Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Learning Task-Driven Control Policies via Information Bottlenecks

Vincent Pacelli and Anirudha Majumdar

***Abstract*—This paper presents a reinforcement learning approach to synthesizing *task-driven* control policies for robotic systems equipped with rich sensory modalities (e.g., vision or depth). Standard reinforcement learning algorithms typically produce policies that tightly couple control actions to the *entirety* of the system's state and rich sensor observations. As a consequence, the resulting policies can often be sensitive to changes in *task-irrelevant* portions of the state or observations (e.g., changing background colors). In contrast, the approach we present here learns to create a task-driven representation that is used to compute control actions. Formally, this is achieved by deriving a policy gradient-style algorithm that creates an *information bottleneck* between the states and the task-driven representation; this constrains actions to only depend on task-relevant information. We demonstrate our approach in a thorough set of simulation results on multiple examples including a grasping task that utilizes depth images and a ball-catching task that utilizes RGB images. Comparisons with a standard policy gradient approach demonstrate that the task-driven policies produced by our algorithm are often significantly more robust to sensor noise and task-irrelevant changes in the environment.**

I. Introduction

The increasing availability of high-resolution sensors has significantly contributed to the recent explosion of robotics applications. For example, high-precision cameras and LIDARs now allow autonomous vehicles to perform tasks ranging from navigating busy city streets to mapping mines and buildings. By far the most common approach adopted by such systems is to utilize as much sensor information as is available in order to estimate the full state of these complex environments; the resulting state estimates are then used by the robot to choose control actions necessary to complete its task. However, this ubiquitous approach does not distinguish between the *task-relevant* and *task-irrelevant* portions of the system's state. The result is an unnecessarily tight coupling between sensor observations and control actions that is often not robust to noise or uncertainty in irrelevant portions of the state. Moreover, these policies tend to have a large computational burden. This computational requirement can manifest either as a state estimator that needs to process large amounts of data in real time, or as a policy learned from sampling a large number of diverse operating environments. The goal of this paper is to address these challenges by synthesizing control policies that only depend on *task-driven representations* of the robot's state. Doing so reduces the coupling between the states and control actions, improves robustness, and reduces computational requirements.

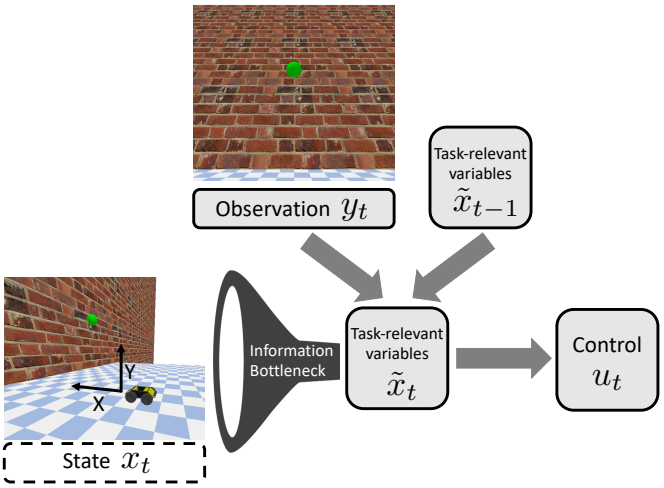


Fig. 1. A depiction of our approach applied to the ball-catching problem described in Section IV-B. Our policy implements an information bottleneck that limits the amount of state information (in this case, the robot and ball position and velocities) extracted from the sensor observations (RGB images) to create a set of task-relevant variables (TRVs) on which the control action (robot velocity) depends. The result is a policy that is more robust to image noise and changes to the brick background texture.

To illustrate the advantages of a task-driven policy, consider a ball-catching example. An agent equipped with a high-resolution camera is tasked with catching a ball (Figure 1). Using its sensor, the agent can attempt to estimate the full state of the system (e.g., ball velocity, ego-motion, wind speed, etc.), feed this information into a physical model for the ball's flight, integrate the model to find where the ball will land, and move to this location to catch it. This approach requires estimating every parameter involved in the ball's motion and can easily be compromised when the ball is far away and difficult to see or when there are visual artifacts such as glare.

Instead, extensive cognitive psychology experiments [11, 29, 36] have demonstrated that humans employ a task-driven strategy known as the gaze heuristic to catch projectiles. This strategy simply modulates the human's speed in order to fix the position of the ball in their visual field. This policy, which only requires minimal sensor information and internal computation, naturally drives the human to the ball's landing position. The gaze heuristic highlights that task-driven control policies are often robust control policies that only depend on small amounts of salient state information. Specifically, the heuristic is naturally *robust to distributional shifts* in irrelevant portions of the environment (e.g., the visual backdrop) while also being adaptable to perturbations like a gust of wind altering the ball's

This work is partially supported by the National Science Foundation [IIS-1755038], the Office of Naval Research [Award Number: N00014-18-1-2873], the Google Faculty Research Award, and the Amazon Research Award. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

The authors are with the Mechanical and Aerospace Engineering department at Princeton University, NJ, 08540, USA {vpacelli, ani.majumdar}@princeton.edu

course. Traditionally, task-driven policies have required hand-engineering for each control problem. This process can be difficult and time consuming. The goal of this paper is to propose a reinforcement learning framework that *automatically synthesizes* task-driven control policies for systems with nonlinear dynamics and high-dimensional observations (e.g., RGB or depth images).

Statement of Contributions. The main technical contribution of this paper is to formulate a reinforcement learning algorithm that synthesizes task-driven control policies. This synthesis is achieved by creating an *information bottleneck* [44] that limits how much state information the policy is allowed to use. We present a reinforcement learning algorithm — referred to as task-driven policy gradient (TDPG) — that leverages the recently-proposed mutual information neural estimator [4] to tractably search for an effective task-driven policy. Finally, we demonstrate that this formulation provides the key advantage of a task-driven approach — robustness to perturbations in task-irrelevant state and sensor variables. This benefit is demonstrated in three examples featuring nonlinear dynamics and high-dimensional sensor models: an adaption of the lava problem from the literature on partially observable Markov decision processes (POMDPs), ball catching using RGB images, and grasping an object using depth images.

A. Related Work

State Estimation and Differentiable Filtering. Classical approaches to controlling robotic systems typically involve two distinct pipelines: one that uses the robot's sensors to estimate its state and another that uses this state estimate to choose control actions. Such an architecture is motivated by the *separation principle* [3] from (linear) control theory and allows one to leverage powerful control-theoretic techniques for robust estimation and control [9, 48]. A recent line of work on *differentiable filtering* [16, 18, 20] has extended this traditional pipeline to elegantly handle rich sensor observations (e.g., images) by *learning* state estimators in an end-to-end manner via deep learning. However, as the gaze heuristic example from Section I demonstrates, full state representations are often overly rich when viewed from the perspective of the task at hand. This is particularly true in settings where representing the full state requires capturing the state of the robot's environment. Instead, our goal is to learn minimalistic *task-driven* representations that are sufficient for control. Such representations can be highly compact as compared to full state representations. Moreover, carefully-constructed task-driven representations have the potential to be robust to sensor noise and changes to irrelevant portions of the robot's environment. Intuitively, this is because uncertainty or noise in irrelevant portions of the sensor observations are filtered out and thus no longer corrupt the robot's actions. We present a theoretical result in Section II along with simulation experiments in Section IV to support this intuition.

