Reachability-Aware Laplacian Representation in Reinforcement Learning

Kaixin Wang [1 *] **Kuangqi Zhou** [2 *] **Jiashi Feng** [3] **Bryan Hooi** [1 4] **Xinchao Wang** [1 2]

Abstract

In Reinforcement Learning (RL), Laplacian Representation (LapRep) is a task-agnostic state representation that encodes the geometry of the environment. A desirable property of LapRep stated in prior works is that the Euclidean distance in the LapRep space roughly reflects the reachability between states, which motivates the usage of this distance for reward shaping. However, we find that LapRep does not necessarily have this property in general: two states having a small distance under LapRep can actually be far away in the environment. Such a mismatch would impede the learning process in reward shaping. To fix this issue, we introduce a Reachability-Aware Laplacian Representation (RA-LapRep), by properly scaling each dimension of LapRep. Despite the simplicity, we demonstrate that RA-LapRep can better capture the inter-state reachability as compared to LapRep, through both theoretical explanations and experimental results. Additionally, we show that this improvement yields a significant boost in reward shaping performance and benefits bottleneck state discovery.

1. Introduction

Reinforcement Learning (RL) seeks to learn a decision strategy that advises the agent on how to take actions according to the perceived states (Sutton & Barto, 2018). The state representation plays an important role in the agent's learning process — a proper choice of the state representation can help improve generalization (Zhang et al., 2018; Stooke et al., 2020; Agarwal et al., 2021), encourage exploration (Pathak et al., 2017; Machado et al., 2017; 2020) and enhance learning efficiency (Dubey et al., 2018; Wu et al., 2019; Wang et al., 2021). In studying the state representation, one direction of particular interest is to learn a task-agnostic representation that encodes transition dynamics of the environment (Mahadevan & Maggioni, 2007; Machado et al., 2021).

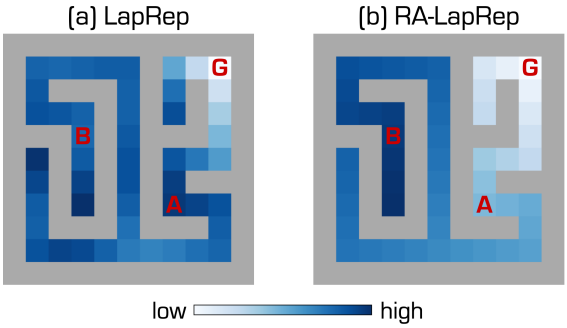

Figure 1. Euclidean distances between each state and the goal state G, under LapRep **(a)** and RA-LapRep **(b)**.

Along this line, the Laplacian Representation (LapRep) has received increasing attention (Mahadevan, 2005; Machado et al., 2017; Wu et al., 2019; Wang et al., 2021; Erraqabi et al., 2022). Specifically, the LapRep is formed by the d smallest eigenvectors of the Laplacian matrix of the graph induced from the transition dynamic (see Section 2.2 for definition). It is assumed in prior works (Wu et al., 2019; Wang et al., 2021) that LapRep has a desirable property: the Euclidean distance in the LapRep space roughly reflects the *reachability* among states (Wu et al., 2019; Wang et al., 2021), *i.e.*, smaller distance implies that it is easier to reach one state from another. This motivates the usage of the Euclidean distance under LapRep for reward shaping (Wu et al., 2019; Wang et al., 2021).

However, there is a lack of formal justification in previous works (Wu et al., 2019; Wang et al., 2021) for this property. In fact, it turns out that the Euclidean distance under LapRep does *not* correctly capture the inter-state reachability in general. Figure 1 (a) shows an example. Under LapRep, a state that has a larger distance (*e.g.*, A) might actually be closer to the goal than another state (*e.g.*, B). Consequently, when the agent moves towards the goal, the pseudo-reward

[*]Equal contribution [1]Institute of Data Science, National University of Singapore, Singapore [2]Department of Electrical and Computer Engineering, National University of Singapore, Singapore [3]ByteDance, Singapore [4]School of Computing, National University of Singapore, Singapore. Correspondence to: Kaixin Wang, Xinchao Wang <kaixin96.wang@gmail.com, xinchao@nus.edu.sg>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

provided by LapRep would give a wrong learning signal. Such a mismatch would hinder the learning process with reward shaping and result in inferior performance.

In this work, we introduce a Reachability-Aware Laplacian Representation (RA-LapRep) that reliably captures the inter-state distances in the environment geometry (see Figure 1 (b)). Specifically, RA-LapRep is obtained by scaling each dimension of the LapRep by the inverse square root of the corresponding eigenvalue. Despite its simplicity, RA-LapRep has theoretically justified advantages over LapRep in the following two aspects. First, the Euclidean distance under RA-LapRep can be interpreted as the *average commute time*, which measures the expected number of steps required in a random walk to navigate between two states. Thus, such distance provides a good measure of reachability. In contrast, to our best knowledge, there lacks a connection between the Euclidean distance under LapRep and the reachability. Second, RA-LapRep is equivalent to the embedding computed by the classic multidimensional scaling (MDS) (Borg & Groenen, 2005), which preserves pairwise distances globally (Tenenbaum et al., 2000). LapRep, on the other hand, preserves only local information (*i.e.*, mapping neighboring states close), since it is essentially Laplacian eigenmap (Belkin & Niyogi, 2003). Thus, LapRep is inherently incompetent for measuring the inter-state reachability.

To further validate the advantages of RA-LapRep over LapRep, we conduct experiments to compare them on two discrete gridworld and two continuous control environments. The results show that RA-LapRep indeed performs much better in capturing the inter-state reachability as compared to LapRep. Furthermore, when used for reward shaping in the goal-reaching tasks, RA-LapRep significantly outperforms LapRep. In addition, we show that RA-LapRep can be used to discover the bottleneck states based on graph centrality measure, and more accurately find the key states than LapRep.

2. Background

Notations. We use boldface letters (*e.g.*, $\mathbf{u}$) for vectors, and calligraphic letters (*e.g.*, $\mathcal{U}$) for sets. For a vector $\mathbf{u}$, $\|\mathbf{u}\|$ denotes its L_2 norm, $\mathrm{diag}(\mathbf{u})$ denotes a diagonal matrix whose main diagonal is $\mathbf{u}$. We use $\mathbf{1}$ to denote an all-ones vector, whose dimension can be inferred from the context.

2.1. Reinforcement Learning

In the Reinforcement Learning (RL) framework (Sutton & Barto, 2018), an agent aims to learn a strategy that advises how to take actions in each state, with the goal of maximizing the expected cumulative reward. We consider the standard Markov Decision Process (MDP) setting (Puterman, 1990), and describe an MDP with a sextuple $(\mathcal{S}, \mathcal{A}, r, P, \gamma, \mu)$. $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The initial state s_0 is generated according to the distribution $\mu \in \Delta_{\mathcal{S}}$, where $\Delta_{\mathcal{S}}$ denotes the space of probability distributions over $\mathcal{S}$. At timestep t, the agent observes from the environment a state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$. Then the environment provides the agent with a reward signal $r(s_t, a_t) \in \mathbb{R}$. The state observation in the next timestep $s_{t+1} \in \mathcal{S}$ is sampled from the distribution $P(s_t, a_t) \in \Delta_{\mathcal{S}}$. We refer to $P \in (\Delta_{\mathcal{S}})^{\mathcal{S}\times\mathcal{A}}$ as the transition functions and $r : \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ as the reward function. A stationary stochastic policy $\pi \in (\Delta_{\mathcal{A}})^{\mathcal{S}}$ specifies a decision making strategy, where $\pi(s, a)$ is the probability of taking action a in state s. The agent's goal is to learn an optimal policy π^* that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_{\pi\in\Pi} \mathbb{E}_{\pi,P} \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1}$$

where Π denotes the policy space and $\gamma \in [0, 1)$ is the discount factor.

2.2. Laplacian representation in RL

The Laplacian Representation (LapRep) (Wu et al., 2019) is a task-agnostic state representation in RL, originally proposed in (Mahadevan, 2005) (known as the proto-value function). Formally, the Laplacian representations for all states are the eigenfunctions of Laplace-Beltrami diffusion operator on the state space manifold. For simplicity, here we restrict the introduction of LapRep to the discrete state case and refer readers to (Wu et al., 2019) for the formulation in the continuous case.

The states and transitions in an MDP can be viewed as nodes and edges in a graph. The LapRep is formed by d smallest eigenvectors of the graph Laplacian (usually $d \ll |\mathcal{S}|$). Each eigenvector (of length $|\mathcal{S}|$) corresponds to a dimension of the LapRep. Formally, we denote the graph as $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ where $\mathcal{S}$ is the node set consisting of all states and $\mathcal{E}$ is the edge set consisting of transitions between states. The Laplacian matrix of graph $\mathcal{G}$ is defined as $L \coloneqq D - A$, where A is the adjacency matrix of $\mathcal{G}$ and $D \coloneqq \mathrm{diag}(A\mathbf{1})$ is the degree matrix (Chung & Graham, 1997). We sort the eigenvalues of L by their magnitudes and denote the i-th smallest one as λ_i. The unit eigenvector corresponding to λ_i is denoted as $\mathbf{v}_i \in \mathbb{R}^{|\mathcal{S}|}$. Then, the d-dimensional LapRep of a state s can be defined as

$$\boldsymbol{\rho}_d(s) \coloneqq (\mathbf{v}_1[s], \mathbf{v}_2[s], \cdots, \mathbf{v}_d[s]), \tag{2}$$

where $\mathbf{v}_i[s]$ denotes the entry in vector $\mathbf{v}_i$ corresponding to state s. In particular, $\mathbf{v}_1$ is a normalized all-ones vector and hence it provides no information about the environment geometry. Therefore we omit $\mathbf{v}_1$ and only consider other dimensions.

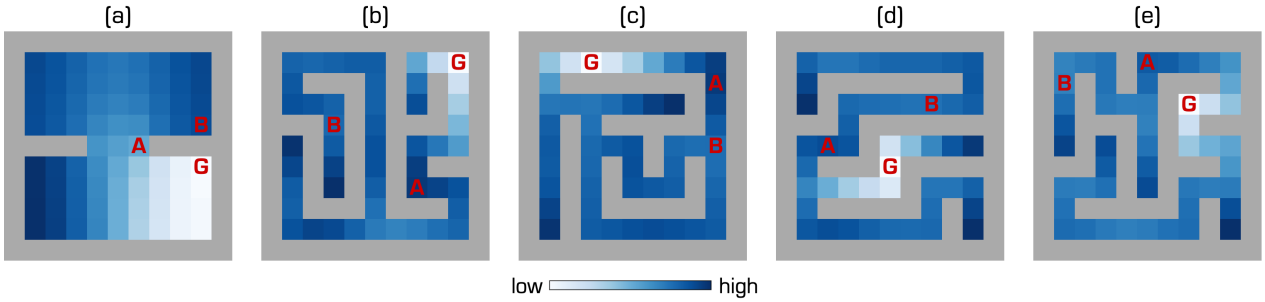

Figure 2. Visualizations of the Euclidean distances under LapRep between all states and the state G.

For environments with a large or even continuous state space, it is infeasible to obtain the LapRep by directly computing the eigendecomposition. To approximate LapRep with neural networks, previous works (Wu et al., 2019; Wang et al., 2021) propose sample-based methods based on the spectral graph drawing (Koren, 2005). In particular, Wang et al. (2021) introduce a generalized graph drawing objective that ensures dimension-wise faithful approximation to the ground truth $\boldsymbol{\rho}_d(s)$.

3. Method

3.1. Reachability-aware Laplacian representation

In prior works (Wu et al., 2019; Wang et al., 2021), LapRep is believed to have a desirable property that the Euclidean distance between two states under LapRep (*i.e.*, $\mathrm{dist}_{\boldsymbol{\rho}}(s, s') := \|\boldsymbol{\rho}_d(s) - \boldsymbol{\rho}_d(s')\|$) roughly reflects the reachability between s and s'. That is, a smaller distance between two states implies that it is easier for the agent to reach one state from the other. Figure 2 (a) shows an illustrative example similar to the one in (Wu et al., 2019). In this example, $\mathrm{dist}_{\boldsymbol{\rho}}(\mathtt{A}, \mathtt{G})$ is smaller than $\mathrm{dist}_{\boldsymbol{\rho}}(\mathtt{B}, \mathtt{G})$, which aligns with the intuition that moving to G from A takes fewer steps than from B. Motivated by this, LapRep is used for reward shaping in goal-reaching tasks (Wu et al., 2019; Wang et al., 2021).

