Intrinsic Control of Variational Beliefs in Dynamic Partially-Observed Visual Environments

Nicholas Rhinehart [1] **Jenny Wang** [1] **Glen Berseth** [1] **John D. Co-Reyes** [1] **Danijar Hafner** [2] **Chelsea Finn** [3] **Sergey Levine** [1]

Abstract

Humans and animals explore their environment and acquire useful skills even in the absence of clear goals, exhibiting intrinsic motivation. The study of intrinsic motivation in artificial agents is concerned with the following question: what is a good general-purpose objective for an agent? We study this question in dynamic partially-observed environments, and argue that a compact and general learning objective is to minimize the entropy of the agent's state visitation estimated using a latent state-space model. This objective induces an agent to both gather information about its environment, corresponding to reducing uncertainty, and to gain control over its environment, corresponding to reducing the unpredictability of future world states. We instantiate this approach as a deep reinforcement learning agent equipped with a deep variational Bayes filter. We find that our agent learns to discover, represent, and exercise control of dynamic objects in a variety of partially-observed environments sensed with visual observations without extrinsic reward.

1. Introduction

Reinforcement learning offers a framework for learning control policies that maximize a given measure of reward – ideally, rewards that incentivize simple high-level goals, such as survival, accumulating a particular resource, or accomplishing some long-term objective. However, extrinsic rewards may be insufficiently informative to encourage an agent to explore and understand its environment, particularly when the environment is *partially-observed*: when the agent has a limited view of its environment. A generalist agent should instead acquire an understanding of its environment before a specific objective or reward is provided. This goal motivates the study of *self-supervised / unsupervised* reinforcement learning: algorithms that provide the agent with an intrinsically-grounded drive to acquire understanding and control of its environment in the absence of an extrinsic reward signal. Agents trained with intrinsic reward signals might accomplish tasks specified via simple and sparse rewards more quickly, or may acquire broadly useful skills that could be adapted to specific task objectives. Our aim is to design an embodied agent and a general-purpose intrinsic reward signal that leads to the agent controlling partially-observed environments when equipped only with a high-dimensional sensor (camera) and no prior knowledge.

A large body of prior methods for self-supervised reinforcement learning focus on attaining *coverage*, typically through novelty-seeking or skill-discovery objectives; see Hafner et al. (2020b) for a survey. As argued in prior work (Friston et al., 2010; Friston, 2013; Fountas et al., 2020; Berseth et al., 2021), a compelling alternative to coverage suited to complex and dynamic environments is to minimize surprise, which incentivizes an agent to control aspects of its environment to achieve homeostasis within it – i.e. constructing and maintaining a niche where it can reliably remain despite external perturbations. We generally expect agents that succeed at minimizing surprise in complex environments to develop similarly complex behaviors; such acquired complex behaviors may be repurposed for other tasks (Berseth et al., 2021). However, these frameworks do not explicitly address the difficulty of controlling partially-observed environments: if an otherwise complex and chaotic environment contains a "dark room" (small reliable niche), an agent could minimize surprise simply by hiding in this room and refusing to make meaningful observations, thereby failing to explore and control the wider surrounding environment.

Consider Fig. 1, which depicts a partially-observed outdoor environment with various flora (trees, vegetables, and grass), fauna (a goat), weather, and an agent. We will discuss three different intrinsic incentives an agent might adopt in this environment. If the agent's incentive is to (*i*) minimize the entropy of its next observation, it will seek the regions with minimal unpredictable variations in flora, fauna, and weather. This is unsatisfying because it merely requires

[1]UC Berkeley [2]University of Toronto [3]Stanford University. Correspondence to: Nicholas Rhinehart <nrhinehart@berkeley.edu>.

Unsupervised Reinforcement Learning Workshop at ICML 2021.

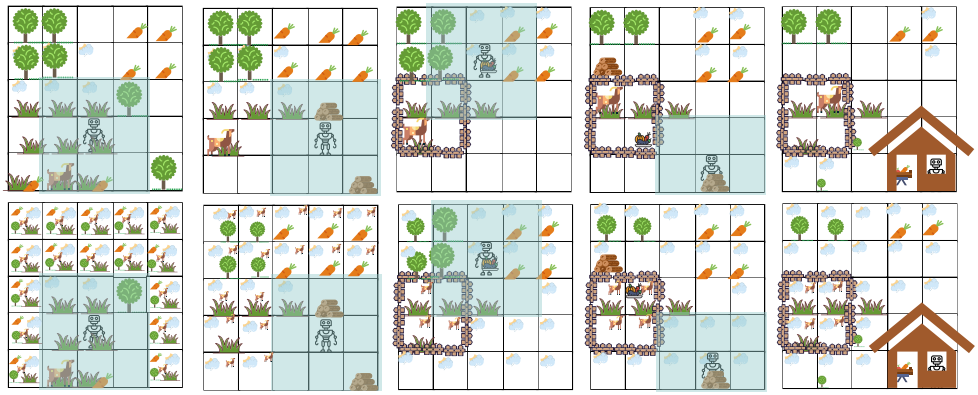


Figure 1: *Top row:* The environment consists of a large number of objects, some of which (e.g., the goat) move and act in unpredictable ways, and are not observed unless the agent is nearby. *Bottom row:* If the agent maintains a latent state space model of the world, it has uncertain beliefs about unobserved objects, particularly those that are dynamic (like the goat). If the agent reduces the long-horizon average entropy of its beliefs, it will first seek out information (e.g., finding the goat), and then modify the environment to limit the range of states the goat can occupy *even when it is no longer observed*, for example by building a fence around it.

avoidance, rather than interaction. Let us assume the agent will maintain a model of its *belief* about a learned *latent state* – the agent cannot observe the *true* state, instead it learns a state representation. Further, let us assume the agent maintains a separate model of the visitation of its latent state – we will refer to this distribution as its *latent visitation*. If the agent's incentive is to (*ii*) minimize the entropy of belief (either at every step or at some final step), the agent will gather information and take actions to make the environment predictable: find and observe the changes in flora, fauna, and weather that are predictable and avoid those that aren't. However, once it has taken actions to make the world predictable, this agent is agnostic to future change – it will not resist predictable changes in the environment. Finally, if the agent's incentive is to (*iii*) minimize the entropy of its latent visitation, this will result in categorically different behavior: the agent will seek both to make the world predictable by gathering information about it and *prevent it from changing*. While both the belief and latent visitation entropy minimization objectives are worthwhile intrinsic motivation objectives to study, we speculate that an agent that is adept at preventing its environment from changing will generally learn more complex behaviors.

We present a concise and effective objective for self-supervised reinforcement learning in dynamic partially-observed environments: minimize the entropy of the agent's latent visitation under a latent state-space model learned from exploration. Our method, which we call Believer, results in an agent that learns to seek out and control factors of variation outside of its immediate observations. We instantiate this framework by simultaneously learning a state-space model as a Deep Variational Bayes Filter along with a policy that employs the model's beliefs. Our experiments show that our method learns to represent and control dynamic entities in partially-observed visual environments with *no extrinsic reward signal*, including in several 3D environments.

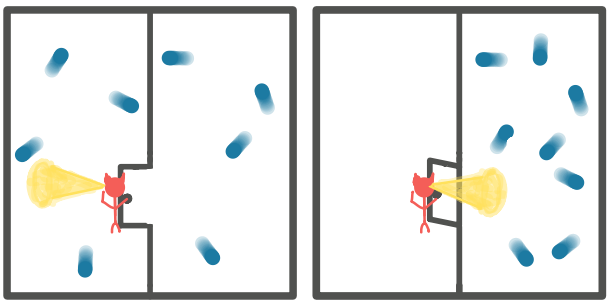


Figure 2: A "demon" gathering information to sort particles, reducing the entropy of the particle configuration.

2. Maxwell's Demon and Belief Entropy

The main concept behind our approach to self-supervised reinforcement learning is that incentivizing an agent to minimize the entropy of its *beliefs* about the world is sufficient to lead it to *both* gather information about the world *and* learn to control aspects of its world. Our approach is partly inspired by a well-known connection between information theory and thermodynamics, which can be illustrated informally by a version of the Maxwell's Demon thought experiment (Maxwell & Pesic, 2001; Leff & Rex, 2014). Imagine a container separated into compartments, as shown in Fig. 2. Both compartments contain gas molecules that bounce off of the walls and each other in a somewhat unpredictable fashion, though short-term motion of these molecules (between collisions) is predictable. The compartments are separated by a massless door, and the agent (the eponymous "demon") can open or close the door at will to sort the particles.[1] By sorting the particles onto one side, the demon appears to reduce the disorder of the system, as measured by the thermodynamic entropy, S, which increases the energy, F

[1] In Maxwell's original example, the demon sorts the particles into the two chambers based on velocity. Our example is closely related to Szilard's engine (Szilard, 1929; Magnasco, 1996).

available to do work, as per Helmholtz's free energy relationship, $F = U - TS$. The ability to do work affords the agent *control over the environment*. This apparent violation of the second law of thermodynamics is resolved by accounting for the agent's information processing needed to make decisions (Bennett, 1982). By concentrating the particles into a smaller region, the number of states each particle visits is reduced. Therefore, this illustrates an example environment in which reducing the entropy of the visitation distribution results in an agent gaining the ability to do work. In the same way that Maxwell's demon accumulates free energy via information-gathering and manipulating of its environment, we would expect self-supervised agents guided by belief entropy minimization to accumulate the equivalent of potential energy in their corresponding sequential decision processes, which would lead them to gain control.