End-to-end Learning of Policies. Deep reinforcement learning approaches have the ability to learn control policies in an *end-to-end* manner [14, 15, 21, 26, 27, 40, 49]. Such end-to-end approaches learn to create representations that are tuned to the task at hand by exploiting statistical regularities in the robot's observations, dynamics, and environment. However, these methods do not explicitly attempt to learn representations that are task-driven (i.e., representations that filter out portions of the sensor observations that are irrelevant to the task). As a result, policies trained via standard deep RL techniques may be sensitive to changes in irrelevant portions of the robot's environment (e.g., changes to the background color in a ball-catching task). In contrast, the deep RL-based approach we present seeks to explicitly learn task-driven representations that filter out irrelevant factors. Our simulation results in Section IV empirically demonstrate that our approach is robust to such *distributional shifts*.

Information Bottlenecks. Originally developed in the information theory literature, *information bottlenecks* [44, 2, 25, 43] allow one to formalize the notion of a "minimal-information representation" that is sufficient for a given task (e.g., a prediction task in the context of supervised learning). Given an input random variable X and a target random variable Y, one seeks a representation $\tilde{X}$ that forms a Markovian structure $X \rightarrow \tilde{X} \rightarrow Y$. The representation is chosen to minimize the mutual information between X and $\tilde{X}$ while still maintaining enough information to predict Y from $\tilde{X}$. Recent work has sought to adapt this theory for synthesizing task-driven representations for control [30, 1]. In [30], a model-based approach for automatically synthesizing task-driven representations via information bottlenecks is presented. However, this approach is limited to settings with an explicit model of the robot's sensor and dynamics. This prevents the approach from being applied to systems with rich sensing modalities (e.g. RGB or depth images), for which one cannot assume a model. In contrast, the approach we present here *learns* to create a task-driven representation via reinforcement learning and is directly applicable to settings with rich sensing. In [1], the authors use information bottlenecks to define minimal state representations for control tasks involving high-dimensional sensor observations (this approach is also related to the notion of *actionable information* [38, 39] in vision; see [1] for a discussion). However, [1] does not present concrete algorithms for learning task-driven representations. In contrast, we present a policy gradient algorithm based on *mutual information neural estimation* (MINE) [4]. Moreover, we note that our definition of a task-driven policy differs from that of [1]; the representations we learn seek to create a bottleneck between sensor observations and control actions as opposed to finding a minimal representation for predicting costs.

Exploration in RL via Mutual Information Regularization. The information bottleneck principle has also been used to improve sample efficiency in RL by encouraging exploration [12, 45]. More generally, there is a line of work on endowing RL agents with *intrinsic motivation* by maximizing the mutual information between the agent's control actions and states [24, 32, 33, 42, 46, 47]. Our focus instead is on improving robustness and generalization of policies; the information bottleneck-based approach we present is aimed at learning task-driven representations for control. This is achieved by minimizing the mutual information between the

agent's states and learned task-relevant variables.

II. Defining Task-Driven Policies

In this section, we formalize our notion of a task-driven control policy. Our definition is in terms of a reinforcement learning problem whose solution produces a set of task-relevant variables (TRVs) and a policy that depends only on these variables. We begin by formulating the problem as a finite-horizon partially-observable Markov decision process (POMDP) [41]. The robot's states, control actions, and sensor observations at time $t \in \{0, \dots, T\}$ are denoted by $x_t \in \mathcal{X}_t$, $u_t \in \mathcal{U}_t$, and $y_t \in \mathcal{Y}_t$ respectively. The robot dynamics and sensor model are denoted by the conditional distributions $p(x_{t+1}|x_t, u_t)$ and $s(y_t|x_t)$ respectively. We do not assume knowledge of either of these distributions. The functions $c_0(x_0, u_0), \dots, c_{T-1}(x_{T-1}, u_{T-1})$ describe the cost at each time step with a terminal cost specified by $c_T(x_t, u_t) \coloneqq c_T(x_T)$. Our goal is to find a policy $\pi_t(u_t|y_t)$ that solves

$$\underset{\pi_t(u_t|y_t)}{\text{minimize}} \ \mathbb{E}\left[c(\tau)\right] \coloneqq \mathbb{E}\left[\sum_{t=0}^{T} c_t\right], \tag{1}$$

when run online in a test environment. Throughout this paper, we use $\mathbb{E}[\cdot]$ to denote the expectation; it is subscripted with a distribution when necessary for clarity.

To achieve our goal of learning a task-driven policy, we choose the policy to have the recurrent structure illustrated in Figure 1. We refer to $\tilde{x}_t \in \tilde{\mathcal{X}}_t$ as the *task-relevant variables* (TRVs) and search for two conditional distributions: $q(\tilde{x}_t|\tilde{x}_{t-1}, y_t), \pi(u_t|\tilde{x}_t)$. The first specifies a (stochastic) mapping from the previous TRVs and the current observation to the current TRVs. The second conditional distribution computes the control actions given the current TRVs. These will be parameterized by neural networks in the following sections. The overall policy can thus be expressed as:

$$\pi(u_t|y_t, \tilde{x}_{t-1}) = \int_{\tilde{\mathcal{X}}_t} \pi(u_t|\tilde{x}_t) q(\tilde{x}_t|\tilde{x}_{t-1}, y_t) \ \mathrm{d}\tilde{x}_t.$$

Our goal is to learn TRVs that form a *compressed representation* of the state that is sufficient for the purpose of control. To formalize this, we leverage the theory of information bottlenecks [44] to limit how much information $\tilde{x}_t$ contains about x_t. This is quantified using the mutual information

$$\mathbb{I}[x_t; \tilde{x}_t] \coloneqq \mathbb{D}\left[p_t(x_t, \tilde{x}_t) \| p_t(x_t) q_t(\tilde{x}_t)\right] \tag{2}$$

between the state x_t and TRVs $\tilde{x}_t$ at each time step. Here, $\mathbb{D}[\cdot\|\cdot]$ is the Kullback-Leibler (KL) divergence [8]. Intuitively, minimizing the mutual information corresponds to learning TRVs that are as independent of the state as possible. Thus, the TRVs "filter out" irrelevant information from the state. Thus, we would like to find a policy that solves:

$$\underset{\substack{q(\tilde{x}_t|\tilde{x}_{t-1}, y_t) \\ \pi_t(u_t|\tilde{x}_t)}}{\text{minimize}} \ \mathcal{J} \coloneqq \beta \mathbb{E}\left[c(\tau)\right] + \sum_{t=0}^{T} \mathbb{I}[x_t; \tilde{x}_t]. \tag{3}$$

Here, β can be viewed as a Lagrange multiplier, and the above problem can be interpreted as minimizing the total information contained in $\tilde{x}_t$ subject to an upper bound on the expected cost over the time horizon.