However, little justification is provided in previous works (Wu et al., 2019; Wang et al., 2021) for this argument (*i.e.*, the Euclidean distance under LapRep captures the inter-state reachability). In fact, we find that it does not hold in general. As shown in Figure 2 (b-e), $\mathrm{dist}_{\boldsymbol{\rho}}(\mathtt{A}, \mathtt{G})$ is larger than $\mathrm{dist}_{\boldsymbol{\rho}}(\mathtt{B}, \mathtt{G})$, but A is clearly closer to G than B. As a result, when the agent moves towards the goal, $\mathrm{dist}_{\boldsymbol{\rho}}$ might give a wrong reward signal. Such a mismatch hinders the policy learning process when we use this distance for reward shaping.

In this paper, we introduce the following Reachability-Aware Laplacian Representation (RA-LapRep):

$$\boldsymbol{\phi}_d(s) := \left(\frac{\mathbf{v}_2[s]}{\sqrt{\lambda_2}}, \frac{\mathbf{v}_3[s]}{\sqrt{\lambda_3}}, \cdots, \frac{\mathbf{v}_d[s]}{\sqrt{\lambda_d}} \right), \tag{3}$$

which can fix the issue of LapRep and better capture the reachability between states. We provide both theoretical explanation (Section 3.2) and empirical results (Section 4) to demonstrate the advantage of RA-LapRep over LapRep.

3.2. Why RA-LapRep is more desirable than LapRep?

In this subsection, we provide theoretical groundings for RA-LapRep from two aspects, which explains why it better captures the inter-state reachability than LapRep.

First, we find that the Euclidean distance under RA-LapRep is related to a quantity that measures the expected random walk steps between states. Specifically, let $\mathrm{dist}_{\boldsymbol{\phi}}(s, s') := \|\boldsymbol{\phi}_d(s) - \boldsymbol{\phi}_d(s')\|$ denote the Euclidean distance between states s and s' under RA-LapRep. When $d = |\mathcal{S}|$, $\mathrm{dist}_{\boldsymbol{\phi}}(s, s')$ has a nice interpretation (Fouss et al., 2007): it is proportional to the square root of the average commute time between states s and s', *i.e.*,

$$\mathrm{dist}_{\boldsymbol{\phi}}(s, s') \propto \sqrt{n(s, s')}. \tag{4}$$

Here the average commute time $n(s, s')$ measures the expected number of steps required in a random walk to navigate from s to s' and back (see Appendix A for the formal definition). Thus, $\mathrm{dist}_{\boldsymbol{\phi}}(s, s')$ provides a good quantification of the concept of reachability. Additionally, with the proportionality in Eqn. (4), RA-LapRep can be used to discover bottleneck states (see Section 4.3 for a detailed discussion and experiments). In contrast, to the best of our knowledge, the Euclidean distance under LapRep (*i.e.*,$\mathrm{dist}_{\boldsymbol{\rho}}(s, s')$) does not have a similar interpretation that matches the concept of reachability.

Second, we show that RA-LapRep preserves global information while LapRep only focuses on preserving local information. Specifically, we note that RA-LapRep is equivalent (up to a constant factor) to the embedding obtained by classic Multi-Dimensional Scaling (MDS) (Borg & Groenen, 2005) with the squared distance matrix in MDS being $D^{(2)}_{ij} = n(i, j)$ (Fouss et al., 2007) (see Appendix A.2 for a detailed derivation). Since classic MDS is known to preserve pairwise distances globally (Tenenbaum et al., 2000), the Euclidean distance under RA-LapRep is then a good

fit for measuring the inter-state reachability. In comparison, the LapRep is only able to preserve local information. This is because, when viewing the MDP transition dynamic as a graph, the LapRep is essentially the Laplacian eigenmap (Belkin & Niyogi, 2003). As discussed in (Belkin & Niyogi, 2003), the Laplacian eigenmap only aims to preserve local graph structure for every single neighborhood in the graph (*i.e.*, mapping neighboring states close), while making no attempt in preserving global information about the whole graph (*e.g.*, pairwise geodesic distances between nodes (Tenenbaum et al., 2000)). Therefore, the Euclidean distance under LapRep is inherently not intended for measuring the reachability between states, especially for distant states.

3.3. Approximating RA-LapRep

We note that the theoretical reasonings in the above subsection are based on $d = |\mathcal{S}|$. In practice, however, for environments with a large or even continuous state space, it is infeasible to have $d = |\mathcal{S}|$ and hence we need to take a small d. One may argue that, using a small d would lead to approximation error: when $d < |\mathcal{S}|$, the distance $\mathrm{dist}_{\boldsymbol{\phi}}(s, s')$ is not exactly proportional to $\sqrt{n(s, s')}$. Fortunately, the gap between the approximated $\tilde{n}(s, s')$ and the true $n(s, s')$ turns out to be upper bounded by $C \sum_{i=d+1}^{|\mathcal{S}|} \frac{1}{\lambda_i}$, where C is a constant and the summation is over the $|\mathcal{S}| - d$ largest eigenvalues. Thus, this bound will not be very large. We will empirically show in Section 4.2 that a small d is sufficient for good reward shaping performance and further increasing d does not yield any noticeable improvement.

Furthermore, even with a small d, it is still impractical to obtain RA-LapRep via directly computing eigendecomposition. To tackle this, we follow (Wang et al., 2021) to approximate RA-LapRep with neural networks using sample-based methods. Specifically, we first learn a parameterized approximation $f_i(\cdot\,;\theta)$ for each eigenvector $\mathbf{v}_i$ by optimizing a generalized graph drawing objective (Wang et al., 2021), *i.e.*, $f_i(s\,;\theta) \approx \mathbf{v}_i[s]$, where θ denotes the learnable parameters of the neural networks. Next, we approximate each eigenvalue λ_i simply by

$$\lambda_i = \mathbf{v}_i^\top L \mathbf{v}_i \approx \mathbb{E}_{(s,s')\in\mathcal{T}} \left(f_i(s\,;\hat{\theta}) - f_i(s'\,;\hat{\theta}) \right)^2, \quad (5)$$

where $\hat{\theta}$ denotes the learned parameters, and $\mathcal{T}$ is the same transition data used to train f. Let $\tilde{\lambda}_i$ denote the approximated eigenvalue. RA-LapRep can be approximated by

$$\boldsymbol{\phi}_d(s) \approx \tilde{\boldsymbol{\phi}}_d(s) = \left(\frac{f_2(s\,;\hat{\theta})}{\sqrt{\tilde{\lambda}_2}}, \frac{f_3(s\,;\hat{\theta})}{\sqrt{\tilde{\lambda}_3}}, \cdots, \frac{f_d(s\,;\hat{\theta})}{\sqrt{\tilde{\lambda}_d}} \right). \quad (6)$$

In experiments, we find this approximation works quite well and is on par with using the true $\boldsymbol{\phi}_d(s)$.

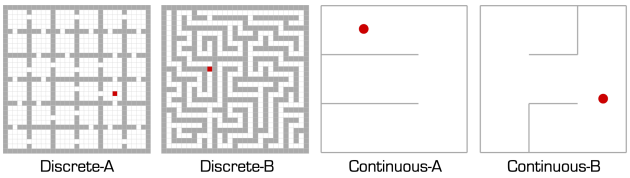

Figure 3. Environments used in our experiments (agents shown in red, and walls in grey).

4. Experiments

In this section, we conduct experiments to validate the benefits of RA-LapRep compared to LapRep. Following (Wu et al., 2019; Wang et al., 2021), we consider both discrete gridworld and continuous control environments in our experiments. Figure 3 shows the layouts of environments used. We briefly introduce them here and refer readers to Appendix B for more details. In discrete gridworld environments, the agent takes one of the four actions (*up*, *down*, *left*, *right*) to move from one cell to another. If hitting the wall, the agent remains in the current cell. In continuous control environments, the agent picks a continuous action from $[-\pi, \pi)$ that specifies the direction along which the agent moves a fixed small step forward. For all environments, the observation is the agent's (x, y)-position. We use $d = 10$ as default for our experiments and include an analysis on varying d in Section 4.2.

4.1. Capturing reachability between states

In this subsection, we evaluate the learned RA-LapRep and LapRep in capturing the reachability among states. We also include the ground-truth RA-LapRep for comparison. Specifically, for each state s, we compute the Euclidean distance between s and an arbitrarily chosen goal state s_{goal}, under learned LapRep $\tilde{\boldsymbol{\rho}}$, learned RA-LapRep $\tilde{\boldsymbol{\phi}}$, and ground-truth RA-LapRep $\boldsymbol{\phi}$ (*i.e.*, $\mathrm{dist}_{\tilde{\boldsymbol{\phi}}}(s, s_{\text{goal}})$, $\mathrm{dist}_{\tilde{\boldsymbol{\rho}}}(s, s_{\text{goal}})$ and $\mathrm{dist}_{\boldsymbol{\phi}}(s, s_{\text{goal}})$). Then, we use heatmaps to visualize the three distances. For continuous environments, the heatmaps are obtained by first sampling a set of states roughly covering the state space, and then performing interpolation among sampled states.

We train neural networks to learn RA-LapRep and LapRep. Specifically, we implement the two-step approximation procedure introduced in Section 3.3 to learn RA-LapRep, and adopt the method in (Wang et al., 2021) to learn LapRep. Details of training and network architectures are in Appendix C. The ground truth RA-LapRep $\boldsymbol{\phi}$ is calculated using Eqn. (3). For discrete environments, the eigenvectors and eigenvalues are computed by eigendecomposition; For continuous environments, the eigenfunctions and eigenvalues are approximated by the finite difference method with 5-point stencil (Peter & Lutz, 2003).

