Learning State Representations with Robotic Priors

Rico Jonschkowski · Oliver Brock

Abstract Robot learning is critically enabled by the availability of appropriate state representations. We propose a robotics-specific approach to learning such state representations. As robots accomplish tasks by interacting with the physical world, we can facilitate representation learning by considering the structure imposed by physics; this structure is reflected in the changes that occur in the world and in the way a robot can effect them. By exploiting this structure in learning, robots can obtain state representations consistent with the aspects of physics relevant to the learning task. We name this prior knowledge about the structure of interactions with the physical world *robotic priors*. We identify five robotic priors and explain how they can be used to learn pertinent state representations. We demonstrate the effectiveness of this approach in simulated and real robotic experiments with distracting moving objects. We show that our method extracts task-relevant state representations from high-dimensional observations, even in the presence of task-irrelevant distractions. We also show that the state representations learned by our method greatly improve generalization in reinforcement learning.

Keywords robot learning · reinforcement learning · representation learning · prior knowledge

1 Introduction

Roboticists and AI researchers are striving to create versatile robots, capable of autonomously solving a wide range of tasks. As different aspects of the environment might be important for every one of these tasks, robots must have versatile, task-general sensors, leading to high-dimensional sensory input. High-dimensional observations, however, present

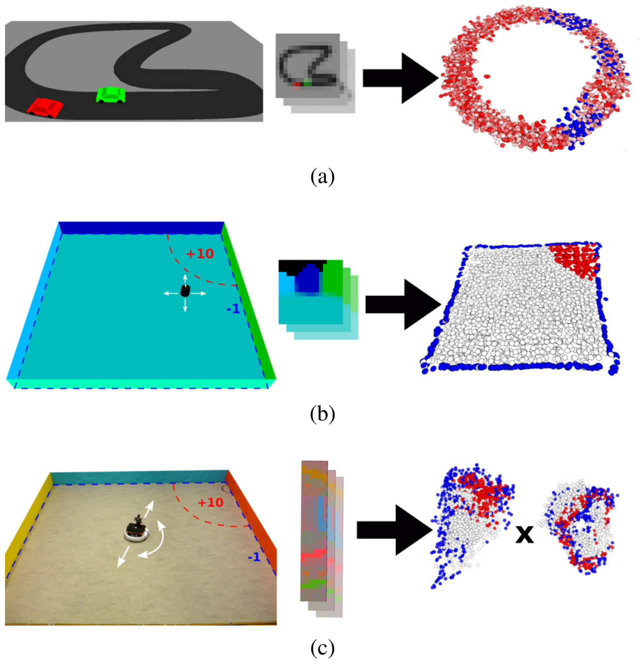


Fig. 1 State representations learned from visual input in (a) a simulated slot car racing task, (b) a simulated (and simplified) navigation task, and (c) a navigation task with a real robot. For (a) and (b) the state representations clearly capture the pertinent information for each task, i.e., the position of the slot car on the circular race track and the location of the mobile robot, respectively. For (c) the learned state representation is five-dimensional and, therefore, difficult to inspect visually. The plot shows two projections of the state space onto the first principal components.

a challenge for perception and learning. This seems unnecessary, as every single task can be mastered by only considering those aspects of the high-dimensional input that are pertinent to it. To build task-general robots, it is, therefore, necessary to extract from the high-dimensional sensor data only those features pertinent to solving the task at hand.

Feature engineering is probably the most common approach to this challenge in robotics. The mapping from ob-

Rico Jonschkowski (✉) · Oliver Brock
Robotics and Biology Laboratory, Technische Universität Berlin
E-mail: rico.jonschkowski@tu-berlin.de

servations to states is designed by hand, using human intuition. Feature engineering has enabled robots to learn and solve complex tasks. But the downside of this approach is that we have to define an observation-state-mapping for every task that we want the robot to learn.

Representation learning methods use machine learning instead of human intuition to extract pertinent information from high-dimensional observations. This approach does not require specific knowledge about the task. Instead, it uses general assumptions about the problem structure. However, the price for its generality is the huge amount of data and computation required to extract useful state representations. In robotics, data acquisition is costly and slow. Thus, existing representation learning approaches are difficult to apply.

Fortunately, robots do not have to solve the general representation learning problem. Robots can exploit substantial knowledge about their environment. Every robotic task consists of interactions with the real world, which is governed by the laws of physics. As the robot's internal state captures properties of the world, the same laws that apply to the physical world must also apply to this internal state representation. Robots can learn representations that are consistent with physics by exploiting prior knowledge about interacting with the physical world, which we term *robotic priors*. We believe that formulating and incorporating robotic priors are the key to effective representation learning in robotics.

In this paper, we formulate five robotic priors (see Section 4.1). We describe how to incorporate them into state representation learning by defining a loss function over state representations, which measures their inconsistency with the robotic priors (see Section 4.2). And we explain how, by minimizing this loss function, our method finds state representations that allow robots to effectively learn new tasks using standard reinforcement learning (see Section 4.3). We analyze our approach in depth in simulated robotic tasks based on visual observations: a slot car racing task with two cars and a navigation task (see Figure 1). We verify these results in an experiment with a real robot in a similar navigation task (see Section 5). These tasks contain distractors that influence the visual observations of the robot but that are irrelevant for the task. With our method, the robot learns to extract a low-dimensional state representation from high-dimensional visual observations that ignores these task-irrelevant distractions. We show that this change of representation greatly simplifies the subsequent learning problem and thereby substantially improves reinforcement learning performance. We also demonstrate that state representations that were learned for one task can also be transferred to new tasks.

This paper is an extended and revised version of a conference publication [14]. It presents a refined version of our method, adding regularization, including a different state similarity metric, and using non-uniform weighting of the loss-terms. We also changed the reinforcement learning method for evaluating the state representation. Additionally, we reimplemented our approach. The new implementation allows a more efficient use of training data by sampling more experience pairs to estimate the pair-wise loss terms and can handle a higher number of input dimensions. Using this new implementation of the updated method, we have replicated our previous experiments. As a result of these changes, the performance in the reinforcement learning experiment increased dramatically. Additionally, we have extended the experimental section to present additional insights into the learning process (see Section 5.1) to show the potential of our approach for transfer learning (see Section 5.6), and to verify our results on a real robot (see Section 5.7).

2 Related Work

2.1 Task-Specific, Generic, and Robotic Priors

In Bayesian statistics, the word "prior" refers to the prior probability distribution that is multiplied by the data likelihood and then normalized to compute the posterior. In this way, priors represent knowledge, before taking into account the data, as probability distributions. Following others in the field of representation learning [1], we use the word *prior* more broadly in an analogous fashion. In the context of this paper, a prior represents knowledge about a class of learning problems that is available before taking into account data from a specific problem instance. We do not restrict ourselves to represent these priors in the form of probability distributions[1]. We will now look at different domains, for which priors can be defined.

Many robotic tasks have been successfully solved using reinforcement learning, from ball-in-a-cup to inverted helicopter flight [16]. However, these approaches typically require human engineering, relying on what we call *task-specific priors*, priors that apply only to a specific task. One way of introducing task-specific priors is feature engineering: defining a mapping from observations to task-relevant states by hand.

Work in the area of representation learning strives to remove the need for feature engineering by automatically extracting pertinent features from data. The power of this approach has been empirically demonstrated in tasks such as speech recognition [29], object recognition [18], and natural language processing [5]. All of these examples substantially improve on the best previous methods based on engineered representations. To achieve these results, the representation

[1] Note, however, that we construct a loss function based on the priors that are proposed in this paper. This loss could be interpreted as negative logarithm of a prior probability distribution such that minimizing this loss function is analogous to maximizing the posterior.

learning methods use generic priors, big data, and massive computation.

According to Bengio et al. [1], the key to successful representation learning is the incorporation of “many general priors about the world around us.” They proposed a list of *generic priors* for artificial intelligence and argue that refining this list and incorporating it into a method for representation learning will bring us closer to artificial intelligence. This is exactly what we would like to do in the context of robotics. However, we believe that the problem of artificial intelligence is too broad and that therefore generic priors are too weak. We attempt to find stronger priors, incorporating more of the structure inherent to the problem, by focusing specifically on robotic tasks. Hence, we call such priors *robotic* priors. These priors only pertain to tasks requiring physical interactions with the world.

Scholz et al. [28] follow the same idea of using physics based priors to learn a forward model (or transition function) of the world. Instead of using a generic hypothesis space for the forward model, they use a restricted parametric hypothesis space based on physics. This can be very helpful for robotic tasks because we know that the right forward model must lie in this restricted hypothesis space. However, this work assumes to already have a suitable state representation consisting of poses and velocities as well as knowledge about the exact semantics of every state-dimension. Our work focuses on how a robot could learn to extract such a state representation from sensory input.

2.2 State Representation Learning

State representation learning is an instance of representation learning for interactive problems. The goal of state representation learning is to find a mapping from observations—or, more generally, from histories of interactions—to states that allow choosing the right actions. State representation learning is related to the problem of abstraction selection in hierarchical reinforcement learning, which deals with selecting a state representation from a given set [35, 17] or choosing a subset of state dimensions from a given representation [4]. In contrast to this work, we want to emphasize the need to construct state representations from sensory input without relying on predefined task-specific state features.

Note that state representation learning is more difficult than the standard dimensionality reduction problem, which is addressed by multi-dimensional scaling [19] and other methods [27, 34, 7] because they require knowledge of distances or neighborhood relationships between data samples in state space. The robot, on the other hand, does not know about semantic similarity of sensory input beforehand. In order to know which observations correspond to similar situations with respect to the task, it has to solve the reinforcement learning problem (see Section 3), which it cannot solve without a suitable state representation. The question is: What is a good objective for state representation learning? We will now look at different objectives that have been proposed in the literature and relate them to our robotic priors (which we will define in Section 4).

Compression of Observations: Lange et al. [20] obtain state representations by compressing observations using deep autoencoders. This approach relies on the prior that there is a simple (low-dimensional) state description and on the prior that this description is a compression of the observations. While we use the same simplicity prior, we believe that it is important to also take time, actions, and rewards into account.

Temporal Coherence: Slow feature analysis [36] finds a mapping to states that change as slowly as possible, guided by the prior that many properties in our world change slowly over time. This method has been used to identify a representation of body postures of a humanoid robot [8] as well as for solving reinforcement learning tasks with visual observations [21]. Luciw and Schmidhuber [23] showed that slow feature analysis can approximate proto-value functions [24], which form a compact basis for all value functions. Incorporating the same prior, dimensionality reduction methods have used temporal distance to estimate neighborhood relationships [11].

Slowness or temporal coherence is an important robotic prior that our method also relies on. However, the purpose of the state is to provide a sufficient statistic based on which the robot can choose the right actions. Therefore, the actions of the robot must also be considered for learning the state representation. The following methods and ours take this valuable information into account.

Predictive and Predictable Actions: These approaches find state representations in which actions lead to simple, predictable transformations. Action respecting embeddings, proposed by Bowling et al. [3], aim at a state space in which actions are distance-preserving. Sprague’s [32] predictive projections find a representation such that actions applied to similar states result in similar state changes. Predictive state representations, proposed by Littman et al. [22], define states as success probabilities for a set of tests, where a test is a prediction about future observations conditioned on future actions. Boots et al. [2] showed how predictive state representations can be learned from visual observations. As we will see, these ideas are related to the proportionality prior, the causality prior, and the repeatability prior in this paper.

