D Mujoco control environment where the agent takes actions to reach a goal location while being impeded by obstacles. We also investigate **Distractor Pointmaze** where the background in Pointmaze has been replaced with photorealistic distractions in the form of video clips. Finally, **Habitat** (Savva et al., 2019b) is a complex 3D environment where the agent must navigate through scans of human environments to reach a goal location. Additional environment details are in Appendix H (number of obstacles, goal/grid size, randomization, etc.) and all other relevant hyperparameters are in Appendix I.

5.1 Baselines

We compare the performance of our method against representation learning pretraining methods used in prior RL works that utilize control-, contrastive- and reconstruction-based objectives.

Single-Step Inverse (SSI): This baseline uses the single-step objective learned using Equation 7 with $k = 1$ to learn a state representation. This demonstrates whether simply learning a myopic action-centric inverse dynamics representation is sufficient for good performance. In general, this representation performs surprisingly well.

Agent Centric Representations for Offline RL (ACRO) (Lamb et al., 2022): This method is equivalent to SSI with $k \neq 1$. When $k > 1$, this means that the model must learn to identify the first action taken from a pair of states. While this allows the model to capture longer-term relationships, it also limits how effective it can be when trained with random actions.



(a) Nav2D (b) Pointmaze (c) Distractor Pointmaze (d) Habitat

Figure 3: Training **Performance Curves** for downstream RL learning in (a) **2D navigation**, (b) **Pointmaze navigation**, (c) **Distractor Pointmaze** and (d) **Habitat**. Mean and 95% confidence interval are plotted over 5 trials with different random seeds for each domain.

β-**Variational Autoencoder** (bVAE) (Higgins et al., 2016): This method evaluates a classic compressed state reconstruction method for representation learning. While popularized with video, it has been applied to RL with marginal success. In general, reconstruction can struggle to pick up fine-grained changes such as the movement of the agent.

Contrastive Unsupervised Representations for Reinforcement Learning (CURL) (Laskin et al., 2020): This method uses data augmentation with a contrastive objective to learn a representation. In this work, we used random noise augmentations because of the importance of identifying small features (the location of the agent).

Vanilla RL (Schulman et al., 2017; Mnih et al., 2013): Trains either Deep Q-networks (DQN) in Gridworld, or Proximal Policy Optimization (PPO) in the remaining domains, from scratch.

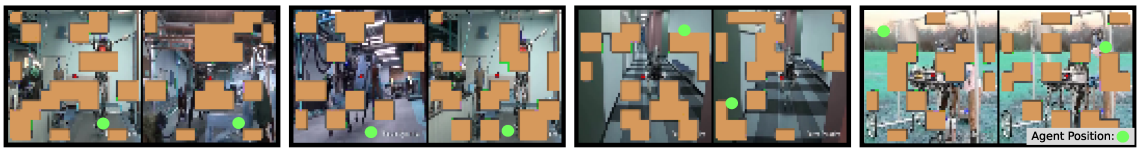
5.2 Downstream Learning

To evaluate downstream learning we first gather an offline dataset of random action state transitions, with sizes recorded in Table 2. State encoder $\phi(s)$ is trained with Equation 10, which is used to initialize the policy $\pi_\theta(\cdot|\phi(s))$. This fine-tuning strategy proved to be the best performing empirically, though future work could investigate freezing the encoder, the technique used in Lamb et al. (2022), as we discuss in Appendix E. Figure 3 illustrates the comparison of the action-bisimulation encoder to baseline encodings.

As we can see in Figure 3, learning with the action-bisimulation representation outperforms other methods in terms of sample efficiency, even k-step controllability (ACRO), by a substantial margin. This provides evidence for **hypotheses 1 and 2**, that action-bisimulation learns useful representations which compare well with other methods. That reconstruction and data augmentation-based methods bVAE and CURL perform poorly is not unexpected: in this domain, the agent is often small, so these methods achieve low reconstruction loss even when they omit the most important element: the agent position. On the other hand, SSI captures the agent position but is highly myopic, limiting transfer to downstream tasks. We hypothesize ACRO struggles because it relies on predicting action from two states separated by k timesteps, which is ill-posed, especially when using a dataset of random actions. Additional details on baselines can be found in Appendix G.

5.3 Background Distractors

In this section we evaluate **hypothesis 3**: whether the action-bisimulation encoding is robust to distractors. We assess this through a modified Pointmass environment with a photorealistic visual background. The foreground, that is the agent, goal position, and obstacles, remain the same. We visualize the distractor environments in Figure 4, where the agent has been exaggerated.



(a) Action-bisimulation (b) Single-Step (c) ACRO (d) β-VAE

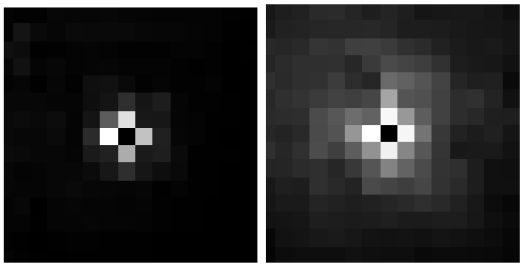
Figure 4: **States close together in embedding space** drawn from the Distractor Pointmaze domain. Notice that action-bisimulation captures both the agent location and the local region of obstacles, while other methods are distracted by the background (ACRO, bVAE) or only capture one-step relations (single step). The agent is exaggerated in these images so it is easier to locate—in reality, it is quite hard to detect because of the distractors.

Figure 3c shows that adding background distractors dramatically widens the gap between action-bisimulation and other methods. These backgrounds make vanilla RL, reconstruction and data-augmentation-based methods struggle wildly since these methods have no built-in robustness. They also have a significant effect, even on the fixed-step models, ACRO, and SSI. For single-step models, we hypothesize this is because pretraining causes the agent to mostly ignore obstacles since they have a limited myopic effect. For ACRO, the correlated background images appear to confuse the k-step prediction. For action-bisimulation, by contrast, there is only a marginal difference.

We also illustrate how a few representative methods map together states in Figure 4. In these plots, two nearby states are sampled and visualized. As we can see, action-bisimulation and single-step encodings encode the agent position, but action-bisimulation also maps regions of similar local obstacles together. Beta-VAE (bVAE) encodings are trained with reconstruction; the encodings largely ignore the agent in favor of matching similar backgrounds. Interestingly, ACRO also maps similar backgrounds together. We think this is because of the correlation between subsequent frames in the video, though this is worth further investigation.