In Section III, we develop a reinforcement learning algorithm for tackling (3). We refer to the resulting policies as *task-driven policies*. Note that the mutual information is invariant under bijective transforms of the random variables for which it is computed [4, 8]. As a result, the optimum value of $\mathcal{J}$ remains unchanged for equivalent representations of the robot and its environment.

Similar to the gaze heuristic (ref. Section I), we expect that a task-driven policy as defined above will be a robust policy. This benefit is conferred to our policy by minimizing the mutual information between states and TRVs. Intuitively, a policy is closer to being open-loop when less state information is present in $\tilde{x}_t$. The more open-loop the policy is, the less it is impacted by changes in the state or sensor distributions between environments. In [30], this intuition is formalized using the theory of *risk metrics* with the following theorem:

Theorem II.1. *Define the* entropic risk metric *[37, Example 6.20]:*

$$\rho_\beta\left[c_t\right] \coloneqq \frac{1}{\beta} \log\Big[\mathbb{E}_{p_t} \exp(\beta c_t)\Big]. \tag{4}$$

Let $\check{p}_t(x_t, \tilde{x}_t, u_t)$ be any distribution satisfying:

$$\begin{aligned} &\beta \mathbb{D}[\check{p}_t(x_t, \tilde{x}_t, u_t) \| p_t(x_t, \tilde{x}_t, u_t)] \\ &\leq \mathbb{D}[p_t(x_t, \tilde{x}_t) \| p_t(x_t) q_t(\tilde{x}_t)]. \end{aligned} \tag{5}$$

Then, the online expected cost is bounded by a combination of the entropic risk and mutual information:

$$\mathbb{E}_{\check{p}_t}\left[c(\tau)\right] \leq \sum_{t=0}^{T} \Big(\rho_\beta\left[c_t\right] + \mathbb{I}[x_t; \tilde{x}_t]\Big). \tag{6}$$

The entropic risk is a functional that is similar to the expectation, but also accounts for higher moments of the distribution, e.g. its variance. The parameter β controls how much the metric weights the expected cost versus the higher moments, and $\lim_{\beta \to 0} \rho_\beta[c_t] = \mathbb{E}[c_t]$. Optimizing ρ_β can be difficult because computing its gradient requires computing $q(\tilde{x}_t)$ at each time step. Therefore, we optimize $\mathcal{J}$, a first-order approximation of (6). In section Section IV, we demonstrate in multiple examples that minimizing our objective $\mathcal{J}$ produces a robust policy that generalizes beyond its training environment.

Finally, we remark that we can alternatively minimize $\mathbb{I}[\tilde{x}_t; y_t]$, instead of $\mathbb{I}[x_t; \tilde{x}_t]$. However, this is not consistent with Theorem II.1. Moreover, as described in Section III, the mutual information will be estimated empirically and in many cases (see Section IV), the dimension of the observation space is significantly larger than that of the state space. As a result, estimating $\mathbb{I}[\tilde{x}_t; y_t]$ is often more challenging.

III. Learning Task-Driven Policies

This section discusses finding a policy that approximately solves (3) within a reinforcement learning (RL) framework. If the mutual information term is removed from the objective $\mathcal{J}$, standard RL techniques such as policy gradient (PG) (e.g. [34, 35]) would be sufficient. However, the mutual information

and its gradient are known to be difficult quantities to estimate. A number of tractable upper and lower bounds have been proposed recently to provide means for optimizing objectives containing a mutual information term [31]. We elected to use the recently-proposed mutual information neural estimator (MINE) [4] due to its accuracy and ease of implementation. We then derive a PG-style algorithm in Section III-B.

A. Mutual Information Neural Estimator (MINE)

The MINE is based on the Donsker-Varadhan (DV) variational representation of the KL divergence [13, Theorem 2.3.2]. For any two distributions P, Q defined on sample space Ω, the KL divergence between them can be expressed as:

$$\mathbb{D}[P\|Q] = \sup_{F:\Omega\to\mathbb{R}} \mathcal{J}_{dv} \coloneqq \mathbb{E}_P F - \log \mathbb{E}_Q[\exp(F)]. \tag{7}$$

The supremum is taken over all functions F such that the two expectations are finite. Let $\hat{\mathbb{E}}_P^N$ denote the empirical expectation computed with N i.i.d. samples from P. We can estimate the KL divergence by replacing the function class over which the supremum is taken by a family of neural networks F^θ parameterized by $\theta \in \Theta$:

$$\hat{\mathbb{D}}^N[P\|Q] \coloneqq \sup_{\theta\in\Theta} \underbrace{\hat{\mathbb{E}}_P^N F^\theta - \log \hat{\mathbb{E}}_Q^N \left[\exp\left(F^\theta\right)\right]}_{\hat{\mathcal{J}}_{dv}}. \tag{8}$$

This approximation is a *strongly consistent* underestimate of (7) (see [4]). In short, this means that through choice of appropriate network structure and sample size, $\hat{\mathbb{D}}^N[P\|Q]$ can approximate $\mathbb{D}[P\|Q]$ arbitrarily closely.

Since the mutual information is a KL divergence, we can approximate (2) using the above KL approximation as:

$$\hat{\mathbb{I}}^N[x_t;\tilde{x}_t] \coloneqq \hat{\mathbb{D}}^N\left[p_t(x_t,\tilde{x}_t)\|p_t(x_t)q_t(\tilde{x}_t)\right]. \tag{9}$$

This is the MINE. The algorithm for computing the MINE estimate is presented in Algorithm 1. At a high-level, the algorithm attempts to find neural network parameters θ in order to maximize $\hat{\mathcal{J}}_{dv}$ in (8) via stochastic gradient descent. We use the notation $\mathbb{E}_\perp[\cdot]$ to denote the expectation taken using $p_t(x_t)q_t(\tilde{x}_t)$. The expectation computed using the joint distribution $p_t(x_t,\tilde{x}_t)$ is unsubscripted. These expectations are approximated by sampling two minibatches of size $B < N$ uniformly from the training batch. The minibatches are denoted in Algorithm 1 by indicies $j_1,\dots,j_B$ and $m_1,\dots,m_B$ respectively. The gradient of $\hat{\mathcal{J}}_{dv}$ is

$$\nabla_\theta \hat{\mathcal{J}}_{dv} = \hat{\mathbb{E}}^B\left[\nabla_\theta F^\theta\right] - \frac{\hat{\mathbb{E}}_\perp^B\left[\nabla_\theta F^\theta \exp\left(F^\theta\right)\right]}{\hat{\mathbb{E}}_\perp^B[\nabla_\theta \exp\left(F^\theta\right)]}. \tag{10}$$

This gradient, which is computed using the minibatches, is used to update θ with stochastic gradient descent (or a similar optimizer like ADAM [22]). For additional details on training the MINE, see [4].