3. Control and Information Gathering via Belief Entropy Minimization

Preliminaries. Our goal in this work will be to design self-supervised reinforcement learning methods in partially observed settings, which can acquire complex behaviors that both gather information and gain control over their environment. To this end, we will formulate the learning problem in the context of a discrete-time partially-observed controlled Markov process, also known as a controlled hidden Markov process (CHMP), which corresponds to a POMDP without a reward function. The CHMP is defined by a state space $\mathcal{S}$ with states $\mathbf{s} \in \mathcal{S}$, action space $\mathcal{A}$ with actions $\mathbf{a} \in \mathcal{A}$, transition dynamics $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, observation space Ω with observations $\mathbf{o} \in \Omega$, and emission distribution $O(\mathbf{o}_t|\mathbf{s}_t)$. The agent is a policy $\pi(\mathbf{a}_t|\mathbf{o}_{\leq t})$; it does *not* observe $\mathbf{s}$.

We denote the undiscounted finite-horizon state visitation as $d^\pi(\mathbf{s}) \doteq {}^1\!/_T \sum_{t=0}^{T-1} \Pr_\pi(\mathbf{s}_t = \mathbf{s})$, where $\Pr_\pi(\mathbf{s}_t = \mathbf{s})$ is the probability that $\mathbf{s}_t = \mathbf{s}$ after executing π for t steps. Using $d^\pi(\mathbf{s})$, we can quantify the average *disorder* of the environment with the Shannon entropy, $H(d^\pi(\mathbf{s}))$. Prior work proposes observational surprise minimization ($\min_\pi - \log \hat{p}(\mathbf{o})$) as an intrinsic control objective (Friston, 2009; Ueltzhöffer, 2018; Berseth et al., 2021); in Berseth et al. (2021) (SMiRL), the agent models the state visitation distribution, $d^\pi(\mathbf{s})$ with $\hat{p}(\mathbf{s})$, which it computes by assuming access to $\mathbf{s}$. In environments in which there are natural sources of variation outside of the agent, this incentivizes the SMiRL agent, fully aware of these variations observed through $\mathbf{s}$, to take action to control them. In a partially-observed setting, SMIRL's model becomes $\hat{p}(\mathbf{o})$, which generally will enable the agent to ignore variations that it can prevent from appearing in its observations. We observe this phenomenon in our experiments.

The main question that we tackle in the design of our algorithm is: how can we formulate a general and concise objective function that can enable an RL agent to gain control over its partially-observed environment, in the absence of any user-provided task reward? Consider the following partially-observed "TwoRoom" environment, depicted in Fig. 3. The environment has two rooms: an empty ("dark") room on the left, and a "busy" room on the right, the latter containing two moving particles that move around until the agent "tags" them, which stops their motion. Intuitively, an agent that aims to gather information and gain control of its environment should search for the moving particles to find out where they are. However, it is difficult to observe both particles at the same time. A more effective strategy is to "tag" the particles – then, their position remains fixed, and the agent will know where they are at all times, even when they are not observed. This task can be seen as a simple analogy for more complex settings that can occur in natural environments, where we might want agents to arrange an environment in an orderly fashion. We can view this task as a rough analogy for a version of the previously-discussed Maxwell's Demon thought experiment.

Representing variability with latent state-space models. In order for the agent to represent the dynamic components of the environment observed from images, our method involves learning a latent state-space model (LSSM) (Watter et al., 2015; Krishnan et al., 2015; Karl et al., 2016; Maddison et al., 2017; Hafner et al., 2018; Mirchev et al., 2018; Wayne et al., 2018; Vezzani et al., 2019; Lee et al., 2019; Das et al., 2019; Hafner et al., 2019; Mirchev et al., 2020; Rafailov et al., 2021). We intermittently refer to these dynamic components as "factors of variation" to distinguish the model's representation of variability in the environment (latent state) from the true variability (true state). At timestep t, the LSSM represents the agent's current *belief* as $q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$, where $\mathbf{z}_t$ is the model's latent state. We defer the description of the LSSM learning and architecture to Section 4, and now motivate how we will use the LSSM for constructing an intrinsic control objective.

Belief entropy and latent visitation entropy. Consider a policy that takes actions to minimize the entropy of the belief $H(q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}))$. This corresponds to the agent performing *active state estimation* (Feder et al., 1999; Williams, 2007; Kreucher et al., 2005), and is equivalent to taking actions to maximize expected latent-state information gain $I(\mathbf{o}_t, \mathbf{z}_t|\mathbf{o}_{<t}, \mathbf{a}_t)$ (Aoki et al., 2011). However, active state estimation is satisfied by a policy that simply collects informative observations, as it does not further incentivize actions to "stabilize" the environment by preventing the latent state from changing. Analogous to the standard definition of undiscounted state visitation, consider the undiscounted latent visitation: $d^\pi(\mathbf{z}) \doteq {}^1\!/_T \sum_{t=0}^{T-1} \Pr_\pi(\mathbf{z}_t = \mathbf{z})$, where $\Pr_\pi(\mathbf{z}_t = \mathbf{z}) = \mathbb{E}_\pi q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ (the expected belief after executing π for t timesteps). Our goal is to minimize $H(d^\pi(\mathbf{z}))$, as this corresponds to *stabilizing the agent's be-*

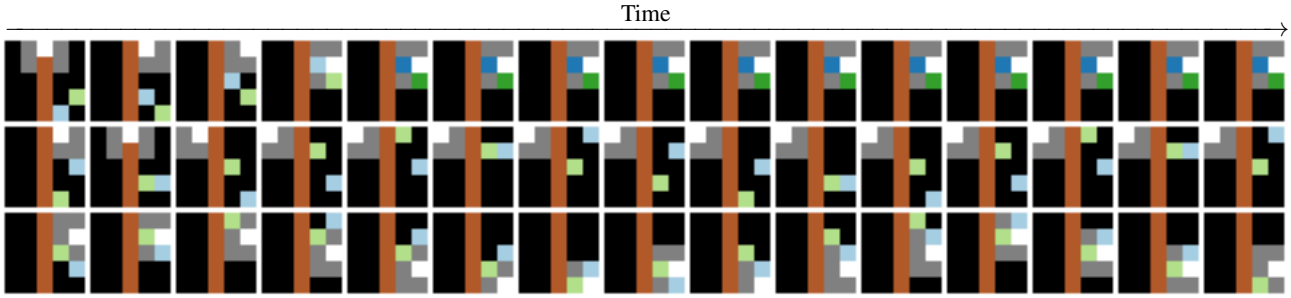


Figure 3: Comparison of several approaches on the TwoRoom environment. The agent, in white, can view a limited area around it, in grey, and can stop particles within its view and darken their color. The vertical wall, in brown, separates particles (blue and green) in the "busy room" (on right) from the "dark room" (on left). *Top*: Our approach seeks out the particles and stops them. *Middle:* The observational surprise minimization method in Berseth et al. (2021) leads the agent to frequently hide in the dark room, leaving the particles unstopped. *Bottom:* Latent-state infogain leads the agent to find and observe the particles, but not stop them.

liefs, which incentivizes both reducing uncertainty in each $q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$, as well as constructing a niche such that each $q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ concentrates probability on the same latent states.

Discovering factors of variation. In order for belief entropy minimization to incentivize the agent to control entities in the environment, the LSSM's belief must represent the underlying state variables in some way and model their uncertain evolution until either observed or controlled. For example, the demon in the thought experiment in Section 2 would have no incentive to gather the particles if it did not know that they existed. While sufficient random exploration may result in a good-enough LSSM, making this approach generally practical requires a suitable exploration strategy to collect the experience necessary to train an LSSM that represents all of the underlying factors of variation. To this end, we learn a separate exploratory policy to maximize *expected model information gain*, similar to Schmidhuber (2010); Houthooft et al. (2016); Gheshlaghi Azar et al. (2019); Sekar et al. (2020). Expected information gain about model parameters $\boldsymbol{\theta}$ is relative to a set of prior experience $\mathcal{D}$ and a partial trajectory $\mathbf{h}_{t-1}$, given as $I(\mathbf{o}_t; \boldsymbol{\theta}|\mathbf{h}_{t-1}, \mathcal{D}) = \mathbb{E}_{\mathbf{o}_t}\mathrm{KL}(p(\boldsymbol{\theta}|\mathbf{o}_t, \mathbf{h}_{t-1}, \mathcal{D}) \,||\, p(\boldsymbol{\theta}|\mathbf{h}_{t-1}, \mathcal{D}))$. Note that model information gain is *distinct* from the information an agent may gather to reduce its belief entropy $H(q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}))$ within the current episode. Computing the full model parameter prior $p(\boldsymbol{\theta}|\mathbf{h}_{t-1}, \mathcal{D})$ and posterior $p(\boldsymbol{\theta}|\mathbf{o}_t, \mathbf{h}_{t-1}, \mathcal{D})$ is generally computationally expensive, and also requires evaluating an expectation over observations – instead, we approximate this expected information gain following a method similar to Sekar et al. (2020): we use an ensemble of latent dynamics models, $\mathcal{E} = \{p_{\theta_i}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\}_{i=1}^K$ to compute the variance of latent states estimated by $q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. We build the ensemble throughout training using the method of Izmailov et al. (2018). Thus, the exploration reward is given as: $r_e = \mathrm{Var}_{\{\theta_i\}}[\log p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})|\mathbf{z}_t \sim q_\phi]$.