5.4 Captured Representations

To investigate **hypothesis 4**: how well the action-bisimulation encodings capture multi-step relationships, we provide qualitative visualizations comparing the multi-step and single-step encodings. Figures 5 is a **perturbation map**, which visualizes how much the representation changes when a single obstacle is placed at a particular location, compared with the base representation. Figure 5 illustrates the contrast between the myopia of the single-step encoder compared with the range of the multi-step encoder.



(a) (b)

Figure 5: **Perturbation map of single step vs action-bisimulation** shows encoder distance in 2D Navigation when obstacles are toggled at all locations around the agent (located at the center). Brightness at a pixel indicates the size of the change of representation. **Left: The Single Step** encoder myopically captures only directly adjacent obstacles. **Right: The Multi Step** encoder captures more distant obstacles.

In Appendix D, we provide several additional qualitative results demonstrating how the action-bisimulation representation captures multi-step relations, including perturbation plots of how the sensitivity changes with c, the tradeoff parameter, and the representation difference from near-vs-far perturbations. Furthermore, sensitivity to perturbations is environment-dependent: if the environment has a fixed structure such as a corridor or maze, unreachable obstacle perturbations will be mapped close together in the action-bisimulation space.

6 Conclusion

Controllability-capturing encodings for reinforcement learning are a promising direction for representation pretraining since they can be learned without reward but are still able to filter out uncontrollable distractors. However, existing methods either only capture short-term controllability or are dependent on demonstration data, which has implicit task bias. We introduce the action-bisimulation encoding, which builds off of myopic representations by enforcing recursive invariance to learn a supervision-free multi-step controllability representation. The empirical results in this work demonstrate how these encodings can be used to improve the sample efficiency, especially in domains with significant background distractors. The primary limitation of this method is the inverse dynamics single-step model, which might not capture *all* controllable features, just a subset. This can result in the representation being agnostic to important task elements. A more in-depth discussion of limitations is included in Appendix F. Altogether, action-bisimulation is a novel invariance relation for capturing controllability from offline data that removes expert performance requirements and smoothly handles long-horizon controllability.

7 Acknowledgements

This work has taken place in part in the Safe, Correct, and Aligned Learning and Robotics Lab (SCALAR) at The University of Massachusetts Amherst. SCALAR research is supported in part by the NSF (IIS-2323384), AFOSR (FA9550-20-1-0077), and the Center for AI Safety (CAIS). The work was supported by the National Defense Science & Engineering Graduate (NDSEG) Fellowship sponsored by the Air Force Office of Science and Research (AFOSR). Special thanks to collaborators Stephen Guigere, Harshit Sikchi, Alex Levine, Siddhant Agarwal, Rudolf Lioutikov, Yuchen Cui, Akanksha Saran, Wonjoon Goo, Daniel Brown, Prasoon Goyal, Christina Yuan, and Ajinkya Jain for their fruitful conversations and timely help.

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pp. 2915–2923. PMLR, 2016.

Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.

David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *Aaai/iaai*, pp. 119–125, 2002.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Marc Bellemare, Joel Veness, and Michael Bowling. Investigating contingency awareness using atari 2600 games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pp. 864–871, 2012.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Martin Bertran, Walter Talbott, Nitish Srivastava, and Joshua Susskind. Efficient embedding of semantic similarity in control policies via entangled bisimulation. *arXiv preprint arXiv:2201.12300*, 2022.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10069–10076, 2020.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34:30113–30126, 2021.

Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-aware exploration in reinforcement learning. *arXiv preprint arXiv:1811.01483*, 2018.

Caleb Chuck, Supawit Chockchowwat, and Scott Niekum. Hypothesis-driven skill discovery for hierarchical deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5572–5579. IEEE, 2020.

Caleb Chuck, Kevin Black, Aditya Arjun, Yuke Zhu, and Scott Niekum. Granger-causal hierarchical skill discovery. *arXiv preprint arXiv:2306.09509*, 2023.

Francisco Cruz, Sven Magg, Cornelius Weber, and Stefan Wermter. Training agents with interactive reinforcement learning and contextual affordances. *IEEE Transactions on Cognitive and Developmental Systems*, 8(4):271–284, 2016.

Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 13, pp. 124–131, 1997.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, volume 20, pp. 162–169, 2004.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

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

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pp. 8407–8426. PMLR, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2016.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm van Seijen, Remi Tachet des Combes, et al. Agent-controller representations: Principled offline rl with rich exogenous information. *arXiv preprint arXiv:2211.00164*, 2022.

Matt Jones and Fabián Canas. Integrating reinforcement learning with models of representation learning. In *Proceedings of the annual meeting of the cognitive science society*, volume 32, 2010.

Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pp. 752–757, 2005.

Tobias Jung, Daniel Polani, and Peter Stone. Empowerment for continuous agent—environment systems. *Adaptive Behavior*, 19(1):16–39, 2011.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34:4764–4777, 2021.

Mete Kemertas and Allan Douglas Jepson. Approximate policy iteration with bisimulation metrics. *Transactions on Machine Learning Research*, 2022.

Khimya Khetarpal, Zafarali Ahmed, Gheorghe Comanici, David Abel, and Doina Precup. What can i do here? a theory of affordances in reinforcement learning. In *International Conference on Machine Learning*, pp. 5243–5253. PMLR, 2020.

George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Anurag Koul, Shivakanth Sujit, Shaoru Chen, Ben Evans, Lili Wu, Byron Xu, Rajan Chari, Riashat Islam, Raihan Seraj, Yonathan Efroni, Lekan Molu, Miro Dudik, John Langford, and Alex Lamb. Pclast: Discovering plannable continuous latent states, 2023.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J Foster, Lekan P Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2022.

Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2010.

Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing (preliminary report). In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 344–352, 1989.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Alexander Levine, Peter Stone, and Amy Zhang. Multistep inverse is not all you need, 2024.

Andrew Levy, Sreehari Rammohan, Alessandro Allievi, Scott Niekum, and George Konidaris. Hierarchical empowerment: Towards tractable empowerment-based skill-learning. *arXiv preprint arXiv:2307.02728*, 2023.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Weijian Liao, Zongzhang Zhang, and Yang Yu. Policy-independent behavioral metric-based representation for deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8746–8754, Jun. 2023. doi: 10.1609/aaai.v37i7.26052. URL https://ojs.aaai.org/index.php/AAAI/article/view/26052.