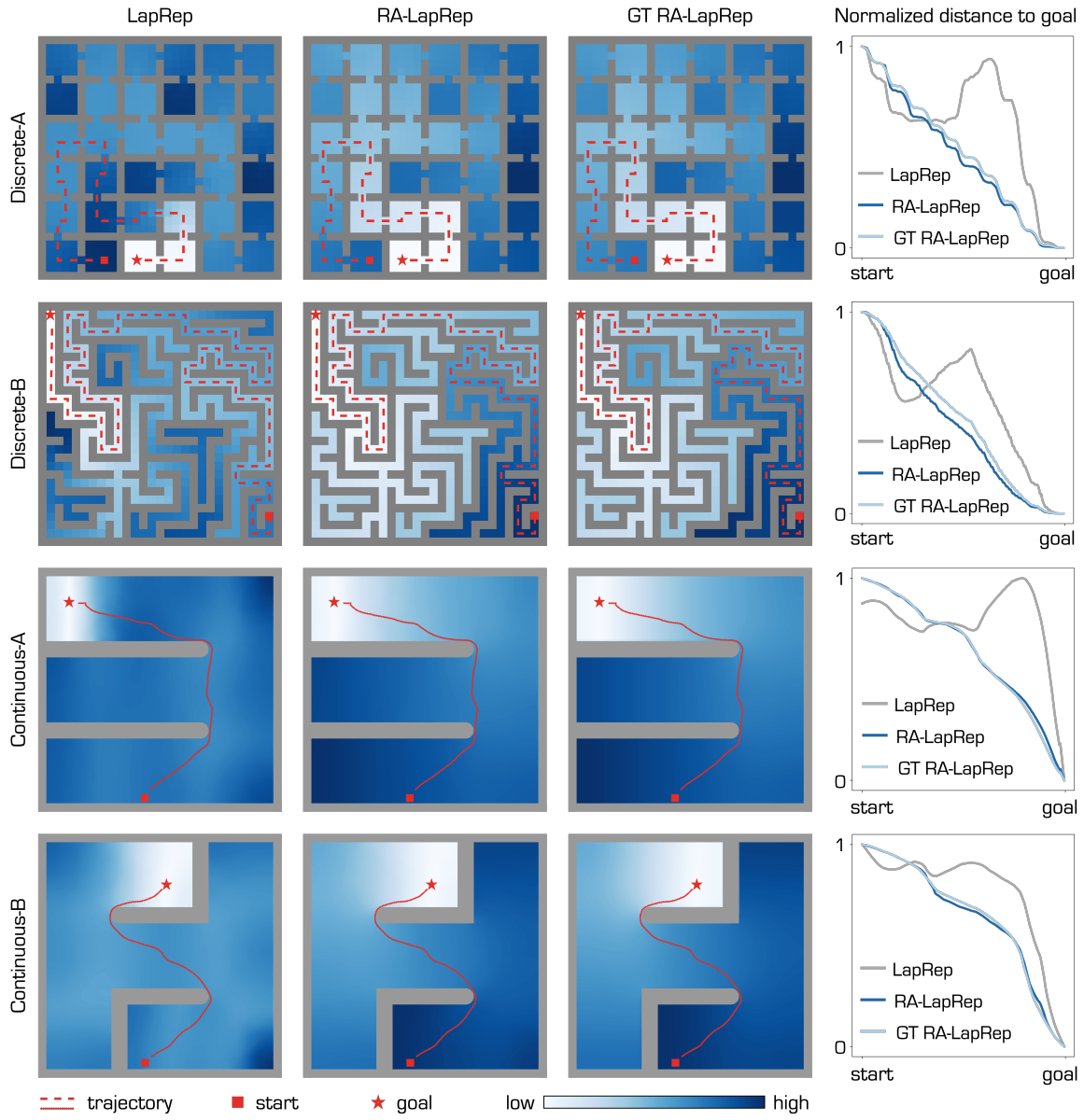

Figure 4. **Left 3 columns**: Visualizations of the Euclidean distances between all states and the goals in the four environments, under learned LapRep, learned RA-LapRep, and ground-truth RA-LapRep. Example trajectories are shown in red. **Right**: Line charts of the distance values for states in the trajectories (normalized to $[0, 1]$), where the states are sorted by temporal order.

The visualization results are shown in Figure 4. For clearer comparison, we highlight in each environment an example trajectory, and plot the distance values along each trajectory in the line charts on the right. As we can see, for both discrete and continuous environments, as the agent is moving towards the goal, the distances under the learned LapRep are increasing in some segments of the trajectories. This is contradictory to the belief that the Euclidean distances under LapRep reflect the inter-state reachability. In contrast, the distances under RA-LapRep decrease monotonically, which accurately reflects the reachability between current states and the goals. Apart from the highlighted trajectories, similar observations can be obtained for other trajectories or goal positions (see Appendix D.1). Besides, the distances under the learned RA-LapRep are very close to those under the ground-truth one, indicating the effectiveness of our approximation approach. To see how much error truncating the dimension (*i.e.*, using $d = 10$ instead of the full dimension) incurs, we plot the distances calculated using all dimensions of the ground-truth RA-LapRep for the two discrete environments. As shown in Figure 12, the error caused by the truncation is small, indicating that the upper

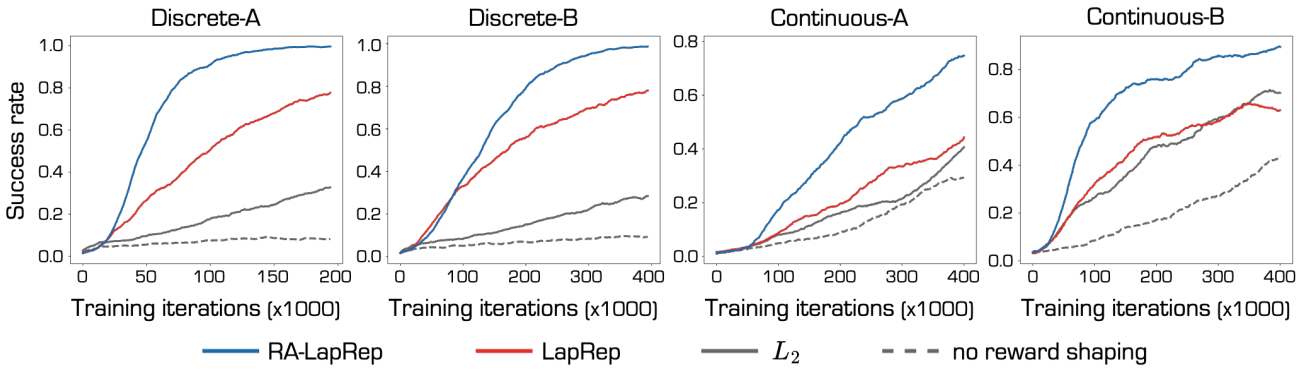

Figure 5. Reward shaping results in goal-reaching tasks, with different choices of the shaped reward.

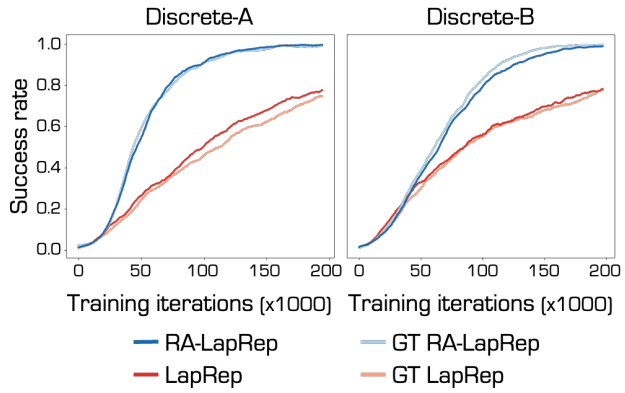

Figure 6. Comparison in the reward shaping results among the learned representations and the ground-truth ones.

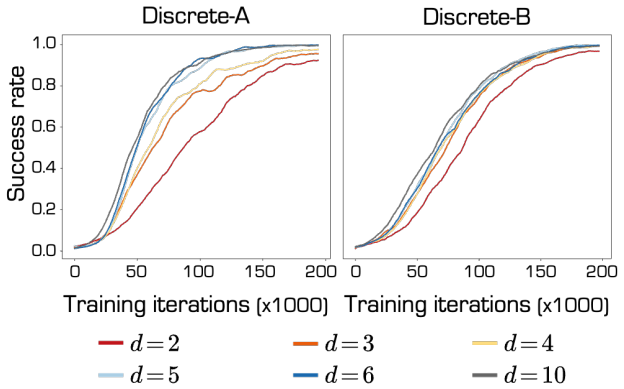

Figure 7. Comparison in the reward shaping results among using different d for the learned RA-LapRep.

bound of the gap is pretty tight.

4.2. Reward shaping

The above experiments show that the RA-LapRep better captures the reachability between states than the LapRep. Next, we study if this advantage leads to higher performance for reward shaping in goal-reaching tasks.

Following (Wu et al., 2019; Wang et al., 2021), we define the shaped reward as

$$r_t = 0.5 \cdot r_t^{\text{env}} + 0.5 \cdot r_t^{\text{dist}}. \tag{7}$$

Here r_t^{env} is the reward obtained from the environment, which is set to 0 when the agent reaches the goal state and -1 otherwise. For discrete environments, r_t^{env} is simply formalized as $r_t^{\text{env}} = -\mathbb{1}[s_{t+1} \neq s_{\text{goal}}]$. For continuous environments, we consider the agent to have reached the goal when its distance to the goal is within a small preset radius ϵ, *i.e.*, $r_t^{\text{env}} = -\mathbb{1}[\|s_{t+1} - s_{\text{goal}}\| > \epsilon]$. The pseudo-reward r_t^{dist} is the negative distance under the learned representations:

$$\begin{aligned} r_t^{\text{dist}} &= -\text{dist}_{\tilde{\boldsymbol{\rho}}}(s_{t+1}, s_{\text{goal}}) \quad \text{for LapRep}, \\ r_t^{\text{dist}} &= -\text{dist}_{\tilde{\boldsymbol{\phi}}}(s_{t+1}, s_{\text{goal}}) \quad \text{for RA-LapRep}. \end{aligned} \tag{8}$$

As in (Wu et al., 2019; Wang et al., 2021), we also include two baselines: L_2 shaping, *i.e.*, $r_t^{\text{dist}} = -\|s_{t+1} - s_{\text{goal}}\|$, and no reward shaping, *i.e.*, $r_t = r_t^{\text{env}}$. Following (Wang et al., 2021), we consider multiple goal positions for each environment (see Appendix C), in order to minimize the bias brought by goal positions. The final results are averaged across different goals and 10 runs per goal.

As shown in Figure 5, on both discrete and continuous environments, RA-LapRep outperforms LapRep and the other two baselines by a large margin. Compared to LapRep, using RA-LapRep for reward shaping is more sample efficient, reaching the same level of performance in fewer than half of the steps. We attribute this performance improvement to the fact that the Euclidean distance under RA-LapRep more accurately captures the inter-state reachability.

Comparing to the ground-truth To find out if the neural network approximation limits the performance, we also use the ground-truth RA-LapRep and LapRep for reward shaping, and compare the results with the learned representations. From Figure 6, we can see that the performance of the learned RA-LapRep is as good as the ground-truth one, indicating the effectiveness of neural network approxi-

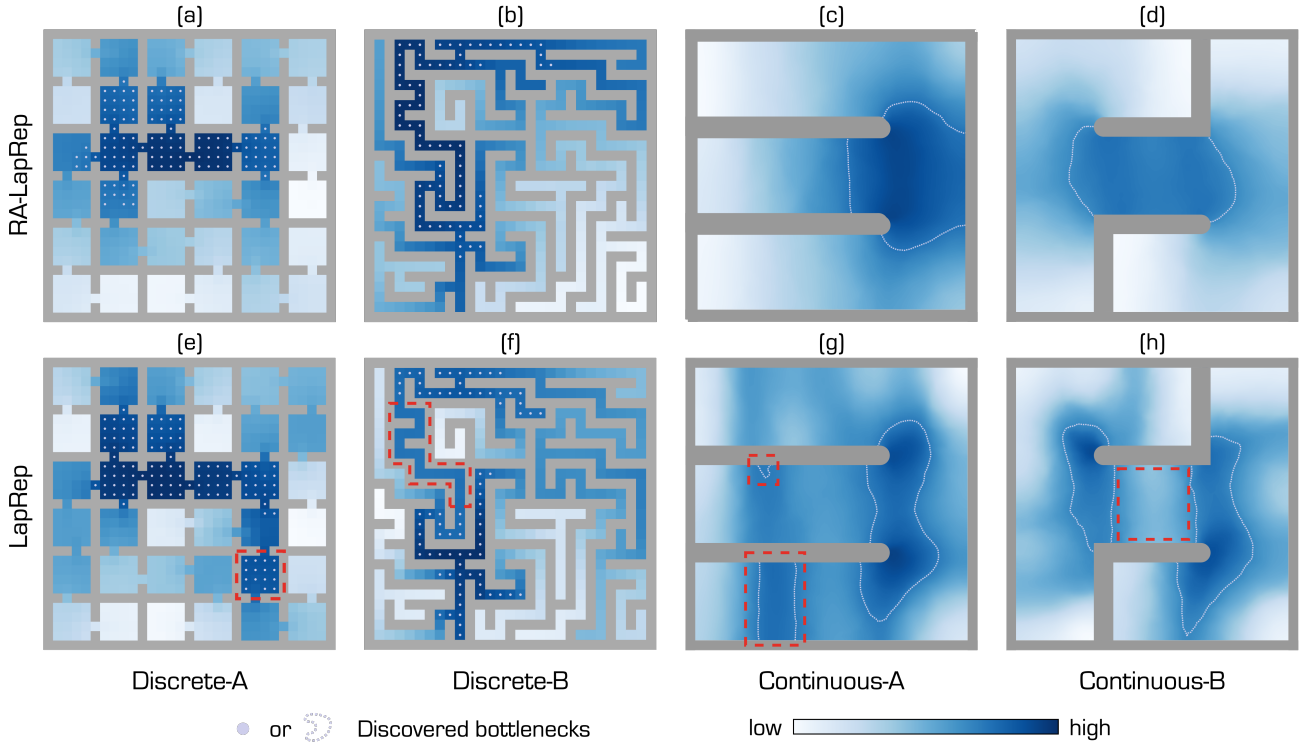

Figure 8. Bottleneck discovery results of the learned RA-LapRep and LapRep. Discovered bottleneck states are those marked as dots (for discrete environments) or those within the contour lines (for continuous environments).

mation. Besides, the learned LapRep performs comparably to the ground-truth one, suggesting that the inferior performance of LapRep is not due to poor approximation.

