State Representation Learning with Robotic Priors for Partially Observable Environments

Marco Morik Divyam Rastogi Rico Jonschkowski[1] Oliver Brock

***Abstract*— We introduce *Recurrent State Representation Learning* (RSRL) to tackle the problem of state representation learning in robotics for partially observable environments. To learn low-dimensional state representations, we combine a Long Short Term Memory network with robotic priors. RSRL introduces new priors with landmarks and combines them with existing robotics priors from the literature to train the representations. To evaluate the quality of the learned state representation, we introduce validation networks that help us better visualize and quantitatively analyze the learned state representations. We show that the learned representations are low-dimensional, locally consistent, and can approximate the underlying true state for robot localization in simulated 3D maze environments. We use the learned representations for reinforcement learning and show that we achieve similar performance as training with the true state. The learned representations are robust to landmark misclassification errors.**

I. Introduction

To solve complex tasks in the real world, robots are equipped with sensors providing high-dimensional observations [1]–[3]. However, for a particular task, only a small part of this observation space is relevant. Thus, to facilitate task learning, we can encode the task-relevant information by learning a low-dimensional state representation for the robot [4]–[7].

To learn a low-dimensional state representation, most robots rely on observations of their direct surrounding. In *partially observable* environments, this local information is not sufficient to determine the current state of the robot.

This leads us to two main challenges when trying to learn a low-dimensional state representation for partially observable environments: we need (1) to memorize the information of past observations, and (2) to learn the state representation without requiring information about the true state.

In our approach, we use the hidden state a Long Short Term Memory (LSTM) network [8] to to maintain a memory of past observations. Furthermore, we use the idea of *robotic priors* [5] to learn state representations. These priors encode information about the interaction between the robot and the environment in the physical world by enforcing pairwise constraints between states. However, the existing priors are not powerful enough to learn representations when trained with complex models, such as LSTMs. We mitigate this by using a *reference point prior* [7].

We propose two concrete implementations of the *reference point prior—landmark* and *similar point*—and combine them with existing robotic priors to solve state representation learning in partially observable environments. For the experiments in this paper, we use the ground truth state to determine which samples are close to a reference point, but we believe that our method can also be applied without such supervision if the robot is able to recognize some of the states that it has seen before.

We validate our approach on a robot localization task in simulated 3D maze environments. Using both qualitative comparison and quantitative metrics, we are able to show that the learned state representation is (1) low-dimensional, (2) locally consistent, (3) encodes the connectivity of the environment, and (4) approximates the true state.

Furthermore, to test the utility of the learned states, we solve a reinforcement learning (RL) task to navigate a robot inside a maze. Despite the fact that our method is semi-supervised, the agent trained on the learned states performs at par with an agent trained directly on the true state.

II. Related Work

State representation learning (SRL) is the problem of extracting a low-dimensional representation from high-dimensional observations [5]. The goal of SRL is to learn representations which allow the robot to solve certain tasks. Most approaches in the literature combine SRL with RL. These approaches can be divided into two main categories: (1) learning a representation by optimizing for SRL and RL end-to-end or (2) using priors for learning a representation and then using the learned representation in RL.

End-to-end learning approaches optimize the state representation and the policy simultaneously. Since these approaches are very data intensive [4], they have been combined with auxiliary losses for learning complex tasks [9], [10]. In our work, we focus on losses that enable data-efficient learning of state representations without taking the end-to-end signal into account.

The second approach is to first learn a representation by optimizing an objective, defined by a set of priors, and then using it for RL. There are a variety of objectives to shape the state representation.

Autoencoders: Vincent et al. [11] used an autoencoder to learn a low-dimensional representation by compressing the observations. The learned states from autoencoders were successfully used to solve tasks using reinforcement learning [1], [12], [13]. However, the autoencoder also learns task irrelevant features which are necessary to reconstruct the observation.

All authors are with the Robotics and Biology Laboratory at Technische Universtität Berlin, Germany

[1]Now at Robotics at Google, USA

Predictive State Representations: Instead of compressing the observations, some authors used predictive state representations [14] to learn a state by predicting the future observations in partially observable environments. The learned states were used in a reinforcement learning framework [15]. These methods use relatively generic priors, while our approach makes stronger assumptions that are specific to robotics.

Robotic Priors: In robotics, the environment and the robot obeys the laws of physics. The idea of robotic priors is to encode this knowledge into a loss function which is minimized during training to shape the state space [5]. Jonschkowski and Brock [5] showed that the learned representation was resistant against visual distractions and could be used to solve RL tasks with improved learning performance and generalizability. However, the method assumed full observability of the environment. Lesort et al. [7] noticed that a state space learned only with these priors could be unstable. When trained on multiple trajectories, trajectories which were close together in the underlying true state space were mapped to different regions in the learned state space. Therefore, they introduced the *reference point prior* which pulled states at a reference point together. However, they only used one reference point in their approach which is insufficient to train more complex models (e.g LSTMs) in partially observable environments.