Qiyuan Liu, Qi Zhou, Rui Yang, and Jie Wang. Robust representation learning by clustering with bisimulation metrics for visual reinforcement learning with distractions. *arXiv preprint arXiv:2302.12003*, 2023.

Bogdan Mazoure, Ahmed M Ahmed, Patrick MacAlpine, R Devon Hjelm, and Andrey Kolobov. Cross-trajectory representation learning for zero-shot generalization in rl. *arXiv preprint arXiv:2106.02193*, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 163–172, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019a.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9339–9347, 2019b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Satinder Singh, Michael James, and Matthew Rudary. Predictive state representations: A new theory for modeling dynamical systems. *arXiv preprint arXiv:1207.4167*, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *International Conference on Machine Learning*, pp. 11214–11224. PMLR, 2020a.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020b.

Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. *European Conference on Computer Vision (ECCV)*, 2022.

Yuanyi Zhong, Alexander Schwing, and Jian Peng. Disentangling controllable object through video prediction improves visual reinforcement learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3672–3676. IEEE, 2020.

A Convergence and Causal Properties

In this section, we extend some of the convergence properties that apply to reward-based bisimulation metrics to the action-based bisimulation metrics. Then, we prove that an optimized representation is agnostic to causally irrelevant components: elements that do not affect control and cannot be affected by control.

A.1 Fixed point convergence

First, we demonstrate that our action bisimulation metric converges to a fixed point. This proof follows a similar pattern to that found in Agarwal et al. (2021).

Theorem A.1. *Let $\mathcal{M}$ be the space of bounded pseudometrics on $\mathcal{S}, \mathcal{A}$. Define operator $\mathcal{F} : \mathcal{M}$ based on the action-bisim distance metric in Theorem 6:*

$$\mathcal{F}(d)(s_i, s_j) = d_{ss}(s_i, s_j) + c \cdot E_{a\sim U(\mathcal{A})}[W_1(d)(P(\cdot|s_i, a), P(\cdot|s_j, a))].$$

Then $\mathcal{F}$ is a contraction mapping and has a unique fixed point for a bounded dist.

Proof: See Appendix B.1. ■

A.2 Agnostic to Behavior Irrelevant Components

Just because there is an optimal fixed point does not imply that this optimal fixed point is useful. Even using a trivial single-step embedding ψ which maps all states to zero will still satisfy the convergence. However, if we assume that $\psi(s)$, the single-step representation, captures only action-relevant information between S and S', the myopic state information, then we can show that the learned representation captures a subset of the control relevant state features only.

First, we assume a uniform behavior policy:

Assumption A.2. The distribution of $\pi(a|s)$ is uniform (uniform distribution denoted $U(\mathcal{A})$), and therefore not conditioned on S:

$$P(a|S) = \frac{1}{|\mathcal{A}|} \quad \forall a$$

.

This is because otherwise, the behavior policy could introduce relationships between states and actions that are not present as a result of control. Now we turn to the properties of the single-step encoder. Using the abuse of notation where $\psi(S)$ is the random variable representing state, we make the following assumption about the single-step model:

Assumption A.3. $\psi : \mathcal{S} \to \mathcal{Z}_{\text{ss}}$ captures a minimum sufficient representation between S, S' and A:

$$\begin{aligned} \psi \coloneqq \arg\min_{\psi} \quad & I\left(S; \psi(S)\right) \\ \text{s. t.} \quad & d_{KL}\left(P\left(A|[\psi(S), \psi(S')]\right) \| P\left(A|[S, S']\right)\right) = 0, \end{aligned} \tag{11}$$

where $d_{KL}(\cdot\|\cdot)$ is the KL divergence between two distributions. Then this question denotes that $\psi(s)$ captures as little information about the current state as possible (the first term), the conditional

distribution over A from $[\psi(S), \psi(S')]$ is the same as that using $[S, S']$. Notice that the terms in this assumption are approximated in single step encoder training (Equation 7). The inverse dynamics prediction approximates the KL constraint, and the encoding regularization ensures minimal remaining information.

Before using this assumption, we first define what kind of information our representation should be agnostic to. Suppose that there is a partitioning of the state features (analogous to causal feature sets in (Zhang et al., 2020a)) where one set is controllable S^c, and any feature not part of that set is S^u. The sets can be imagined as sets of causal variables, where the concatenation of these sets produces the complete state space S. These sets can be defined as follows:

Definition A.4. State S can be decomposed into controllable feature set S^c and uncontrollable feature set S^u that completely describe S (bidirectional entropy is 1). These partitions have the property that the transition dynamics of S^c are independent of the transition dynamics of S^u, and the transition dynamics of S^u are independent of S^c and A:

$$\begin{aligned} P(S^{u'}|S, A) &= P(S^{u'}|S^u) \\ P(S^{c'}|S, A) &= P(S^{c'}|S^c, A) \\ H(S|S^c, S^u) &= H(S^c, S^u|S) = 1. \end{aligned} \tag{12}$$

The encoder will compress action-irrelevant components (elements of S^u), which are components with no undirected path in the causal graph connected to actions. By compression, we mean that states that vary only according to these elements will share the same encoding.

Theorem A.5. ***Action-Bisimulation Control Relevance:*** *Suppose that $\phi : \mathcal{S} \to \mathcal{Z}$ maps observations to a latent action bisimulation representation where $\|\phi(s_i) - \phi(s_j)\|_1 = d_{a\text{-}bisim}(s_i, s_j, \psi, \phi)$ using a ψ described in Definition A.3. Z, the distribution of encodings has no information about action-irrelevant components: $I(Z; S^u) = 0$.*

Proof: See Appendix B.3. ■

B Proofs

B.1 Fixed point proof

Theorem (reproduced). Let $\mathcal{M}$ be the space of bounded pseudometrics on $\mathcal{S}, \mathcal{A}$. Define operator $\mathcal{F} : \mathcal{M}$ based on the action-bisim distance metric in Theorem 6:

$$\mathcal{F}(d)(s_i, s_j) = d_{\text{ss}}(s_i, s_j) + c \cdot E_{a \sim U(\mathcal{A})}[W_1(d)(P(\cdot|s_i, a), P(\cdot|s_j, a))].$$

then $\mathcal{F}$ is a contraction mapping and has a unique fixed point for a bounded dist.