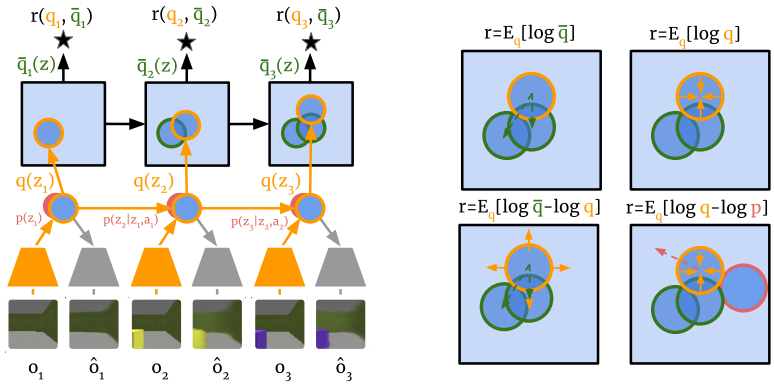
4. The Believer Algorithm

Now we describe how we implement and combine the various components of our method into a learning algorithm for minimizing belief visitation entropy in CHMPs (reward-less POMDPs). The main components are the latent-state space model, the latent visitation model, and the exploration and control policies.

Latent state-space model. In our CHMP setting, the agent only has access to partial observations $\mathbf{o}$ of the true state $\mathbf{s}$. In order to estimate a representation of states and beliefs, we employ a sequence-based Variational Autoencoder to learn latent variable belief, dynamics, and emission models. We formulate the variational posterior to be $q_\phi(\mathbf{z}_{1:T}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T}) = \prod_{t=1}^T q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ and the generative model as $p(\mathbf{z}_{1:T}, \mathbf{o}_{1:T}|\mathbf{a}_{1:T}) = \prod_{t=1}^T p_\theta(\mathbf{o}_t|\mathbf{z}_t)p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$. Denoting $\mathbf{h}_t \doteq (\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$, the log-evidence of the model and its lower-bound are:

$$\log\ p(\mathbf{o}_{1:T}|\mathbf{a}_{1:T-1}) = \log \mathbb{E}_{\mathbf{z}_{1:T}\sim p(\mathbf{z}_{1:T}|\mathbf{a}_{1:T})}\left[\prod_{t=1}^T \log p(\mathbf{o}_t|\mathbf{z}_t)\right]$$
$$\geq \mathcal{L}(\phi, \theta) = \sum_{t=1}^T \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}_t|\mathbf{h}_t)}\left[p_\theta(\mathbf{o}_t|\mathbf{z}_t)\right] - \mathop{\mathbb{E}}_{q_\phi(\mathbf{z}_{t-1}|\mathbf{h}_{t-1})}\left[\mathrm{KL}(q_\phi(\mathbf{z}_t|\mathbf{h}_t) \,||\, p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right] \quad (1)$$

Given a dataset $\mathcal{D} = \{(\mathbf{o}_{1:T}, \mathbf{a}_{1:T})_i\}_{i=1}^N$, Eq. 1 is used to train the model via $\max_{\phi,\theta} \mathbb{E}_{U(\mathcal{D})}\mathcal{L}(\phi, \theta)$. The focus of our work is not to further develop the performance of LSSMs; our method could be further improved with advances in the particular LSSM employed. In practice, we implemented an LSSM in PyTorch (Paszke et al., 2019) similar to the categorical LSSM architecture described in (Hafner et al., 2020a). In this case, both the belief prior and belief posterior are distributions formed from products of K_1 categorical distributions over K_2 categories: $g(\mathbf{z}; \mathbf{v}) = \prod_{\kappa_1=1}^{K_1}\prod_{\kappa_2=1}^{K_2} \mathbf{v}_{\kappa_1,\kappa_2}^{\mathbf{z}_{\kappa_1,\kappa_2}}$, where the vector of $K_1 \cdot K_2$ parameters, $\mathbf{v}$, is predicted by neural networks: $\mathbf{v}_{\text{posterior}} = f_\phi(\mathbf{o}_{\leq t}, \mathbf{a}_t)$ for the posterior, and $\mathbf{v}_{\text{prior}} = f_\theta(\mathbf{o}_{<t}, \mathbf{a}_t)$ for the prior. We found it effective



(a) Latent state-space and visitation models. (b) Intrinsic rewards.

Figure 4: Figure of latent-state space model and rewards. *Left:* The model observes images, $\mathbf{o}_t$ to inform beliefs about latent states, $q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$, and observes actions to make one-step predictions $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$. Each belief is used to update the latent visitation, $\bar{q}_{t'}(\mathbf{z})$. *Right:* The beliefs and latent visitations can be combined into various reward functions. The solid arrows denote directions of belief expansion and contraction incentivized by rewards; the dotted arrows denote directions of belief translation incentivized by rewards.

to construct f_ϕ and f_θ following the distribution-sharing decomposition described in (Karl et al., 2016; 2017; Das et al., 2019), in which the posterior is produced by transforming the parameters of the prior with a measurement update. We implemented the prior as an RNN, and the posterior as an MLP, and defer further details to the supplementary website.

Latent visitation model. Our agent cannot, in general, evaluate $d^\pi(\mathbf{z})$ or $H(d^\pi(\mathbf{z}))$; at best, it can approximate them. We do so by maintaining a within-episode estimate of $d^\pi(\mathbf{z})$ by constructing a mixture across the belief history samples $\bar{q}_{t'}(\mathbf{z}) = {}^1\!/_{t'} \sum_{t=0}^{t'} q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. This corresponds to a mixture (across time) of single-sample estimates of each $\mathbb{E}_\pi q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. Given $\bar{q}_{t'}(\mathbf{z})$, we have an estimate of the visitation of the policy, and we can use this as part or all of the reward signal of the agent. To implement $\bar{q}_{t'}(\mathbf{z})$, we experimented both approximating it by averaging each $\mathbf{v}$ and with recording every belief, and found that simply recording each belief sufficed.

4.1. Belief-based objectives

We now describe several rewards that can be constructed with the LSSM, and how they connect to our main objective. In what follows, we denote $q_\phi(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) = q_t(\mathbf{z}_t)$ for brevity. While our primary objective is "niche creation" by minimizing the expected surprise of the latent visitation model, we also explore other intrinsic objectives that can be computed in terms of the LSSM and the latent visitation model. These rewards are visualized in Fig. 4.

Certainty. The entropy of the current belief measures the agent's certainty of the latent state, and by extension, about the aspects of the environment captured by the latent. Minimizing the belief entropy increases the agent's certainty. It is agnostic to predictable changes, and penalizes unpredictable changes. We define the *certainty reward* as the negative belief entropy:

$$r_t^c \doteq -\mathrm{H}(q_t(\mathbf{z}_t)) = \mathbb{E}_{q_t(\mathbf{z}_t)}[\log q_t(\mathbf{z}_t)] \tag{2}$$

Niche Creation: Our primary objective is the expected surprise of the latent visitation distribution, which measures how many states the agent believes it could have been in during the episode so far. Minimizing this reduces the number of visited environment states and increases the agent's certainty. We define the *niche creation reward* as the negative cross entropy of the visitation under the belief:

$$r_t^{nc} \doteq -\mathrm{H}(q_t(\mathbf{z}_t), \bar{q}_{t'}(\mathbf{z})) = \mathbb{E}_{q_t(\mathbf{z}_t)}[\log \bar{q}_{t'}(\mathbf{z})]. \tag{3}$$

Niche Expansion. Instead of minimizing the latent visitation entropy altogether, we can add an entropy bonus for the current belief to encourage exploration and thus potentially find a broader niche. The results in bringing the current belief towards the current latent visitation distribution. We define the *niche expansion reward* as the negative KL divergence:

$$r_t^{ne} \doteq -\mathrm{KL}(q_t(\mathbf{z}_t)||\bar{q}_{t'}(\mathbf{z})) = \mathbb{E}_{q_t(\mathbf{z}_t)}[\log \bar{q}_{t'}(\mathbf{z}) - \log q_t(\mathbf{z}_t)]. \tag{4}$$

State Infogain. The latent state information gain (not to be confused with the model information gain described in Section 3) measures how much more certain the belief is compared to its temporal prior. Gaining information does not always coincide with being certain, because an infogain agent may cause chaotic events in the environment with outcomes that it only understands partially. As a result, it has gained more information than standing still but has also become less certain. We define the *infogain reward* as the KL divergence:

$$\begin{aligned} r_t^i &\doteq \mathrm{KL}(q_t(\mathbf{z}_t)||p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})) \\ &= \mathbb{E}_{q_t(\mathbf{z}_t)q_{t-1}(\mathbf{z}_{t-1})}[\log q_t(\mathbf{z}_t) - \log p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})]. \end{aligned} \tag{5}$$

4.2. Algorithm summary

Conceptual pseudocode for our method is presented in Alg. 1. The algorithm begins by initializing the LSSM, q_ϕ and p_θ, as well as two separate policies: one trained to collect difficult-to-predict data with the exploration objective with rewards defined by r_e, π_e, and one trained to maximize one of the intrinsic control objectives as defined by Eqs. (2) to (5).