In this work, we introduce *Recurrent State Representation Learning* (RSRL) which combines the idea of robotic priors with LSTMs to handle partial observability and learns task relevant state representations.

III. Recurrent State Representation Learning

To learn state representations in partially observable environments, we use LSTMs to maintain a memory of past observations. LSTMs retain this memory by using a hidden state. We use the LSTMs to implement the mapping Φ, which uses the observation o_t, action a_t and the hidden state $\hat{h}_t$ at time step t to extract state $\hat{s}_t$ (and hidden state $\hat{h}_{t+1}$),

$$\hat{s}_t, \hat{h}_{t+1} = \Phi(o_t, a_t, \hat{h}_t)$$

We denote the learned states by $\hat{s} \in \hat{\mathcal{S}}$ and the underlying true state by $s \in \mathcal{S}$. We introduce the network architecture used to learn Φ in Section III-A.

To train the LSTM network, we use robotic priors. The priors used in this work are introduced in Section III-B and the training procedure in shown in Algorithm 1.

A. LSTM based Model

Our model consists of a convolutional network and a LSTM recurrent neural network. The model architecture is shown in Figure 1. The dimensionality of the output layer determines the dimensionality of the learned state space which depends on the task.

B. Robotic Priors

The mapping Φ is learned using robotic priors. Each prior is encoded in a loss function, which is minimized during training, to enforce a state space consistent with these priors. We will introduce all the priors used in this section.

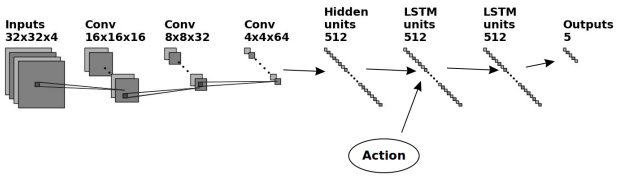


Fig. 1: The network architecture that is used for our experiments. The text above each layer indicates its size. From left to right, we have three convolutional layers, one fully connected layer, two LSTM layers and one fully connected output layer.

Temporal Coherence: The *temporal coherence prior* has been used for several state representation learning tasks [5], [16]–[18]. It encodes the idea that things in the real world move slowly by minimizing the distance between consecutive states. We encode this prior with the loss:

$$\mathcal{L}_{\text{Temp}} = \mathbb{E}(||\hat{s}_{t+1} - \hat{s}_t||^2)$$

Variation: The *variation prior* was introduced by Jonschkowski et al. [6]. As task-relevant information vary, randomly selected pairs of states should not be similar. This prior is encoded by the following loss:

$$\mathcal{L}_{\text{Var}} = \mathbb{E}(e^{-||\hat{s}_a - \hat{s}_b||^2})$$

Proportionality: To enforce the proportionality of state changes to actions in the state representation, Jonschkowski and Brock [5] introduced the *proportionality prior*. We extend this prior for continuous actions. In this formulation, the magnitude of the state change should be proportional to the action. This is encoded by the loss:

$$\mathcal{L}_{\text{Prop}} = \mathbb{E}((||\hat{s}_{t+1} - \hat{s}_t|| - e^{\beta}||a_t||)^2)$$

The proportionality factor e^{β} is part of the learning problem and β is selected based on the data.

Reference Point: The *temporal coherence* and *proportionality prior* are only affecting consecutive states, i.e. state pairs in the same trajectory, whereas the *variation prior* affects all state pairs. This means that states from different trajectories are pushed apart by the *variation prior* and not pulled together by any other prior. To connect states from different trajectories, Lesort et. al [7] introduced the *Reference Point Prior*. They used one *landmark*[1] as an anchor and minimized the distance between learned states, when the true state was equal to the landmark. However, the true state space $\mathcal{S}$ is often large and not all trajectories visit any specific reference point.

To tackle this problem, we generalize this prior to consider multiple landmarks and compute the k most similar state pairs within a landmark's range. Since we do not need the true state in our loss computation, we can, in practice, determine the similar states around a landmark with methods like semantic localization [19] or landmark recognition [20]–[22].

[1]We use the term *landmark* instead of *reference point* [7].