Proof:
First, we utilize a lemma that is proved in Agarwal et al. (2021), which allows us to apply a powerful inequality to the bisimulation-esque pseudometric defined in Equation 6 in Appendix A.

Lemma B.1. *Inequality for two pseudometrics d, d' and probability distributions P_X, P_Y:*

$$W_1(d)(P_X, P_Y) \leq \|d - d'\| + W_1(d')(P_X, P_Y). \tag{13}$$

See Lemma B.1 proof in Agarwal et al. (2021).

Then, use Banach fixed point theorem:

$$\begin{aligned}
&\mathcal{F}(d)(x,y) - \mathcal{F}(d')(x,y) = \\
&\quad = c \cdot E_{a\sim U(\mathcal{A})}[W_1(d)(P(\cdot|s_i,a),P(\cdot|s_j,a))] - E_{b\sim U(\mathcal{A})}[W_1(d')(P(\cdot|s_i,b),P(\cdot|s_j,b))] \\
&\quad = c \cdot E_{a\sim U(\mathcal{A})}[W_1(d)(P(\cdot|s_i,a),P(\cdot|s_j,a))] - W_1(d')(P(\cdot|s_i,b),P(\cdot|s_j,b))] \\
&\quad \leq c \cdot E_{a\sim U(\mathcal{A})}[\|d-d'\| + W_1(d')(P(\cdot|s_i,a),P(\cdot|s_j,a))] - W_1(d')(P(\cdot|s_i,b),P(\cdot|s_j,b))]\,. \\
&\qquad\qquad \text{Applying Lemma 13} \\
&\quad = c \cdot E_{a\sim U(\mathcal{A})}[\|d-d'\|] \\
&\quad = c \cdot \|d-d'\|
\end{aligned}$$

Since $\mathcal{F}(d)(x,y) - \mathcal{F}(d')(x,y) \leq c \cdot \|d-d'\|$, $\mathcal{F}$ is a contractive mapping for $c < 1$ and has unique fixed point d^*. ■

B.2 Causal Parititon proof

Assumption B.2. ψ captures the information bottleneck representation between S^t, S^{t+k} and A_k:

$$\arg\min \psi I(S^t, S^{t+k}; \psi(S), \psi(S^{t+k})) - \beta I(\psi(S), \psi(S^{t+k}); A_k) \tag{14}$$

Then, the following theorem holds:

Theorem B.3. ***Action Bisimulation Partitions:*** *If we partition observations using the action bisimulation metric where the single-step representation optimizes Equation 14, then the action bisimulation partitions correspond to a subset of the causal feature set for current and future actions.*

Proof:
Suppose u is a feature along which action bisimulation partitions, but is not part of the causal feature set for current and future actions.

First, consider the case of current actions: then by definition, this will increase $I(S^t, S^{t+k}; \psi(S), \psi(S^{t+k}))$, because it will be a component of state encoded by the embedding. However, since it is not part of the causal feature set, it will not increase $\beta I(\psi(S), \psi(S^{t+k}); A_k)$. Thus, it will not satisfy the optimal embedding specified in Equation 14 for the single step embedding, which will increase the base-case loss in Equation 10, $\|\psi(s_i) - \psi(s_j)\|$.

Second, consider that in the case where u encodes information about future actions, suppose at time horizon k. This will increase the loss in the second term for Equation 10. This can be seen by unrolling the distance across k steps, where the l1 loss is used in the Wasserstein distance.

Thus, u cannot exist while also being an optimal solution, meaning it could not be a feature along which the action bisimulation partitions. ■

This connection allows us to make statements about what information the encoder compresses. The encoder will compress action-irrelevant components, which are components with no undirected path in the causal graph connected to actions so that these states are encoded together.

B.3 Action-Bisimulation Control Relevance Proof

Theorem (reproduced). **Action-Bisimulation Control Relevance:** Suppose that $\phi : \mathcal{S} \to \mathcal{Z}$ maps observations to a latent action bisimulation representation where $\|\phi(s_i) - \phi(s_j)\|_1 = d_{\text{a-bisim}}(s_i, s_j)$. Z, the distribution of encodings has no information about action-irrelevant components: $I(Z; S^u) = 0$.

Proof:
s_i and s_j are two states which only differ according to features in S^u. We demonstrate for any s_i, s_j, S^u, $\phi(s_i) = \phi(s_j)$.

Lemma B.4. $I(\psi(S); S^u) = 0$ *for any* $\psi(\cdot)$ *that satisfies Assumption A.3*

Proof of Lemma B.4:
We start by demonstrating that a $\psi(S)$ with zero mutual information with S^u can still satisfy $d_{KL}\left(P\left(A|[\psi(S),\psi(S')]\right)\|P\left(A|[S,S']\right)\right)=0$, the distribution matching property of Equation 14 by demonstrating that the distributions have no dependence on S^u:

$$\begin{aligned}
P(A|S,S') &= \frac{P(S'|S,A)P(A|S)}{P(S'|S)} \\
&= \frac{P(S^{u'}|S,A)P(S^{c'}|S,S^{u'},A)P(A|S)}{P(S^{u'}|S)P(S^{c'}|S,S^{u'})} \\
&= \frac{P(S^{u'}|S^u)P(S^{c'}|S^c,A))P(A|S)}{P(S^{u'}|S^u)P(S^{c'}|S^c)} \\
&\qquad \text{Applying Definition A.4} \\
&= \frac{P(S^{c'}|S^c,A))U(\mathcal{A})}{P(S^{c'}|S^c)} \\
&\qquad \text{Applying Assumption A.2}
\end{aligned} \tag{15}$$

Where $U(\mathcal{A})$ is the uniform distribution over actions. By removing the dependence of $P(A|S,S')$ on S^u, this means that $d_{KL}(P(A|S,S')\|P(A|\psi(S),\psi(S')))=0$ for all $\phi(S)$ where the distribution differs only according to S^u.