B. Task-Driven Policy Gradient Algorithm

We now describe our procedure for leveraging MINE within a policy gradient (PG) algorithm for tackling (3). As described in Section II, the policy is parameterized by two neural networks: a recurrent network $q^\phi(\tilde{x}_t|\tilde{x}_{t-1},y_t)$ that outputs the TRVs and a feedforward network $\pi^\psi(u_t|\tilde{x}_t)$ that outputs the actual control action. Here $\phi\in\Phi, \psi\in\Psi$ represent the parameters for each network. In our implementation, both networks output the mean and diagonal covariance of multivariate Gaussian distributions. During training, we assume access to the states visited by the robot in order to compute the mutual information. Online execution, however, only requires access to the observations.

Algorithm 1 Mutual Information Neural Estimator (MINE)

procedure TRAIN-MINE($\theta, \{(x^n,\tilde{x}^n)\}_{n=1}^N$)
  repeat
    Sample joint minibatch: $\{(x^{j_b},\tilde{x}^{j_b})\}_{b=1}^B$.
    Sample marginal minibatch: $\{\tilde{x}^{m_b}\}_{b=1}^B$.
    Compute $\hat{\mathbb{E}}^B\left[\nabla_\theta F^\theta\right]$ with $\{(x^{j_b},\tilde{x}^{j_b})\}_{b=1}^B$.
    Compute $\frac{\hat{\mathbb{E}}_\perp^B\left[\nabla_\theta F^\theta\right]}{\hat{\mathbb{E}}_\perp^B[\nabla_\theta \exp(F^\theta)]}$ with $\{(x^{j_b},\tilde{x}^{m_b})\}_{b=1}^B$.
    Update θ using minibatches and (10).
  until Convergence of $\hat{\mathcal{J}}_{dv}$.
return θ
end procedure

Computing the gradient of our objective (3) is difficult due to the presence of the mutual information, so the MINE is used to approximate this term.[1] This approximation allows us to derive a learning algorithm similar to policy gradient. Using the well-known identity

$$\nabla_\psi \pi^\psi(u_t|\tilde{x}_t) = \pi^\psi(u_t|\tilde{x}_t)\nabla_\psi \log \pi^\psi(u_t|\tilde{x}_t), \tag{11}$$

the gradient of the expected cost with respect to ψ is given by

$$\nabla_\psi \mathbb{E}\left[c(\tau)\right] = \mathbb{E}\left[c(\tau)\sum_{t=0}^T \nabla_\psi \log \pi^\psi(u_t|\tilde{x}_t)\right]. \tag{12}$$

Repeating this process for ϕ yields an analogous formula with $q^\phi(\tilde{x}_t|\tilde{x}_{t-1},y_t)$ replacing $\pi^\psi(u_t|\tilde{x}_t)$:

$$\nabla_\phi \mathbb{E}\left[c(\tau)\right] = \mathbb{E}\left[c(\tau)\sum_{t=0}^T \nabla_\phi \log q^\phi(\tilde{x}_t|\tilde{x}_{t-1},y_t)\right]. \tag{13}$$

Finally, it remains to calculate $\nabla_{\phi,\psi}\hat{\mathbb{I}}^N[x_t;\tilde{x}_t]$. Though it is more tractable to optimize the MINE instead of the true mutual information, computing the gradient of the MINE with respect to the policy parameters is not entirely straightforward. The complexity lies in the fact that the gradient of the MINE with respect to these parameters depends on knowing or estimating the marginal distributions $p_t(x_t)$ and $q_t(\tilde{x}_t)$, both of which depend on ϕ and ψ. To approximate the MINE gradient we fix θ to the converged MINE parameters from Algorithm 1, which yields:

$$\hat{\mathbb{I}}^N[x_t;\tilde{x}_t] = \hat{\mathbb{E}}^N[F^\theta] - \log \hat{\mathbb{E}}_\perp^N[\exp(F^\theta)]. \tag{14}$$

[1] We note that, in order to employ the bound (6), we need to employ an overestimate of $\mathbb{I}[x_t;\tilde{x}_t]$. In Section IV, we demonstrate that in practice, minimizing the MINE is a practical approximation that produces robust policies.

Since the neural networks used to parameterize the policy produce Gaussian distributions, we can represent $\tilde{x}_t$ as

$$\tilde{x}_t = \mu^\phi(\tilde{x}_{t-1}, y_t) + A^\phi(\tilde{x}_{t-1}, y_t)\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I). \quad (15)$$

This is similar to the reparameterization trick for variational autoencoders [23]. Here, μ^ϕ, A^ϕ are computed using the mean and covariance output from the network $q^\phi(\tilde{x}_t|\tilde{x}_{t-1}, y_t)$. Since the dynamics and sensor model are unknown, the approximations that x_t and y_t are independent of ϕ and $x_t, y_t, \tilde{x}_t$ are independent of ψ are made. With these fixed, the gradient of (14) with respect to ϕ can be computed by storing $\epsilon_0, \ldots, \epsilon_{t-1}$ as a part of each rollout and backpropogating using (15).

The task-driven policy gradient (TDPG) algorithm is outlined in Algorithm 2. Our learning process involves training $T-1$ MINE networks, whose parameters are denoted $\theta_0, \ldots, \theta_{T-1}$. For clarity in future sections, we refer to an iteration of the outer optimization loop as a policy epoch and an iteration of the optimization loop in Algorithm 1 as a MINE epoch. During each policy epoch, we rollout N trajectories using the current policy parameters ϕ, ψ. We then update each set of MINE parameters using Algorithm 1. Once the MINE networks converge, they are used to approximate $\mathcal{J}$ and optimize the policy. This is repeated until the empirical estimate of $\mathcal{J}$, which is given by $\hat{\mathcal{J}} \coloneqq \beta\hat{\mathbb{E}}^N[c(\tau)] + \sum_{t=0}^{T}\hat{\mathbb{I}}^N[x_t; \tilde{x}_t]$ converges.

Implementation Details. It remains to specify how to select β in a principled manner. Returning to the perspective described in Section II, we treat $\mathcal{J}$ as the Lagrangian for minimizing the information shared between x_t and $\tilde{x}_t$ subject to an upper bound on the maximum expected cost the policy is allowed. Then, we sweep through a set of values for β and select the policy from the epoch with the lowest MINE estimate that also satisfies the specified limit on the empirical expected cost. This strategy produces the policy estimated to have the least state information present in the TRVs while satisfying our performance constraint.

It is likely that each MINE network is initialized to a poor estimate of the mutual information. In order to improve the initial estimate of the mutual information and its gradient, additional MINE epochs are used during the first policy epoch. In the following section, we will specify the number of additional epochs used. Moreover, as discussed in [4], using minibatches to estimate the MINE gradient in (10) leads to a biased estimate of the gradient. Replacing the denominator in (10) with the exponential moving average (EMA) of its value compensates for this bias.[2] This technique is used in some examples in the following section. We also limit the policy networks to output Gaussian distributions with diagonal covariances.