The problem with these methods—and all other methods discussed so far—is that they generate task-general state representations. This is problematic when the robot lives in a complex environment and there is no common state representation that works for all tasks. Therefore, we will use the reward to focus on the task-specific aspects of the observations and ignore information irrelevant for the task.

Interleaving State Representation Learning and Reinforcement Learning: The approaches presented so far learn state representations first to then use them for reinforcement learning. We will now discuss approaches that combine these steps. Piater et al. [26] use decision trees to incrementally discriminate between observations with inconsistent state-action values according to the reinforcement learning algorithm. This method is comparable to an earlier approach of Singh et al. [31], which minimizes the error in the value function by clustering states. Menache et al. [25] also adapt the state representation during reinforcement learning; they represent the state as a set of basis functions and adapt their parameters in order to improve the value function estimate.

Methods in this category rely on causality of values. They assume that the value is attributable to the state. To compute the value, they must solve the reinforcement learning problem. These steps can be decoupled by factorizing the value function into the reward function and the state transition function. This is used by the following approaches, and also by ours.

Simultaneously Learning the Transition Function: In earlier work [13], we proposed to learn the state transition function together with the state representation to maximize state predictability while simultaneously optimizing temporal coherence. A drawback of this approach is that it ignores the reward and, therefore, cannot distinguish task-relevant from irrelevant information.

Simultaneously Learning Transition Function and Reward Function: Some approaches jointly learn an observation-state-mapping, a transition function, and a reward function, differing in their learning objective. Hutter [9] proposes minimizing the combined code length of the mapping, transition function, and reward function. Duell et al. [6] learn these functions to predict future rewards conditioned on future actions. Jetchev et al. [12] maximize state predictability and reward discrimination to learn a symbolic state representation.

These approaches build models of state transitions and rewards to enforce state predictability and reward discrimination. Contrary to this approach, we define our learning objective in terms of distances between state-samples, similar to the idea of multi-dimensional scaling [19]. In this way, we can assure the existence of transition and reward functions for the state representation without having to model them explicitly.

3 State Representation Learning

Reinforcement learning is the problem of learning how to maximize future rewards from interacting with the environment. The standard formalization of this problem is a Markov decision process (MDP). Solving an MDP means finding a policy π that selects the right action a_t based the current state s_t in order to maximize future rewards $r_{t+1:\infty}$ [33]. The term "Markov" refers to the following property of the state: given the current state, all future state transitions and rewards are independent of past interactions. In other words: the state summarizes all information from past interactions that the robot needs to select the best action.

This framework can be generalized to partially observable Markov decision processes (POMDPs), where the robot cannot directly perceive its state s_t but only an observation o_t that depends on s_t [15]. To determine which actions to choose, the robot must take the entire history of observations, actions, and rewards into account. The MDP and POMDP frameworks are able to describe the interactive loop between the robot and the physical world. They are, therefore, well-suited for formalizing many learning problems in robotics.

As robots must rely on their task-general sensors, they cannot directly perceive their task-specific state. Instead, they must extract the state from sensory observations. *State representation learning* is the problem of learning to extract a state description from a history of interactions (i.e. previous observations, actions, and rewards) in order to enable efficient learning of the policy. We formalize this problem as learning a mapping ϕ to the current state, such that $s_t = \phi(o_{1:t}, a_{1:t-1}, r_{1:t})$. Given s_t, the robot selects the next action according to its policy: $a_t = \pi(s_t)$.

In this paper, we focus on a special instance of this problem assuming that the state can be estimated from a single observation. The reduces the state representation learning problem to learning an observation-state-mapping ϕ, where $s_t = \phi(o_t)$ (see Figure 2).

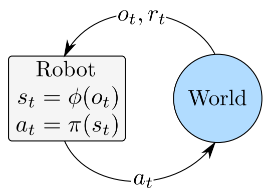


Fig. 2 The robot-world-interaction. At time t, the robot computes the state s_t from its observation o_t using observation-state-mapping ϕ. It chooses action a_t according to policy π with the goal to maximize future rewards $r_{t+1:\infty}$.

To make this simplification, we must assume that the observation has the Markov property. Technically, this turns the problem into a Markov decision process such that we could use the raw observation as state. However, as we will see later in this paper, the distance metric in the observation space (e.g., camera images) often does not enable the robot to learn the task. Even with Markov observations, robots must learn more useful state representations.

Note that sensory input of robots is often local and, therefore, observations are non-Markov for many robotic tasks. For some tasks, we can make observations Markov by including sensory inputs from multiple time steps in o_t. However, the dimensionality of o_t multiplies with the number of time steps that are taken into account and, ultimately, the state can depend on the entire history of observations, actions, and rewards. Alternatively, one could learn a state-filter instead of a observation-state-mapping. With every new action, observation, and reward, the filter would update the state. We think that it is crucial to investigate these extensions in future work. For now, we assume Markov observations and restrict ourselves to learning an observation-state-mapping. In the next section, we outline how we address this problem in a robotics-specific way.

4 State Representation Learning in Robotics

In this section, we present our approach to state representation learning in robotics. First, we list and explain five robotic priors. Then, we formulate state representation learning as an optimization problem by turning our robotic priors into a loss function. Finally, we turn the theory into a method that minimizes this loss function, thereby learning a state representation that reflects the priors.

4.1 Robotic Priors

The interaction between the robot and the real world is structured by the laws of physics. From this fact, we can derive robotic priors that capture characteristics of all robotic tasks based on physical interaction.

Simplicity Prior: For a given task, only a small number of world properties are relevant. This prior is related to Occam's razor, a widely accepted principle in science. In our context, Occam's razor will favor state representations that exclude irrelevant information, thereby leading to a lower-dimensional reinforcement learning problem and improving generalization.

Temporal Coherence Prior: Task-relevant properties of the world change gradually over time. This prior is related to Newton's first law of motion. Physical objects have inertia and change their velocity only gradually as a result of external forces. However, temporal coherence also applies to more abstract properties than physical motion, as most changes in the world occur gradually. The temporal coherence prior favors state representations that obey this principle as the robot transitions between states.

Proportionality Prior: The amount of change in task-relevant properties resulting from an action is proportional to the magnitude of the action. This prior results from Newton's second law of motion, $F = m \cdot a$. If an action represents the application of a certain force on an object of a fixed mass, the acceleration evoked by this force is constant. This holds true for robot movements and physical interactions with other objects but also generalizes to more abstract processes. As the robot does not know the magnitudes of its actions beforehand, this prior enforces that multiple executions of the same action result in the same amount of state change.

Causality Prior: The task-relevant properties, together with the action, determine the reward. This and the next prior resemble Newton's third law of motion or, more generally, causal determinism. If the same action leads to different rewards in two situations, these situations must differ in some task-relevant property and should thus not be represented by the same state. Consequently, this prior favors state representations that include the relevant properties to distinguish these situations.

Repeatability Prior: The task-relevant properties and the action together determine the resulting change in these properties. This prior is analogous to the previous one—for states instead of rewards—and also results from Newton's third law of motion. This principle is enforced by favoring state representations in which the consequences of actions are similar if they are repeated in similar situations. The repeatability prior and the causality prior together constitute the Markov property of states.

Note that most of these priors are defined in terms of actions and rewards. Thus, they do not apply to passive systems that can only observe but not act. These priors are also not generic artificial intelligence priors applicable to *all* tasks and environments, as artificial environments can be very different from our world, e.g., not obeying Newton's laws of motion. However, restricting the problem space to the physical world allows us to define useful priors.

But even in the physical world, there are still counterexamples for each prior. Proportionality does not hold when the robot hits a wall and its position remains constant even though it attempts to move with a certain velocity. Causality is violated due to sensory aliasing when the robot cannot distinguish two situations with different semantics. Repeatability is contradicted by stochastic actions. As we will see, all of these counterexamples are present in the experiments in this paper and our method is robust against them.

Our approach can handle such counterexamples because the robotic priors are not strict assumptions that have to hold for the method to work. Instead, our method finds a state representation that is consistent with these priors as much as possible. As the robotic priors capture the general structure of interactions with the physical world, they are useful, even in the presence of counterexamples.

4.2 Formulation as an Optimization Problem

We will now turn the robotic priors into a loss function L that is minimized when the state representation is most consistent with the priors. We construct loss terms for all robotic priors (except for the simplicity prior, see below) and define L as their weighted sum

$$L(D,\hat{\phi}) = \omega_{\text{t}} L_{\text{temporal coherence}}(D,\hat{\phi}) + \omega_{\text{p}} L_{\text{proportionality}}(D,\hat{\phi}) + \omega_{\text{c}} L_{\text{causality}}(D,\hat{\phi}) + \omega_{\text{r}} L_{\text{repeatability}}(D,\hat{\phi}).$$

Each of these terms is computed for an observation-state-mapping $\hat{\phi}$ and data of the robot interacting with the world, $D = \{o_t, a_t, r_t\}_{t=1}^{n}$, which consist of sensory observations o, actions a, and rewards r for n consecutive steps. The observation-state-mapping $\hat{\phi}$ is then learned by minimizing $L(D,\hat{\phi})$ (the $\hat{}$ indicates that ϕ changes during learning).

By linearly combining these loss terms, we assume independence between the robotic priors. They could also be combined non-linearly, but the existence of independent counterexamples for each individual prior supports our assumption. There is a weight ω assigned to each loss term because the typical magnitude varies significantly between these terms. The weights can counteract this imbalance. Additionally, the weighting allows to stress the importance of individual priors by using a higher weight for the corresponding loss term.

We will now describe how the individual robotic priors are defined as loss terms. For better readability, we will write $\hat{s}_t$ instead of $\hat{\phi}(o_t)$ when we refer to the state at time t according to the observation-state-mapping $\hat{\phi}$.

Simplicity Loss: The simplicity prior is not formulated as a loss term but implemented by enforcing the state representation to be of fixed low dimensionality. Future work should explore formulations of this prior as a loss term, e.g., defined in terms of sparsity, the number of dimensions, or other properties of the state representation.

Temporal Coherence Loss: States must change gradually over time. We denote the state change as $\Delta\hat{s}_t = \hat{s}_{t+1} - \hat{s}_t$. The temporal coherence loss is the expected squared magnitude of the state change.

$$L_{\text{temp.}}(D,\hat{\phi}) = \mathbf{E}\Big[\|\Delta\hat{s}_t\|^2\Big].$$

Proportionality Loss: If the robot has performed the same action at times t_1 and t_2, the states must change by the same magnitude $\|\Delta\hat{s}_{t_1}\| = \|\Delta\hat{s}_{t_2}\|$.

The proportionality loss term is the expected squared difference in magnitude of state change after the same action was applied.

$$L_{\text{prop.}}(D,\hat{\phi}) = \mathbf{E}\Big[(\|\Delta\hat{s}_{t_2}\| - \|\Delta\hat{s}_{t_1}\|)^2 \;\Big|\; a_{t_1} = a_{t_2}\Big].$$

Causality Loss: Two situations at times t_1 and t_2 must be dissimilar if the robot received different rewards in the following time step, even though it had performed the same action, $a_{t_1} = a_{t_2} \wedge r_{t_1+1} \neq r_{t_2+1}$.

The similarity of two states is 1 if the states are identical and approaches 0 with increasing distance between them. To compute the similarity of state pairs, we can use any differentiable similarity function. Previously, we have proposed to use the exponential of the negative distance, $e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|}$, following research from psychology [30]. However, the bell shaped function, $e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2}$, is a reasonable alternative and in our experience leads to improved performance.