Algorithm 1: Training a recurrent state representation.

```
O: observations, A: actions, S: true state
Function RSRL(O, A, S):
    Preprocessing and initialization
    for e = 1 : Epochs do
        for b = 1 : Batches do
            O_B, A_B, S_B ← sample batch
            list_var ← random pairs
            list_ref ← Reference(S_B)
            Ŝ_B ← Φ(O_B, A_B)
            L ← Loss(S_B, list_var, list_ref)
            UpdateModel(Φ, L)
        check for early stopping
```

In our work, we use the true state s to find pairs of similar states around a landmark. To study the effect of the number of landmarks on the state representation, we propose two different formulations of this *reference point prior*.

1) *Landmark Prior*: Given a set $\mathcal{R}$ of l landmarks, for each landmark s_l, we gather all states in a certain range around the landmark. From these states, we select the k most similar pairs of states (s_a, s_b) from different trajectories. The loss minimizes the distance between these k pairs of states in the learned state space $\hat{\mathcal{S}}$:

$$\mathcal{L}_{\text{Ref}} = \mathbb{E}(||\hat{s}_a - \hat{s}_b||^2 \mid s_a \approx s_b \approx s_l;\ s_l \in \mathcal{R})$$

   In this work, we distribute the landmarks evenly across our environment and we define them using the true state s. We analyze the effect of different number of landmarks in Section V-C.
2) *Similar Point Prior*: To test the upper bound of the *reference point prior*, the *similar point prior* selects the k most similar pairs (s_a, s_b) regardless of any landmark. In our work, these are the pairs with the smallest distance in true state space $\mathcal{S}$. The prior is encoded by the loss $\mathcal{L}_{\text{Ref}}$ by implicitly assuming landmarks on every state.

Note, that we only included priors independent of the reward. Therefore, the learned state representation can be used for several tasks in the same environment.

C. Algorithm

An overview of the training procedure is presented in Algorithm 1. The Reference function returns the k pairs of points for the *reference point prior*. These are either computed with the *landmark* or *similar point prior*. Even though we used the true state to compute these pairs of points (s_a, s_b) in Reference, we only need the indices of these pairs $(a, b) \in list_{ref}$ to compute $\mathcal{L}_{\text{Ref}}$ and no information about the true state. We train the state representation by minimizing the weighted sum of the priors. The final loss function used for training is:

$$\mathcal{L}_{\text{RSRL}} = w_1\mathcal{L}_{\text{Temp}} + w_2\mathcal{L}_{\text{Var}} + w_3\mathcal{L}_{\text{Prop}} + w_4\mathcal{L}_{\text{Ref}} \quad (1)$$

where $w_i,\ i \in \{1, \dots, 4\}$ are the weights associated with each prior. We choose the weights w_i such that the gradients for all the individual losses are approximately equal in the beginning of training.

D. Supervised Baseline Model

In order to provide an upper baseline for the comparison of our representation learning models, we also train a supervised model. The supervised model has the same network architecture but we replace the computation of the loss in Equation (1) with the squared distance to the true state s:

$$\mathcal{L}_{\text{Sup}} = \frac{1}{n_B \cdot m_B} \sum_{i=1}^{n_B} \sum_{j=1}^{m_B} (\|\hat{s}_{i,j} - s_{i,j}\|_2^2)$$

where n_B is the batch size and m_B is the sequence length used in training the model.

E. Quantitative Evaluation Metrics

For quantitative assessment of our models, we use two methods:

KNN-Score: This score has been used by several authors [5], [7] for evaluating the learned state space. The KNN-Score calculates for each learned state $\hat{s}_i \in \hat{\mathcal{S}}$, the k nearest neighbors in the learned state space $\hat{\mathcal{S}}$. It then calculates the average squared distance of $s_i \in \mathcal{S}$ to the previously calculated neighbors in the true state space $\mathcal{S}$. The KNN score can be written as:

$$\text{KNN-Score}(\hat{s}) = \frac{1}{nk} \sum_{i=1}^{n} \sum_{\hat{s}_j \in \text{KNN}(\hat{s}_i, k)} \|s_i - s_j\|_2^2$$

where $k = 20$ for our experiments.

Validation Networks: Since the semantics of the learned state space $\hat{\mathcal{S}}$ are unknown, it is difficult to visualize the dimensions of $\hat{\mathcal{S}}$. Therefore, we introduce validation networks that try to reconstruct the true state s from the learned state $\hat{s}$. We can use the validation networks not only to generate a score but also to get more insight into the learned state space by displaying the reconstructed state. We learn the mapping $\Psi : \hat{\mathcal{S}} \rightarrow \mathcal{S}$ with a small supervised feed forward neural network. We train two different validation networks:

ValCon (Ψ_{ValCon}): This network Ψ_{ValCon} directly predicts the true state s given the learned state $\hat{s}$ as input. The loss for this network is computed with the weighted squared distance between the true state s_i and the predicted true state $\tilde{s}_i$:

$$\mathcal{L}_{\text{ValCon}} = \frac{1}{n_B} \sum_{i=1}^{n_B} \alpha_i \|\tilde{s}_i - s_i\|_2^2$$

where α_i performs a weighting of the training dataset to boost under-represented areas of the environment.