Finally, since we are solving a finite-horizon control problem, the mappings $q^\phi(\tilde{x}_t|\tilde{x}_{t-1}, y_t), \pi^\psi(u_t|\tilde{x}_t)$ may optionally chosen to be time varying, i.e. use separate values of ϕ, ψ for each time step. With a rich enough network structure, this should have no impact on the policy since the network can always encode t in $\tilde{x}_t$. In practice, we found that this increased flexibility can help achieve a lower training cost in some cases at the cost of more computation per epoch.

[2]The EMA is a filter defined on a sequence $f_0, \ldots, f_t$ by the recursive relationship $\hat{f}_t = (1-\alpha)a_t + \alpha\hat{f}_{t-1}, \hat{f}_0 = f_0$.

Algorithm 2 Task-Driven Policy Gradient (TDPG)

repeat
  Rollout batch of N trajectories: $\{(x_t^n, \tilde{x}_t^n, u_t^n)_{t=0}^T\}_{n=0}^N$.
  for $t = 0, \ldots, T-1$ **do**
    $\theta_t \leftarrow$ TRAIN-MINE$(\theta_t, \{x_t^n, \tilde{x}_t^n)\}_{n=1}^N)$
  end for
  Update ϕ, ψ using rollout batch and (12), (13), (15).
until Convergence of $\hat{\mathcal{J}}$.

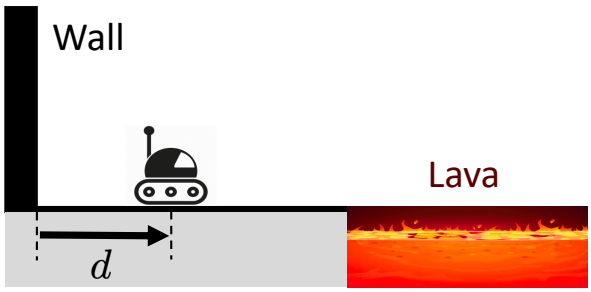


Fig. 2. An illustration of the Lava Problem described in Section IV-A. The robot (a double integrator) needs to navigate to a target state without moving so far right that the robot falls into the lava.

IV. Examples

In this section, we apply the algorithm described in Section III to three problems: (i) a continuous state and action version of the "Lava Problem" from the POMDP literature, (ii) a vision-based ball-catching example, and (iii) a grasping problem with depth-image observations. For each of these problems, we present thorough simulation results demonstrating that the task-driven policy is robust to distributional shifts in the sensor model and testing environment. We compare our method against a policy with the same parameterization as ours, but trained using a standard policy gradient method in order to minimize the expected cost associated with the problem. The source code for these examples is available here: `github.com/irom-lab/trc-nn`

A. Lava Problem

The first problem we consider is a continuous state and action version of the Lava problem (Figure 2) [5, 10, 19, 30], which is a common example for evaluating robust solutions to POMDPs. This scenario involves a robot navigating to a goal location along a line segment between a wall and a lava pit. The robot is modeled as a time-discretized double integrator, i.e. its state $x_t = [d_t \ v_t]^\mathrm{T}$ evolves with dynamics $x_{t+1} = [d_t + v_t \ v_t + u_t]^\mathrm{T}$. Here d_t is the displacement from the wall (in meters) and v_t is the robot's velocity (in meters per second). The goal is to navigate to the state $g = [3 \ 0]^\mathrm{T}$ within a time horizon of $T = 5$ steps. However, d_t is limited to the interval $[0, 5]$m. If the robot collides with the wall located at $d = 0$, then its velocity is set to 0m/s as well. If the robot's position exceeds $d_t = 5$m, then the robot falls into hot lava, where it is unable to move any further. At training time, the robot is provided with a high-quality estimate of its state. This is modeled by the choice $y_t \sim \mathcal{N}(x_t, \sigma^2 I)$, where $\sigma^2 = 0.0001$

Scenario	Policy Gradient		Task-Driven PG	
	Mean	Std.	Mean	Std.
Training	**31.04**	**7.191**	36.18	23.22
Sensor Noise σ^2 =1E-3	**31.58**	**8.632**	36.55	25.02
Sensor Noise σ^2 =1E-2	**35.69**	**15.68**	36.00	22.32
Sensor Noise σ^2 =1E-1	66.29	36.48	**36.20**	**21.85**
Sensor Noise σ^2 =1E0	172.40	88.08	**37.39**	**25.60**

TABLE I
PERFORMANCE OF PG AND TDPG POLICIES IN TRAINING AND TESTING LAVA ENVIRONMENTS

and $\mathcal{N}$ denotes the Gaussian distribution. The cost function for this problem is $c_t(x_t, u_t) = \|x_t - g\|_2$ for $t = 0, \ldots, T-1$ and $c_T = 100\,\|x_T - g\|_2$. The robot is initialized with $v_t = 0$ and d_t uniformly distributed between 0 and 5 meters.

Training Summary. Both $q^\phi(\tilde{x}_t|\tilde{x}_{t-1}, y_t)$ and $\pi^\psi(u_t|\tilde{x}_t)$ have two hidden layers with 64 units and use exponential linear unit (ELU) nonlinearities [6]. We choose a two-dimensional space $\tilde{\mathcal{X}}_t$ of TRVs. In this example, we found the performance of all learned policies improved when the parameters ϕ and ψ were allowed to be time-varying. The MINE networks used in this example contain two hidden layers with 32 units each and ELU nonlinearities. A batch of 500 rollouts is used for each epoch, and a minibatch size of 50 is used for each MINE epoch, and an EMA with α =5E-5 was used to compute the MINE gradient. The learning rates in this problem are 8E-4 for the policy networks and 5E-5 for the MINE network. All policies were trained for 300 epochs, with a computation time of about 10 seconds per epoch (about 50 minutes total for each policy). For this example, all computation was carried out on an Intel i9-7940X. Policies were trained with $\beta \in \{25^{-1}, 50^{-1}, 75^{-1}, 100^{-1}\}$ and the upper limit on the expected cost for selecting the policy was 40.

Policy Evaluation. We compare the resulting policy with one trained to minimize only the expected cost using policy gradient. The two policies are compared in Figure 3, and the statistics of their performance in different environments (i.e. different levels of sensor noise) are presented in Table I. Due to the presence of low sensor noise in the training environment, the PG policy finds it optimal to drive the robot directly towards the goal state. Interestingly, the TDPG algorithm finds a *qualitatively different* strategy. In particular, TDPG recovers the robust open-loop behavior described in [5, 10, 30]: regardless of initial position, the robot moves left until it collides with the wall, then moves right to the goal state. As the sensor noise σ^2 is increased, the performance of the PG policy degrades rapidly. For example, if the sensor reports the robot is to the left of the goal when it is really to the right of the goal, it can fall in the lava. In contrast, the TDPG is unaffected by the increased sensor noise.