Therefore, we define the causality loss as the expected similarity of the state pairs for which the same action leads to different rewards.

$$L_{\text{caus.}}(D,\hat{\phi}) = \mathbf{E}\Big[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \;\Big|\; a_{t_1} = a_{t_2}, r_{t_1+1} \neq r_{t_2+1}\Big].$$

Repeatability Loss: States must be changed by the actions in a repeatable way. If the same action was applied at times t_1 and t_2 and these situations are similar (have similar state representations), the state change produced by the actions should be equal, not only in magnitude but also in direction.

We define the repeatability loss term as the expected squared difference in the state change following the same action, weighted by the similarity of the states.

$$L_{\text{rep.}}(D,\hat{\phi}) = \mathbf{E}\Big[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \|\Delta\hat{s}_{t_2} - \Delta\hat{s}_{t_1}\|^2 \;\Big|\; a_{t_1} = a_{t_2}\Big].$$

4.3 Our Method

We will now show how a linear mapping from observations to states can be learned by minimizing the loss function.

Choosing the Weights: The weights in the loss function balance the relative importance of the different robotic priors. We believe that all proposed priors are important. Therefore, we chose the weights such that they provide gradients with similar magnitudes for the tasks we are interested in: $\omega_{\text{t}} = 1$, $\omega_{\text{p}} = 5$, $\omega_{\text{c}} = 1$, $\omega_{\text{r}} = 5$.

Note that little effort was put into choosing these parameters because the method is robust against a range of weight assignments. Simple uniform weighting also worked well in all of our experiments [14].

Approximating the Loss Function: We compute the expected values in the loss function by taking the mean over training samples. For the proportionality loss, the causality loss, and the repeatability loss, this would require taking into account all $O(n^2)$ pairs of training samples. For reasons of computational efficiency, we approximate each of these computations by only considering a subset of all pairs: For each training sample, we randomly choose 10 other samples such that the resulting pairs fulfill the conditions of the respective loss term. The mean is then taken over this subset of training sample pairs.

Learning a Linear Observation-State-Mapping: Our method learns a linear observation-state-mapping,

$$\hat{s}_t = \hat{\phi}(o_t) = \hat{W}(o_t - \mu_o),$$

where μ_o is the mean of all observations during training and $\hat{W}$ is a weight matrix that is adapted by performing gradient descent on the approximated loss function L. Linear functions form a very limited hypothesis space, but this method can easily be extended to non-linear functions approximators, e.g., using feature expansion, kernel approaches, or artificial neural networks.

Note that subtracting the mean observation is equivalent to subtracting the mean state and does not affect the loss function, which is defined over relative and not over absolute properties of the state samples. The only purpose of this step is to center the state representation. This can be useful for subsequent reinforcement learning depending on which form of function approximation is used (neural networks, for example, rely on mean centered input data).

Regularization: Without regularization, this learning scheme can lead to overfitting. Minimizing the loss for the training data can potentially increase the loss for unseen test data. In this case, the learned observation-state-mapping would not generalize well to new data. We approach this problem by introducing a L1 regularization on the weights of the observation-state-mapping. We find the best observation-state-mapping by solving the following optimization:

$$\hat{\phi} = \operatorname*{argmin}_{\phi} \left[L(D, \phi) + \lambda l_1(\phi) \right], \quad \text{where} \quad l_1(\phi) = \sum_{i,j} |W_{i,j}|.$$

The regularization parameter λ can vary greatly depending on the task. However, it can be estimated using cross-validation. In all our experiments, we do this by using the first 80% of the data for training and the remaining 20% as validation set. Then we do a grid search over different values for $\lambda \in \{0, 0.0003, 0.001, 0.003, ..., 3, 10\}$ and pick the value that achieved the lowest loss on the validation data after 100 steps of gradient descent. Using this value, we learn the final state representation on the entire data set for 100 steps.

Gradient Descent: The representation learning process starts with initializing the weight matrix W with small random weights, uniformly chosen from the interval $[-0.05, 0.05]$. Using this observation-state-mapping, it projects all experienced observations into the state space, computes the loss of this state representation and the gradient with respect to W, and changes W accordingly. This process is repeated, in our case for 100 learning steps. We dynamically adapt the step size for each weight dimension using improved Rprop with weight backtracking [10].

Analytical Gradient: The gradient can be computed analytically by partial derivatives with respect to all parameters $\hat{W}_{i,j}$ for each loss term. The weighted sum of these terms, together with the gradient of the regularization term, gives the total gradient.

$$\frac{\partial}{\partial \hat{W}_{i,j}} L_{\text{temp.}}(D, \hat{\phi}) = \mathbf{E}\Big[2 \underbrace{(\hat{s}_{i,t+1} - \hat{s}_{i,t})}_{\Delta \hat{s}_{i,t}} \underbrace{(o_{j,t+1} - o_{j,t})}_{\Delta \hat{o}_{j,t}} \Big].$$

$$\frac{\partial}{\partial \hat{W}_{i,j}} L_{\text{prop.}}(D, \hat{\phi}) = \mathbf{E}\Big[2\left(\|\Delta \hat{s}_{t_2}\| - \|\Delta \hat{s}_{t_1}\|\right) \Big(\frac{\Delta \hat{s}_{i,t_2} \Delta o_{j,t_2}}{\|\Delta \hat{s}_{t_2}\|} - \frac{\Delta \hat{s}_{i,t_1} \Delta o_{j,t_1}}{\|\Delta \hat{s}_{t_1}\|} \Big) \Bigm| a_{t_1} = a_{t_2} \Big].$$

$$\frac{\partial}{\partial \hat{W}_{i,j}} L_{\text{caus.}}(D, \hat{\phi}) = \mathbf{E}\Big[-2e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} (\hat{s}_{i,t_2} - \hat{s}_{i,t_1}) (o_{j,t_2} - o_{j,t_1}) \Bigm| a_{t_1} = a_{t_2}, r_{t_1} \neq r_{t_2} \Big].$$

$$\frac{\partial}{\partial \hat{W}_{i,j}} L_{\text{rep.}}(D, \hat{\phi}) = \mathbf{E}\Big[2e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \Big((\Delta \hat{s}_{i,t_2} - \Delta \hat{s}_{i,t_1}) (\Delta o_{j,t_2} - \Delta o_{j,t_1}) - (\hat{s}_{i,t_2} - \hat{s}_{i,t_1}) (o_{j,t_2} - o_{j,t_1}) \|\Delta \hat{s}_{t_2} - \Delta \hat{s}_{t_1}\|^2 \Big) \Bigm| a_{t_1} = a_{t_2} \Big].$$

Reinforcement Learning Method: To evaluate the utility of learned state representations, we used a standard reinforcement learning method that can handle continuous state spaces: fitted Q-iteration based on normalized radial basis function (RBF) features that are repeated for every action.

We generate these features by applying the k-means algorithm on all state samples to find 100 radial basis function centers. We set the standard deviation of the radial basis functions to be half of the average distance from a center to the closest next center. The activation of the radial basis functions are normalized such that they sum to one. This vector of state features is repeated for every action and stacked to build, for example, a 2500 dimensional feature vector when there are 25 discrete actions. In this state-action-feature vector, all elements are zero except for the radial basis function activations for the corresponding action.

The state-action-value-function, or Q-function, describes the value of applying a certain action in a certain state. We define it as a linear function of the radial basis feature vector, $\hat{Q}(s,a) = \hat{\beta}^T f_{\text{RBF}}(s,a)$. We initialize $\hat{\beta}$ with small random values and perform one step of Q-iteration to estimate the state-action-values of all training samples.

$$\forall_t : \hat{\mathbb{Q}}(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_a \left[\hat{Q}(s_{t+1}, a) \right],$$

where $\gamma = 0.9$ is used to exponentially discount future rewards. We then fit the linear function $\hat{Q}$ to the estimated state-action-values $\hat{\mathbb{Q}}(s_t, a_t)$ by assigning $\hat{\beta}$ to the vector that minimizes the sum of squared errors:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_t \Big(\underbrace{\beta^T f_{\text{RBF}}(s_t, a_t)}_{Q(s_t,a_t)} - \hat{\mathbb{Q}}(s_t, a_t) \Big)^2.$$

We alternate Q-iteration and Q-fitting until convergence. The resulting Q-function allows the robot to greedily choose the actions with the highest value, $\hat{\pi}(s) = \underset{a}{\operatorname{argmax}} \left[\hat{Q}(s,a) \right]$.

5 Experiments

In this section, we extensively evaluate our method in simulated and real robotic tasks with visual observations. We test how well our method can map 768-dimensional visual observations (16×16 pixels for red, green, and blue) into a two-dimensional or five-dimensional state space.

First, we look at the learning process in detail to understand how the state representation, the loss, and the gradient evolve. Then, we analyze learned state representations to gain insight into the capabilities of our approach. We start by comparing learned state representations for a simple navigation task[2] when the robot observes the scene from different perspectives, having either an egocentric view or a top-down view. The results show that, in both cases, our method learns a mapping to the same pertinent dimensions.

Next, we investigate in a slot car racing task[3] how our method can handle task-irrelevant distractors. To the best of our knowledge, this is the first time that this problem is addressed in state representation learning, even though we view it as essential (in any real-world environment, the observations of robots will be subject to task-irrelevant distractions). We will see that our method can separate task-relevant properties of the observation from irrelevant information. Then, we will also introduce distractors into the navigation task and see that the performance of our method is not influenced by them. After that, we analyze how the state representations for both tasks change if they are given more dimensions than necessary to solve the task. The results show that, in one of the tasks, our method can identify the minimal state dimensionality irrespective of the distractors.

Finally, we demonstrate that the learned state representations can substantially improve the performance of subsequent reinforcement learning. We explain how this effect results from improved generalization. Additionally, we show that state representations learned for one task can also enable reinforcement learning in other related tasks. In the end, we verify our results from simulation in an experiment with a real robot.

5.1 Learning Process

During learning, the observation-state-mapping is continually changed according to the gradient of the loss function to minimize the loss function of the resulting state representation. In this experiment, we will look at this learning process in detail for one run of the simple navigation task.

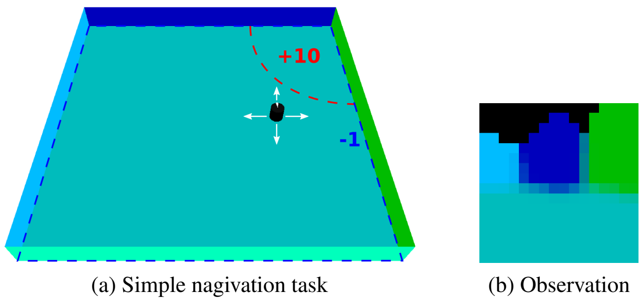


Fig. 3 Simple navigation task with fixed orientation. The robot gets positive reward in the top right corner and negative reward when it runs into a wall. (b) shows the observation in scene (a).

Simple Navigation Task: In the simple navigation task (see Figure 3a), the robot is located in a square-shaped room of size 45×45 units with 4-units-high walls of different colors. The robot has a height and diameter of 2 units. The orientation of the robot is fixed but it can control its up-down and left-right velocity choosing from $[-6, -3, 0, 3, 6]$ units per time step. The robot thus has 25 discrete actions. These actions are subject to Gaussian noise with 0 mean and standard deviation of 10% of the commanded velocity. The task of the robot is to move to the top right corner without hitting a wall. If the distance to this corner is less than 15 units, the robot gets a reward $+10$ unless it is touching a wall, in which

[2] The navigation task is based on similar experiments in the literature [2, 32].

[3] The slot car racing task is inspired by an experiment of Lange et al. [20].