Now, consider any $\tilde{\psi}(S)$ where $I(\tilde{\psi}(S);S^u)=\alpha>0$. We have already shown that the $\tilde{\psi}(S)$ distributional dependence is unnecessary to satisfy the KL constraint. Thus, any dependence on S^u will increase the mutual information $I(S;\tilde{\psi}(s))$. This means that for any single step encoding $\tilde{\psi}(\cdot)$, there exists a lower cost $\psi(\cdot)$ which has no dependence on S^u, since any dependence on S^u is unnecessary to satisfy the KL constraint. Thus any $\psi(\cdot)$ that satisfies Assumption A.3 has the property $I(\psi(S);S^u)=0$.

Lemma B.5. *For any $s_i,s_j\in S$ which differ only according to features in S^u,*

$$\|\psi(s_i)-\psi(s_j)\|=0$$

Proof of Lemma B.5:
The consequence of Lemma B.4 is that the zero mutual information indicates:

$$\psi(s_i)=\psi(s_j) \qquad \forall s_i,s_j \quad \text{s. t.} \quad s_i \text{ and } s_j \text{ differ only according to features in } S^u$$

This follows from the definition of mutual information, where $I(X,Y)=0$ implies that X is independent of Y. If two variables are independent, then any change of one variable will not change the other variable. As a result, $\|\psi(s_i)-\psi(s_j)\|=0$. ■

Finally, we can complete the proof by unrolling the multi-step objective for any two states s_i,s_j which differ only according to S^u:

$$\begin{aligned}
d_{\text{a-bisim}}(s_i,s_j,\psi,\phi) &= (1-c)\cdot\|\psi_(s_i)-\psi_(s_j)\|_1+c\cdot\mathbb{E}_{a\sim U(\mathcal{A})}\left[W_1(p(\phi(s_i),a),p(\phi(s_j),a))\right] \\
&= c\cdot\mathbb{E}_{a\sim U(\mathcal{A})}\left[W_1(p(\phi(s_i),a),p(\phi(s_j),a))\right] \\
&= c\cdot E_{a\sim\mathcal{U}(\mathcal{A})}\left[\int_{s_i'\sim p(\phi(s_i),a),s_j'\sim p(\phi(s_j),a)} d_{\text{a-bisim}}(s_i',s_j')\delta s_i'\delta s_j'\right]
\end{aligned}$$

Notice that unrolling $d_{\text{a-bisim}}(s_i',s_j')$ gives $(1-c)\cdot\|\psi_(s_i')-\psi_(s_j')\|_1+c\cdot\mathbb{E}_{a\sim U(\mathcal{A})}\left[W_1(p(\phi(s_i'),a),p(\phi(s_j'),a),d)\right]$. Using Definition A.4 demonstrates that s_i' and s_j' must also only differ according to features in S^u. By induction, this difference holds for all timesteps, which demonstrates that $d_{\text{a-bisim}}(s_i,s_j,\psi,\phi)=0$. Since $\phi(\cdot)$ is defined as matching $d_{\text{a-bisim}}(s_i,s_j,\psi,\phi)$, this implies that $\phi(s_i)=\phi(s_j)\quad\forall s_i,s_j\in S$ that differ only according to S^u. ■

C Alternative Base Case Representations

This section introduces the single step contrastive alternative to the encoder introduced in Section 4.1, as well as the k-step generalization, where the existing methods can be seen as $k=1$.

C.1 Contrastive Representations

Contrastive representations approximate the lower bound of the mutual information between two signals, in this case the state transition (s, s') and the action a. Mutual information is the degree to which knowledge about (s, s') encodes information about a, which is defined by: $I((s, s'); a) = H((s, s')) - H((s, s')|a)$, where H is the Shannon entropy. InfoNCE (Oord et al., 2018) is a popular contrastive method for computing a lower bound of this statistic based on Noise Contrastive Estimation (Gutmann & Hyvärinen, 2010). Like inverse dyanmics, define the learned state encoder as $\psi_\theta(s) : \mathcal{S} \to \mathcal{Z}_{ss}$. Define action encoder to map to the concatenated space of state encodings $[\mathcal{Z}_{ss}, \mathcal{Z}_{ss}]$, where square brackets represent concatenation: $\psi_{\eta,\mathcal{A}}(a) : \mathcal{A} \to [\mathcal{Z}_{ss}, \mathcal{Z}_{ss}]$. Finally, a pairwise distance operator $d(z_1, z_2) : \mathcal{Z}_{ss} \times \mathcal{Z}_{ss} \to \mathbb{R}$. In our experiments $d(\cdot, \cdot)$ was the l2 distance. The InfoNCE objective is as follows:

$$L_{\text{infoNCE}}(\mathcal{D}, \theta, \eta) = E_{(s,a^+,s')\sim\mathcal{D}}\left[\frac{e^{d([\psi_\theta(s),\psi_\theta(s')],\psi_{\eta,\mathcal{A}}(a^+))}}{\sum_{\tilde{a}\in\{a^-,a^+\}} e^{d([\psi_\theta(s),\psi_\theta(s')],\psi_{\eta,\mathcal{A}}(\tilde{a}))}}\right]. \tag{16}$$

a^+ denotes the positive sample, which is the actual action taken in state s. a^- represents the negative samples, which are the alternative actions not taken in s. Optimizing the loss in Equation 16 will learn a representation encoding action-relevant components. In practice, the contrastive representations did not perform as well as the inverse dyanmics-based ones, and future work is investigating the reason for this in detail.

C.2 K-step Base Cases for Action-Bisimulation

In this work, we primarily investigate a base encoder $\psi_\theta(\cdot)$ trained using s, a, s', where s and s' are subsequent states. Prior work (Lamb et al., 2022) has investigated training encoders two states k steps apart and predicting the first action. While it may seem like longer-term controllability can be captured by simply increasing k, choosing a fixed horizon introduces a clear limitation: Determining the inverse dynamics between states when k is small is well-defined but myopic, but when k gets large, there may no longer be enough information between the state at t and $t + k$ to provide meaningful information about the action. For this to be well defined, there must be a meaningful correlation between the current action and the state k steps into the future. This correlation does not exist if the actions are random and the agent can return to states that it has been to before. As a result, in practice k-step controllability is limited to the offline RL setting, where some meaningful trajectories are provided to the agent (Islam et al., 2022). This means that in practice k-step methods are not fully unsupervised.

Depending on the nature of the offline data, the k-step extension can be combined with Action-bisimulation, where instead of choosing a large c (i.e. $c > 0.9$), the single-step encoders can be replaced with k-step encoders. This has the potential to significantly increase the degree to which the action-bisimulation encoder ϕ_η can capture long-term controllability.