Varying the dimension d As mentioned in Section 3.3, the theoretical approximation error of using a smaller d is not very large. Here we conduct experiments to see if this error causes significant performance degradation. Specifically, we vary the dimension d from 2 to 10, and compare the resulting reward shaping performance. As Figure 7 shows, the performance first improves as we increase d, and then plateaus. Thus, a small d (*e.g.*, $d = 10$, compared to $|\mathcal{S}| > 300$) is sufficient to give a pretty good performance.

4.3. Discovering bottleneck states

The bottleneck states are analogous to the key nodes in a graph, which allows us to discover them based on the graph centrality measure. Here we consider a simple definition of the centrality (Bavelas, 1950):

$$\mathrm{cent}(s) = \left(\sum_{s' \in \mathcal{S}} \mathrm{dist}(s, s') \right)^{-1}, \tag{9}$$

where dist is a generic distance measure. The states with high centrality are those where many paths pass, hence they can be considered bottlenecks. Since the Euclidean distance under RA-LapRep more accurately reflects the inter-state reachability, we aim to see if it benefits bottleneck discovery.

Specifically, we calculate $\mathrm{cent}(s)$ for all states with dist being the Euclidean distance under the learned RA-LapRep or the learned LapRep, and take top 20% states with lowest $\mathrm{cent}(s)$ as the discovered bottlenecks. For continuous environments, the summation in Eqn. (9) is calculated over a set of sampled states. Figure 8 visualizes the computed $\mathrm{cent}(s)$ and highlights the discovered bottleneck states. In this top row of Figure 8, we can see that the bottlenecks discovered by the learned RA-LapRep are essentially the states that many paths pass through. In comparison, the results of using LapRep are not as satisfactory. For one thing, as highlighted (with dashed red box) in Figure 8 (e) and (g), some of the discovered states are actually not in the center of the environment (*i.e.*, where most trajectories pass), which does not match the concept of bottleneck states. For another, as highlighted in Figure 8 (f) and (h), some regions that should have been identified are however missing. We note that the vertical asymmetry in Figure 8 (g) is not due to truncating the dimension. We create the same visualization using the ground-truth eigenfunctions instead of the learned ones As shown in Figure 16 in the appendix, the heatmap is vertically symmetric. Thus, the asymmetry in Figure 8 (g) can be attributed to imperfect approximation.

5. Limitation

5.1. Environments with directed underlying graph

One implicit assumption in our and previous works (Wu et al., 2019; Wang et al., 2021) is that the underlying graph is undirected, which implies that the actions are reversible. In practical settings such as robot control, this is often not the case. To tackle this limitation, one way is generalizing the notion of RA-LapRep to directed graphs, for example, by taking inspiration from effective resistance on directed graphs (Young et al., 2016; Boley et al., 2011; Fitch, 2019).

However, it is a highly non-trivial challenge. First, it is not straightforward to give a proper definition for RA-LapRep in directed cases. Moreover, due to the complex-valued eigenvalues of directed graph Laplacian matrices, designing an optimization objective to approximate the eigenvectors (as done in (Wu et al., 2019; Wang et al., 2021)) would be difficult. Despite the challenges, generalization to directed graphs would be an interesting research topic and worth an in-depth study beyond this work.

5.2. High-dimensional environments

In this work, we use 2D mazes because such environments allow us to easily examine whether the inter-state reachability is well captured. For applying our method to more complex environments such as Atari (Bellemare et al., 2013), we foresee some non-trivial challenges. For example, most games contain irreversible transitions. So when dealing with these environments, we may need to first generalize our method to directed graphs. Nevertheless, as a first step towards applying RA-LapRep to high-dimensional environments, we conduct experiments by replacing the 2D (x, y) position input with the high-dimensional top-view images input. The experiment results (see Appendix F) show that, with high-dimensional input, the learned RA-LapRep is still able to accurately reflect the reachability among states and significantly boost the reward shaping performance. This suggests that learning RA-LapRep with high-dimensional input on more complex environments is possible, and we will continue to explore along this direction in future works.

5.3. Assumption on uniform state coverage

As in (Wu et al., 2019; Wang et al., 2021), in our work, it is assumed that a set of transition data that roughly uniformly cover the state space can be pre-collected. This may be infeasible in practical cases. One may wonder how the learned RA-LapRep would be affected when the uniformly full state coverage assumption breaks. To investigate this, we conduct ablative experiments (learning RA-LapRep and reward shaping) by manipulating the distribution of collected data. The results show that, the learned RA-LapRep is robust to moderate changes in the data distribution, w.r.t. both its capacity in capturing the reachability and its reward shaping performance. Only when the distribution is too non-uniform, the resulting graph will be disconnected and the performance will degrade. Please refer to Appendix E for details about the experiments setup and results.

6. Related Works

Our work is built upon prior works on learning Laplacian representation with neural networks (Wu et al., 2019; Wang et al., 2021). We introduce RA-LapRep that can more accurately reflect the inter-state reachability. Apart from Laplacian representations, another line of works also aims to learn a representation that captures the inter-state reachability (Hartikainen et al., 2020; Savinov et al., 2019; Zhang et al., 2020). However, their learned reachability is not satisfying. Both our work and the reachability network (Savinov et al., 2019) view the adjacent states as positive pairs and the distant states as negative pairs, for shaping the representation. The reachability network is trained to classify whether two states are adjacent (within a preset radius) or far away. While being flexible, this method is largely based on intuition. In comparison, our approach is more theoretically grounded and ensures that the distance between two states reflects the reachability between them. As the Figure 11 in (Zhang et al., 2020) shows, even for a very simple two-room environment, the adjacency learned by the reachability network (Savinov et al., 2019) is not good (note the adjacency score between s_1 and s_2).

While the adjacency network (Zhang et al., 2020) shows some improvements over the reachability network (Savinov et al., 2019), the results are still not satisfying. For example in their Figure 11, the adjacency score between s_1 and s_2 is lower than the one between s_1 and the door connecting two rooms (it is supposed to be higher, since s_1 is farther from s_2 than from the door). In comparison, our results are verified in more complicated mazes and do not have such issues. More importantly, the adjacency network (Zhang et al., 2020) needs to maintain an adjacency matrix, which limits its applicability to environments with large or even continuous state spaces.

The dynamic distance method (Hartikainen et al., 2020) also focuses on approximating the commute time between two states. They directly learn a parameterized function to predict the temporal difference between two states sampled from the same trajectory. However, the learned distance is only visualized for a simple S-shaped maze. It is hard to tell whether this method can still learn good representations in environments with more complex geometry. In comparison, our method is able to accurately reflect the inter-state reachability in complex environments (such as Discrete-A and Discrete-B in our paper).

Apart from reward shaping, Laplacian representations have also found applications in option discovery (Machado et al., 2017; 2018; Jinnai et al., 2019; Wang et al., 2021). Note that our RA-LapRep can still be used for option discovery and would yield same good results as LapRep (Machado et al., 2017; Wang et al., 2021), since the dimension-wise scaling (in Eqn. 3) does not change the eigen-options. Regarding bottleneck state discovery, there are prior works (Şimşek & Barto, 2008; Moradi et al., 2010) that adopt other centrality measures in graph theory (*e.g.*, betweenness) to find bottleneck states for skill characterization.

7. Conclusion

Laplacian Representation (LapRep) is a task-agnostic state representation that encodes the geometry structure of the environment. In this work, we point out a misconception in prior works that the Euclidean distance in the LapRep space can reflect the reachability among states. We show that this property does not actually hold in general, *i.e.*, two distant states in the environment may have small distance under LapRep. Such issue would limit the performance of using this distance for reward shaping (Wu et al., 2019; Wang et al., 2021).

To fix this issue, we introduce a Reachability-Aware Laplacian Representation (RA-LapRep). Compared to LapRep, we show that the Euclidean distance RA-LapRep provides a better quantification of the inter-state reachability. Furthermore, this advantage of RA-LapRep leads to significant performance improvements in reward shaping experiments. In addition, we also provide theoretical explanation for the advantages of RA-LapRep.

Acknowledgement

This research is supported by the National Research Foundation Singapore under its AI Singapore Programme (Award Number: AISG2-RP-2021-023).

References

Agarwal, R., Machado, M. C., Castro, P. S., and Bellemare, M. G. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qda7-sVg84.

Bavelas, A. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950. doi: 10.1121/1.1906679. URL https://doi.org/10.1121/1.1906679.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. doi: 10.1162/089976603321780317.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47: 253–279, 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613/jair.3912.

Boley, D., Ranjan, G., and Zhang, Z.-L. Commute times for a directed graph using an asymmetric laplacian. *Linear Algebra and its Applications*, 435(2):224–242, 2011. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2011.01.030. URL https://www.sciencedirect.com/science/article/pii/S0024379511000668.

Borg, I. and Groenen, P. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Chung, F. and Graham, F. *Spectral Graph Theory*. Conference Board of Mathematical Sciences. Conference Board of the mathematical sciences, 1997. ISBN 9780821803158. URL https://books.google.com.sg/books?id=4IK8DgAAQBAJ.

Şimşek, O. and Barto, A. Skill characterization based on betweenness. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper/2008/file/934815ad542a4a7c5e8a2dfa04fea9f5-Paper.pdf.

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T., and Efros, A. Investigating human priors for playing video games. In *International Conference on Machine Learning*, pp. 1349–1357. PMLR, 2018.

Erraqabi, A., Machado, M. C., Zhao, M., Sukhbaatar, S., Lazaric, A., Denoyer, L., and Bengio, Y. Temporal abstractions-augmented temporally contrastive learning: An alternative to the laplacian in rl. *arXiv preprint arXiv: Arxiv-2203.11369*, 2022.

Fitch, K. E. Effective resistance preserving directed graph symmetrization. *SIAM J. Matrix Anal. Appl.*, 40(1):49–65, 2019. doi: 10.1137/18M1172892. URL https://doi.org/10.1137/18M1172892.

Fouss, F., Pirotte, A., Renders, J.-m., and Saerens, M. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007. doi: 10.1109/TKDE.2007.46.

Göbel, F. and Jagers, A. Random walks on graphs. *Stochastic Processes and their Applications*, 2 (4):311–336, 1974. ISSN 0304-4149. doi: https://doi.org/10.1016/0304-4149(74)90001-5. URL https://www.sciencedirect.com/science/article/pii/0304414974900015.

Hartikainen, K., Geng, X., Haarnoja, T., and Levine, S. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lmhaVtvr.

Jinnai, Y., Park, J. W., Machado, M. C., and Konidaris, G. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2019.

Kemeny, J. G. and Snell, J. L. *Finite Markov chains: with a new appendix" Generalization of a fundamental matrix"*. Springer, 1983.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980.

Koren, Y. Drawing graphs by eigenvectors: theory and practice. *Computers & Mathematics with Applications*, 49(11-12):1867–1888, 2005.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016. URL http://arxiv.org/abs/1509.02971.

Machado, M. C., Bellemare, M. G., and Bowling, M. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pp. 2295–2304. PMLR, 2017.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Bk8ZcAxR-.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-based exploration with the successor representation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 5125–5133. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5955.

Machado, M. C., Barreto, A., and Precup, D. Temporal abstraction in reinforcement learning with the successor representation. *arXiv preprint arXiv:2110.05740*, 2021.

Mahadevan, S. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 553–560, 2005.