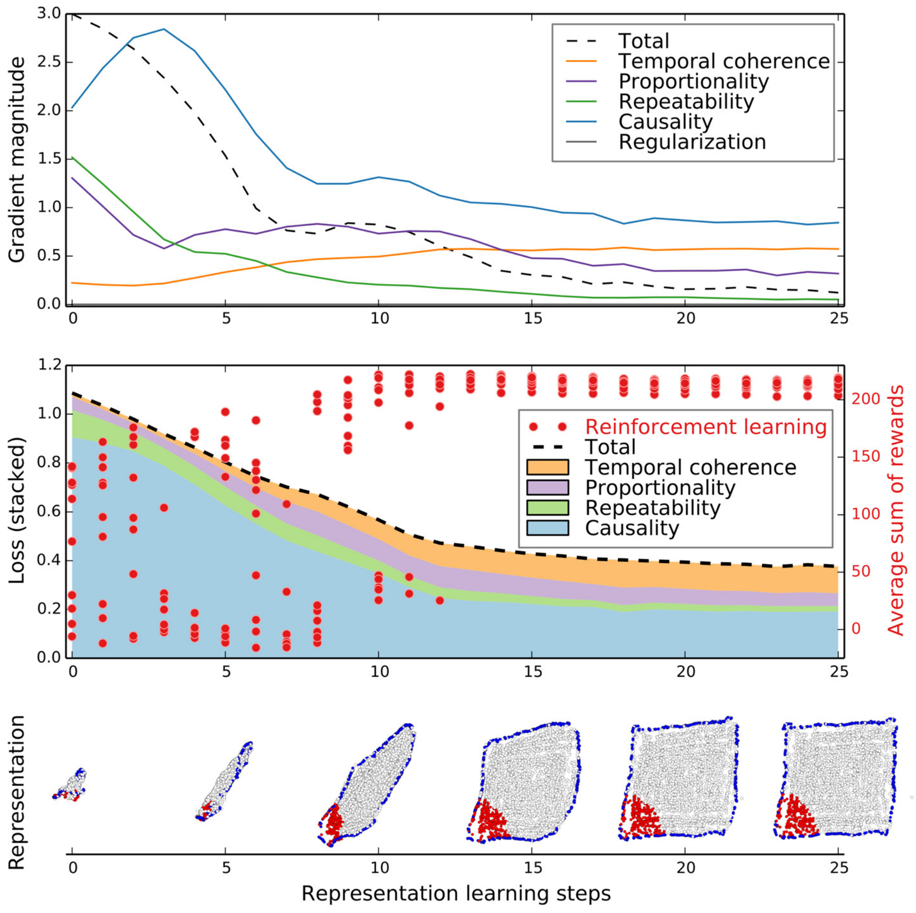


Fig. 4 State representation learning process. The plots show how the gradient magnitude (top), the loss on validation samples (middle), the reinforcement learning success (middle), and the state representation (bottom) change during learning. Every red dot in the middle plot shows the evaluation of one policy based on the learned state representation. Every dot in the bottom plot represents one observation mapped into the state space. The colors denote the reward that was received in that situation (red $= +10$, blue $= -1$, regularization: $\lambda = 0.03$).

case it gets a negative reward of -1. The robot perceives its environment through a camera with a wide angle lens (field of view $300°$). The 16×16-pixel RGB image is represented as a 768-dimensional observation vector. The example observation (see Figure 3b) shows the dark blue wall in the middle and the green and the light blue wall on either side of the image.

Experimental design: The robot explored its environment performing 5000 random actions. Based on this experience, it learned a mapping from the 768-dimensional observation space to a two-dimensional state representation. We interrupted the learning process after every learning step, applied a standard reinforcement learning method to learn a policy, and evaluated it for 20 episodes of 25 steps starting from different initial positions. We repeated this reinforcement learning and evaluation cycle ten times for every learning step due to the randomness in the reinforcement learning method.

Results—Loss Gradient Unfolds State Representation: The learning process starts from a randomly initialized observation-state-mapping and the corresponding state representation (see bottom left in Figure 4). Every learning step changes the mapping and unfolds the state representation. When this process converges, the state representation resembles the task-relevant properties of the world—in this case the location of the robot (see bottom right in Figure 4).

The observation-state-mapping is changed following the loss gradient (see dashed line in top plot in Figure 4). This gradient is the sum of the loss term gradients, each of which can point in a different direction (see colored lines in top plot, Figure 4). The causality gradient pushes apart states that lead to different rewards after performing the same action. It is most active in the beginning when all states are near each other. The temporal coherence gradient pulls consecutive states closer together. It becomes more active when the state representation is spread out. The repeatability gradient enforces deterministic state changes and is also most active in the beginning when the state representation is still very chaotic. The proportionality gradient equalizes distances across the state space. Its activation declines as the representation increasingly complies with this objective.

Throughout the training process, the causality loss and gradient are larger then the corresponding terms of the other priors. This results from the fact that causality directly opposes the other priors. Causality can be optimized by expanding the state space while all other robotic priors can be optimized by collapsing all states to a single point. To reach a balance, the causality gradient must be as large as the other gradients combined.

In the beginning, the different gradient terms do not (entirely) oppose each other and there is a large total gradient according to which the observation-state-mapping is modified. Following the total gradient, its magnitude decreases until a local minimum of the loss function is reached. This is where the different gradients point in opposing directions, the total gradient is zero, and the state representation converges.

Note that the resulting state representation discriminates between states that have different delayed rewards although immediate rewards are identical. For example, most locations in the room have the same immediate reward, zero, but they are still represented by different states. It is important to differentiate between them because they lead to different delayed rewards after performing a number of actions. Our method discriminates between these situations due to a synergy of the causality prior and the repeatability prior. The causality prior discriminates between different immediate rewards. The repeatability prior enforces deterministic state changes and thereby propagates this discrimination to situations from which states with different rewards can be reached. The temporal coherence prior and the proportionality prior make this process more robust using additional knowledge.

Results—Loss as Proxy for Reward: As the loss decreases during learning, the reinforcement learning success increases (see middle plot, Figure 4). In the beginning, the reinforcement success varies greatly between different evaluations. This is due to the randomness in the reinforcement learning method we used, especially in random assignment of radial basis function centers. But during the learning process this changes in two ways.

First, the reinforcement learning failures (with total reward less than 50) stop after learning step twelve. These failures are due to incorrect generalization from experience to new situations. They are caused by a state representation that does not capture the pertinent dimensions of the task. With a good state representation, however, every execution of the reinforcement learning method generalizes well.

Second, the successful reinforcement learning trials (with more than total reward of 50) become better. There is a strong correlation between this increase and the loss decrease.

Both results show that minimizing the loss function can be a good proxy objective for maximizing reward when learning a state representation. If we wanted to use the reward as objective directly, every representation learning step would involve learning policies and evaluating them by performing actions in the real world. Even worse, the number of required evaluations increases exponentially with the number of parameters in the observation-state-mapping, because we would have to compute a gradient with that many dimensions. The loss function, on the other hand, allows to directly compute the gradient.

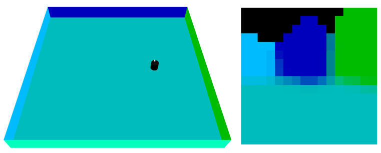
5.2 Invariance to Perspective

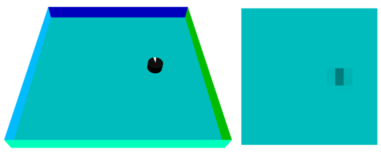
To investigate whether our method is invariant to perspective, we test it in two versions of a simple navigation task with different visual observations, viewing the scene from the robot's perspective and viewing the scene from the top. In both versions, the robot learns a state representation that reflects its location which is exactly the information required to solve the task.

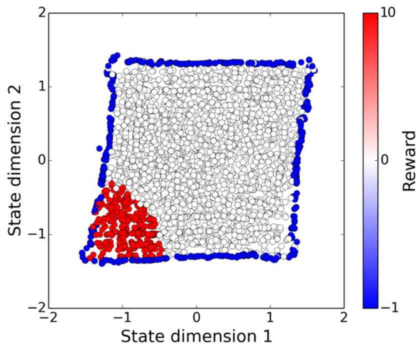
Perspectives in Simple Navigation Task: The simple navigation task, as described in section 5.1, has egocentric observations from a camera on the robot. We will refer to this task as egocentric view version of the simple navigation task (see Figure 5a). The top-down view version of this task is identical to this version, except for the observations which are images of the entire scene taken from the top by a static camera. In these images, the robot shows as a dark spot against the background. For the robot to be always visible, we had to increase its diameter to 4 units in this version of the task (see Figure 5b).

Experimental Design: We performed the following experiment for both versions of the task. The robot explored its environment performing 5000 random actions and learned a mapping from the 768-dimensional observation space to a two-dimensional state representation based on this experience.

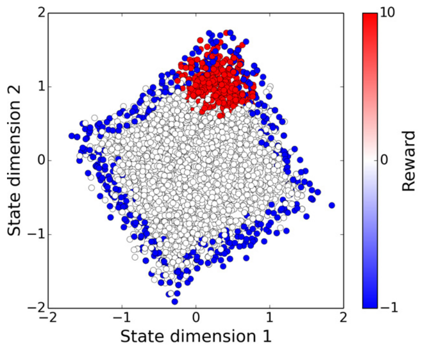
Results—Equivalent State Representations: To compare the learned state representations, we have plotted the state estimates for 5000 training steps based on egocentric view observations (see Figure 5c) and based on top-down view observations (see Figure 5d). In both cases, the samples form a square in state space, suggesting that the state is an estimate of the location of the robot in the square room. We can show that this is in fact what is learned by coloring each state sample according to the reward that the robot received in this state during training. The result resembles very accurately the reward definition in Figure 3a. The learned state representation is equivalent in both versions of the task. There are two orthogonal axes in the state representations that correspond to the coordinates of the robot. Of course, these axes in state space do not have to align between experiments; they can be rotated or flipped.


(a) Egocentric view version

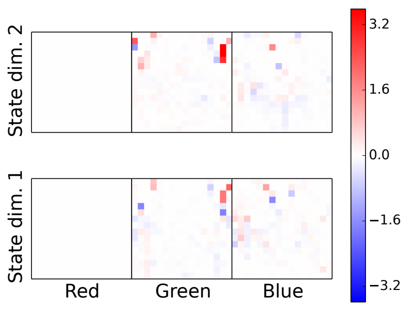

(b) Top-down view version



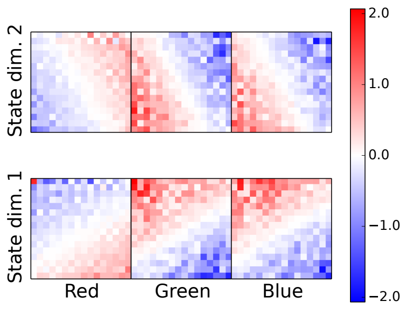
(c) State representation (egocentric)



(d) State representation (top-down)



(e) Observation-state-mapping (egocentric)



(f) Observation-state-mapping (top-down)

Fig. 5 Results for two versions of the simple navigation task with fixed orientation. The observation of the robot is either an egocentric view (a) or a top-down view (b) of the scene. (c) and (d) show the representations of 5000 training samples in state space for both versions. Each dot results from applying the learned observation-state-mapping $\phi^\star$ to an observation. The color relates the state samples to the reward received in that situation during training. (e) and (f) display the learned observation-state-mappings, i.e., the respective weight matrices which linearly project pixel values in the different color channels to state dimensions. Regularization and final loss on validation samples: egocentric version, $\lambda = 0.03$, $L = 0.36$; top-down version, $\lambda = 0.1$, $L = 0.52$.