Formally, instead of the tuple (s, a, s'), we use the tuple $(s^{(t)}, a^{(t)}, s^{(t+k)})$. Then, we can represent the k-step regularized inverse dynamics loss (adapting from Equation (7)) with:

$$L_{ssr}(\mathcal{D}, \theta, \eta) = -\sum_{(s^{(t)},a^{(t)},s^{(t+k)})\sim\mathcal{D}} \log f_{\eta,\text{forward}}(a^{(t)}|\psi_\theta(s^{(t)}), \psi_\theta(s^{(t+k)})) + \beta\left(\|\psi_\theta(s^{(t)})\|_1 + \|\psi_\theta(s^{(t+k)})\|_1\right).$$

Notice that if a large k is chosen, this can run into the same issues as other fixed-k methods, where the distribution of actions can affect the features captured by the single step model.

Similarly, we can replace the InfoNCE representation by replacing a^+ with $\mathbf{a}^+$, which is the actual *sequence* of actions between $s^{(t)}$ and $s^{(t+k)}$, instead of just the first action. We can also replace

a^- with $\mathbf{a}^-$, which is a sequence of actions different from the actual one. This gives the k-step representation of Equation (16):

$$L_{\text{infoNCE}}(\mathcal{D}, \theta, \eta) = E_{(ss^{(t)}, \mathbf{a}^+, s^{(t+k)}) \sim \mathcal{D}} \left[\frac{e^{d([\psi_\theta(s^{(t)}), \psi_\theta(s^{(t+k)})], \psi_{\eta,\mathcal{A}}(\mathbf{a}^+))}}{\sum_{\tilde{\mathbf{a}} \in \{\mathbf{a}^-, \mathbf{a}^+\}} e^{d([\psi_\theta(s^{(t)}), \psi_\theta(s^{(t+k)})], \psi_{\eta,\mathcal{A}}(\tilde{\mathbf{a}}))}} \right]. \tag{17}$$

C.3 Adaptive Regularization for Minimal Representation

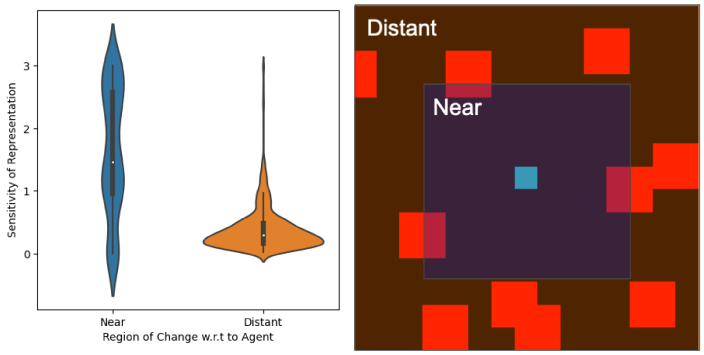
To train an encoder with the loss described in Eq. 7, it is necessary to choose a regularizing constant β beforehand. We found it was possible (and sometimes easier from a hyper-parameter search perspective) to adapt the β parameter to the current performance of encoder ψ_θ. We changed β throughout training according to the accuracy of the inverse dynamics predictions, lowering the regularization constant when the accruacy was low and raising it when the accuracy was high. The intuition is that if accuracy is low, then the representation needs to be less minimal and so we need to regularize less heavily. We calculated the regularization constant β_i where i is the training iteration with:

$$\beta_i = \beta_{\max}(1 - \exp(-4\alpha_{i-1}^2)),$$

where $\beta_{\max}$ is the maximum regularization constant and α_{i-1} is the action prediction accuracy during the previous iteration. This trick did not significantly impact our results, but lessened the hyper-parameter search.

D Additional Qualitative Results

This section describes several other qualitative results that demonstrate the properties of the encodings learned using action-bisimulation as compared to other encoding methods. We first provide qualitative results describing how the representation is sensitive not only to the agent's location but also to the local obstacles. This distinction is valuable since encoding agent position can often be sufficient to already significantly improve downstream RL performance. To generate the plot, we randomly generate obstacles either near the agent (left) or far from the agent (right), where near and distance are described below. When the changes are near the agent, there is a large variation in representation distance. On the other hand, distant perturbations make little difference to the representation of the agent.

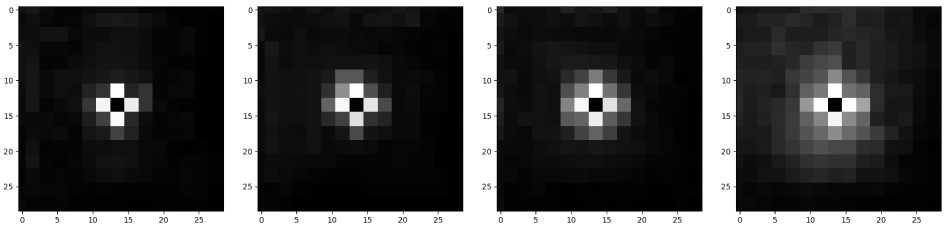


(a) (b)

Figure 6: **Left: Violin Plot** shows how the representation is sensitive to changes in obstacles near and distant to the agent. **Right: Sample observation** illustrates the *near* and *distant* regions with respect to the agent in the center.

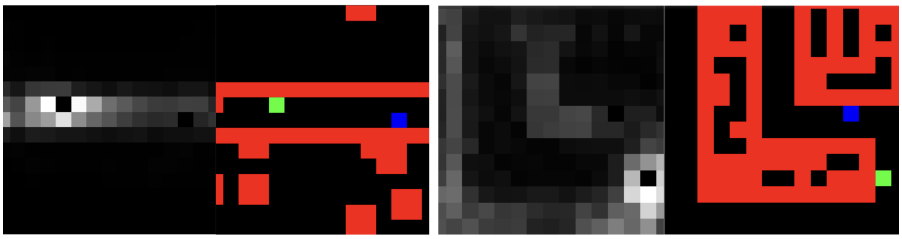
Figure 7 shows how this dropoff varies as the value of c, the discount factor in Equation 6, changes. The multi-step encoder gracefully increases in sensitivity with greater c, though a very large c can make it unstable. Fundamentally, the possible sequences of actions grows exponentially, especially

when trained with random actions, which is why selecting a value of c which ensures some dropoff ensures that the action-bisimulation representation does not become too off-policy.