Mahadevan, S. and Maggioni, M. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(74):2169–2231, 2007. URL http://jmlr.org/papers/v8/mahadevan07a.html.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Moradi, P., Shiri, M. E., and Entezari, N. Automatic skill acquisition in reinforcement learning agents using connection bridge centrality. In Kim, T.-h., Vasilakos, T., Sakurai, K., Xiao, Y., Zhao, G., and Ślęzak, D. (eds.), *Communication and Networking*, pp. 51–62, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-17604-3.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017.

Peter, K. and Lutz, A. *For the Beginning: The Finite Difference Method for the Poisson Equation*, pp. 19–45. Springer New York, New York, NY, 2003. ISBN 978-0-387-21762-8. doi: 10.1007/0-387-21762-2_2. URL https://doi.org/10.1007/0-387-21762-2_2.

Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Savinov, N., Raichuk, A., Vincent, D., Marinier, R., Pollefeys, M., Lillicrap, T., and Gelly, S. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkeK3s0qKQ.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319. URL https://www.science.org/doi/abs/10.1126/science.290.5500.2319.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Wang, K., Zhou, K., Zhang, Q., Shao, J., Hooi, B., and Feng, J. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pp. 11003–11012. PMLR, 2021.

Wu, Y., Tucker, G., and Nachum, O. The laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJlNpoA5YQ.

Young, G. F., Scardovi, L., and Leonard, N. E. A new notion of effective resistance for directed graphs - part I: definition and properties. *IEEE Trans. Autom. Control.*, 61(7):1727–1736, 2016. doi: 10.1109/TAC.2015.2481978. URL https://doi.org/10.1109/TAC.2015.2481978.

Zhang, A., Satija, H., and Pineau, J. Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*, 2018.

Zhang, T., Guo, S., Tan, T., Hu, X., and Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5f3b8d720f34ebebceb7765e447268b-Abstract.html.

A. Definitions and Derivations

Definition A.1 (*Average First-Passage Time* (Fouss et al., 2007; Kemeny & Snell, 1983))**.** Given a finite and connected graph $\mathcal{G}$, the *average first-passage time* $m(j|i)$ is defined as the average number of steps required in a random walk that starts from node i, to reach node j for the first time.

Definition A.2 (*Average Commute Time* (Fouss et al., 2007; Göbel & Jagers, 1974))**.** Given a finite and connected graph $\mathcal{G}$, the *average commute time* $n(i,j)$ is defined as the average number of steps required in a random walk that starts from node i, to reach node j for the first time and then go back to node i. That is, $n(i,j) = m(j|i) + m(i|j)$.

A.1. Connection between RA-LapRep and the average commute time

Let us first set up some notations. We denote the pseudo-inverse of the graph Laplacian matrix L as L^+. We denote the i-th smallest eigenvalue (sorted by magnitude) of L^+ as λ_i^+, and the corresponding unit eigenvector as $\mathbf{u}_i$. Recall that we denote the i-th smallest eigenvalue (sorted by magnitude) of L as λ_i, and the corresponding unit eigenvector as $\mathbf{v}_i$. Assuming the graph is connected, we have the following correspondence between the eigenvectors/eigenvalues of L^+ and those of L:

$$\begin{aligned} \mathbf{u}_1 &= \mathbf{v}_1, \\ (\mathbf{u}_2, \mathbf{u}_3, \cdots, \mathbf{u}_{|\mathcal{S}|}), &= (\mathbf{v}_{|\mathcal{S}|}, \mathbf{v}_{|\mathcal{S}|-1}, \cdots, \mathbf{v}_2) \\ \lambda_1^+ &= \lambda_1, \\ (\lambda_2^+, \lambda_3^+, \cdots, \lambda_{|\mathcal{S}|-1}^+) &= (\frac{1}{\lambda_{|\mathcal{S}|}}, \frac{1}{\lambda_{|\mathcal{S}|-1}}, \cdots, \frac{1}{\lambda_2}). \end{aligned} \tag{10}$$

In particular, $\mathbf{u}_1 = \mathbf{v}_1$ is a normalized all-ones vector and $\lambda_1^+ = \lambda_1 = 0$.

Let $U = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{|\mathcal{S}|})$ and $\Lambda = \mathrm{diag}(\lambda_1^+, \lambda_2^+, \cdots, \lambda_{|\mathcal{S}|}^+)$. We use $\mathbf{e}_s \in \mathbb{R}^{\mathcal{S}}$ to denote a standard unit vector with i-th entry being 1. Fouss et al. (2007) show the connection between the average commute time and the eigenvectors of the pseudo-inverse of Laplacian matrix L^+,

$$n(s, s') = V_{\mathcal{G}} \|\Lambda^{\frac{1}{2}} U^\top \mathbf{e}_s - \Lambda^{\frac{1}{2}} U^\top \mathbf{e}_{s'}\|^2. \tag{11}$$

where $V_{\mathcal{G}}$ is the volume of graph $\mathcal{G}$ (*i.e.*, sum of the node degrees). Since $\lambda_1^+ = 0$, we can obtain

$$n(s, s') = V_{\mathcal{G}} \left\| \left(\sqrt{\lambda_2^+}\mathbf{u}_2, \cdots, \sqrt{\lambda_{|\mathcal{S}|}^+}\mathbf{u}_{|\mathcal{S}|} \right)^\top \mathbf{e}_s - \left(\sqrt{\lambda_2^+}\mathbf{u}_2, \cdots, \sqrt{\lambda_{|\mathcal{S}|}^+}\mathbf{u}_{|\mathcal{S}|} \right)^\top \mathbf{e}_{s'} \right\|^2. \tag{12}$$

Based on Eqn (10), we can get

$$n(s, s') = V_{\mathcal{G}} \left\| \left(\frac{\mathbf{v}_{|\mathcal{S}|}}{\sqrt{\lambda_{|\mathcal{S}|}}}, \cdots, \frac{\mathbf{v}_2}{\sqrt{\lambda_2}} \right)^\top \mathbf{e}_s - \left(\frac{\mathbf{v}_{|\mathcal{S}|}}{\sqrt{\lambda_{|\mathcal{S}|}}}, \cdots, \frac{\mathbf{v}_2}{\sqrt{\lambda_2}} \right)^\top \mathbf{e}_{s'} \right\|^2. \tag{13}$$

Therefore, when $d = |\mathcal{S}|$, we have

$$n(s, s') = V_{\mathcal{G}} \|\boldsymbol{\phi}_d(s) - \boldsymbol{\phi}_d(s')\|^2 = V_{\mathcal{G}} \left(\mathrm{dist}_{\boldsymbol{\phi}}(s, s')\right)^2, \tag{14}$$

and that is $\mathrm{dist}_{\boldsymbol{\phi}}(s, s') \propto \sqrt{n(s, s')}$.

A.2. Equivalence between RA-LapRep and MDS

Given an input matrix of pairwise dissimilarities between n items, classic MDS (Borg & Groenen, 2005) outputs an embedding for each item, such that the pairwise Euclidean distances between embeddings preserve the pairwise dissimilarities as well as possible.

Specifically, given the squared dissimilarity matrix $D^{(2)} \in \mathbb{R}^{n \times n}$, MDS first applies doubly centering on $D^{(2)}$:

$$B = -\frac{1}{2} J D^{(2)} J, \tag{15}$$

where $J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is a centering matrix. Next, MDS calculates the eigendecomposition of B. Let Λ_+ denote the diagonal matrix containing the eigenvalues greater than 0, and Q_+ denote the matrix made of corresponding eigenvectors. Then the embedding matrix is given by $X = Q_+\Lambda_+^{\frac{1}{2}}$.

Let N denote the matrix containing pairwise average commute time, *i.e.*, $[N]_{ij} = n(i,j)$. Then we have

$$-\frac{1}{2}[JNJ]_{ij} = -\frac{1}{2}\left[n(i,j) - \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} n(i,k) - \frac{1}{|\mathcal{S}|}\sum_{h=1}^{|\mathcal{S}|} n(h,j) + \frac{1}{|\mathcal{S}|^2}\sum_{h=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} n(h,k)\right]. \tag{16}$$

Fouss et al. (2007) show that

$$n(i,j) = V_{\mathcal{G}}(l_{ii}^+ + l_{jj}^+ - 2l_{ij}^+) \tag{17}$$

where $l_{ij}^+ = [L^+]_{ij}$. Thus, we can get

$$\begin{aligned} -\frac{1}{2}[JNJ]_{ij} &= -\frac{1}{2}V_{\mathcal{G}}\left[-2l_{ij}^+ + \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} 2l_{ik}^+ + \frac{1}{|\mathcal{S}|}\sum_{h=1}^{|\mathcal{S}|} 2l_{hj}^+ - \frac{1}{|\mathcal{S}|^2}\sum_{h=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} 2l_{hk}^+\right] \\ &= V_{\mathcal{G}}\left[l_{ij}^+ - \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} l_{ik}^+ - \frac{1}{|\mathcal{S}|}\sum_{h=1}^{|\mathcal{S}|} l_{hj}^+ + \frac{1}{|\mathcal{S}|^2}\sum_{h=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} l_{hk}^+\right]. \end{aligned} \tag{18}$$

Since L^+ is doubly centered, *i.e.*, the sum of each row or each column is 0 (see Appendix A.1.2 in (Fouss et al., 2007)), we can obtain

$$-\frac{1}{2}[JNJ]_{ij} = V_{\mathcal{G}} l_{ij}^+, \tag{19}$$

that is, $B = V_{\mathcal{G}} L^+$ with $D^{(2)} = N$. Therefore, the embeddings from MDS is equivalent to the embeddings $U\Lambda^{\frac{1}{2}}$ in Eqn. (11) (up to a constant $\sqrt{V_{\mathcal{G}}}$). Based on the derivations in previous subsection, we can see the equivalence between RA-LapRep and the embeddings from MDS.

B. Environment descriptions

The two discrete environments used in our experiments are built with MiniGrid (Chevalier-Boisvert et al., 2018). Specifically, the `Discrete-A` environment is a 29×29 grid with 391 states, and the `Discrete-B` environment is a 31×31 grid with 611 states. The agent has 4 four actions: *moving left*, *moving right*, *moving up* and *moving down*. We consider two kinds of raw state representations : (x, y) position and top-view image of the grid. To pre-process the input for network training, we scale (x, y) positions to the range $[-0.5, 0.5]$, and the top-view image to the range $[0, 1]$.

The two continuous environments used in our experiments are built with MuJoCo (Todorov et al., 2012). Specifically, both `Continuous-A` and `Continuous-B` are of size 15×15. A ball with diameter 1 is controlled to navigate in the environment. At each step, the agent takes a continuous action (within range $[-\pi, \pi)$) that specifies a direction, and then move a small step forward along this direction (step size set to 0.1). We consider the (x, y) positions as the raw state representations, and also scale them to the range $[-0.5, 0.5]$ in pre-processing.

C. Experiment details

C.1. Learning the representations (LapRep and RA-LapRep)

To learn LapRep, we follow the training setup in (Wang et al., 2021) (see their Appendix E.1). Briefly speaking, we first collect a dataset of transitions using a uniformly random policy with random starts, and then learn LapRep on this dataset with mini-batch gradient decent. We use the same network architectures as in (Wang et al., 2021) and the Adam optimizer (Kingma & Ba, 2015). Other configurations are summarized in Table 1. After learning LapRep, we approximate RA-LapRep with the approach introduced in Section 3.3.

Table 1. Configurations for learning LapRep.

	Discrete environments	Continuous environments
Dataset size (in terms of steps)	100,000	1,000,000
Episode length	50	500
Training iterations	200,000	400,000
Learning rate	1e-3	1e-3
Batch size	1024	8192
LapRep dimension d	10	10
Discount sampling	0.9	0.95

C.2. Reward shaping

Following (Wu et al., 2019; Wang et al., 2021), we train the agent in goal-achieving tasks using Deep Q-Network (DQN) (Mnih et al., 2013) for discrete environments, and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) for continuous environments. We use the (x, y) position observation in both discrete and continuous cases. For DQN, we use the same fully-connected network as in (Wang et al., 2021). For DDPG, we use a two-layer fully connected neural network with units $(400, 300)$ for both the actor and the critic. The detailed configurations are summarized in Table 2. As mentioned in Section 4.2, we consider multiple goals to minimize the bias brought by the goal positions. The locations of these goals are shown in Figure 9.