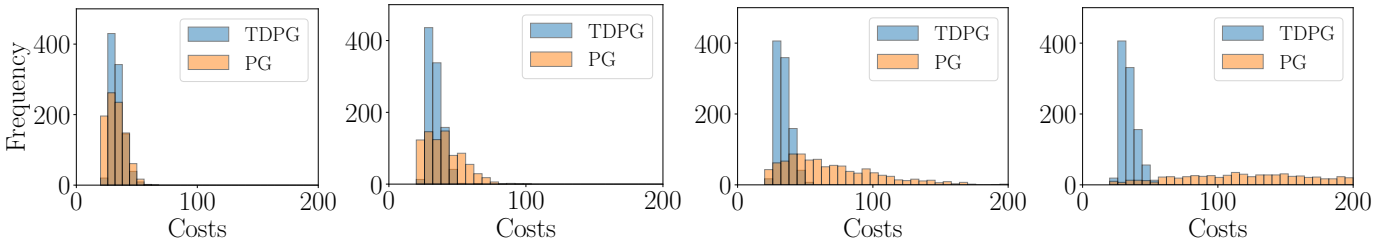
B. Vision-Based Ball Catching

Next, we consider a ball-catching example inspired by the gaze heuristic discussed in Section I (see Figure 1). We formalize this problem by considering a ball confined to a plane with x and y coordinates (b_t^x, b_t^y). The robot is confined to the x-axis and must navigate to b_t^x. The state of the system in this example is given by $x_t = [d_t\ b_t^x\ b_t^y\ v_t^x\ v_t^y]^{\mathrm{T}}$, where d_t represents the robot's displacement along the x-axis, v_t^x represents the ball's velocity along the x-axis, and v_t^y represents the ball's velocity along the y-axis (i.e., the ball's vertical velocity). The dynamics are given by:

$$\begin{bmatrix} d_{t+1} \\ b_{t+1}^x \end{bmatrix} = \begin{bmatrix} d_t + \Delta t \cdot u_t \\ b_t^x + \Delta t \cdot v_t^x \end{bmatrix}, \quad \begin{bmatrix} b_{t+1}^y \\ v_{t+1}^y \end{bmatrix} = \begin{bmatrix} b_t^y + \Delta t \cdot v_t^y \\ v_t^y - \Delta t \cdot g \end{bmatrix},$$

and $v_{t+1}^x = v_t^x$. Here, $g = 9.81\text{m/s}^2$ is gravity, and $\Delta t = \frac{1}{15}\text{s}$ is used to discretize the dynamics of the system in time. The robot's initial position is uniformly distributed over the interval $[-2, 2]$m. The ball is launched from $b_0^x = 8$m and $b_0^y = 1$m with fixed initial velocities $v_0^x = -4.5$m/s and $v_0^y = 7.85$m/s. These initial conditions are chosen such that the ball always spends a fixed number of time steps above the x-axis. The time horizon $T = 25$ is chosen such that T is the last time step the ball remains above the x-axis. The cost function that we are trying to minimize is $c_t(x_t, u_t) = 0.01\,\|u_t\|_2$ for $t = 0, \ldots, T-1$ and $c_T(x_T) = 100\,\|d_T - b_T^x\|_2$, which encourages the robot to be very close to the ball when it lands at the end of the time horizon. The sensor in this scenario is a camera mounted above the robot. This camera provides 64×64 RGB images with values scaled between 0 and 1. We also placed a wall with a red brick texture centered at 10m along the x-axis. All simulations are carried out using PyBullet [7]. A sample observation from the camera is presented in Figure 1, and a video depicting both policies operating in training and testing environments for this scenario is provided in the supplementary material.

Training Summary. In this example, $q^\phi(\tilde{x}_t|\tilde{x}_{t-1}, y_t)$ is parameterized by a network with 2 convolutional layers with 6 output channels, kernel size of 4, and stride of length 2, followed by two fully-connected layers using 32 units each.



(a) $\sigma^2 = 0.001$ (b) $\sigma^2 = 0.01$ (c) $\sigma^2 = 0.1$ (d) $\sigma^2 = 1$

Fig. 3. These histograms compare our policy (TDPG) to the policy found by policy gradient (PG) in the Lava Problem described in Section IV-A by showing the frequencies at which each policy incurs different costs. The sensor noise of the test environment is increased from left to right. As the noise increases, the PG policy performance degrades, while the TDPG policy performance remains almost constant. This is because the TDPG algorithm found a task-relevant open-loop policy that is robust to this kind of disturbance.

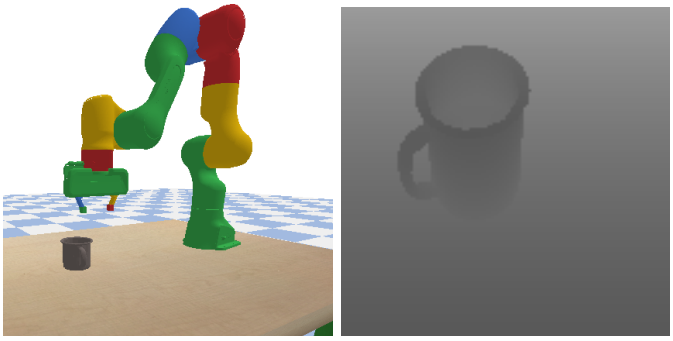
An ELU nonlinearity is applied between convolutional layers. After each fully-connected layer, $\tanh$ nonlinearities were used instead of ELU nonlinearities to prevent the values of $\tilde{x}_t$ from growing unbounded. The dimension of $\tilde{\mathcal{X}}_t$ is 8. The network $\pi^{\psi}(u_t|\tilde{x}_t)$ contains a single linear layer. The MINE networks used in this example contain two hidden layers with 64 units each and ELU nonlinearities. A batch of 200 rollouts is used for each epoch, and a minibatch size of 20 is used to train the MINE network. The learning rates in this problem are 1E-3 for the policy networks and 5E-5 for the MINE network. When training the TDPG policy, it was initialized with the PG solution and trained for 100 policy epochs. Each policy epoch contained 100 MINE epochs with 100,000 MINE epochs used on the first policy epoch, and an EMA with $\alpha = 5\text{E-}5$ was used to compute the MINE gradient. Policies with $\beta = 16^{-1}, 18^{-1}, \ldots, 40^{-1}$ were evaluated, with the upper limit on the expected cost placed at 24. Rollouts were computed on an 3.7GHz i7-8700K CPU while all optimization was done using an Nvidia Titan Xp. Each policy epoch took about 45 seconds to compute.

Policy Evaluation. We consider two kinds of testing environments. All statistics for the test environments were computed using 1000 rollouts. In the first group of examples, we add random noise to each pixel; the noise is sampled from $\mathcal{N}(0, \sigma^2)$, where σ is varied between experiments. To ensure that the observed image is a valid RGB image after adding noise, we normalize the values to lie between 0 and 1. For each experiment, the mean cost and distance between the robot and the ball on the x-axis is presented in Table II. As the level of image noise increases, the performance of the PG policy deteriorates dramatically while the TDPG policy's performance remains largely unchanged.