Results—Different Observation-State-Mappings: In the two versions of the task, the sensory observations of the robot are very different. Nevertheless, it learned a mapping from these observations to the task-relevant dimensions: the location of the robot. Note that the mappings from observations to states must be very different to result in this identical state representation (see Figures 5e and 5f).

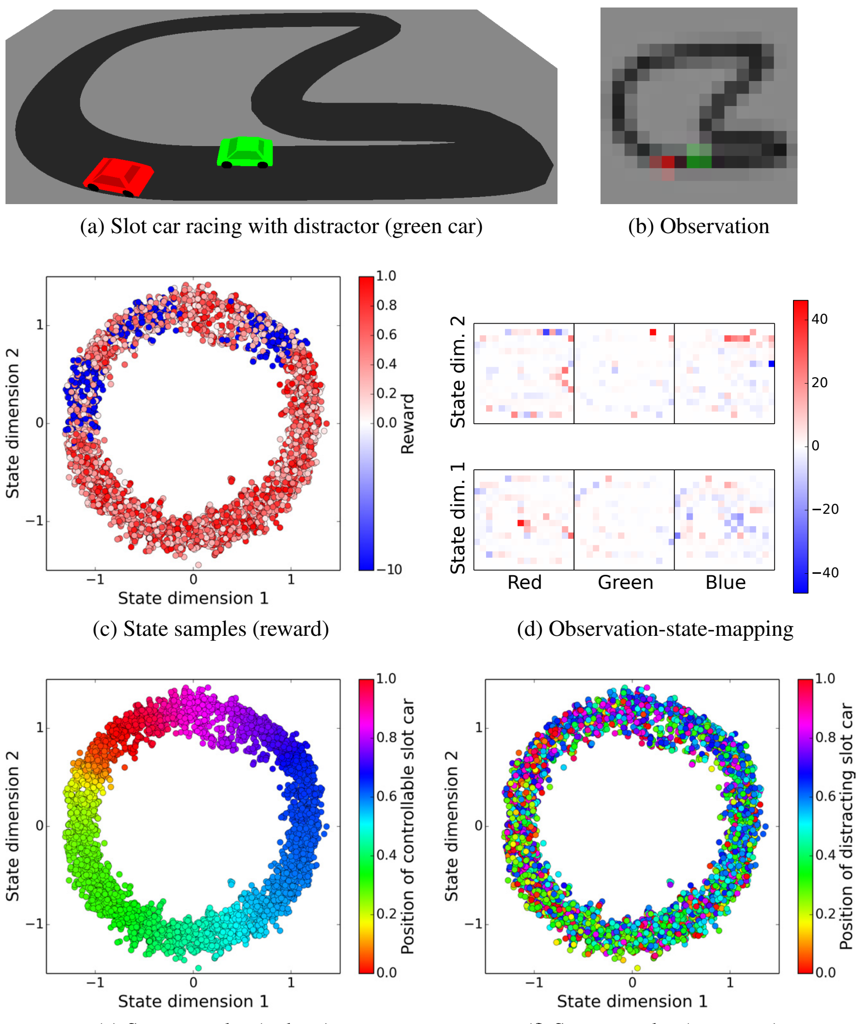
Results—Loss Explains Quality Difference: Even though the state representations for both versions are equivalent in principle, they are of different quality. The state representation based on egocentric observations is much clearer compared to the more blurry state representation based on top-down view observations, where situations with different rewards are mapped to similar states. Reinforcement learning based on such a representation will be less successful compared to the state representation based on egocentric observations. Presumably, the low resolution top-view observations cannot be (linearly) mapped to the location of the robot with high accuracy because they do not contain sufficient information.

This quality difference is also reflected in the final loss on validation samples L. $L = 0.36$ for the egocentric version and $L = 0.52$ for the top-down version of the task. This reconfirms that the defined loss is a sensible quality measure for state representations.

5.3 Ignoring Distractors

In this experiment, we test whether our method distinguishes task-relevant properties of the observations from irrelevant information. First, we investigate this in a slot car racing task with two cars. While the robot observes two slot cars, it can only control one of them. The other car does not play a role in this task apart from potentially distracting the robot. The robot does not know beforehand which car is relevant for the task. Second, we extend the simple navigation task with task-irrelevant distractors and test our method in this scenario.

The Slot Car Racing Task: An example scene from the slot car racing task is shown in Figure 6a. The robot can control



(a) Slot car racing with distractor (green car)

(b) Observation

(c) State samples (reward)

(d) Observation-state-mapping

(e) State samples (red car)

(f) State samples (green car)

Fig. 6 Results for the slot car racing task (a) with visual observations (b). Plots (c), (e), and (f) show the learned state representation. The color relates state samples to reward (c), the position of the relevant car (e), and the position of the distractor (f). Plot (d) shows the weight matrix of the learned observation-state-mapping. The regularization and final validation loss are $\lambda = 0.001$ and $L = 0.49$.

the velocity of the red car, choosing from $[0.01, 0.02, \ldots, 0.1]$ units per time step. The velocity is subject to zero mean Gaussian noise with standard deviation of 10% of the commanded velocity. The robot's reward is proportional to the commanded velocity—unless the car goes too fast in a sharp turn and is thrown off the track. In this case, the robot gets a negative reward of -10. The robot cannot control the green slot car. The velocity of this car is chosen randomly from the same range as for the red car. The green slot car does not influence the reward of the robot or the movement of the red car. The robot observes the scene from the top through a 16×16-pixel RGB image (see Figure 6b).

Experimental design: The robot explored randomly for 5000 time steps and then learned a mapping from 768-dimensional observations to two-dimensional states.

Results—Relevant Information Extracted: To understand the learned state representation, we have plotted the states of the 5000 training steps with one dot per state sample (see Figure 6c). The states form a circle which corresponds to the topology of the track. We have colored the state samples according to the reward. The three blue clusters in this plot correspond to the three sharp turns on the track, where the robot had routinely driven too fast during training and thus received negative reward. We also colored the states according to the ground truth position of the red slot car (see Figure 6e) and the green slot car (see Figure 6f). The figures show that the position along this circle in state space corresponds to the position of the controllable slot car on the track. One round of the red slot car corresponds to a circular trajectory in state space. Our method was able to distinguish task-relevant from irrelevant information in the observations and, at the same time, found a compressed representation of these pertinent properties.

Results—Two-Dimensional Representation of Position: The position of the slot car on the track is one-dimensional and,

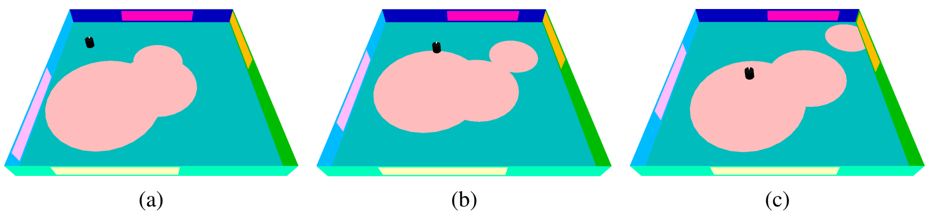
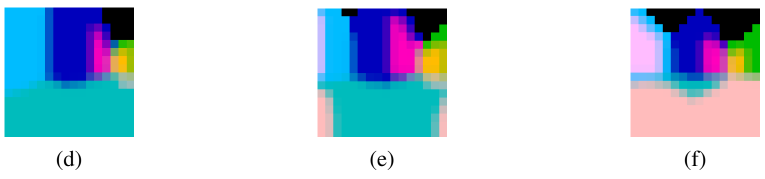
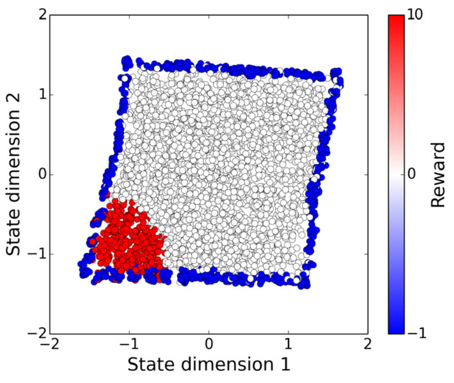
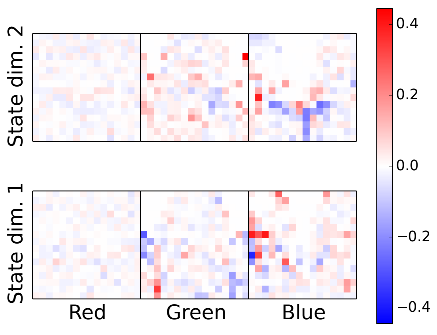



Fig. 7 Distractors in the simple navigation task. (a-c) show three situations at an interval of ten time steps. The robot is exploring while the distractors move randomly. (d-f) show the corresponding observations (note how they are influenced by the distractors).



(a) State representation



(b) Observation-state-mapping

Fig. 8 Results for simple navigation task with distractors. Regularization and final validation loss are $\lambda = 0.3$ and $L = 0.36$.

thus, could be captured in one-dimensional state representation. However, such a representation would not conform with the temporal coherence prior. The positions at the beginning and the end of the track would be maximally apart in such a representation, even though they are actually very close together due to the circularity of the track. To represent a circular one-dimensional property, we need a two-dimensional Euclidean space.

Simple Navigation Task with Distractors: To verify the results from the slot car racing task in the simple navigation task, we added seven distractors to this task: three circles on the floor and four rectangles on the walls that move randomly (see Figure 7). The distractors are observed by the robot but do not influence its movement or reward. They are irrelevant for the task and should thus not be included in the state representation[4]

Experimental design: The robot explored randomly for 5000 time steps and then learned a mapping from 768-dimensional observations to a two-dimensional state.

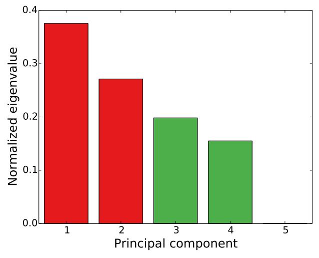
[4] The colors of the distractors were chosen to be equal to their background in the green and the blue color channel of the RGB image. They can be ignored by not taking into account the observation dimensions that correspond to the red color channel.

Results—Identical State Representation: The state representation that the robot learned in the presence of visual distractors captures the task-relevant properties and entirely ignores the distractors. In fact, the learned representation is identical to the one learned without the presence of distractors (compare Figures 8a and 5c). Also, the validation loss after training is 0.36 in both tasks.

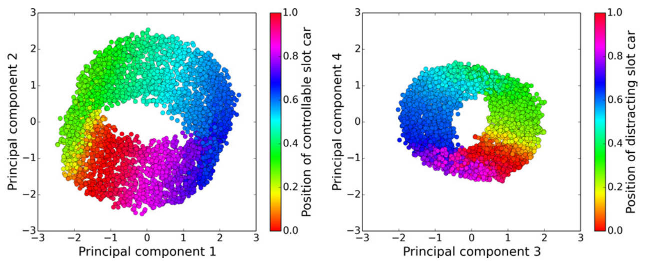
Results—Different Observation-State-Mappings: There are major differences in the learned observation-state-mappings between the task with and without distractors (compare Figures 8b and 5e). In the task including the distractors, the weights are more evenly distributed across the image. In this task, the highest weight is smaller by an order of magnitude compared to the task without distractors. This makes sense because the observations vary much more in this task, leading to stronger regularization ($\lambda = 0.3$ instead of 0.03) and a more robust mapping.