C.3. Resource types and computation cost

Our experiments are run on Linux servers with Intel® Core™ i7-5820K CPU and NVIDIA Titan X GPU. For discrete environments with (x, y) positions as observations, each run of representation learning requires about 600MB GPU memory and takes about 50 minutes. Each run of reward shaping requires about 600MB GPU memory and takes about 20 minutes (for `Discrete-A`) or 40 minutes (for `Discrete-B`). For discrete environments with top-view images as observations, each run of representation learning requires about 900MB GPU memory and takes about 1 hour. For continuous environments, each run of representation learning requires about 4GB GPU memory and takes about 11 hours. Each run of reward shaping requires about 900MB GPU memory and takes about 1.5 hours.

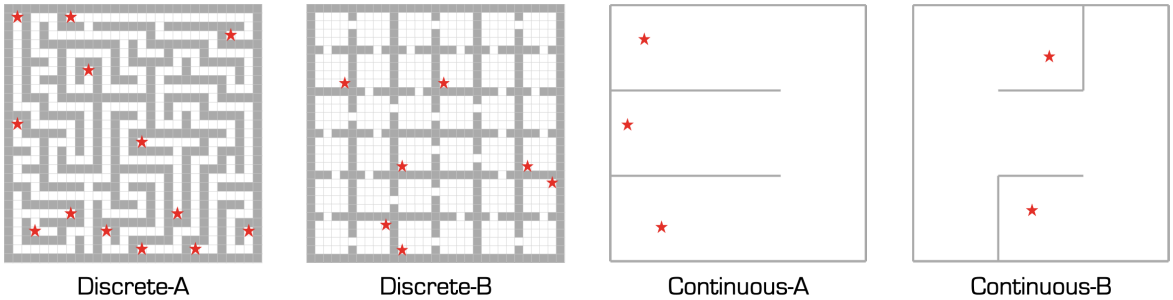

Figure 9. Goal positions for reward shaping experiments. Each red star represents a goal.

Table 2. Configurations for reward shaping.

DQN	
Timesteps	200,000 (`Discrete-A`) 400,000 (`Discrete-B`)
Episode length	150
Optimizer	Adam
Learning rate	1e-3
Learning starts	5,000
Training frequency	1
Target update frequency	50
Target update rate	0.05
Replay size	100,000
Batch size	128
Discount factor γ	0.99

DDPG	
Timesteps	500,000
Episode length	1,000
Optimizer	Adam
Learning rate	1e-3
Learning starts	100,000
Target update rate	0.001
Replay size	200,000
Batch size	2,048
Discount factor γ	0.99
Action noise type	Gaussian noise
Gaussian noise σ	0.5

D. Additional results

D.1. Capturing reachability between states

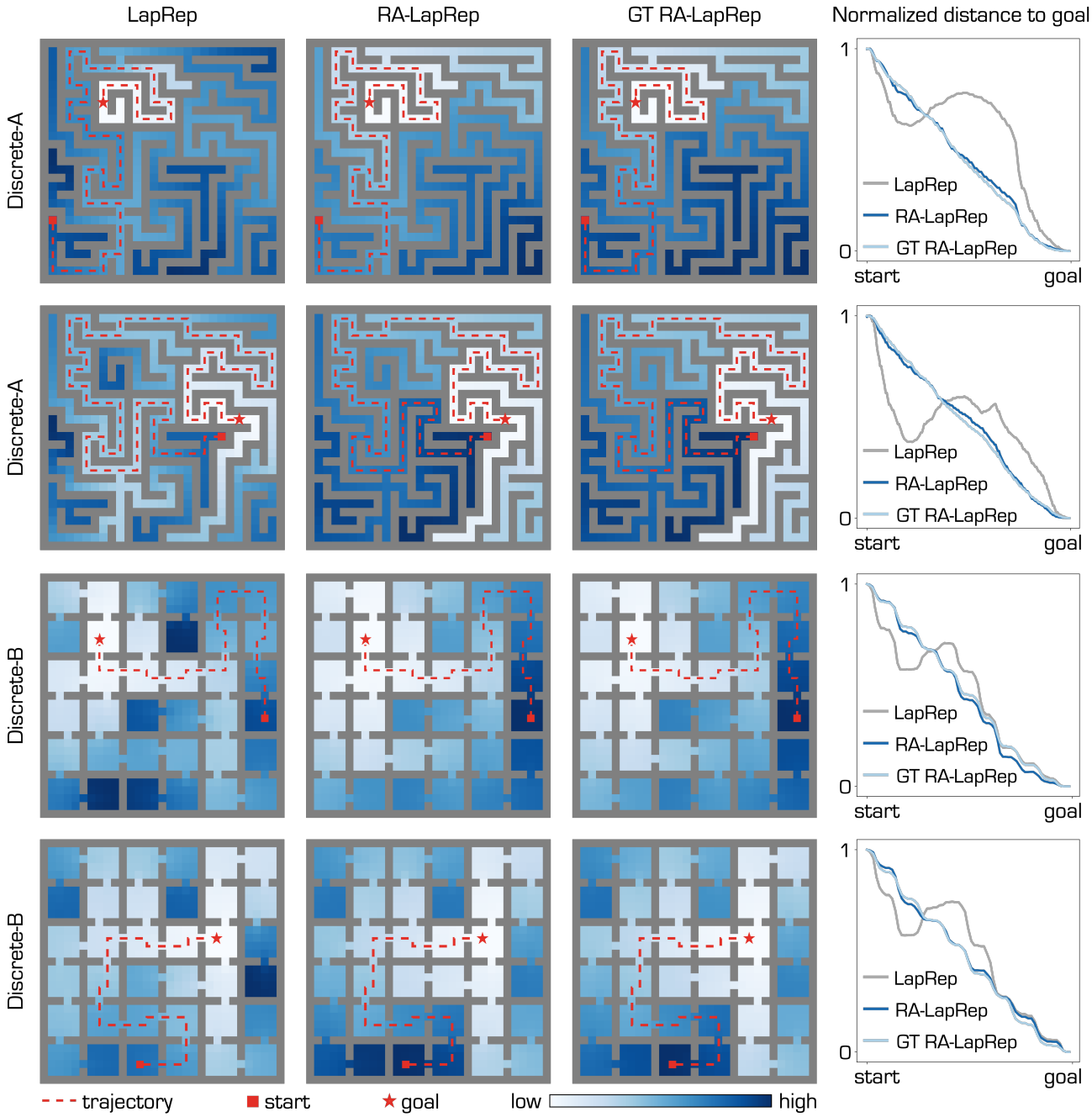

Figure 10. **Left 3 columns**: Visualization of the Euclidean distance between all states and the goals in discrete environments. For each environment, two additional trajectories (different from the one in the main paper) are shown in red. **Right**: Normalized distance values for states in the trajectories.

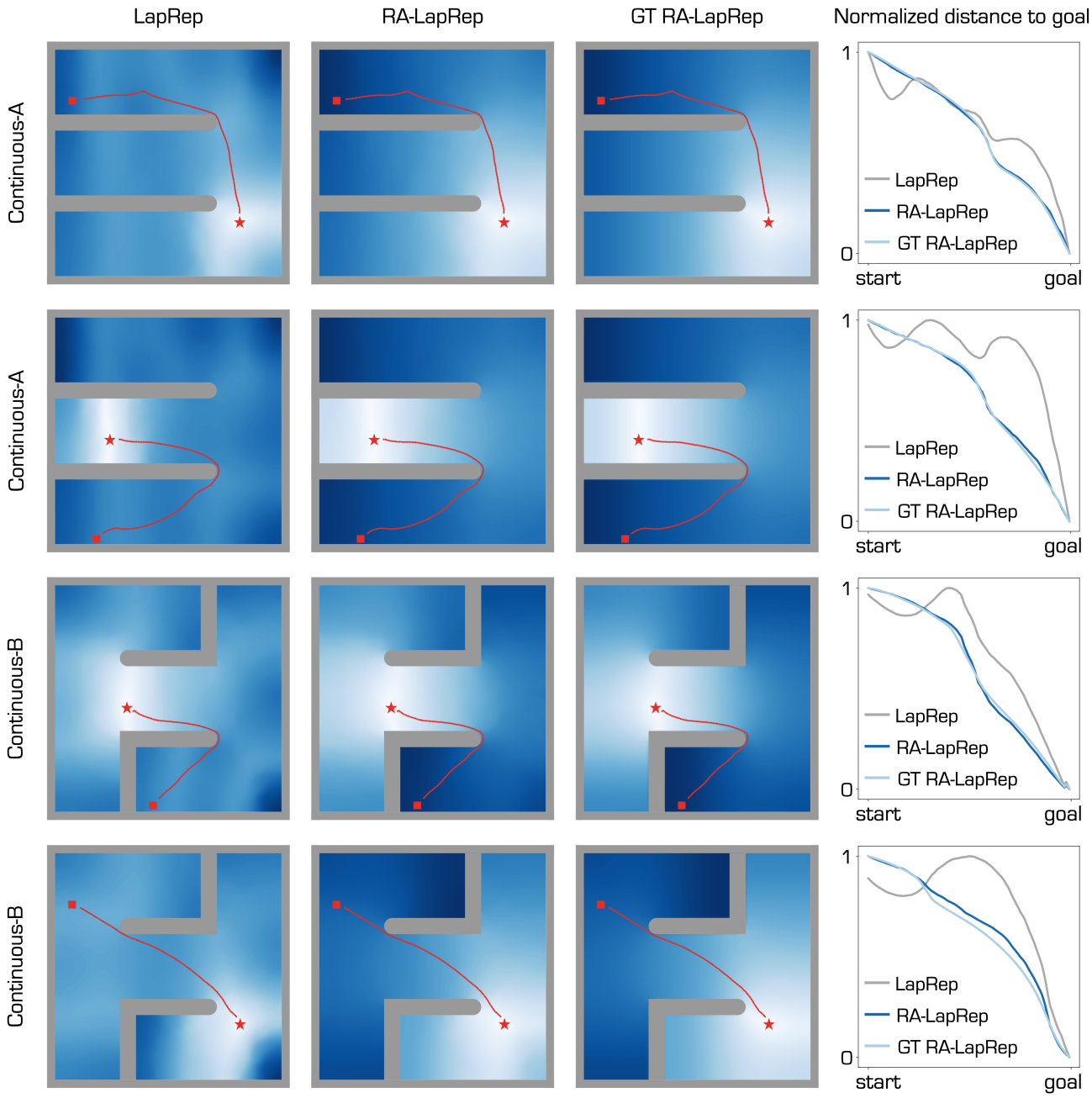

Figure 11. **Left 3 columns**: Visualization of the Euclidean distance between all states and the goals in continuous environments. For each environment, two additional trajectories (different from the one in the main paper) are shown in red. **Right**: Normalized distance values for states in the trajectories.

D.2. Distances calculated with different tractories

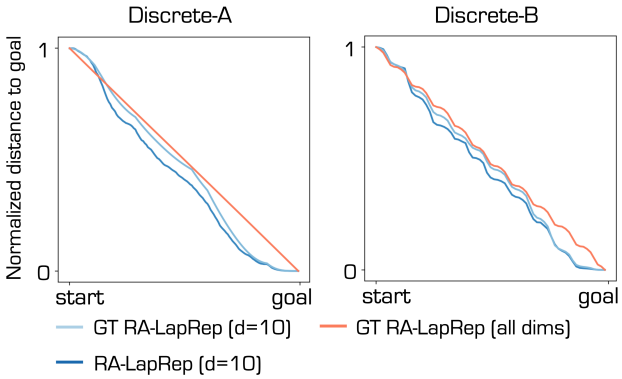

Figure 12. The Euclidean distances between all states and the goals for trajectories in Figure 4 (normalized to $[0, 1]$), where the states are sorted by temporal order.