We represent each policy $\pi(\mathbf{a}_t|\mathbf{v}_{\text{posterior},t})$ as a two-layer fully-connected MLP with 128 units. Recall that $\mathbf{v}_{\text{posterior}}$ is the vector of posterior parameters, which enables the policy to use the memory represented by the LSSM. We do not back-propagate the policies' losses to the LSSM for simplicity of implementation, although prior work does so in the case of a single policy (Lee et al., 2019). Our method is agnostic to the subroutine used to improve the policies. In our implementation, we employ PPO (Schulman et al., 2017).

Algorithm 1 Believer

procedure BELIEVER(Env; K, M, N, L)
  Initialize $\pi_c, \pi_e, q_\phi, p_{\{\theta_i\}_{i=1}^K}, \mathcal{D} \leftarrow \emptyset$.
  for episode $= 0, \ldots, M$ **do**
    $\mathcal{D}_e \leftarrow \text{Collect}(N, \text{Env}, \pi_e)$
    $\mathcal{D}_c \leftarrow \text{Collect}(N, \text{Env}, \pi_c)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_e \cup \mathcal{D}_c$
    `// LSSM fitting step.`
    Update q_ϕ, p_θ w. Eq. 1, $\mathcal{D}$ for L rounds.
    Update π_e w. PPO on $\mathcal{D}_e$ with rewards r_e
    `// Intrinsic control opt. step.`
    Update π_c w. PPO on $\mathcal{D}_c$ with rewards defined by one of Eqs. (2) to (5)
=0

5. Experiments

Our experiments are designed to answer the following questions: **Q1: Intrinsic control capability**: Does our latent visitation-based self-supervised reward signal cause the agent to stabilize partially-observed visual environments with dynamic entities more effectively than prior self-supervised stabilization objectives? **Q2: Properties of Believer objectives**: What types of emergent behaviors does each belief-based objective described in Section 4.1 evoke?

In order to answer these questions, we identified environments with the following properties **(i):** partial observability, **(ii)**: dynamic entities that the agent can affect, and **(iii)**: high-dimensional observations. Because many standard RL benchmarks do not contain the significant partial-observability that is prevalent in the real world, it is challenging to answer these questions with them. Instead, we create several environments, and employ several existing environments we identified to have these properties. In what follows, we give an overview of the experimental settings and conclusions. We defer comprehensive details to the supplementary website.

5.1. Environments

(a) TwoRoom Large Environment. (b) Vizdoom Defend The Center. (c) One Room Capture 3D.

TwoRoom. As previously described, this environment has two rooms: an empty ("dark") room on the left, and a "busy" room on the right, the latter containing moving particles that move around unless the agent "tags" them, which permanently stops their motion, as shown in Fig. 5a. The agent can observe a small area around it, which it receives as an image. In this environment, control corresponds to finding and stopping the particles. The action space is $\mathcal{A} = \{\text{left, right, up, down, tag, no-op}\}$, and the observation space is normalized RGB-images: $\Omega = [0, 1]^{3\times30\times30}$. An agent that has significant control over this environment should tag particles to reduce the uncertainty over future states. To evaluate policies, we use the average fraction of total particles locked, the average fraction of particles visible, and the discrete true-state visitation entropy of the positions of the particles, $H(d^\pi(s_d))$. We employed two versions of this environment, with details provided in the supplementary website. In the large environment, the agent observes a 5x5 area around it as an image, and the busy room contains 5 particles.

VizDoom DefendTheCenter. The VizDoom DefendTheCenter environment shown in Fig. 5b is a circular arena in which a stationary agent, equipped with a partial field-of-view of the arena and a weapon, can rotate and shoot encroaching monsters (Kempka et al., 2016). The action space is $\mathcal{A} = \{\text{turn left, turn right, shoot}\}$, and the observation space is normalized RGB-images: $\Omega = [0, 1]^{3\times64\times64}$. In this environment, control corresponds to reducing the number of monsters by finding and shooting them. We use the average original environment return, (for which *no policy* has access to during training), the average number of killed monsters at the end of an episode, and the average number of visible monsters to measure the agent's control.

OneRoomCapture3D The MiniWorld framework is a customizable 3D environment simulator in which an agent perceives the world through a perspective camera (Chevalier-

Boisvert, 2018). We used this framework to build the environment in Fig. 5c which an agent and a bouncing box both inhabit a large room; the agent can lock the box to stop it from moving if it is nearby, as well as constrain the motion of the box by standing nearby it. In this environment, control corresponds to finding the box and either trapping it near a wall, or tagging it. The action space is $\mathcal{A} = \{\text{left } 20°, \text{right } 20°, \text{forward, backward, tag}\}$, and observation space is normalized RGB-images: $\Omega = [0, 1]^{3\times64\times64}$. We use the average fraction of time the box is captured, the average time the box is visible, and continuous (Gaussian-estimated) true-state visitation entropy of the box's position $H(d^\pi(s_d))$ to measure the agent's ability to reduce entropy of the environment.

5.2. Comparisons

In order to answer **Q1**, we compare to SMiRL (Berseth et al., 2021) and a recent empowerment method (Zhao et al., 2021) that estimates the current empowerment of the agent from an image. We use the empowerment estimate as a reward signal for training a policy. In order to answer **Q2**, we ensured each environment was instrumented with the aforementioned metrics of visibility and control, and deployed Algorithm 1 separately with each of Eqs. (2) to (5), as well a with simple sum of Eq. (3) and Eq. (5). Finally, we compare to a "random policy" that chooses actions by sampling from a uniform distribution over the action space, as well as an "oracle policy" that has access to privileged information about the environment state in each environment. We perform evaluation at $5e6$ environment steps with 50 policy rollouts per random seed, with 3 random seeds for each method (150 rollouts total).

Our primary results are presented in Tables 1 and 2. We observe the Niche Creation+Infogain and Niche Expansion rewards to yield policies that exhibit a high degree of control over each environment – finding and stopping the moving objects, and finding and shooting the monsters, in the *complete absence of any extrinsic reward signal*. The Infogain-based agent generally seeks out and observes, but does not interfere with, the dynamic objects; the high Visibility metric and low control metrics (Lock, Capture, Kill) in each environment illustrates this. Qualitative results of this phenomenon are illustrated in Fig. 6, in which the infogain signal is high when stochastically-moving monsters are visible, and low when they are not. We observe that in these partially observed environments the method of Berseth et al. (2021) tends to learn policies that hide from the dynamic objects in the environment, as indicated by the low control and visibility values of the final policy. Furthermore, we observe the method of Zhao et al. (2021) not to exhibit controlling behavior in these partially-observed environments, instead it views the dynamic objects in the TwoRoom and OneRoomCapture3D Environments somewhat more frequently than the random policy. We present videos of all policies in the supplementary website.

Method	TwoRoom Environment		
	Obj. Lock↑	Obj. Visible↑	$H(d^\pi(s_d))\downarrow$
Niche Creation+Infogain	$0.92_{\pm 0.01}$	$\mathbf{0.94}_{\pm 0.01}$	$0.33_{\pm 0.03}$
Niche Expansion, Eq. (4)	$\mathbf{0.95}_{\pm 0.00}$	$0.66_{\pm 0.03}$	$\mathbf{0.22}_{\pm 0.02}$
Niche Creation, Eq. (3)	$0.50_{\pm 0.05}$	$0.46_{\pm 0.05}$	$1.06_{\pm 0.09}$
Certainty, Eq. (2)	$0.06_{\pm 0.02}$	$0.01_{\pm 0.00}$	$1.86_{\pm 0.04}$
Infogain, Eq. (5)	$0.00_{\pm 0.00}$	$0.55_{\pm 0.00}$	$1.95_{\pm 0.02}$
SMiRL	$0.25_{\pm 0.04}$	$0.27_{\pm 0.03}$	$1.52_{\pm 0.06}$
Empowerment	$0.00_{\pm 0.00}$	$0.46_{\pm 0.03}$	$1.95_{\pm 0.03}$
Random	$0.61_{\pm 0.03}$	$0.28_{\pm 0.01}$	$1.19_{\pm 0.06}$
Oracle	$0.98_{\pm 0.00}$	$0.91_{\pm 0.01}$	$0.13_{\pm 0.01}$
	OneRoomCapture3D Environment		
	Obj. Captured↑	Obj. Visible↑	$H(d^\pi(s_d))\downarrow$
Niche Creation+Infogain	$\mathbf{0.84}_{\pm 0.08}$	$\mathbf{0.88}_{\pm 0.02}$	$-0.06_{\pm 0.16}$
Niche Expansion, Eq. (4)	$0.73_{\pm 0.03}$	$0.67_{\pm 0.04}$	$\mathbf{-0.96}_{\pm 0.21}$
Niche Creation, Eq. (3)	$0.39_{\pm 0.04}$	$0.01_{\pm 0.00}$	$1.15_{\pm 0.16}$
Certainty, Eq. (2)	$0.29_{\pm 0.04}$	$0.16_{\pm 0.03}$	$0.66_{\pm 0.24}$
Infogain, Eq. (5)	$0.57_{\pm 0.03}$	$0.54_{\pm 0.04}$	$0.97_{\pm 0.14}$
SMiRL	$0.46_{\pm 0.04}$	$0.20_{\pm 0.02}$	$-0.02_{\pm 0.28}$
Empowerment	$0.49_{\pm 0.04}$	$0.22_{\pm 0.02}$	$0.21_{\pm 0.25}$
Random	$0.54_{\pm 0.03}$	$0.16_{\pm 0.01}$	$0.12_{\pm 0.22}$
Oracle	$0.95_{\pm 0.00}$	$0.96_{\pm 0.00}$	$-2.58_{\pm 0.23}$

Table 1: Policy evaluation in TwoRoom and OneRoomCapture3D. Means and their standard errors are reported; grey shading denotes a variant of our method, bolding denotes where a method achieves the best mean performance under a metric. We observe that the Niche Expansion and Niche Creation+Infogain objectives lead the agent to seek out and stabilize the dynamic objects substantially more effectively than other methods.