5.4 Mapping to a Higher-Dimensional State Space

In the previous experiments, we gave the robot an appropriate number of dimensions for the state representation. In this section, we investigate what happens in these same examples when the robot learns state representations with more dimensions than necessary. To this end, we repeated the experiments for the slot car task and for the simple navigation task with distractors, only now learning a five-dimensional instead of a two-dimensional state representation. After exploration for 5000 time steps and state representation learning, we took the 5000×5-matrix M containing the estimated states for these experiences and performed a principal component analysis of this matrix.



(a) Eigenvalues of state samples



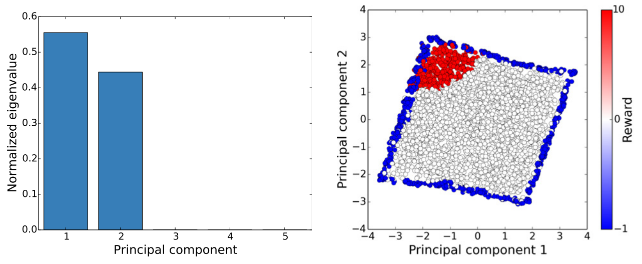
(b) Projected state samples (red car) (c) Projected state samples (green car)

Fig. 9 Results for the slot car task with a five-dimensional state space ($\lambda = 0.003$, $L = 0.45$).

Results—Alternative Explanations for the Reward: In the slot car task, the state sample matrix M has rank four. There are two larger eigenvalues and two smaller eigenvalues (see Figure 9a). If we project the state samples on their first two principal components, we can see that the dimensions with the larger eigenvalues correspond to the position of the controllable red slot car on the race track (see Figure 9b). The third and fourth principal component correspond to the position of the non-controllable green slot car (see Figure 9c).

If the green car is irrelevant for the task, why is it represented in the state? The robot maximizes state dissimilarity between situations where it received different rewards even though it performed the same action. If the robot chooses the same velocity but the slot car is thrown off one time while it stays on track another time, it makes the states of these two situations dissimilar. The most powerful discrimination between these situations is the position of the red slot car. But sometimes small differences in position or the stochasticity of the actions can make the difference between the two outcomes. The robot thus finds alternative explanations like the position of the green slot car. The eigenvalues show that this property has a lower impact on the state than the position of the controllable red slot car. Our method includes these alternative explanations if there are enough dimensions in the state space. When the state space is limited, the method focuses on pertinent dimensions as shown in Section 5.3.

Results—Identifying the Dimensionality of the Task: For the navigation task, we find that all but the first two eigenvalues of M are close to zero (see Figure 10a). The rank of the matrix is effectively two. This means that all state samples lie on a plane in the five-dimensional state space. We can visualize this plane by projecting the state samples on their first two principal components (see Figure 10b). The state space again corresponds to the location of the robot just as in the two-dimensional experiment. Apparently, the robot does not need to include additional properties of the world into the state representation in order to explain the reward. Thus, even with a five-dimensional state space, the robot learns that the task is two-dimensional and captures only those properties of its observation ignoring the distractors.



(a) Eigenvalues of state samples (b) Projected state samples

Fig. 10 Results for the navigation task with a five-dimensional state space ($\lambda = 1$, $L = 0.36$).

5.5 Improved Performance in Reinforcement Learning

The preceding experiments have shown some promising features of our method. But in the end, the quality of state representations can only be measured by their utility for subsequent learning. In this experiment, we will see that our method can substantially improve reinforcement learning performance and that it needs very few data to do so.

Extended Navigation Task: To construct a more challenging task for this experiment, we extended the navigation task by allowing the robot to change its orientation. The robot can turn and move forwards or backwards choosing its rotational velocity from $[-30, -15, 0, 15, 30]$ degrees per time step and its translational velocity from $[-6, -3, 0, 3, 6]$ units per time step for a total of 25 actions. All actions are subject

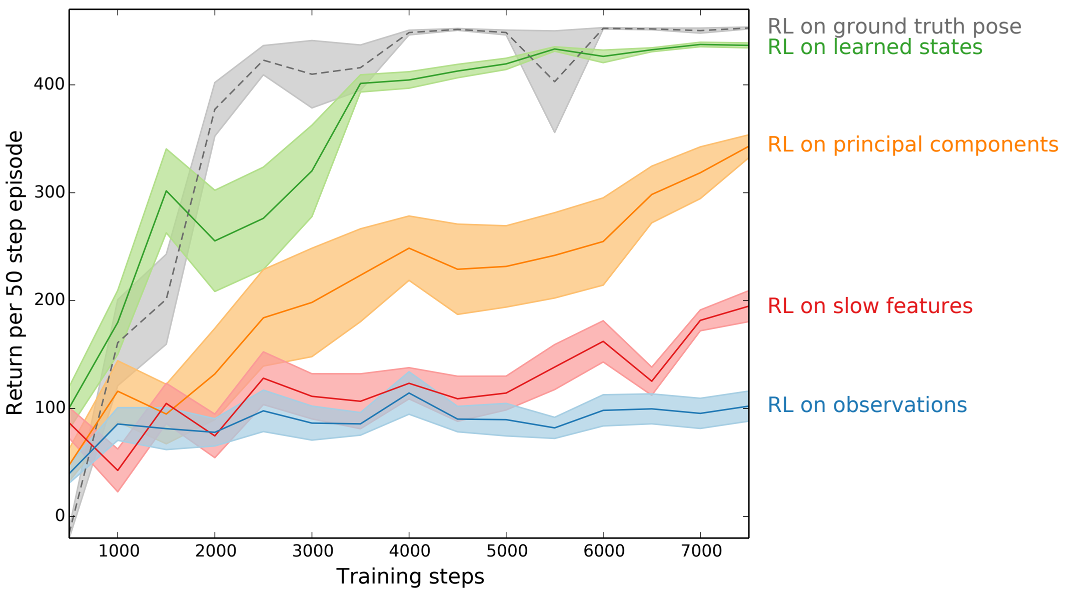


Fig. 11 Reinforcement learning (RL) curves for different state representations. Lines show means, surfaces display their standard errors.

to Gaussian noise with 0 mean and 10% standard deviation. This task includes the same visual distractors described before. The objective of the task also did not change. The robot must move to within 15 units of the top right corner, where it gets a reward of 10 unless it runs into a wall, in which case it gets a reward of -1 (see Figure 12).

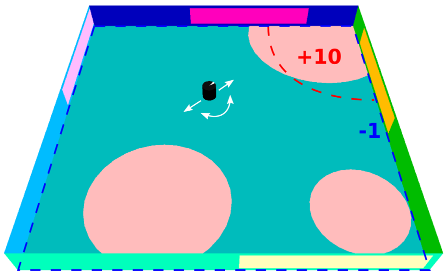


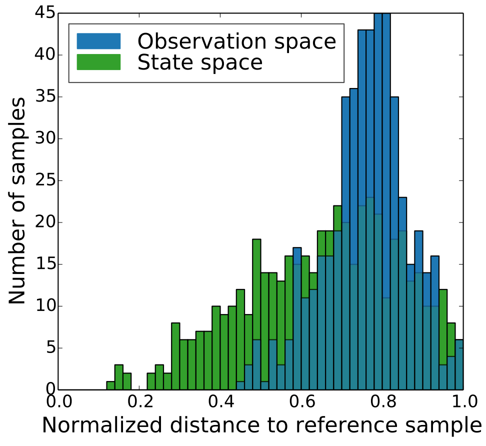
Fig. 12 Extended navigation task with distractors.

Experimental design: The robot explored its environment ϵ-greedy. With probability 0.9, it performed a random action, otherwise it performed the best action according to its policy. After every 500 time steps, the exploration was interrupted. From its accumulated experience, the robot learned an observation-state-mapping and a policy which it used in the following 500 exploration steps. After learning, the robot was also tested for 20 episodes, each consisting of 50 steps starting from random initial configurations. Based on these tests, we computed the average sum of rewards. This cycle of exploration, learning, and testing was carried out until the robot had explored for 7500 time steps. The entire learning experiment was repeated ten times.

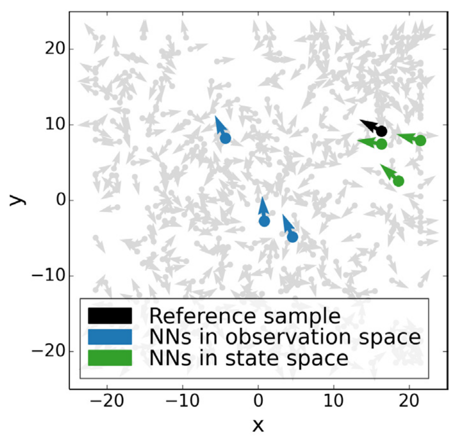
We performed the experiment multiple times with the same reinforcement learning method and different state representations: the five-dimensional state representation from our method, the five slowest features of the observations (computed using linear slow feature analysis [36]), the first five principal components of the observations, and the raw 768-dimensional observation. To get an upper bound on the reinforcement learning performance, we also compared with a simpler version of this task without distractors in which the robot has access to its ground truth pose. In this case it uses its position (normalized to $[-1, 1]$) and the cosine and sine of its orientation as state, which we consider an optimal representation for this task.

Results—Improved Generalization: We want to start analyzing the results in Figure 11, by comparing our method (green) against performing reinforcement learning directly on the raw observations (blue). These results show that the robot was not able to learn the task by directly applying reinforcement learning on the observations. When tested, the robot usually rotated in place or did not move at all. Using the state representation found by our method, however, it learned this task getting an average reward of about 430 per 50 step episode. This corresponds to reaching the goal area after 7 steps on average without ever running against a wall. Where does this difference come from? Our method basically acts as a regularization on the learning problem by extracting the right information from sensory input. This leads to better generalization from experiences to new situations.

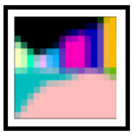
Virtually every learning algorithm, including the one we used for reinforcement learning, generalizes according to the smoothness prior, which is: “similar inputs lead to similar outputs” or in this case “similar states require similar actions”. Therefore, experiences are generalized to situations that have a similar representation where similarity is usually defined in terms of Euclidean distance. How do the distance



(a) Histogram of distances

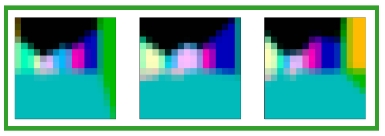


(b) Ground truth pose of NNs

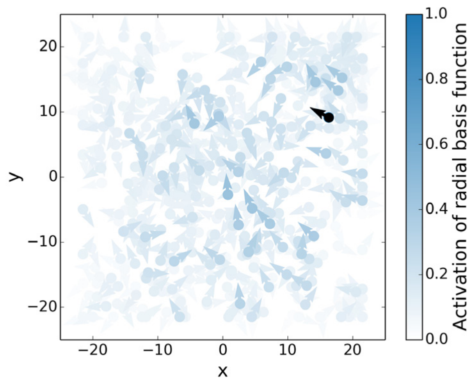


(c) Ref.