D.3. Reward shaping results with error bars plotted

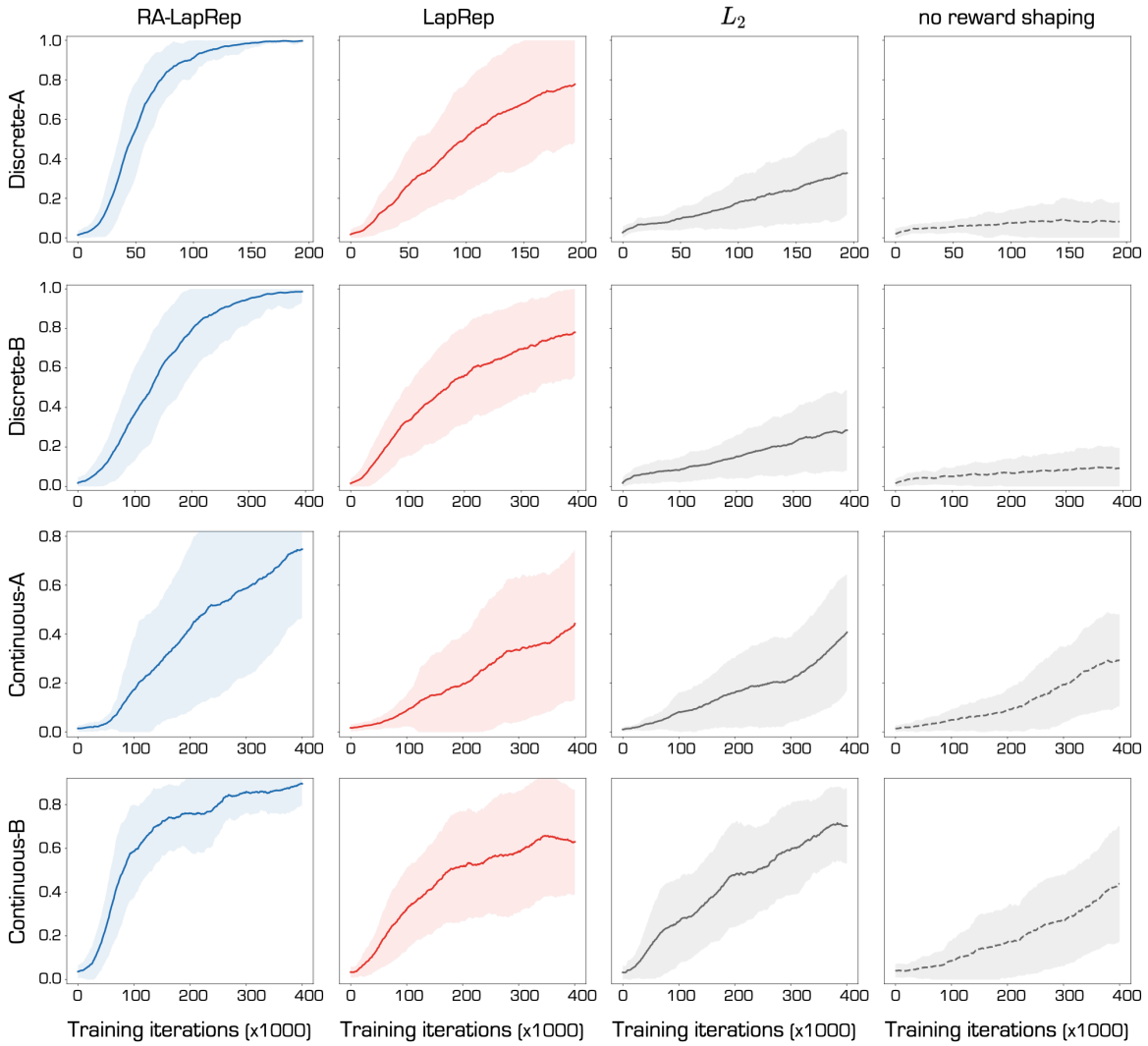

Figure 13. Reward shaping results of different methods, with standard deviation plotted as shaded area.

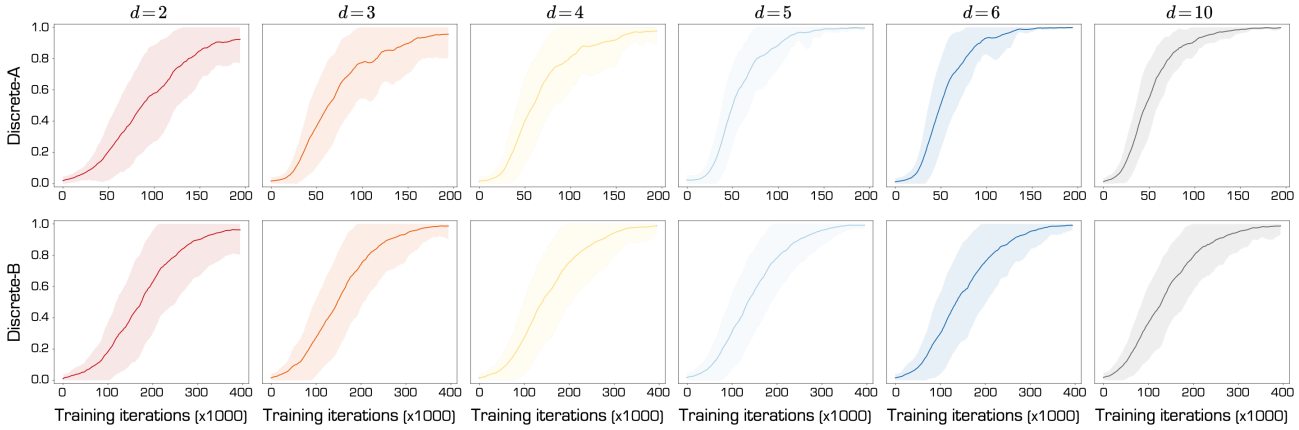

Figure 14. Reward shaping results of using different number of dimensions, with standard deviation plotted as shaded area.

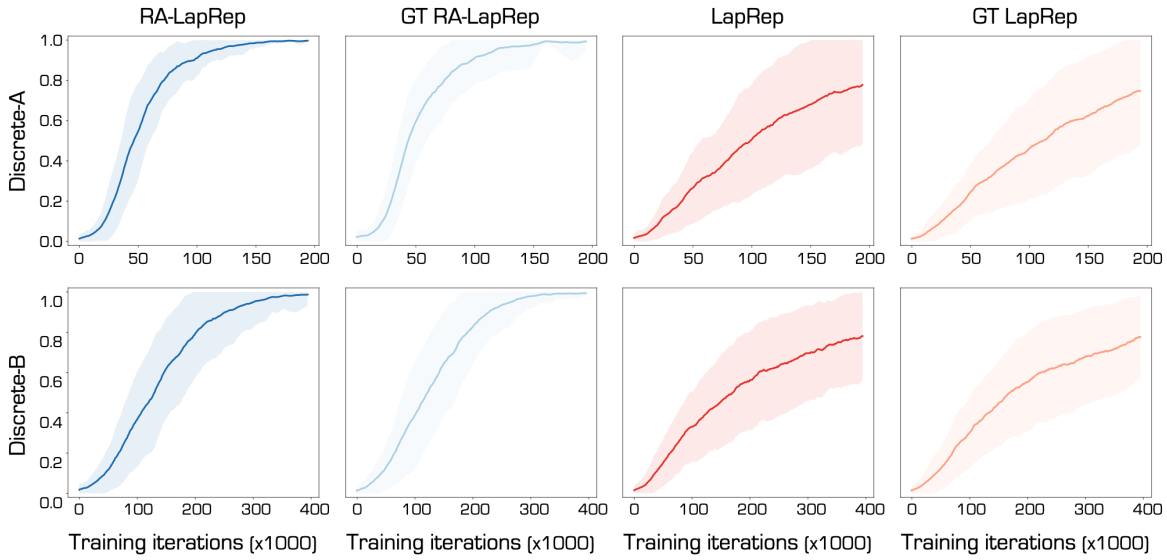

Figure 15. Reward shaping results of using learned or ground truth representations, with standard deviation plotted as shaded area.

D.4. Bottleneck discovery using the ground-truth LapRep and RA-LapRep

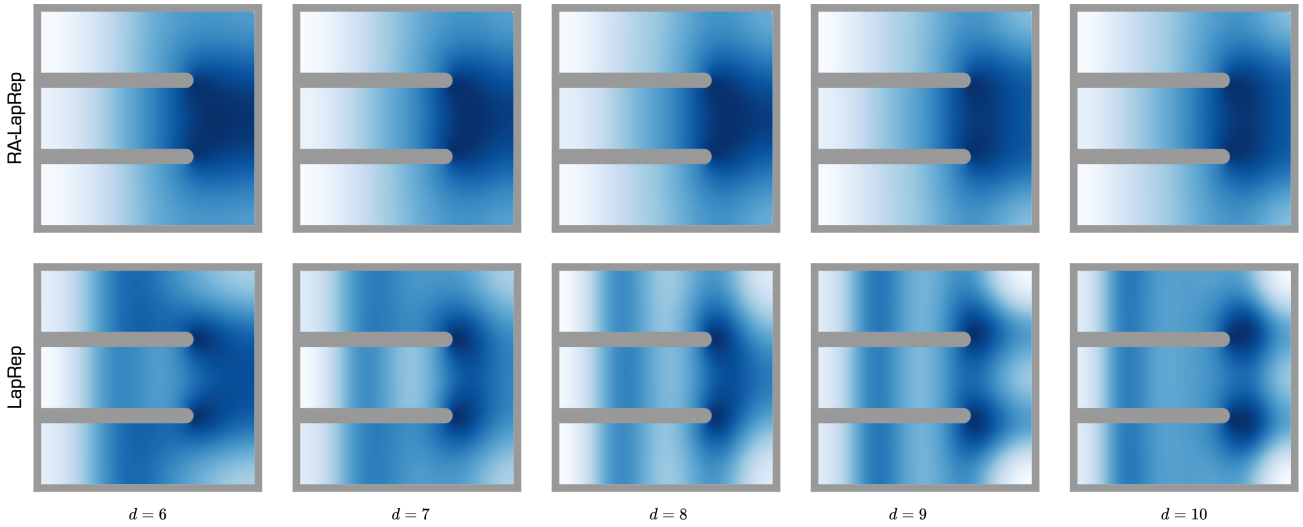

Figure 16. Bottleneck discovery results of the ground-truth LapRep and RA-LapRep.

E. Ablation study on uniform state coverage

We conduct ablative experiments to study the learned RA-LapRep would be affected when the uniformly full state coverage assumption breaks, by manipulating the distribution of the collected data. For easy comparison, we use `Discrete-A` environment since it has a visually clear structure. Specifically, we sample the agent's starting position in each episode from a distribution that can be controlled with a temperature parameter $\tau \in [0, +\infty)$. When $\tau = 0$, it reduces to a uniform distribution. As τ increases, more probability mass will be put on the dead end rooms (see Fig. 17, which visualizes the distribution of the collected data under different τ).

For each $\tau \in \{0, 0.1, 0.3, 0.6, 0.9, 1.5, 3.0, 10.0\}$, we learn RA-LapRep with the corresponding collected data. Note that $\tau = 0$ is the setting we used in the main text to learn RA-LapRep. Hyper-parameters for training are the same as those used in Sec. 4.1. Similar to Fig. 4, we visualize in Fig. 19 the Euclidean distances under RA-LapRep learned using different τ. We can see that, for $0 \leqslant \tau \leqslant 3.0$, the distance is robust to the changes in the data distribution, demonstrating that the RA-LapRep can still capture the inter-state reachability. When τ becomes too large (*e.g.*, $\tau = 10$), the method will fail. This is because the unvisited area is too large, which results in a disconnected graph and hence violates the assumption behind the methods in (Wu et al., 2019; Wang et al., 2021). Furthermore, we conduct reward shaping experiments using the learned RA-LapRep under different τ. As Fig. 18 shows, the performance only drops a little when τ increases from 0 to 3.0. This again shows that our approach is robust to moderate changes in the distribution of the collected data.