ValDis (Ψ_{ValDis}): To analyze whether the learned state space contains the structure of the environment, we discretize our environment into tiles. We replace the output layer of Ψ_{ValCon} by a layer with the number of nodes equal to the

number of tiles, N_{tiles}, followed by a softmax layer. This network Ψ_{ValDis} is trained to predict the probability of the learned state $\hat{s}$ being in each of the tiles using the cross entropy loss given by:

$$\mathcal{L}_{\text{ValDis}} = \frac{1}{n_B} \sum_{i=1}^{n_B} \alpha_i \sum_{j=1}^{N_{tiles}} y_i^j \log(\hat{y}_i^j)$$

where $\hat{y}_i^j$ and y_i^j are the probabilities of $\hat{s}_i$ and s_i being in tile j respectively. We implement the validation networks Ψ_{ValCon} and Ψ_{ValDis} with two hidden layers of 64 hidden nodes each.

With each of these networks, we calculate an error:

Euclidean Error (E_{Pos}): The euclidean error is defined by the average euclidean distance between the true state s and the state $\tilde{s}$ reconstructed by the network Ψ_{ValCon}.

$$E_{\text{Pos}} = \frac{1}{n} \sum_{i=1}^{n} \|\tilde{s}_i - s_i\|_2$$

where n is the size of the test dataset.

Tile Distance (E_{Geod}): The tile distance is defined by the average geodesic distance between the tile predicted by the network Ψ_{ValDis} for a state $\hat{s}_i$ and the tile of the true state s_i. This distance is particularly useful for environments such as mazes.

IV. Experimental Setup

We performed our experiments with the DeepMind Lab environment [23] using two different mazes: small and large. In each of the mazes, we removed posters, to make the environment partially observable. The true state s for the robot in the maze was the 4-dimensional pose $(x, y, \sin(\theta), \cos(\theta))$, where x, y are the euclidean coordinates and θ is the orientation. The observations o_t were $32 \times 32 \times 4$ RGBD images. The actions a_t were encoder values which were computed using the true state difference $a_t = \epsilon(s_t - s_{t-1})$ with normal distributed noise $\epsilon \sim \mathcal{N}(1, 0.1)$. The dimensionality of our learned state space was 5. In comparison to the 4 dimensional true state space, the extra dimension in our learned state space helped the network to avoid local optima and improve the quality of the learned representation. Note that the states are learned unsupervised. Therefore, we have no control which dimensions contain information about orientation or position.

We used a simple exploration policy to collect datasets with two different sampling frequencies, 1 Hz and 10 Hz. The dataset for each maze contained $n = 2000$ trajectories, where each trajectory had $m = 100$ steps. Since the number of steps were same for both sampling frequencies, the 1 Hz dataset explored more parts of the maze within one trajectory, whereas the 10 Hz dataset was necessary for solving the reinforcement learning task. The small maze with one trajectory each from both datasets is shown in Figure 2. The large maze is shown in Figure 4a.

For each maze, we trained four models implementing the *landmark prior* using $l \in \{5, 16, 32, 64\}$ landmarks ($\Phi_{\text{lm5}}, \Phi_{\text{lm16}}, \Phi_{\text{lm32}}, \Phi_{\text{lm64}}$), and one model using the *similar point prior* Φ_{sp}. Additionally, we trained a supervised model

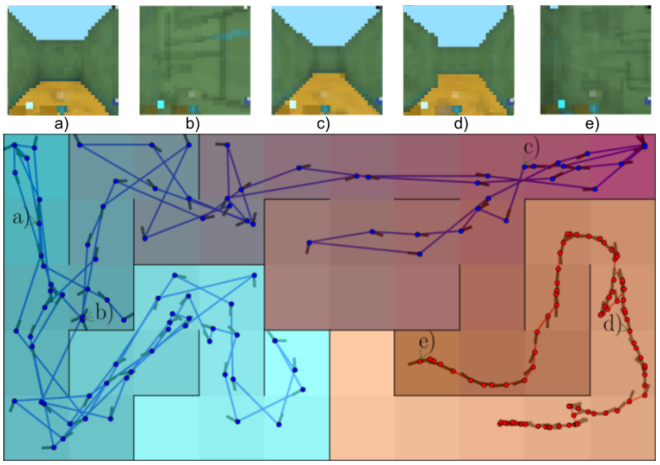
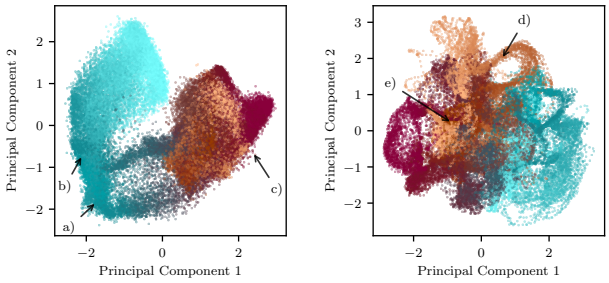