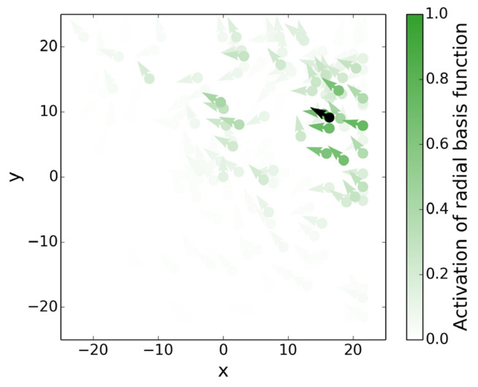
(d) NNs in obs. space



(e) NNs in state space



(f) Generalization in obs. space



(g) Generalization in state space

Fig. 13 Distances and generalization in the observation space (blue) compared to the learned state space (green). (a) shows the distances of different observation samples to the reference sample (c). (d) and (e) display the nearest neighbors (NNs) of the reference sample. The ground truth pose of these samples is displayed in (b). (f) and (g) illustrate how the distance metric affects generalization.

metrics implied by the observation space and the state space compare?

We examined this with another experiment. In this experiment, the robot explored randomly for 5000 steps and learned a five-dimensional state representation based on this experience. For the following analysis, we used every tenth of the training samples (to make sure that they are sufficiently different from each other). We chose one of these samples as a reference and computed the distances between the reference sample and all other samples both in observation space and in state space. This analysis reveals two important differences that have a huge impact on generalization.

First, the distances from the reference to every other sample have a much smaller variance in observation space compared to state space. In other words: Every sample is roughly equally far away in observation space. The nearest neighbor (NN) of the reference is almost half as far away as the most distant sample (see Figure 13a). This effect is caused by the high dimensionality of the observation space. In this space, everything is far apart. One would need an exponential number of samples to adequately fill the space. This poses a big problem for the smoothness prior in subsequent learning. In order to generalize to the most similar situations, one will automatically also generalize to other less similar situations because their distances from the reference are alike.

Second, the nearest neighbors of the reference sample in observation space can have very different semantics with respect to the task. To illustrate this, we show the ground truth pose of the reference sample and its three nearest neighbors in both observation space and state space (see Figure 13b). The nearest neighbors in state space correspond to similar poses such that it makes sense to generalize knowledge about which action to take from one situation to its neighbors. For the nearest neighbors in observation space, this is not the case. The reference observation (see Figure 13c) and its nearest neighbors in observation space (see Figure 13d) exemplify why observations from these different poses are close in observation space: The arrangement of distractors in their visual input coincides. The alignment of the walls, on the other hand, which provides information about the robot's location, differs greatly from the reference observation.

Note that this second issue is not specific to the pixel-representation. The same applies to different image representations: histogram of colors, histogram of gradients, point-features, and in fact every generic representation. Generic representations must include these task-irrelevant distractors because for other tasks, for example "follow the red rectangle", this is the important information and other information is irrelevant and distracting.

Both issues violate the smoothness prior that is used by the reinforcement learning method. The consequences for generalization are shown in Figures 13f and 13g. The color shows the strength of the generalization from the reference sample to every other sample and vice versa (computed as the activation of a radial basis function centered at the reference sample).

Results—Pertinent State Representation: The two baselines, principal component analysis (orange) and slow feature analysis (red), also improve reinforcement learning performance to some degree due to the low dimensionality of the state representation. However, these methods also suffer from being distracted by information in the visual input that is not task-related.

Both methods cannot distinguish relevant from irrelevant information because they are purely based on observations. Our method, on the other hand, takes actions and rewards into account and is, therefore, able to learn state representations that pertain to the task.

The reason why principal component analysis still performs reasonably well is that it finds those dimensions that can explain the largest variances in the observations and the robot's pose is one important factor for that.

Results—Almost as Useful as Ground Truth Pose: We now compare the results of our approach to the upper bound of the reinforcement learning method—using the ground truth pose of the robot as state (gray, dashed line, see Figure 11). We have to keep in mind how much easier this task is compared to coping with 768-dimensional visual observations influenced by distractors. Still, our method is able to learn a state representation that is almost as useful for reinforcement learning as the true pose of the robot. The final difference in average reward corresponds to taking one additional step until reaching the goal. The state representation learned by our method was almost as good as the best omniscient state representation that we could think of.

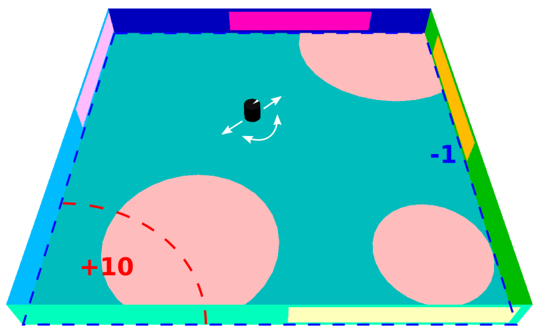
Results—Little Training Data Required: Interestingly, our method required few data to learn useful representations. Even at the very beginning, the learning curves based on the ground truth pose (gray) and based on the learned representation (green) are comparable. This shows that our method needs less training data to learn a useful state representation than the reinforcement learning method needs to learn the right policy.

5.6 Transfer Learning

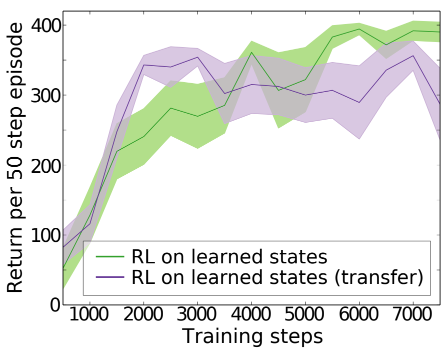
Ultimately, versatile robots must be able to learn a multitude of tasks, many of which are related. For effective learning, experience must be reused and transferred between tasks. For different tasks, the robot often needs to extract different information from its sensory input. If, for example, the robot is given the task to follow an object instead of moving to a fixed location, the absolute pose of the robot becomes meaningless and an object that was a distraction for the navigation task becomes important for the new task. While different tasks generally require different state representations, many tasks are also related, for example moving to different locations in the same environment. Such related tasks share or partly share useful state representations, for example the robot pose. Other tasks are composed of subtasks, for example moving to a certain location, picking something up, and then delivering it to another location. When tasks are combined from subtasks, their state representations could also be combined from the subtasks' state representations, which would allow to incrementally learn even very complex state representations.

Such an incremental learning scheme requires that the learned state representations are general enough to be reused in higher level tasks. This is not self-evident for state representations learned for a specific task. Navigating to the top right corner, for example, only requires to know the direction and the distance to this corner. While these features form a useful state representation for this specific task, they do not enable the robot to navigate to other locations. Interestingly, our previous experiments suggest that this is not the kind of state representation that our method produces. Instead, the learned state representations capture the task dimensions in a general way even though they were learned from data for a specific task. In this experiment, we want to quantify how reusable these state representations are across related tasks. We measure this by the utility of applying an observation-state-mapping that was learned for the extended navigation task in a set of new related tasks.

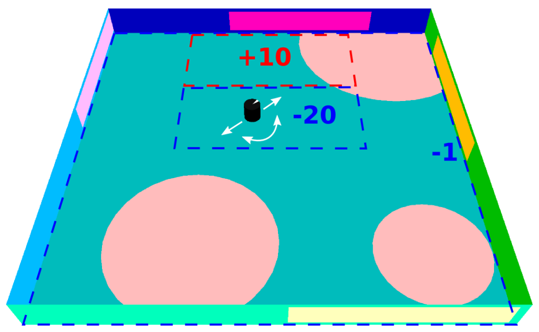
Three New Tasks: Task A is identical to the extended navigation task with distractors, except that the goal area has been changed to the opposite corner (see Figure 14a). In task B, the goal area is a rectangle in the top of the room. In this



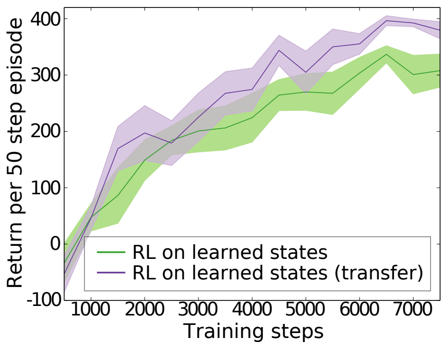
(a) Task A



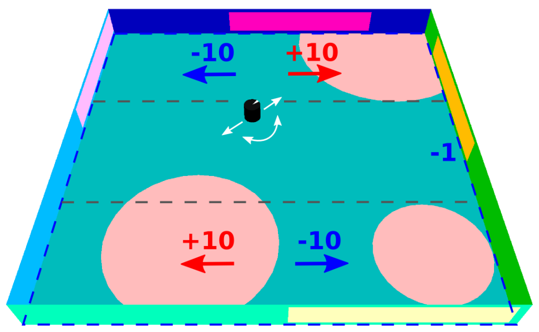
(b) Learning curves for task A



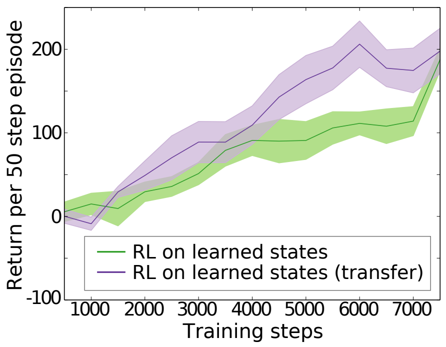
(c) Task B



(d) Learning curves for task B



(e) Task C



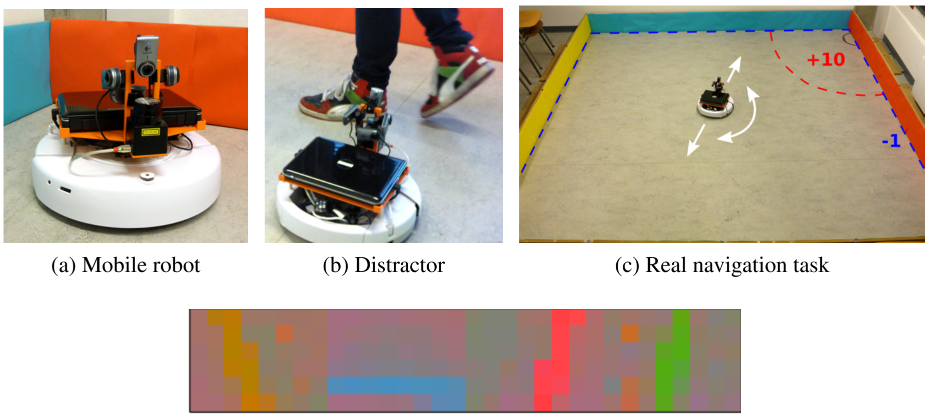
(f) Learning curves for task C

Fig. 14 Transferring learned state representations from the extended navigation task to new tasks. Lines show means, surfaces display mean estimate standard errors.

area, the robot receives a reward of $+10$. Next to this rectangle, there is another rectangular area in which the robot receives negative reward of -20 (see Figure 14c). In task C, the robot receives reward of $+10$ for moving right and negative reward of -10 for moving left when it is in the top third of the room. In the bottom third of the room, this is reversed: the robot is rewarded positively for moving left and negatively for moving right (see Figure 14e).

Experimental design: In this experiment, we measure the utility of a transferred state representation for reinforcement learning based on the cumulative reward obtained by the robot. We use the same design as in Section 5.5, except that the observation-state-mapping is not learned continually during the trial in which it is tested. Instead, the state representation is learned from 5000 random exploration steps. The learned observation-state-mapping is then used in the new tasks without changing it. We compare the utility of state representations learned specifically for the new tasks and those transferred from the extended navigation task.