In the second set of experiments, we change the texture of the backdrop (i.e. the wall in the background) for testing. This change is not task-relevant to catching the ball as long as the ball can be identified within the frame. Each texture is presented in Figure 4. The TDPG policy outperforms the PG policy in the seven testing environments except background 7. This texture has a similar hue (red) to the training environment texture allowing PG to perform well. The TDPG policy, however, outperforms PG in environments whose textures consist of hues that are different than the ones seen when training. Together with the previous experiments, this result suggests that the TDPG policy is more robust to observations that are qualitatively different in task-irrelevant ways than those provided to the policy during training. A video depicting the second set of experiments is available here: `www.youtube.com/watch?v=Mwv0kkRveas`

Scenario	Policy Gradient		Task-Driven PG	
	Cost	Dist. (m)	Cost	Dist. (m)
Training	**8.702**	**0.085**	18.577	0.184
Sensor Noise $\sigma = 0.10$	**19.12**	**0.189**	19.48	0.193
Sensor Noise $\sigma = 0.15$	46.68	0.464	**20.12**	**0.199**
Sensor Noise $\sigma = 0.20$	124.63	1.244	**21.59**	**0.214**
Sensor Noise $\sigma = 0.25$	187.53	1.874	**27.11**	**0.274**
Test Background 1	182.1	1.828	**122.7**	**1.225**
Test Background 2	214.3	2.141	**79.90**	**0.797**
Test Background 3	138.7	1.385	**26.62**	**0.264**
Test Background 4	95.38	0.952	**28.11**	**0.279**
Test Background 5	82.01	0.818	**36.95**	**0.367**
Test Background 6	208.5	2.083	**166.4**	**0.166**
Test Background 7	**9.180**	**0.090**	14.92	0.147

TABLE II
MEAN COST AND FINAL DISTANCE OF PG AND TDPG POLICIES IN TRAINING AND TESTING BALL-CATCHING ENVIRONMENTS



(a) Franka Emika Panda (b) Sensor Observation

Fig. 5. (a) A third-person perspective of the grasping scenario described in Section IV-C using PyBullet. (b) An upscaled example of the 128×128 depth image forming the sensor observation for this scenario.

C. Grasping using a Depth Camera

In this final example, we consider the task of grasping and lifting an object (a mug) from a table using a Franka Emika Panda simulated with PyBullet (see Figure 5). This example is particularly interesting for studying task-driven policies due to the (approximate) radial symmetry of the mug about the vertical axis. Intuitively, this symmetry renders the orientation of the mug about the vertical axis largely irrelevant to the task of grasping the mug. This is due to the fact that the robot does not need to know the precise orientation of the mug; it only needs to know enough about the orientation in order to estimate if the handle will directly interfere with the gripper's location when the grasp is performed. A task-driven policy will remain largely unaffected by changes in the mug's orientation and should therefore generalize to the full set of rotations.

In this grasping problem, the state of the system $x_t \in \mathbb{R}^4$ contains the position of the object and its rotation about the z-axis (which is oriented normal to the table surface) in radians. The control action $u_t \in \mathbb{R}^4$ specifies the position and orientation of the end effector of the arm when the grasp occurs. After

(a) Training (b) Test 1 (c) Test 2 (d) Test 3 (e) Test 4 (f) Test 5 (g) Test 6 (h) Test 7

Fig. 4. Textures used as backdrops in different ball-catching environments as part of the example described in Section IV-B.

grasping, the arm attempts to move the end effector vertically. A cost of 0 is awarded if the arm successfully lifts the object more than 0.05 meters above the table and a cost of 1 is assigned otherwise. The observation is a 128×128 depth image. An example observation is included in Figure 5. The initial state of the object is sampled uniformly from the set $[0.45 \text{ m}, 0.55 \text{ m}] \times [-0.05 \text{ m}, 0.05 \text{ m}] \times \{0 \text{ m}\} \times [-\frac{\pi}{4}, \frac{\pi}{4}]$.

Training Summary. We parameterized the policy in the following manner. The network $q^{\phi}(\tilde{x}_t|\tilde{x}_{t-1}, y_t)$ contains two convolutional layers with 6 output channels. The first uses a kernel size of 6 and the second uses a kernel size of 4; both use a stride length of 2. The convolutional layers are followed by two fully-connected layers of sizes 128 and 64. An ELU linearity is used between convolutional layers and $\tanh$ nonlinearities are used after linear layers. The size of $\tilde{\mathcal{X}}_t$ is 16. The network $\pi(u_t|\tilde{x}_t)$ contains two fully connected layers with 64 units each and an ELU nonlinearity between them. The output of this network is then divided by 10 and added to a nominal control action of $[0.5, 0, 0.06, \frac{\pi}{6}]$. This scaling and translation is done to bias the policy to select grasping locations near the object to speed up learning early in the optimization process. Again, as in Section IV-B, we initialize the TDPG solution to the PG solution at the start of training and learned policies with values of $\beta \in \{5^{-1}, 10^{-1}, 15^{-1}, 20^{-1}\}$ for 30 epochs.

In this example, the MINE proved particularly noisy and took longer to converge. To combat this, we increased the number of MINE epochs to 500,000 on the first policy epoch with 100,000 MINE epochs used on following policy epochs. To determine the value of the MINE, we applied an EMA with parameter $\alpha = 0.9$. No EMA was used for computing MINE gradients. The epoch with the lowest filtered MINE estimate with an expected cost below 0.15 was used for testing. Each epoch took approximately 5 minutes to compute. Rollouts were computed on an Intel 3.7GHz i7-8700K in parallel and optimization was done on a Titan Xp.

Policy Evaluation. The test results for this example are summarized in Table III. Again, all reported statistics are computed using 1000 trials. In the first testing environment, the set of angles the mug is placed at is expanded from $[-\frac{\pi}{4}, \frac{\pi}{4}]$ to $[0, 2\pi]$. The expected cost (i.e., grasp failure rate) of the PG policy increased twice as much as the TDPG policy in these new testing environments. In the second testing environment, the set of angles the object is placed at is again $[-\frac{\pi}{4}, \frac{\pi}{4}]$, but the x and y values were sampled from the larger set of $[0.425 \text{ m}, 0.575 \text{ m}] \times [-0.075 \text{ m}, 0.075 \text{ m}]$. In this setting, both policies perform equally poorly. This result supports our hypothesis that the rotation of the mug is largely unimportant for the task of lifting the mug because the TDPG policy was able to generalize to new mug angles, but not to the task-relevant translational coordinates of the mug. Meanwhile, the PG policy exhibits overfitting to task-irrelevant state information and is impacted poorly by its changes.