Fig. 2: The small maze with two trajectories: the left (blue) sampled with 1 Hz and the right (red) sampled with 10 Hz. Above the maze, there are 5 RGB observations shown with the corresponding true states annotated in the trajectories. Observations $a)$ and $c)$ look very similar but are from opposite parts of the maze which shows that the environment is partially observable. Neighboring tiles in the maze have a similar color.



(a) **(b)**

Fig. 3: The first two principle components of the five-dimensional state representation $\hat{\mathcal{S}}$ learned by Φ_{lm64} for the (a) 1 Hz dataset and (b) 10 Hz dataset. Using the coloring scheme shown in Figure 2, we see that the state space is locally consistent and contains the structure of the maze. We also marked the representation of the sample images from Figure 2.

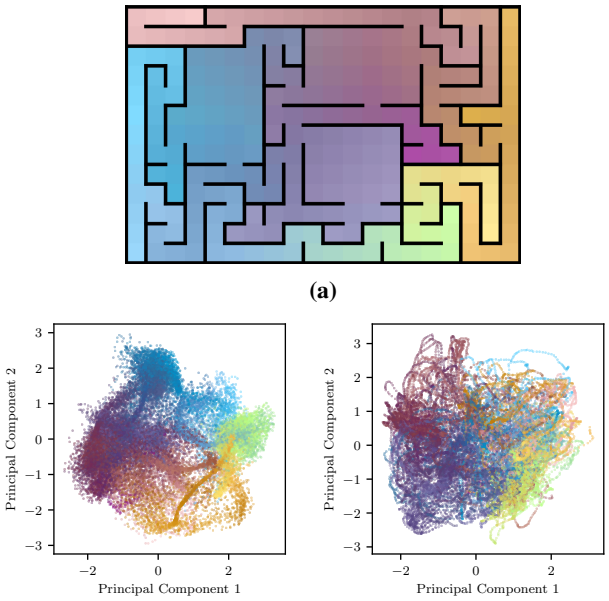
Φ_{sv}, which we used as an upper baseline for the unsupervised trained models.

V. Experiments and Validation

In this section, we show that a five-dimensional representation learned with RSRL in partially observable environments is locally consistent and connected (Section V-A), can be used to reconstruct the true state (Section V-B), and can solve a reinforcement learning task (Section V-D). Furthermore, we study the sensitivity of the priors to the number of landmarks in the learned state representation (Section V-C). At last, we also show that our learned models are robust to errors in landmark classification (Section V-E).

A. Local Consistency and Connectivity

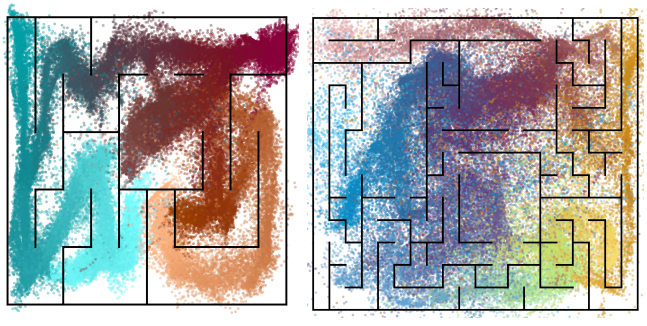
To show the local consistency of the learned state space $\hat{\mathcal{S}}$, we visualize the first two principal components of $\hat{\mathcal{S}}$. We



(a) (b) (c)

Fig. 4: The large maze with a neighboring color scheme is shown in (a). Using this color scheme, we show the principal components of the state space $\hat{\mathcal{S}}$ learned by Φ_{lm64} for the (b) 1 Hz dataset and (c) 10 Hz dataset for the large maze. We see that the sample frequency has a great impact on the consistency of the learned state space. Although the learned state space for both models is locally consistent, the structure of the maze is better represented in b) than in c). This is because, in b), a lower sampling rate means that the robot explore more parts of the maze and has a greater chance of encountering a landmark.

color the states according to their true states s. If the learned state space maps similar colored states together, then it is locally consistent. If we additionally see a smooth color transition, then the learned state space also contains the connectivity of the environment.