6. Related Work

Much of the previous work on learning without extrinsic rewards has been based either on (i) exploration (Chentanez et al., 2005; Oudeyer et al., 2007; Oudeyer & Kaplan, 2009), or (ii) some notion of intrinsic control, such as empowerment (Klyubin et al., 2005a; Mohamed & Jimenez Rezende, 2015; Karl et al., 2017). Exploration approaches include those that maximize model prediction error or improvement (Schmidhuber, 1991; Lopes et al., 2012; Stadie et al., 2015; Pathak et al., 2017), maximize model uncertainty (Houthooft et al., 2016; Still & Precup, 2012; Shyam et al., 2018; Pathak et al., 2019; Gheshlaghi Azar et al., 2019), maximize state visitation (Bellemare et al., 2016; Fu et al., 2017; Tang et al., 2017; Hazan et al., 2019), maximize surprise (Schmidhuber, 1991; Achiam & Sastry, 2017; Sun et al., 2011), and employ other novelty-based exploration bonuses (Lehman & Stanley, 2011; Burda et al., 2018; Kim et al., 2018; 2019). Our method can be combined with prior exploration techniques to aid in optimizing our proposed objective, and in that sense our work is largely orthogonal to prior exploration methods.

Prior works on intrinsic control include empowerment maximization (Klyubin et al., 2005a;b; Mohamed & Rezende, 2015), observational surprise minimization (Friston, 2009;

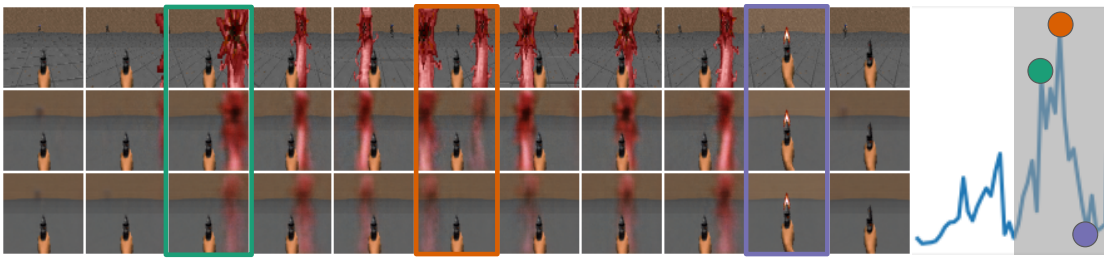


Figure 6: Visualization of a sequence in the VizDoom DefendTheLine environment. *Row 1:* The image provided to the agent. *Row 2:* The agent's reconstruction of a sample from q. *Row 3:* The agent's one-step image forecast. *Right:* The state infogain signal, $\mathbb{E}_q[\log q - \log p]$. Each colored rectangle identifies a keyframe that corresponds to a colored circle on the infogain plot. The infogain signal measures how much more certain the belief is compared to its temporal prior; when stochastic events happen (monster appears nearby), the signal is high; when the next image is predictable (monster disappears when shot), the signal is low.

Method	DefendTheCenter Environment		
	Env. Return↑	Monster Kills↑	Visible↑
Niche Creation+Infogain	1964.4±05.42	0.00±0.00	**1.16**±0.005
Niche Expansion, Eq. (4)	**2506.4**±36.26	**28.2**±1.50	0.94±0.015
Niche Creation, Eq. (3)	2182.2±14.12	10.0±0.55	0.76±0.017
Certainty, Eq. (2)	1958.0±35.63	07.2±0.73	0.92±0.028
Infogain, Eq. (5)	1928.6±35.51	00.2±00.2	1.15±0.008
SMiRL (Berseth et al., 2021)	1918.7±53.80	7.58±0.15	0.89±0.009
Empowerment (Zhao et al., 2021)	2161.8±29.06	16.2±2.35	0.86±0.064
Random policy	2113.8±37.60	14.2±0.73	0.90±0.026
Oracle policy	2550.8±55.90	28.0±1.52	0.80±0.035
	TwoRoom-Large Environment		
	Obj. Lock↑	Obj. Visible↑	$H(d^\pi(s_d))\downarrow$
Niche Creation+Infogain	**0.665**±0.013	**0.501**±0.013	**2.831**±0.083
SMiRL (Berseth et al., 2021)	0.110±0.020	0.087±0.016	3.417±0.022
Random policy	0.271±0.024	0.138±0.011	3.404±0.063
Oracle policy	0.885±0.005	0.464±0.017	1.016±0.043

Table 2: Policy evaluation in VizDoom and TwoRoom-Large. Means and their standard errors are reported; grey shading denotes a variant of our method, bolding denotes where a method achieves the best mean performance under a metric. We observe that the Niche Expansion objective in VizDoom and Niche Creation+Infogain objective in TwoRoom-Large lead the agent to seek out and stabilize the dynamic objects substantially more effectively than other methods.

Friston et al., 2009; Ueltzhöffer, 2018; Berseth et al., 2021; Parr & Friston, 2019), and skill discovery (Barto et al., 2004; Konidaris & Barto, 2009; Gregor et al., 2016; Eysenbach et al., 2018; Sharma et al., 2019; Xu et al., 2020). Observational surprise minimization seeks policies that make *observations* predictable and controllable, and is closely connected to entropy minimization, as entropy is defined to be the expected surprise. In Friston et al. (2010), the notion of Free Energy Minimization corresponds to minimizing *observational* entropy, and states that the entropy of hidden states in the environment is bounded by the entropy of sensory observations. However, the proof assumes a diffeomorphism to hold between states and observations, which is explicitly violated in CHMPs and any real-world setting, as agents cannot perceive the state of anything outside their egocentric sensory observations. Similarly, empowerment (Klyubin et al., 2005b; Karl et al., 2015; 2017; Zhao et al., 2021) is a measure of the degree of control an agent has over future *observations*, whereas state visitation entropy is a measure of the degree of the control over the *underlying environment state*. Our approach seeks to infer and gain control over a representation of the environment's state, as opposed to the agent's observations. We demonstrate environments where minimizing observational surprise and maximizing empowerment leads to degenerate solutions that ignore important factors of variation, whereas our approach identifies and controls them.

Representation learning methods have been explored in a variety of prior work, including, but not limited to, (Lange & Riedmiller, 2010; Watter et al., 2015; Karl et al., 2016; Nair et al., 2018; Zhang et al., 2018; Hafner et al., 2018; Lee et al., 2019). Our approach employs a representation learning method to build a latent state-space model (Watter et al., 2015; Krishnan et al., 2015; Karl et al., 2016; Hafner et al., 2018; Mirchev et al., 2018; Wayne et al., 2018; Vezzani et al., 2019; Lee et al., 2019; Das et al., 2019; Hafner et al., 2019; Mirchev et al., 2020; Rafailov et al., 2021).

7. Discussion

We presented a method, Believer, for intrinsically motivating an agent to discover, represent, and exercise control of dynamic objects in a partially-observed environments sensed with visual observations. We found that our method approached expert-level performance on several environments and substantially surpassed prior work in its unsupervised control capability. While our experiments represent a proof-of-concept that illustrates how latent state belief entropy minimization can incentivize an agent to both gather information and gain control over its environment, there are a number of exciting future directions. First, our method is inspired by a connection between thermodynamics and information theory, but the treatment of this connection is informal. Formalizing this connection could lead to an improved theoretical understanding of how Believer and other intrinsic motivation methods can lead to desirable behavior, and perhaps allow deriving conditions on environments under which such desirable behaviors would emerge. Second, Believer and other surprise-minimizing intrinsic motivation objectives are designed to work well in complex environments with unpredictable phenomena: a particularly interesting direction for future work is to scale up such methods in order to study the behavior that emerges.

References

Achiam, J. and Sastry, S. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

Aoki, E. H., Bagchi, A., Mandal, P., and Boers, Y. A theoretical look at information-driven sensor management criteria. In *14th International Conference on Information Fusion*, pp. 1–8. IEEE, 2011.

Barto, A. G., Singh, S., and Chentanez, N. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19. Piscataway, NJ, 2004.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Bennett, C. H. The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 21 (12):905–940, 1982.

Berseth, G., Geng, D., Devin, C. M., Rhinehart, N., Finn, C., Jayaraman, D., and Levine, S. {SM}irl: Surprise minimizing reinforcement learning in unstable environments. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=cPZOyoDloxl.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-Scale Study of Curiosity-Driven Learning. 2018. URL http://arxiv.org/abs/1808.04355.

Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

Chevalier-Boisvert, M. gym-miniworld environment for openai gym. https://github.com/maximecb/gym-miniworld, 2018.

Das, N., Karl, M., Becker-Ehmck, P., and van der Smagt, P. Beta dvbf: Learning state-space models for control from high dimensional observations. *arXiv preprint arXiv:1911.00756*, 2019.

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

Feder, H. J. S., Leonard, J. J., and Smith, C. M. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668, 1999.

Fountas, Z., Sajid, N., Mediano, P. A., and Friston, K. Deep active inference agents using monte-carlo methods. *arXiv preprint arXiv:2006.04176*, 2020.

Friston, K. The free-energy principle: a rough guide to the brain? *Trends in cognitive sciences*, 13(7):293–301, 2009.

Friston, K. Life as we know it. *Journal of the Royal Society Interface*, 10(86):20130475, 2013.

Friston, K. J., Daunizeau, J., and Kiebel, S. J. Reinforcement learning or active inference? *PLOS ONE*, 4(7):1–13, 07 2009. doi: 10.1371/journal.pone.0006421. URL https://doi.org/10.1371/journal.pone.0006421.

Friston, K. J., Daunizeau, J., Kilner, J., and Kiebel, S. J. Action and behavior: a free-energy formulation. *Biological cybernetics*, 102(3):227–260, 2010.

Fu, J., Co-Reyes, J., and Levine, S. Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in neural information processing systems*, pp. 2577–2587, 2017.

Gheshlaghi Azar, M., Piot, B., Avila Pires, B., Grill, J.-B., Altché, F., and Munos, R. World discovery models. *arXiv e-prints*, pp. arXiv–1902, 2019.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020a.

Hafner, D., Ortega, P. A., Ba, J., Parr, T., Friston, K., and Heess, N. Action and perception as divergence minimization. *arXiv preprint arXiv:2009.01791*, 2020b.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. VIME: Variational Information Maximizing Exploration. 2016. URL http://arxiv.org/abs/1605.09674.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Karl, M., Bayer, J., and van der Smagt, P. Efficient empowerment. *arXiv preprint arXiv:1509.08455*, 2015.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Karl, M., Soelch, M., Becker-Ehmck, P., Benbouzid, D., van der Smagt, P., and Bayer, J. Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv:1710.05101*, 2017.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. IEEE, 2016.

Kim, H., Kim, J., Jeong, Y., Levine, S., and Song, H. O. Emi: Exploration with mutual information. *arXiv preprint arXiv:1810.01176*, 2018.

Kim, Y., Nam, W., Kim, H., Kim, J.-H., and Kim, G. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International Conference on Machine Learning*, pp. 3379–3388, 2019.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. All else being equal be empowered. In Capcarrère, M. S., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J. (eds.), *Advances in Artificial Life*, pp. 744–753, Berlin, Heidelberg, 2005a. Springer Berlin Heidelberg. ISBN 978-3-540-31816-3.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135. IEEE, 2005b.

Konidaris, G. and Barto, A. G. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in neural information processing systems*, pp. 1015–1023, 2009.

Kreucher, C., Kastella, K., and Hero Iii, A. O. Sensor management using an active sensing approach. *Signal Processing*, 85(3):607–624, 2005.

Krishnan, R. G., Shalit, U., and Sontag, D. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Lange, S. and Riedmiller, M. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2010.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

Leff, H. S. and Rex, A. F. *Maxwell's demon: entropy, information, computing*. Princeton University Press, 2014.

Lehman, J. and Stanley, K. O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P.-Y. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in neural information processing systems*, pp. 206–214, 2012.

Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. W. Filtering variational objectives. *arXiv preprint arXiv:1705.09279*, 2017.

Magnasco, M. Szilard's heat engine. *EPL (Europhysics Letters)*, 33(8):583, 1996.

Maxwell, J. C. and Pesic, P. *Theory of heat*. Courier Corporation, 2001.

Mirchev, A., Kayalibay, B., Soelch, M., van der Smagt, P., and Bayer, J. Approximate bayesian inference in spatial environments. *arXiv preprint arXiv:1805.07206*, 2018.

Mirchev, A., Kayalibay, B., van der Smagt, P., and Bayer, J. Variational state-space models for localisation and dense 3d mapping in 6 dof. *arXiv preprint arXiv:2006.10178*, 2020.

Mohamed, S. and Jimenez Rezende, D. Variational information maximisation for intrinsically motivated reinforcement learning. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2125–2133. Curran Associates, Inc., 2015.

Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.

Oudeyer, P.-Y. and Kaplan, F. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2): 265–286, 2007.

Parr, T. and Friston, K. J. Generalised free energy and active inference. *Biological Cybernetics*, 113(5):495–513, Dec 2019. ISSN 1432-0770. doi: 10.1007/s00422-019-00805-w. URL https://doi.org/10.1007/s00422-019-00805-w.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven Exploration by Self-supervised Prediction. 2017.

Pathak, D., Gandhi, D., and Gupta, A. Self-Supervised Exploration via Disagreement. 2019.

Rafailov, R., Yu, T., Rajeswaran, A., and Finn, C. Offline reinforcement learning from images with latent space models. *Learning for Decision Making and Control (L4DC)*, 2021.

Schmidhuber, J. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991.

Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv: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*, pp. 8583–8592. PMLR, 2020.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Shyam, P., Jaśkowski, W., and Gomez, F. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018.

Stadie, B. C., Levine, S., and Abbeel, P. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Still, S. and Precup, D. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Sun, Y., Gomez, F., and Schmidhuber, J. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pp. 41–51. Springer, 2011.

Sutskever, I. *Training recurrent neural networks*. University of Toronto Toronto, Canada, 2013.

Szilard, L. Über die entropieverminderung in einem thermodynamischen system bei eingriffen intelligenter wesen. *Zeitschrift für Physik*, 53(11):840–856, 1929.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pp. 2753–2762, 2017.

Ueltzhöffer, K. Deep active inference. *Biological Cybernetics*, 112(6):547–573, 2018.

Vezzani, G., Gupta, A., Natale, L., and Abbeel, P. Learning latent state representation for speeding up exploration. *arXiv preprint arXiv:1905.12621*, 2019.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

Wayne, G., Hung, C.-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

Williams, J. L. *Information Theoretic Sensor Management*. PhD thesis, Massachusetts Institute of Technology, 2007.

Xu, K., Verma, S., Finn, C., and Levine, S. Continual learning of control primitives: Skill discovery via reset-games. *arXiv preprint arXiv:2011.05286*, 2020.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M. J., and Levine, S. Solar: Deep structured latent representations for model-based reinforcement learning. *arXiv preprint arXiv:1808.09105*, 2018.

Zhao, R., Lu, K., Abbeel, P., and Tiomkin, S. Efficient empowerment estimation for unsupervised stabilization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=u2YNJPcQlwq.

8. Experimental Details

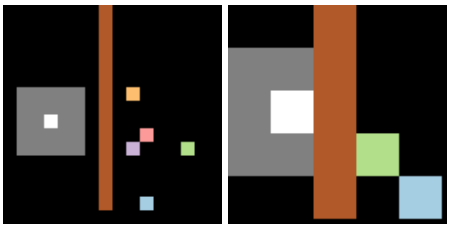
Please visit this anonymous website on which we host videos: https://sites.google.com/view/believer-anonymous/home.

Computational resources. Most experimental results were computed on a Linux Destkop with 32 GiB of RAM, equipped with an AMD Ryzen 7 3800X 8-core CPU and an RTX 2080 Ti GPU.

True-state entropy metric. We approximated $H(d^\pi(s_d))$ by computing an estimated $d^\pi(s_d)$ during the episode. We report the entropy at the final step of the episode, which is when the estimate of $d^\pi(s_d)$ is most precise. Recall that s_d represents the positions of the dynamic objects in the environment. In the TwoRoom environment, $d^\pi(s_d)$ is computed by recording counts. In the OneRoomCapture3D environment, s_d is continuous, and $d^\pi(s_d)$ is computed by fitting a diagonal Gaussian.

8.1. Environment details.

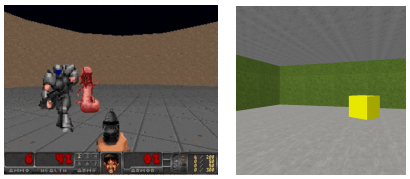
In this section, we elaborate on the details of the environments described in the main text.



(a) TwoRoom Large Environment (b) TwoRoom Environment

Figure 7: TwoRoom Environments. In the large environment, the agent observes a 5x5 area around it as an image, and the busy room contains 5 particles. In the normal environment, the agent observes a 3x3 around it as an image, and the busy room contains 2 particles. In both settings, the particles are initialized to random positions in the busy room at the beginning of each episode.