Scenario	Policy Gradient		Task-Driven PG	
	Mean	Std.	Mean	Std.
Training	0.107	0.309	**0.094**	**0.292**
Increased Rotation	0.182	0.386	**0.132**	**0.339**
Increased Translation	0.367	0.482	**0.358**	**0.480**

TABLE III
PERFORMANCE OF PG AND TDPG POLICIES IN TRAINING AND TESTING GRASPING ENVIRONMENTS

V. DISCUSSION AND CONCLUSION

We presented a novel reinforcement learning algorithm for computing task-driven control policies for systems equipped with rich sensor observations (e.g., RGB or depth images). The key idea behind our approach is to learn a task-relevant representation that contains as little information as possible about the state of the system while being sufficient for achieving a low cost on the task at hand. Formally, this is achieved by using an information bottleneck criterion that minimizes the mutual information between the state of the system and a set of task-relevant variables (TRVs) used for computing control actions. We parameterize our policies using neural networks and present a novel policy gradient algorithm that leverages the recently-proposed mutual information neural estimator (MINE) for optimizing our objective. We refer to the resulting algorithm as task-driven policy gradient (TDPG).

We compare PG and TDPG policies in three experiments: an adaption of the canonical lava problem to continuous state spaces, a ball catching scenario inspired by the gaze heuristic from cognitive psychology, and a depth image-based grasping problem. In the lava example, the TDPG policy exploits the nonlinear dynamics to find a minimal information (open-loop) control policy that is robust to increased sensor noise. In the ball-catching example, the TDPG is also more robust to changes in the sensing model at test time than PG. These changes include both random noise corrupting the images and task-irrelevant structural changes, e.g. altering the textures in the robot's environment. Finally, in the grasping scenario, the TDPG policy generalizes to rotated states of the object not seen during training on which the PG policy struggles. Together, these scenarios validate that our approach to designing task-driven control policies produces robust policies that can operate in environments unseen during training.

Future Work. There are a number of challenges and exciting directions for further exploration. First, we have observed that MINE often results in noisy estimates of the mutual information and can take many epochs to converge. This results in long training times for TDPG. We also employed an approximation of the gradient of MINE with respect to policy parameters (due to the challenges associated with estimating state distributions at each time step, as described in Section III). These observations motivate the exploration of other methods for minimizing the mutual information, eg., Stein variational gradient methods [17, 28]. Another direction for future work is to adapt more advanced on-policy methods (e.g., proximal policy optimization (PPO) [35]) to work with our approach, and potentially explore off-policy methods. We expect that these methods will be even more effective at learning task-driven policies. Finally, an exciting direction for future work is to explore the benefits that our approach affords in terms of sim-to-real transfer. The simulation results in this paper suggest that TDPG can be robust to task-irrelevant features system perturbations. Exploring whether this translates to more robust sim-to-real transfer is a promising future direction.

References

[1] A. Achille and S. Soatto. A separation principle for control in the age of deep learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:287–307, 2018.

[2] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[3] B. Anderson and J. Moore. *Optimal Control: Linear Quadratic Methods*. Courier Corporation, 2007.

[4] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm. Mutual information neural estimation. In *Proceedings of the International Conference on Machine Learning*, pages 531–540, 2018.

[5] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

[6] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

[7] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2018.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[9] G. E. Dullerud and F. Paganini. *A Course in Robust Control Theory: A Convex Approach*, volume 36. Springer Science & Business Media, 2013.

[10] P. R. Florence. Integrated perception and control at high speed. Master's thesis, Massachusetts Institute of Technology, 2017.

[11] G. Gigerenzer. *Gut Feelings: The Intelligence of the Unconscious*. Penguin, 2007.

[12] A. Goyal, R. Islam, D. Strouse, Z. Ahmed, H. Larochelle, M. Botvinick, S. Levine, and Y. Bengio. Transfer and exploration via the information bottleneck. In *Proceedings of the International Conference on Learning Representations*, 2019.

[13] R. M. Gray. *Entropy and Information Theory*. Springer Science & Business Media, 2nd edition, 2011.

[14] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017.

[15] S. Gupta, D. Fouhey, S. Levine, and J. Malik. Unifying map and landmark based representations for visual navigation. *arXiv preprint arXiv:1712.08125*, 2017.

[16] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop KF: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems*, pages 4376–4384, 2016.

[17] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1352–1361, 2017.

[18] R. Jonschkowski, D. Rastogi, and O. Brock. Differentiable particle filters: end-to-end learning with algorithmic priors. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[20] P. Karkus, D. Hsu, and W. S. Lee. Particle filter networks: end-to-end probabilistic localization from visual observations. *arXiv preprint arXiv:1805.08975*, 2018.

[21] P. Karkus, X. Ma, D. Hsu, L. P. Kaelbling, W. S. Lee, and T. Lozano-Pérez. Differentiable algorithm networks for composable robot learning. *arXiv preprint arXiv:1905.11602*, 2019.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

[25] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert. Nonlinear information bottleneck. *Entropy*, 21(12):1181, 2019.

[26] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[27] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)*, 37(4-5):421–436, 2018.

[28] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

[29] P. McLeod, N. Reed, and Z. Dienes. Psychophysics: How fielders arrive in time to catch the ball. *Nature*, 426(6964):244, 2003.

[30] V. Pacelli and A. Majumdar. Task-driven estimation and control via information bottlenecks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[31] B. Poole, S. Ozair, A. v. d. Oord, A. A. Alemi, and G. Tucker. On variational bounds of mutual information. *arXiv preprint arXiv:1905.06922*, 2019.

[32] C. Salge, C. Glackin, and D. Polani. Empowerment and state-dependent noise-an intrinsic motivation for avoiding unpredictable agents. In *Proceedings of the Artificial Life Conference*, pages 118–125. MIT Press, 2013.

[33] C. Salge, C. Glackin, and D. Polani. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.

[34] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1889–1897, 2015.

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

[36] D. M. Shaffer, S. M. Krauchunas, M. Eddy, and M. K. McBeath. How dogs navigate to catch frisbees. *Psychological Science*, 15(7):437–441, 2004.

[37] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2009.

[38] S. Soatto. Steps towards a theory of visual information: Active perception, signal-to-symbol conversion and the interplay between sensing and control. *arXiv preprint arXiv:1110.2053*, 2011.

[39] S. Soatto. Actionable information in vision. In *Machine Learning for Computer Vision*, pages 17–48. Springer, 2013.

[40] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research (IJRR)*, 37(4-5):405–420, 2018.

[41] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

[42] S. Tiomkin, D. Polani, and N. Tishby. Control capacity of partially observable dynamic systems in continuous time. *arXiv preprint arXiv:1701.04984*, 2017.

[43] N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *Proceedings of the IEEE Information Theory Workshop*, pages 1–5. IEEE, 2015.

[44] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37th Allerton Conference on Communication, Control, and Computing*, 1999.

[45] P. Yingjun and H. Xinwen. Learning representations in reinforcement learning: An information bottleneck approach. *arXiv preprint arXiv:1911.05695*, 2019.

[46] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn. Unsupervised visuomotor control through distributional planning networks. *arXiv preprint arXiv:1902.05542*, 2019.

[47] R. Zhao, S. Tiomkin, and P. Abbeel. Learning efficient representation for intrinsic motivation. *arXiv preprint arXiv:1912.02624*, 2019.

[48] K. Zhou and J. C. Doyle. *Essentials of Robust Control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.

[49] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364. IEEE, 2017.