(a) (b)

Fig. 5: Reconstruction of the normalized position $\tilde{\mathcal{S}}$ obtained by Ψ_{ValCon} from the state space $\hat{\mathcal{S}}$ by Φ_{lm64} for the (a) short maze and (b) large maze. The states are colored according to Figure 2 and Figure 4a. For both models, Ψ_{ValCon} mapped most of the states $\hat{s}$ to the correct area. However, the reconstruction for the large maze is less accurate.

We display the first two principal components of the state space for the model Φ_{lm64}, colored with a neighboring color scheme, in Figure 3 for the small maze and in Figure 4 for the large maze. Although there are a few overlapping states, most states are surrounded by states of the same color. The structure of the maze is visible for both mazes, especially for the 1 Hz dataset. Thus, we can conclude that the learned state space $\hat{\mathcal{S}}$ is locally consistent and captures the structure of the environment.

Model	Small Maze (10×5 Tiles)			Large Maze (20×13 Tiles)		
	E_{Geod}	E_{Pos}	KNN	E_{Geod}	E_{Pos}	KNN
Φ_{lm5}	2.38	1.41	2.45	15.05	2.55	2.92
Φ_{lm16}	1.01	0.77	1.36	8.45	1.79	2.55
Φ_{lm32}	0.60	0.57	1.01	2.84	0.94	1.43
Φ_{lm64}	0.42	0.45	0.78	1.93	0.71	1.10
Φ_{sp}	0.29	0.32	0.52	**0.85**	0.35	0.53
Φ_{sv}	**0.20**	**0.02**	**0.04**	1.00	**0.03**	**0.06**

TABLE I: Tile distance (E_{Geod}), euclidean error (E_{Pos}) and KNN-Score (KNN) for all models. In each column, the best model is marked in bold. We can see that the representation improves with increasing landmarks and Φ_{sp} is always the best in the unsupervised models. For the large maze, the tile distance of Φ_{sp} is even lower than the supervised model Φ_{sv}.

B. Reconstructing the true state

We can investigate whether the learned state space can reconstruct the true states by analyzing the reconstructed true state $\tilde{s}$ obtained by Ψ_{ValCon}. This tells us to what extent the learned state space $\hat{\mathcal{S}}$ can reconstruct the true state space $\mathcal{S}$.

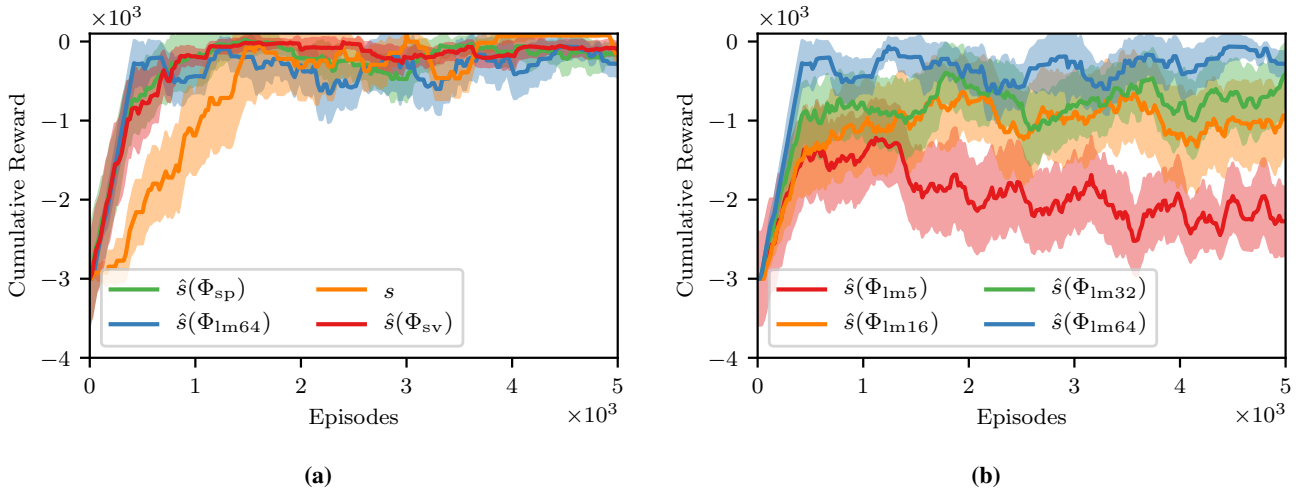
Figure 5 shows the reconstructed normalized position $\tilde{s}$ obtained by Ψ_{ValCon} from the states $\hat{s}$ generated with Φ_{lm64} for the small (left) and large (right) maze. In terms of comparing the reconstructed state with the errors, we can clearly see the effect of a higher euclidean error for the large maze. The reconstructed states of the large maze are more widely distributed and the reconstruction is less accurate than of the small maze. This is to be expected since we are using the same number of landmarks, 64, for both mazes, even though the size of the large maze is five times the size of the small maze. However, even when the exact positional information was not learned in the large maze, we still see most of the structure being extracted.