TwoRoom environment details. As previously described, this environment has two rooms: an empty ("dark") room on the left, and a "busy" room on the right, the latter containing moving particles that move around unless the agent "tags" them, which permanently stops their motion, as shown in Fig. 7. The agent can observe a small area around it, which it receives as an image. In this environment, control corresponds to finding and stopping the particles. The action space is $\mathcal{A} = \{\text{left, right, up, down, tag, no-op}\}$, and the observation space is normalized RGB-images: $\Omega = [0,1]^{3\times30\times30}$. An agent that has significant control over this environment should tag particles to reduce the uncertainty over future states. To evaluate policies, we use the average fraction of total particles locked, the average fraction of particles visible, and the discrete true-state visitation entropy of the positions of the particles, $H(d^\pi(s_d))$. We employed two versions of this environment. In the large environment, the agent observes a 5x5 area around it as an image, and the busy room contains 5 particles. In the normal environment, the agent observes a 3x3 around it as an image, and the busy room contains 2 particles. The large environment consists of an area of 15x15 cells, and the normal environment consists of an area of 5x5 cells. In both settings, the particles are initialized to random positions in the busy room at the beginning of each episode, and episodes last for $T = 100$ timesteps. The particles bounce off the walls, but not each other.



(a) Vizdoom DefendTheCenter. (b) OneRoomCapture3D

Figure 8: Vizdoom Defend The Center and OneRoomCapture3D

VizDoom DefendTheCenter. The VizDoom DefendTheCenter environment shown in Fig. 8 is a circular arena in which an agent, equipped with a partial field-of-view of the arena and a weapon, can rotate and shoot encroaching monsters (Kempka et al., 2016). The action space is $\mathcal{A} = \{\text{turn left, turn right, shoot}\}$, and the observation space is normalized RGB-images: $\Omega = [0,1]^{3\times64\times64}$. In this environment, significant control corresponds to reducing the number of monsters by finding and shooting them. We use the average original environment return (for which *no method* has access to during training) the average number of killed monsters at the end of an episode, and the average number of visible monsters to measure the agent's control. Episodes last for $T = 500$ timesteps.

OneRoomCapture3D The MiniWorld framework is a customizable 3D environment simulator in which an agent perceives the world through a perspective camera (Chevalier-Boisvert, 2018). We used this framework to build the environment in Fig. 8 which an agent and a bouncing box both inhabit a large room; the agent

can lock the box to stop it from moving if it is nearby, as well as constrain the motion of the box by standing nearby it. In this environment, significant control corresponds to finding the box and either trapping it near a wall, or tagging it. When the box is tagged, its color changes from yellow to purple. The action space is $\mathcal{A} = \{\text{turn left } 20°, \text{turn right } 20°, \text{move forward, move backward, tag}\}$, and observation space is normalized RGB-images: $\Omega = [0, 1]^{3\times64\times64}$. We use the average fraction of time the box is captured, the average time the box is visible, and continuous (Gaussian-estimated) true-state visitation entropy of the box's position $H(d^\pi(s_d))$ to measure the agent's ability to reduce entropy of the environment. Episodes last for $T = 150$ timesteps.

9. Implementation Details

Hyperparameters. In Table 3, we provide values of the hyperparameters used in Algorithm 1 and the LSSM architecture described in Table 4.

Architectural details. We provide detailed information on the architectural implementation of the learners in Table 4. The value function used for generalized advantage estimation in PPO uses the same architecture as the policy with decoupled weights, except with a final output size of 1 (scalar).

Optimization. We use RAdam to optimize the policies and LSSM (Liu et al., 2019). Following Hafner et al. (2020a), we use straight-through gradient estimation of samples of the Categorical distributions. This is straightforwardly implemented in PyTorch as shown in Listing 1.

10. OneRoomCapture3d visualization.

In Fig. 9, we analyze the behavior of our intrinsic reward signals over several different sub-episodes in the OneRoomCapture3D environment.

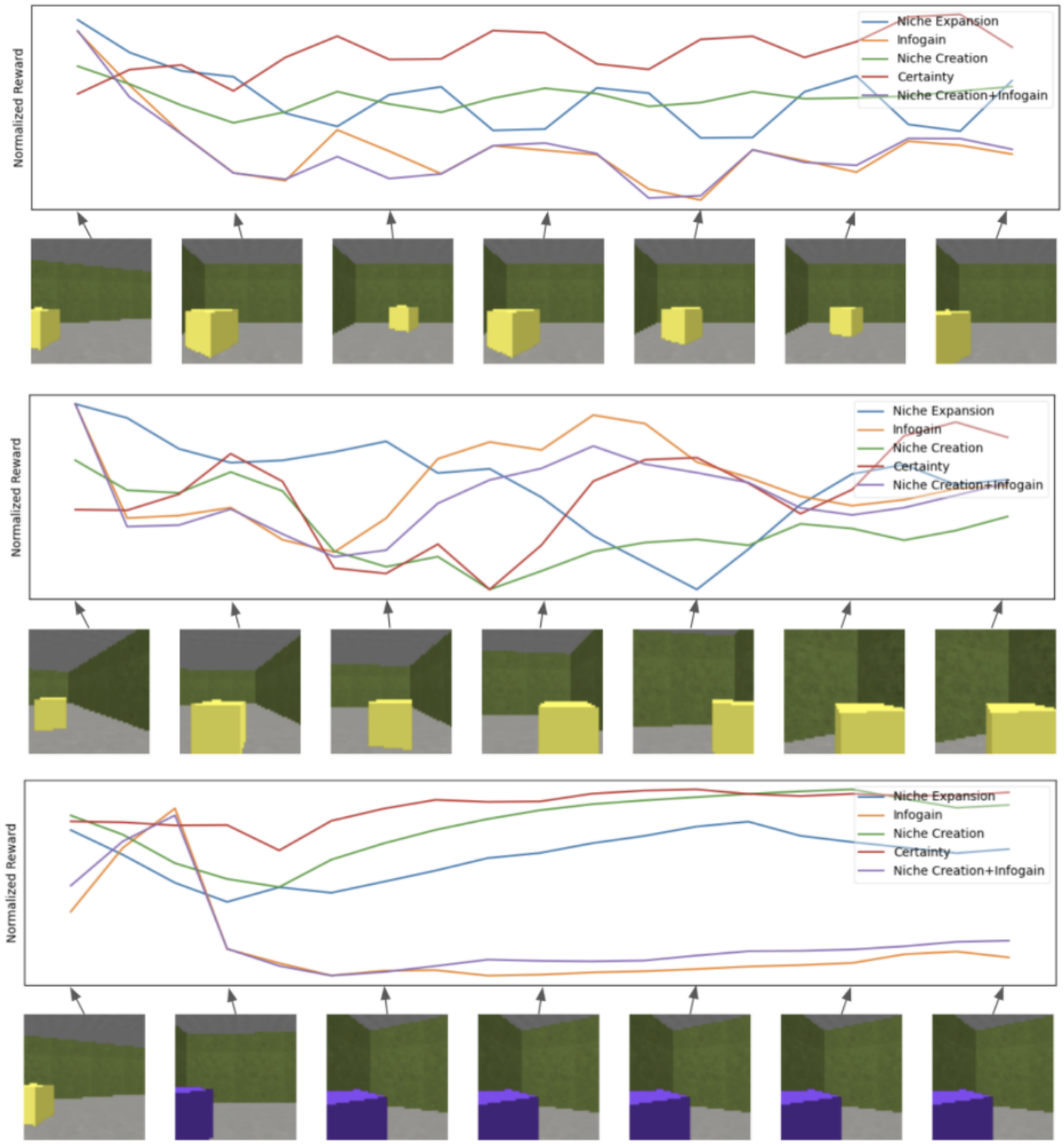


Figure 9: Niche Expansion, Infogain, Niche Creation, Certainty, and Niche Creation+Infogain rewards are plotted for the first 20 steps of select episodes. Rewards are normalized to [0, 1] for each reward across the figures. *Top:* When the agent turns to look at the box without taking actions to capture it, all rewards other than Certainty are relatively low throughout the episode. *Middle:* When the agent moves towards the box to trap it against a wall, Niche Creation and Niche Expansion decrease until the box is trapped, and then they increase; the resulting stable configuration eventually outweighs the preparations needed to trap the box if the episode length is sufficiently long. Infogain and Certainty increase as the box is in view and able to move. *Bottom:* Freezing the box results in low Infogain throughout the episode, however it is highly rewarded by the other rewards.

Hyperparameter	Value	Meaning
	Algorithm 1 hyperparameters	
K	7	Ensemble size
B	32	Minibatch size of LSSM
L	$0.05\|\mathcal{D}\|/B$	Number of minibatches to step the model
M	$1e7/2NT$	Maximum number of total rounds
N	20	Number of episodes to collect per policy per round
T	{100,150,500} (varies)	Episode length in the environment
	LSSM hyperparameters	
H	50	LSSM model training horizon
P	5	Number of latent particles
K_1	16	Number of component categoricals in latent distributions
K_2	16	Number of categories in each component categorical
	Optimization hyperparameters	
α_0	$0.5 \cdot 10^{-4}$	LSSM learning rate with Adam optimizer
α_1	$1.0 \cdot 10^{-4}$	PPO learning rate with Adam optimizer
ϵ_{PPO}	0.2	PPO advantage clipping
γ_0	0.99	PPO discount factor
γ_1	0.90	PPO GAE discount factor
β	1.0	KL loss scaling factor (implicit in Eq. (1))
b_0	True	Whether truncated BPTT (Sutskever, 2013) is used to train the model
j_1, j_2	50	Truncated BPTT horizons

Table 3: Hyperparameters of Algorithm 1, models, and optimization.

.