(a) (b) (c) (d)

Figure 7: We demonstrate the sensitivity of the Action Bisimulation encoder to changes in obstacles around the agent as we change the value of c from left to right with 0.25, 0.75, 0.85, 0.99.



(a) (b)

Figure 8: **Left: Nav2D Corridor** shows how the action-bisimulation representation is sensitive to changes only within the corridor and agnostic to those without. Sensitivity is denoted by brighter pixels in the left subfigure. **Right: Nav2D Maze** demonstrates the action-bisimulation representation's sensitivity in a maze environment.

In Figure 8, we demonstrate how the action-bisimulation metric can learn reprsentations that ignore control-irrelevant information. In the Corridor environment, the agent is never able to leave the interior of the corridor but can always observe the obstacles on the exterior. We see that the representation is sensitive only to changes within the corridor. These results are echoed in the more complex Maze environment where the unreachable obstacles inside the maze's walls have little to no effect on the agent's representation. In fact, we can see that the representation's region of sensitivity almost exactly matches the agent's reachable locations.

E Additional RL Results

The Pointmaze environment which we evaluated with used a set of discrete actions. In general, evaluating $U(\mathcal{A})$, the uniform distribution over actions, is easier with continuous actions. Action-bisimulation can be approximated in continuous contexts simply by sampling some representative number of states. We demonstrate this in a continuous pointmaze environment, where the agent takes continuous 2D directions.

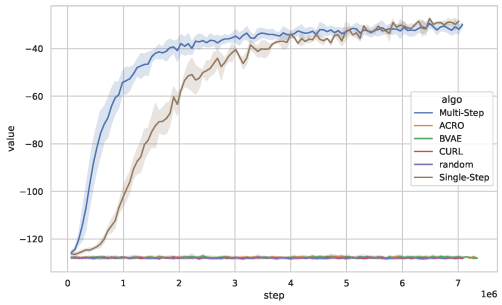


Figure 9: **Continuous Pointmaze performance**: because the action space is more challenging, many of the baselines struggle, especially ACRO, where action ambiguity is heightened.

As we can see, Figure 9 provides evidence that action-bisimulation encodings are not limited to discrete actions.

Environment	2D Navigation	Point-Mass	Habitat
Action-Bisimulation	-14.266 ± 0.509	-30.8 ± 3.7	0.7754 ± 0.005049
Single-step	-12.613 ± 1.992	-29.7 ± 3.9	0.7716 ± 0.1629
ACRO	-49.436 ± 0.342	-127.9 ± 0.61	0.7374 ± 0.0065
β-VAE	-13.880 ± 2.263	-128.4 ± 0.52	0.138 ± 0.0220
CURL	-44.791 ± 7.472	-128.4 ± 0.45	0.7657 ± 0.0193
Vanilla	-14.523 ± 1.560	-128.4 ± 0.052	0.728 ± 0.0294

Table 1: Final Performance Evaluation

F Limitations

The three primary limitations we observed in this work for implementing action-bisimulation are as follows, and we go into further detail in the subsequent subsections:

1. The minimum controllable single-step representation, especially when using learned inverse dynamics can omit controllable information if the action is overrepresented in the state
2. Uncontrollable, but reward-relevant elements must be incorporated into the representation after it is learned.
3. Both the forward model for transitions and the action-bisimulation encoding representation are bootstrapped over the expectation over all actions. This can result in unstable training.
4. When a task does not require much lookahead, action-bisimulation will only provide a marginal benefit.

F.1 Limitations of Minimum Controllable Representation

While Appendix A demonstrates that at convergence the action-bisimulation encoding will capture only action-relevant information, it does not guarantee that it will capture *all* action relevant information. If using the regularized single step loss Equation 7, the method is regularized to capture the *minimal sufficient information* to predict actions. In practice, this can be quite limiting.

For example, consider the scenario where in the top corner of the screen there is a small display of the last action that the agent takes. In this scenario, the inverse dynamics model is likely to learn to only pay attention to this part of the screen, ignoring other components such as the state of the agent. This is because paying attention to this part is a sufficient representation of actions, even though it does not capture all action-relevant information. Information-based methods such

as Equation (16) or generative controllability representations are a possible solution for this, but we have found empirically that they do not seem to learn representations useful for RL. As a result, future work is for investigating action-controllable components that are not necessary for inverse dynamics (action prediction).

F.2 Uncontrollable Reward-relevant components

Another possible limitation is relevant for all controllability pertaining: a representation that captures controllable elements may fail to capture uncontrollable reward-relevant components. For example, consider a goal-based task where the goal is part of the state. The goal itself is not controllable and thus the representation of the goal will not be encoded in an action-bisimulation representation. While the action-bisimulation method is task-agnostic, at least insofar as the initial offline dataset is task-agnostic, it is not a sufficient representation in every task.

This issue can be mitigated by a variety of approaches. The simplest is to simply allow the representation to be modified to be task-specific in the RL training, and we employ this strategy in this work. However, more complex strategies might add an additional channel for task-relevant information, or integrating classic reward-bisimulation to learn the task-relevant components on top of the pre-trained action-bisimulation ones.

F.3 Training Instability of Bootsrapping

One of the challenges when learning an action-bisimulation representation is the inherent bootstrapping where the forward model is trained with $f(\phi(s), a) \to \phi'(s)$, and the encodings themselves are being updated with Equation 10. Since the action distribution in Equation 9 is over the uniform expectation over actions, this can result in instability because of the combinatorial complexity of actions. One way we mitigated this is through the adaptive learning rate of the forward model, but future work should investigate stabilizing the convergence, especially if action-bisimulation is applied to online data.

F.4 Tasks without Lookahead

Finally, while multi-step controllability is a powerful property, not all tasks require this kind of lookahead, and it is not clear that multi-step pretraining would outperform single-step or other baselines in these cases. For example, in the popular Mujoco locomotion domains (Todorov et al., 2012), knowing about future control can often be distracting to the agent—all the relevant information is captured by determining how the current action will affect future actions. Domains where long-term control is useful, such as manipulation, can also be challenging for the current form of action-bisimulation because of the minimal sufficient information property of the single-step losses. Future work is aimed at investigating this in greater detail.