We conclude that our unsupervised models can adequately reconstruct the true state using the validation networks.

C. Influence of the number of landmarks on RSRL

In this section, we want to investigate the sensitivity of our priors to the number of landmarks. Therefore, we quantitatively compare the performance of our models to the supervised baseline Φ_{sv} using the errors introduced in Section III-E.

An overview of our results can be seen in Table I. As expected, increasing the number of landmarks improve the performance as there are more landmarks in different parts of the maze which allows the priors to connect more number of similar states. Thus, the model Φ_{sp} outperformed all models trained using the *landmark prior* and the difference increased with increasing maze size. This is expected since the Φ_{sp} implicitly assumes a landmark at every pair of close



(a) **(b)**

Fig. 6: Reinforcement learning performance over the number of training episodes using representations learned by different models. We write $\hat{s}(\Phi_{\text{sp}})$ when the RL agent observed states $\hat{s}$ learned by Φ_{sp}. In (a), we compare the policy learned on the top three trained models with a policy trained on the true state s. We can see that the policies trained on our losses perform as well as the policy trained on the true state. In (b), we compare the RL performance of Φ_{lm} with different number of landmarks. We see the that the performance of the policy improves with increasing number of landmarks. For each model, we display the mean and standard deviation of four reinforcement learning runs with a sliding window of 20 measurements.

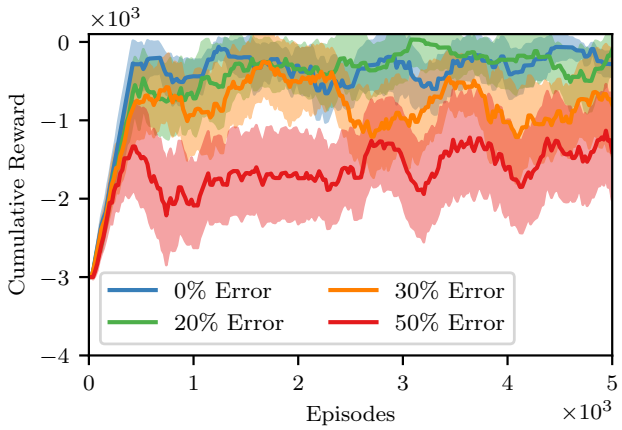


Fig. 7: The performance of an RL agent that used the state representation $\hat{s}(\Phi_{\text{lm64}})$ learned with different errors in landmark classification. We see that the performance remains same untill 20% error, after which it starts to drop.

states. Surprisingly, on the large maze, the Φ_{sp} model even outperformed the baseline Φ_{sv} in terms of the tile distance. This indicates, that in comparison to $\mathcal{S}$, the learned state space $\hat{\mathcal{S}}$ stores the structure in a way that can be better interpreted by a validation network to extract the structure of the maze. This is because points either side of a wall in the true state space can be mapped far away in the learned state space which may make the extraction of the structure of the maze easier with the learned state space.

D. Reinforcement Learning

In the previous sections, we have already shown that our learned state representation is low-dimensional and the local consistency and the structure of the environment is preserved. In this section, we want to show that our learned state representation is close to Markovian by solving a task using the learned states with RL. We used the Deep Double Q-Learning algorithm [24] for our approach. Since Q-learning requires a Markovian state space, if we are able to solve the RL task with Q-Learning, it would show that our learned state space is indeed close to Markovian.

The objective of the RL agent was to solve a navigation task in the small maze. The agent had to reach a fixed goal position from a random starting position. It was given a reward of $+100$ if it reached the goal and a time penalty of -1 for every time step, in addition to a shaping reward of $+10$ if it moved to a tile of the maze that was closer to the goal and -10 if moved to a tile farther away from the goal. Each episode ends after 3000 steps or when the goal is reached.