Listing 1 Straight-through One Hot Categorical implementation.

```
class StraightThroughOneHotCategorical(torch.distributions.OneHotCategorical):
    def rsample(self, *args, **kws):
        return self.sample(*args, **kws) + self.probs - self.probs.detach()
```

Listing 2 MultiCat implementation.

```
def MultiCat(state_logits):
    assert(state_logits.shape[-2:] == (K_1, K_2))
    return torch.distributions.Independent(
              StraightThroughOneHotCategorical(logits=state_logits),
              reinterpreted_batch_ndims=1)
```

Layer	Input [Dimensionality]	Output [Dimensionality]
	Prior $p_\theta(\mathbf{z}_{t+1}\|\mathbf{z}_t, \mathbf{a}_t, g_t) = \prod_{\kappa_1=1}^{K_1}\prod_{\kappa_2=1}^{K_2} \mathbf{v}_{\text{prior},\kappa_1,\kappa_2}^{\mathbf{z}_{t+1,\kappa_1,\kappa_2}} = \text{MultiCat}(\cdot; \mathbf{v}_{\text{prior}})$ (see Listing 2)	
	`//Embed action with affine layer`	
1	$\mathbf{a}_t$ $[A]$	L_a=`Linear(`$\mathbf{a}_t$`)`, $[K_1 \cdot K_2]$
	`//Combine action embedding and latent state`	
2	$\mathbf{z}_t$ $[K_1, K_2]$; L_a $[K_1 \cdot K_2]$	L_{az}=`Concat(Flatten(`$\mathbf{z}_t$`),` L_a`)`, $[2K_1 \cdot K_2]$
	`//RNN transformation of action embedding and latent state`	
3	L_{az}, $[2K_1 \cdot K_2]$; g_t, $[K_1 \cdot K_2]$	g_{t+1}=`GRU(`L_{az}, g_t`)`, $[K_1 \cdot K_2]$
	`//Logits of independent Categorical prior ("MultiCat")`	
4	g_{t+1}, $[K_1 \cdot K_2]$	$\mathbf{l}_{\text{prior}}$=`Linear(`$g_{t+1}$`)`, $[K_1 \cdot K_2]$
	`//Transform logits of all` K_1 `component dists.`	
5	$\mathbf{l}_{\text{prior}}$,$[K_1 \cdot K_2]$	$\mathbf{v}_{\text{prior}}$=`softmax(`$\mathbf{l}_{\text{prior}}$, axis $= K_2$`)`
	Posterior $q_\phi(\mathbf{z}_{t+1} \mid \mathbf{o}_{\leq t+1}, \mathbf{a}_{\leq t}, \mathbf{z}_t, g_t) = \text{MultiCat}(\cdot; \mathbf{v}_{\text{posterior}})$	
	`//Apply CNN to observation.`	
1	$\mathbf{o}_t$, $[3, 64, 64]$	L_{o0}=`ELU(BN(Conv2d(3, 32, 4, 2)))(`$\mathbf{o}_t$`)`
2	L_{o0}, $[32, 31, 31]$	L_{o1}=`ELU(BN(Conv2d(32, 64, 4, 2)))(`L_{o0}`)`
3	L_{o1}, $[64, 14, 14]$	L_{o2}=`ELU(BN(Conv2d(64, 128, 4, 2)))(`L_{o1}`)`
4	L_{o2}, $[128, 6, 6]$	L_{o3}=`ELU(BN(Conv2d(128, 256, 4, 2)))(`L_{o2}`)`
5	L_{o3}, $[256, 2, 2]$	E_o=`Flatten(`L_{o2}`)`, [1024]
	`//Produce observation-specific posterior parameters.`	
6	E_o, [1024]; $\mathbf{a}_t$, [A]	$\mathbf{l}'_{\text{posterior}}$=`Linear(Concat(`$E_o$, $\mathbf{a}_t$`))`, $[K_1 \cdot K_2]$
	`//Produce final posterior parameters as log-space addition to prior parameters.`	
7	$\mathbf{l}'_{\text{posterior}}$, $[K_1 \cdot K_2]$; $\mathbf{l}_{\text{prior}}$, $[K_1 \cdot K_2]$	$\mathbf{l}_{\text{posterior}} = \mathbf{l}'_{\text{posterior}} + \mathbf{l}_{\text{prior}}$, $[K_1 \cdot K_2]$
	`//Transform logits of all` K_1 `component dists.`	
8	$\mathbf{l}_{\text{posterior}}$,$[K_1 \cdot K_2]$	$\mathbf{v}_{\text{posterior}}$=`softmax(`$\mathbf{l}_{\text{posterior}}$, axis $= K_2$`)`, $[K_1 \cdot K_2]$
	Observation Likelihood $p_\theta(\mathbf{o}_t\|\mathbf{z}_t) = \mathcal{N}(\cdot; \mu_o, I)$	
	`//Embed latent state with affine layer to consistently-sized vector.`	
1	$\mathbf{z}_t$, $[K_1 \cdot K_2]$	E_z=`Linear(`$\mathbf{z}_t$`)`, [1024]
	`//Apply transposed-CNN to decode to observation dimensionality.`	
2	E_z, [1024]	L_{z0}=`ELU(BN(ConvTranspose2d(1024, 128, 5, 2)))(`E_z`)`
3	L_{z0}, $[128, 5, 5]$	L_{z1}=`ELU(BN(ConvTranspose2d(128, 64, 5, 2)))(`L_{z0}`)`
4	L_{z1}, $[64, 13, 13]$	L_{z2}=`ELU(BN(ConvTranspose2d(64, 32, 6, 2)))(`L_{z1}`)`
5	L_{z2}, $[32, 30, 30]$	L_{z3}=`ELU(BN(ConvTranspose2d(32, 3, 6, 2)))(`L_{z2}`)`
6	L_{z3}, $[3, 64, 64]$	L_{z4}=`ELU(BN(ConvTranspose2d(32, 3, 6, 2)))(`L_{z3}`)`
	`//Output of final layer is the mean of the observation likelihood.`	
7	L_{z4}, $[3, 64, 64]$	μ_o=`ELU(BN(ConvTranspose2d(3, 3, 1, 1)))(`L_{z4}`)`
	Belief $q_\phi(\mathbf{z}_{t+1} \mid \mathbf{o}_{\leq t+1}, \mathbf{a}_{\leq t}) = q(\mathbf{z}_{t+1}\|\mathbf{h}_{t+1}) = {}^1\!/_P \sum_{p=0}^{P} w_p q_\phi(\mathbf{z}_{t+1,p} \mid \mathbf{o}_{\leq t+1}, \mathbf{a}_{\leq t}, \mathbf{z}_{tp}, g_t)$	
	`//Compute unnormalized log-space particle weights of` P `latent particles`	
1	$\{(\mathbf{z}_{t,p}, \mathbf{z}_{t-1,p})\}_{p=1}^{P}$	$\log \hat{w} = \log \frac{p_\theta(\mathbf{z}_{t+1,p}\|\mathbf{z}_{t,p},\mathbf{a}_t) p_\theta(\mathbf{o}_t\|\mathbf{z}_{t,p})}{q_\phi(\mathbf{z}_{t+1,p}\|\mathbf{o}_{\leq t+1},\mathbf{a}_{\leq t},\mathbf{z}_{t,p})}$, $[P]$
	`//Form belief from weighted mixture over particles.`	
2	w, $[P]$	`MixtureSameFamily(`$\{q_\phi(\mathbf{z}_{t+1,p} \mid \mathbf{o}_{\leq t+1}, \mathbf{a}_{\leq t}, \mathbf{z}_{t,p})\}_{p=1}^{P}, w$`)`
	Latent Visitation Model $\bar{q}_{t'}(\mathbf{z}) = {}^1\!/_{t'} \sum_{t=0}^{t'} q_\phi(\mathbf{z}_t\|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$	
	`//Define uniform mixture weights.`	
1	$\emptyset$	$c_0 = {}^1\!/_{t'}$`torch.ones((1,t'))`, $[1, t']$
	`//Form uniform mixture over previous beliefs.`	
2	$\{q_\phi(\mathbf{z}_t\|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})\}_{t=0}^{t'}$; c_0, $[1, t']$	$\bar{q}_{t'}(\mathbf{z})$ =`MixtureSameFamily(`$\{q_\phi(\mathbf{z}_t\|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})\}_{t=0}^{t'}, c_0$`)`
	Policy $\pi(\mathbf{a}_t\|\mathbf{v}_{\text{posterior}}) =$ `Categorical(`$\cdot; p_a$`)`	
1	$\mathbf{v}_{\text{posterior}}$, $[K_1 \cdot K_2]$	h_0=`tanh(Linear(`$\mathbf{v}_{\text{posterior}}$`))`, [128]
2	h_1, [128]	h_1=`tanh(Linear(`h_0`))`, [128]
	`//Compute categorical action distribution parameters.`	
3	h_2, [128]	$\hat{p}_a$=`Linear(`h_2`)`, $[\|\mathcal{A}\|]$
4	$\hat{p}_a$, $[\|\mathcal{A}\|]$	p_a=`sum(`$\hat{p}_a$`, -1)`, $[\|\mathcal{A}\|]$

Table 4: **Latent state-space model, visitation model, and policy architectural details:** The inputs to the latent state-space model are RGB images $\mathbf{o}_t \in [0, 1]^{3\times64\times64}$ and actions $\mathbf{a}_t \in \{0, 1\}^A$ (one-hot). Pytorch layer notation is used as shorthand. g_t represents the GRU state at t.