G Baseline Details

A detailed description of each of the baselines and the limitations of each. Also a mention of reward-based bisimulation.

H Environment Details

Environment	Pretrain dataset	Evaluation Steps
2D Navigation	1M samples	2M steps
Point-Mass	0.25M samples	7M steps
Habitat	100k samples	2M steps

Table 2: Amount of data used for 2 phases. The pretrain dataset uses random actions and is used to train the encoders, and the evaluation steps is the number of environment steps used to train RL.

H.1 2D Navigation with Obstacles

This environment is a 2D gridworld which consists of a 15×15 grid. The agent has 4 actions, up, down, left and right $[(0, -1), (0, 1), (-1, 0)(1, 0)]$. If the agent moves into an obstacle, or the edge of the screen, its location will not change, otherwise, the direction will be added to the current position. and takes as observation a 3-channel 15×15 image. The first channel encodes the agent position with 1 at the location of the agent, and -1 elsewhere. The second channel encodes obstacles as 1 where there is an obstacle and -1 otherwise, and the last channel encodes the goal. In this version, the goal is always located at the center of the image $(7, 7)$. This is because otherwise a task-agnostic encoding would have to re-learn the goal location. The agent receives a reward of -1 everywhere except the goal, where it receives reward of 0.

Initialization of the environment is as follows: the agent is initialized at a random location. Then obstacles are generated as 20 2×2 obstacles, initialized at random locations. The obstacles can be overlapping, but they cannot be initialized on top of the agent. Finally, the environment checks that there exists a path from the agent to the goal. if there is not, the environment is reinitialized until there is. Each episode is 50 timesteps, after which a new environment is initialized.

H.2 Pointmass

This environment is a modification of the Mujoco Pointmass environment, where a pointmass with the dynamics of a damped linear x and y joint with damping coefficent 1 and friction coefficient 0.5 with navigates through the environment taking a set of four discrete actions, up, down, left, and right. The original environment only included a small number of pre-defined mazes in a 15m $\times$ 15m world. Additionally the original environment directly gives observations of the position of the goal and agent, while this version gives pixel data from a fixed topdown camera. This environment lacked the complexity of controllability in the dynamics we are interested in investigating in this work. Instead, we modified the environment so that 20 2m $\times$ 2m obstacles are randomly arranged in the environment, and added walls to prevent the point from leaving the field of view. The goal is always located in the center of the image. In this environment, the extrinsic reward function is a sparse 0/1 reward for being within $1m$ of the goal, which is the distance traversed by the agent in 1 timestep. The agent takes episodes of 128 time steps.

H.3 Habitat

Habitat (Savva et al., 2019a) is a photorealistic 3D simulator for training embodied agents. The experiments in this paper use five scenes from the `Tiny` partition of the Gibson dataset (Xia et al., 2018), `Andover`, `Azusa`, `Anaheim`, `Ballou`, and `Spotswood`. These scenes were chosen for their high navigational complexity. The observation space is a visually rich RGB+Depth image. Unlike the original Habitat environment, we choose to use an orthographic (as opposed to pinhole) camera placed above the goal in each episode so that the goal location is always at the center of the image observation; using a consistent goal location with respect to the camera is critical as we do not include any other goal information in the observation (in contrast with the traditional PointNav task in Habitat that includes a distance+compass heading sensor to the goal). In the RGB observation, we place a yellow box on top of the agent to indicate its location because the default rendered agent is sometimes the same color as the floor below it; the depth image remains unchanged. The agent and goal are spawned in new locations every episode such that the agent is always in view of the camera; this means that each episode looks at a different part of the scene.

I Hyperparameters

I.1 Nav2D

The network dimensions and architectures used for Nav2D.

Inverse Dynamics Model

Layer Type	Layer Size
Input	1152
Linear & ReLU	256
Linear	256

Action-Bisimulation Parameters

Parameter	Value
Single Step L_1 Penalty	0.0001
Multi Step c	0.99
Learning Rate	0.0001

Reinforcement Learning Parameters

Parameter	Value
Algorithm	DQN
Batch Size L_1 Penalty	32
ϵ_{end}	0.2
ϵ_{start}	0.9
γ	0.99
Learning Rate	0.0001

I.2 Pointmass

The network dimensions and architectures used for the environment.

Encoder parameters

Layer Type	Layer Size	Kernel Size
Input	N/A	64x64x3
Conv2D & ReLU	3x3	32x32x8
Conv2D & ReLU	3x3	16x16x16
Conv2D	8x8	1x1x32

Inverse Dynamics Model

Layer Type	Layer Size
Input	64
Linear & ReLU	256
Linear	32

Actor/Critic Models

Layer Type	Layer Size
Input	32
Linear & ReLU	256
Linear	4/1

Action-Bisimulation Parameters

Parameter	Value
Single Step L_1 Penalty	1.0
Multi Step c	0.75
K Steps	5
Learning Rate	0.0001

Reinforcement Learning Parameters

Parameter	Value
Algorithm	PPO
Batch Size L_1 Penalty	256
Steps Per Rollout	65536
Steps Per Eval	16384
Learning Rate	0.000025

The L_1 penalty is a particularly sensitive parameter, with this incorrectly set the single step model fails to identify relevant features. To train this effectively an adaptive term was used to scale the L_1 regularization term to the listed value as the encoder approached convergence.

I.3 Habitat

Encoder hyperparameters and PPO The network dimensions and architectures used for the Habitat experiments are exact copies of the ResNet18 (He et al., 2015) networks used in the original Habitat PointGoal navigation task (Savva et al., 2019a). For pretraining the encoders, we only trained the visual features encoder of the ResNet18 policy used in Habitat. Further, we used the vanilla implementation of PPO written in Habitat with all default parameters.

Inverse Dynamics Model

Layer Type	Layer Size
Input	2048
Linear & ReLU	256
Linear	256

Action-Bisimulation Parameters

Parameter	Value
Single Step L_1 Penalty	0.0
Multi Step c	0.95
Learning Rate	0.0001

PPO Parameters

Parameter	**Value**
clip_param	0.2
ppo_epoch	4
num_mini_batch	2
value_loss_coef	0.5
entropy_coef	0.01
lr	0.00025
eps	.00001
max_grad_norm	0.5
num_steps	128
hidden_size	512
gamma	0.99
tau	0.95