We trained our policy with the learned states of different models and tested the performance of these policies with a policy trained directly with the true state. In Figure 6a, we can see that the policies trained on the learned states of Φ_{sp} and Φ_{lm64} perform equally good as the policy trained on the true state in terms of cumulative reward and episodes required to reach a good policy. We can clearly see from Figure 6b that the performance improves as the number of landmarks increases. The difference in policy also correlates well to the difference in the euclidean error.

Since Q-learning requires an Markovian state, the results show that the priors are indeed successful in extracting an approximate Markovian state space that can be used for reinforcement learning.

E. Robustness to Landmark Misclassification

Our implementation of the *landmark prior* assumes that we can perfectly identify which states are close to a landmark

in the true state space $\mathcal{S}$. But this is impractical in the real world. To test the robustness of the models against incorrectly classified close states, we use a certain percentage of random pairs of states when computing the closest states in a batch. By increasing the percentage of incorrectly identified close states incrementally, we can see at which point the learned states fail to solve the RL task.

Figure 7 shows how the performance of Φ_{lm64} changes as we increase the percentage of incorrectly classified pairs of states. The performance of the model remains almost the same until 20% classification error and then slowly starts to decrease, which shows that the learned models are robust to misclassification errors.

VI. Conclusion

We introduced *Recurrent State Representation Learning* to solve the problem of state representation learning for partially observable environments with robotic priors. To handle partial observability, we used an LSTM network to maintain a memory of past observations. We used robotic priors to impose constraints for learning task relevant representations. We introduced new variants of the *reference point prior—landmark* and *similar point*—to complement existing robotic priors —*temporal coherence*, *variation* and *proportionality*—and showed how they can be used to learn low-dimensional representations useful for learning.

We also introduced validation networks to provide better quantitative and qualitative metrics to evaluate the learned models. Using these validation networks and reinforcement learning, we showed the learned representations to be low-dimensional, locally consistent, encoding the structure of the environment and the true state space, and close to Markovian. We analyzed the effect of the different number of landmarks on the learned representations. We also validated the robustness of the models to errors in landmark classification.

Even though we use supervision in our implementation of the *landmark prior*, we believe that the robots now are, in fact, capable of recognizing landmarks in their environment without having knowledge of their true state. Thus, our approach can be applied in real-world scenarios.

References

[1] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," *arXiv:1509.06113*, 2015.

[2] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4, pp. 705–724.

[3] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Integrating state representation learning into deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1394–1401, 2018.

[4] J. Munk, J. Kober, and R. Babuška, "Learning state representation for deep actor-critic control," in *IEEE Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 4667–4673.

[5] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, no. 3, pp. 407–428, 2015.

[6] R. Jonschkowski, R. Hafner, J. Scholz, and M. Riedmiller, "PVEs: Position-velocity encoders for unsupervised learning of structured state representations," *arXiv:1705.09805*, 2017.

[7] T. Lesort, M. Seurin, X. Li, N. D. Rodríguez, and D. Filliat, "Unsupervised state representation learning with robotic priors: a robustness benchmark," *arXiv preprint arXiv:1709.05185*, 2017.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv:1611.05397*, 2016.

[10] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, "Learning to navigate in complex environments," *arXiv:1611.03673*, 2016.

[11] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[12] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–8.

[13] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3928–3934.

[14] B. Boots, G. Gordon, and A. Gretton, "Hilbert space embeddings of predictive state representations," *arXiv:1309.6819*.

[15] W. Hamilton, M. M. Fard, and J. Pineau, "Efficient learning and planning with compressed predictive states," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3395–3439.

[16] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, vol. 355, no. 6356, p. 161, 1992.

[17] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[19] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic visual localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6896–6906.

[20] M. Mata, J. M. Armingol, A. de la Escalera, and M. A. Salichs, "A visual landmark recognition system for topological navigation of mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2001, pp. 1124–1129.

[21] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven, "Tour the world: building a web-scale landmark recognition engine," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1085–1092.

[22] J. Cao, Y. Zhao, X. Lai, M. E. H. Ong, C. Yin, Z. X. Koh, and N. Liu, "Landmark recognition with sparse representation classification and extreme learning machine," *Journal of the Franklin Institute*, vol. 352, no. 10, pp. 4528–4545, 2015.

[23] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik *et al.*, "Deepmind lab," *arXiv:1612.03801*, 2016.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI Conference on Artificial Intelligence*, vol. 16, 2016, pp. 2094–2100.

ACKNOWLEDGMENT

We gratefully acknowledge financial support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number 329426068 (Robotics-Specific Machine Learning) and under Germany's Excellence Strategy EXC 2002/1 project number 390523135 (Science of Intelligence). We thank Philippe Giguère for his valuable feedback on the paper.