Results—Successful Transfer of Learned State Representations: We compare the reinforcement learning performance in tasks A, B, and C based on the transferred state representation to the performance based on the state representation learned for each specific task (see Figure 14). Since both state representations lead to similar learning curves, the transferred state representation must be as useful for solving tasks A, B, and C as the state representations specifically learned for each task. These results show that, across this



(a) Mobile robot (b) Distractor (c) Real navigation task

(d) Observation from cameras (left, front, right, back)

Fig. 15 The navigation task with a real robot (c). The observation (d) is a combined image from the down-sampled output of the four RGB cameras attached to the robot (a). (d) shows the yellow wall to the left, the blue wall in front, the red wall to the right, and the green wall in the back of the robot (three cameras are tilted). Our method was applied on this observation without any additional preprocessing. (b) shows a distractor in front of the robot.

set of related tasks, our method is invariant to which specific task it was learned for.

Since task A is equivalent to the extended navigation task, it may be surprising that the reinforcement learning performance in task A is not as good as in the experiment in Section 5.5 (compare green curves in Figures 14b and 11). We think that this performance difference depends on whether the same or different data were used for state representation learning and reinforcement learning. When the learned observation-state-mapping is applied to new data, it will not generalize perfectly. While the learned policy seems to be robust against these deviations during testing, the reinforcement learning method appears to be more sensitive, which might result in a non-optimal policy.

For tasks B and C the learning curves suggest that the performance based on the transferred state representation might even be better than the performance based on the task-specific state representation. At this point, we do not want to speculate about the reason for these differences, but we will further investigate this in future research on state representation learning across related tasks. Overall, these results are very promising as they demonstrate the potential to reuse the learned state representations which is a prerequisite for incrementally combining them to form more complex state representations.

5.7 Verification on a Real Robot

Simulations are a great tool to conduct experiments under ideal conditions. However, simulations always carry the risk of ignoring important aspects of the problem. Therefore, we also need to verify our results in a real world scenario. For a video of this experiment, see:

`https://youtu.be/BolevVGJk18`

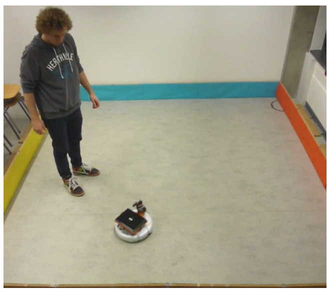
Real navigation task: We tried to replicate the extended navigation task as closely as possible using an iRobot Create equipped with four standard web cams (see Figure 15a). This robot was placed in a square room with yellow, blue, red, and green walls of length 3.2 meters (see Figure 15c). The camera settings (especially contrast and saturation) were chosen such that the colored walls are clearly visible. The images of the four web cams were down-sampled to 8×6 pixels and combined into a single image which the robot received as 576-dimensional observation (see Figure 15d).

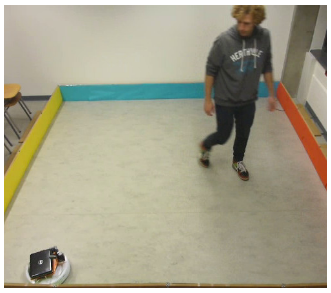
For computing the reward and for analyzing the results, it was necessary to estimate the pose of the robot. This was done using a particle filter based on input from a laser scanner on the robot. Neither the pose information, nor the laser scanner input was used by the robot during the task.

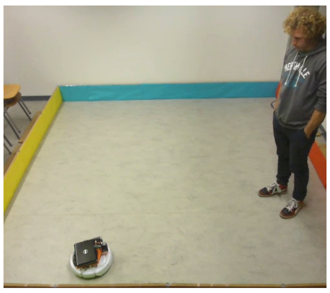
Experimental design: The experimental design was similar to the navigation tasks in simulation. The robot explored ϵ-greedy (see Figure 16d) and was interrupted after every 500 steps to learn an observation-state-mapping and a policy using the same methods and parameters as in the simulation experiments. After 2000 training steps (about 50 minutes), the robot was tested five times starting from different positions in the room. The resulting trajectories are shown in Figure 16e. During the entire experiment, a person was acting as a visual distractor (see Figure 15b). The person walked around randomly, avoiding collisions with the robot, but otherwise moving independently from it.

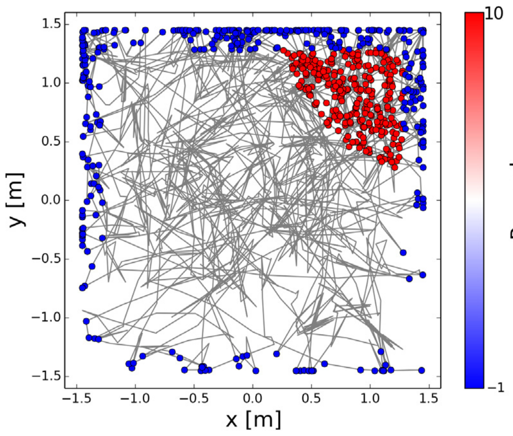
Results—Verification on a Real Robot After 2000 training steps, the robot had learned to move to the top right corner in the presence of the visual distractor (see Figure 16e).

This experiment reproduces our findings from simulation in the real world. Our method is able to extract the right information to solve this task from visual input, even in the presence of distractors. At least in this simple setting, it was able to cope with illumination changes, shadows and occlusion from the distractor, non-Gaussian noise in actions due

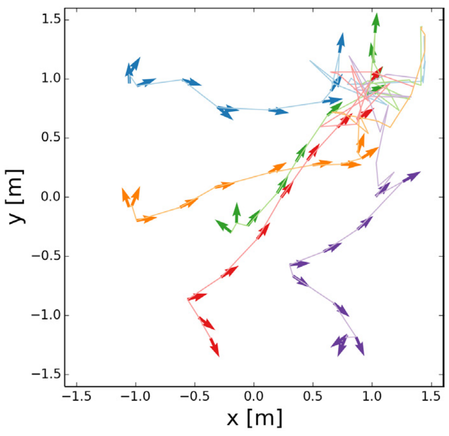

(a)


(b)


(c)



(d) 2000 training steps



(e) Five test episodes

Fig. 16 Training and results on a real robot. (a-c) show the robot and the moving distractor during the training phase. (d) shows the trajectory of the robot during its 2000 training steps. (e) displays five robot trajectories using the learned observation-state-mapping and policy.

to delays in the wireless connection, change of battery level and many other effects that were not modeled in simulation.

6 Conclusion

We have presented a new approach to state representation learning in robotics. The first key idea to this approach is to focus on state representation learning in the physical world instead of trying to solve the general problem of state representation learning in arbitrary (artificial) environments. Reducing the problem domain in this way allows us to use robotics-specific prior knowledge. We strongly believe that such prior knowledge will be vital to create versatile robots.

The second key idea is a specific way to use prior knowledge for state representation learning. We evaluate representations by how consistent they are with our priors about the world. We proposed five robotic priors—simplicity, temporal coherence, proportionality, causality, and repeatability—and showed how they can be turned into an objective for state representation learning.

Our experiments demonstrate that the inclusion of robotic priors can greatly improve learning performance and generalization. The magnitude of this effect is significant, in spite of the generality of the employed priors. We believe that the inclusion of robotic priors will broaden the applicability of learning and lead to an increase in task-versatility of autonomous robotic systems.

The five robotic priors presented here should be viewed as a first exploration into this matter. We believe there is an important need for formulating and evaluating additional priors to reflect the structure inherent to various domains in robotics. Hopefully, this will lead to a broader discussion about what the “right” robotic priors might be and how they could be leveraged in the most effective way to achieve autonomy and task generality in robots.

7 Acknowledgments

We gratefully acknowledge financial support by the Alexander von Humboldt foundation through an Alexander von Humboldt professorship (funded by the German Federal Ministry of Education and Research). We would like to thank Sebastian Höfer, Johannes Kulick, Marc Toussaint, and our anonymous reviewers for their very helpful comments and suggestions.

References

1. Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new per-

spectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

2. Byron Boots, Sajid M. Siddiqi, and Geoffrey J. Gordon. Closing the learning-planning loop with predictive state representations. *International Journal of Robotics Research*, 30(7):954–966, 2011.
3. Michael Bowling, Ali Ghodsi, and Dana Wilkinson. Action respecting embedding. In *22nd International Conference on Machine Learning (ICML)*, pages 65–72, 2005.
4. Luis C. Cobo, Kaushik Subramanian, Charles Lee Isbell Jr., Aaron D. Lanterman, and Andrea Lockerd Thomaz. Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains. *Artificial Intelligence*, 216(1):103–128, 2014.
5. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(8):2493–2537, 2011.
6. Siegmund Duell, Steffen Udluft, and Volkmar Sterzing. Solving partially observable reinforcement learning problems with recurrent neural networks. In *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 709–733. Springer Berlin Heidelberg, 2012.
7. Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006.
8. Sebastian Höfer, Manfred Hild, and Matthias Kubisch. Using slow feature analysis to extract behavioural manifolds related to humanoid robot postures. In *10th International Conference on Epigenetic Robotics*, pages 43–50, 2010.
9. Marcus Hutter. Feature reinforcement learning: Part I: Unstructured MDPs. *Journal of Artificial General Intelligence*, 1(1):3–24, 2009.
10. Christian Igel and Michael Hüsken. Empirical evaluation of the improved RPROP learning algorithms. *Neurocomputing*, 50(1):105 – 123, 2003.
11. Odest Chadwicke Jenkins and Maja J. Matarić. A spatio-temporal extension to ISOMAP nonlinear dimension reduction. In *21st International Conference on Machine Learning (ICML)*, page 56, 2004.
12. Nikolay Jetchev, Tobias Lang, and Marc Toussaint. Learning grounded relational symbols from continuous data for abstract reasoning. In *Autonomous Learning Workshop at the IEEE International Conference on Robotics and Automation*, 2013.
13. Rico Jonschkowski and Oliver Brock. Learning task-specific state representations by maximizing slowness and predictability. In *6th International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems (ERLARS)*, 2013.
14. Rico Jonschkowski and Oliver Brock. State representation learning in robotics: Using prior knowledge about physical interaction. In *Robotics: Science and Systems (RSS)*, 2014.
15. Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.
16. Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.
17. George Konidaris and Andrew G. Barto. Efficient skill learning using abstraction selection. In *21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1107–1112, 2009.
18. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012.
19. Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
20. Sascha Lange, Martin Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2012.
21. Robert Legenstein, Niko Wilbert, and Laurenz Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8):e1000894, 2010.
22. Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1555–1561, 2002.
23. Matthew Luciw and Juergen Schmidhuber. Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In *22nd International Conference on Artificial Neural Networks and Machine Learning (ICANN)*, pages 279–287, 2012.
24. Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10):2169–2231, 2007.
25. Ishai Menache, Shie Mannor, and Nahum Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1): 215–238, 2005.

26. Justus Piater, Sébastien Jodogne, Renaud Detry, Dirk Kraft, Norbert Krüger, Oliver Kroemer, and Jan Peters. Learning visual representations for perception-action systems. *International Journal of Robotics Research*, 30(3):294–307, 2011.
27. Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
28. Jonathan Scholz, Martin Levihn, Charles Isbell, and David Wingate. A physics-based model prior for object-oriented MDPs. In *31st International Conference on Machine Learning (ICML)*, 2014.
29. Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pages 437–440, 2011.
30. Roger N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820): 1317–1323, 1987.
31. Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 361–368, 1995.
32. Nathan Sprague. Predictive projections. In *21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1223–1229, 2009.
33. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
34. Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
35. Harm van Seijen, Shimon Whiteson, and Leon J. H. M. Kester. Efficient abstraction selection in reinforcement learning. *Computational Intelligence*, 30(4):657–699, 2014.
36. Laurenz Wiskott and Terrence J. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.