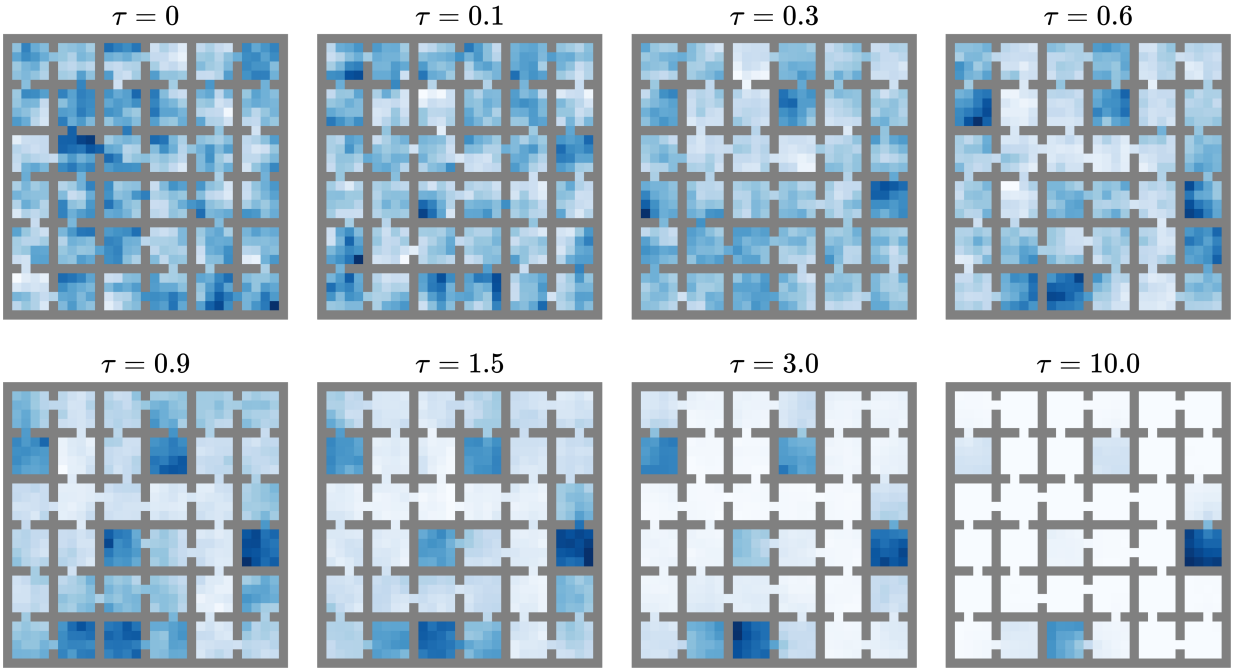

Figure 17. Visualization of the visitation counts of the collected data under different τ.

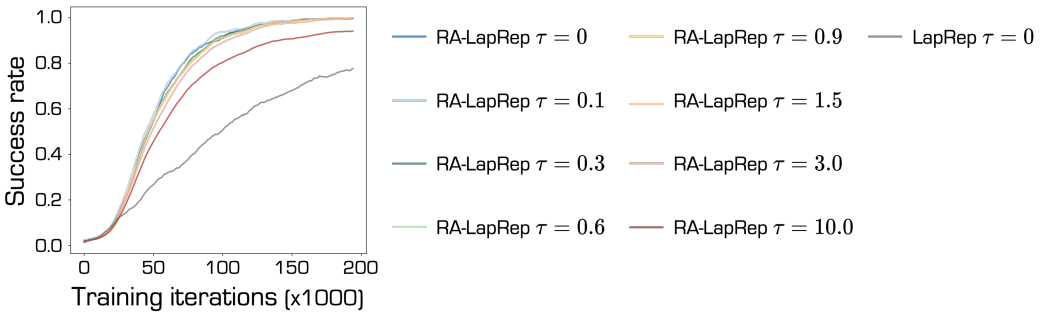

Figure 18. Comparison of the reward shaping results using the learned RA-LapRep under different τ.

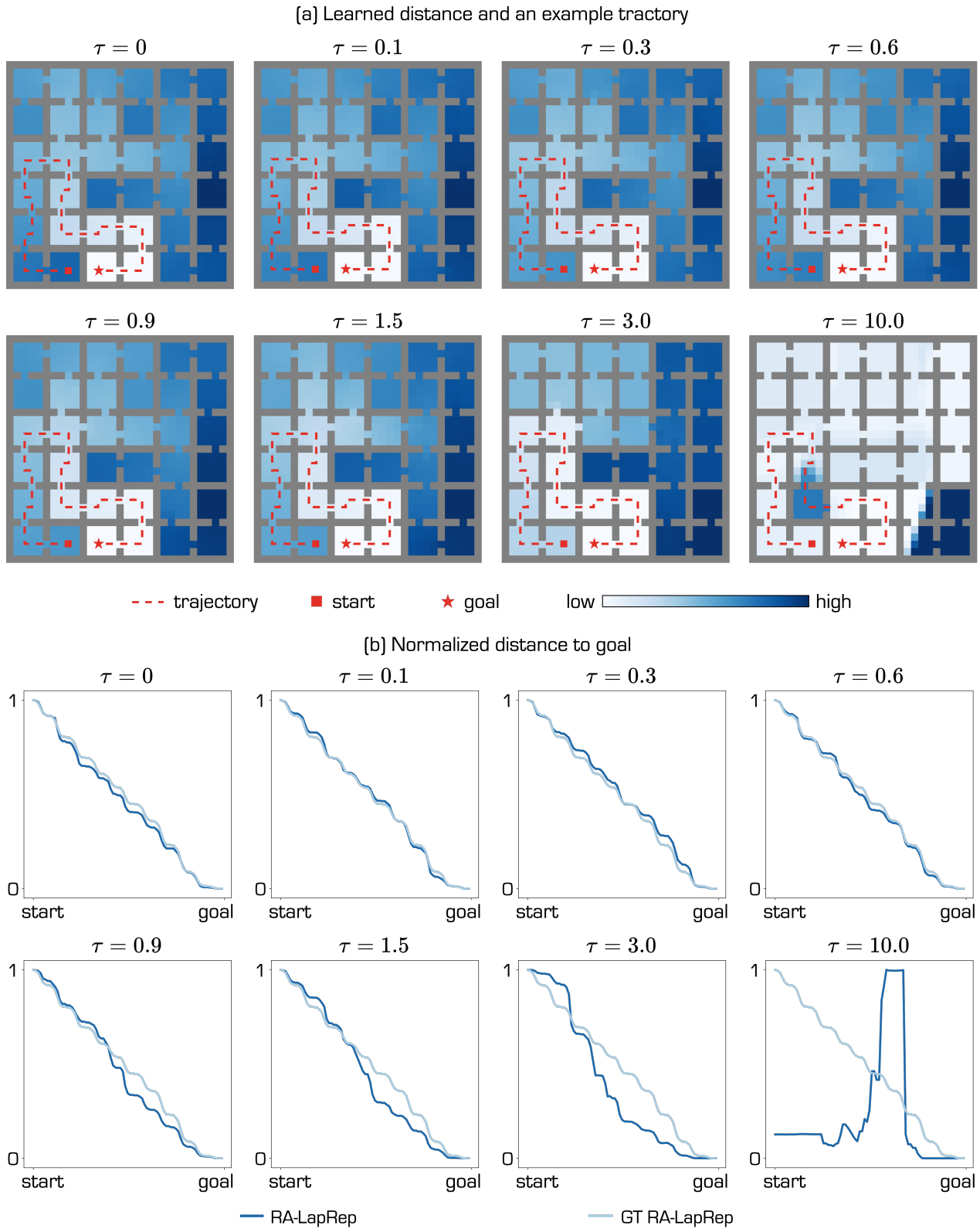

Figure 19. **(a)**: Visualizations of the Euclidean distances between all states and the goals under learned learned RA-LapRep for different τ. Example trajectories are shown in red. **b**: Line charts of the distance values for states in the trajectories (normalized to $[0, 1]$), where the states are sorted by temporal order.

F. Evaluation with top-view image observation

We train LapRep and RA-LapRep with top-view image observations on two discrete environments. The distances under learned representations are visualized in Figure 20 and Figure 21. The reward shaping results are shown in Figure 22.

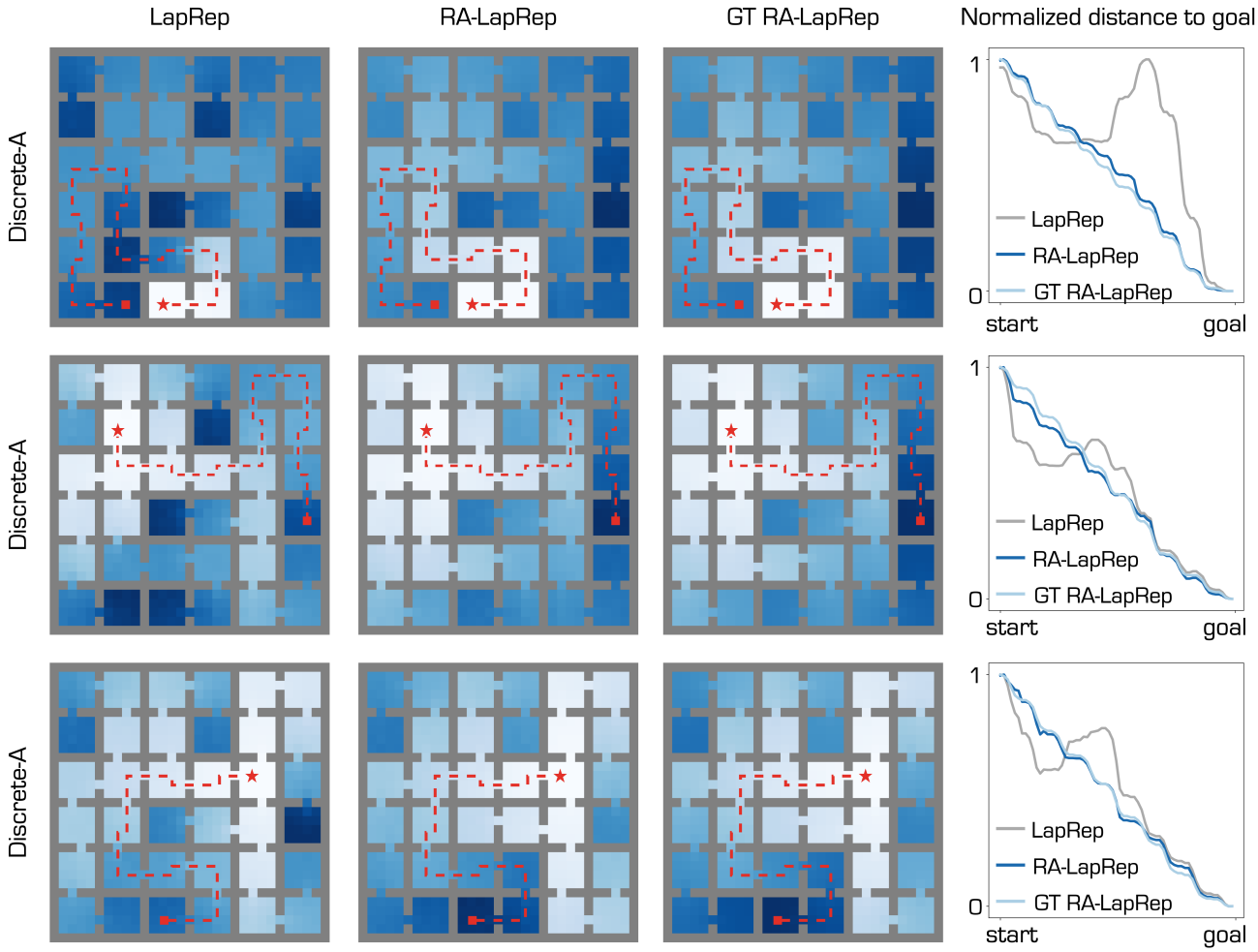

Figure 20. **Left 3 columns**: Visualization of the Euclidean distance between all states and the goals in `Discrete-A` environment, when the representations are learned from top-view image observations. **Right**: Normalized distance values for states in the trajectories.

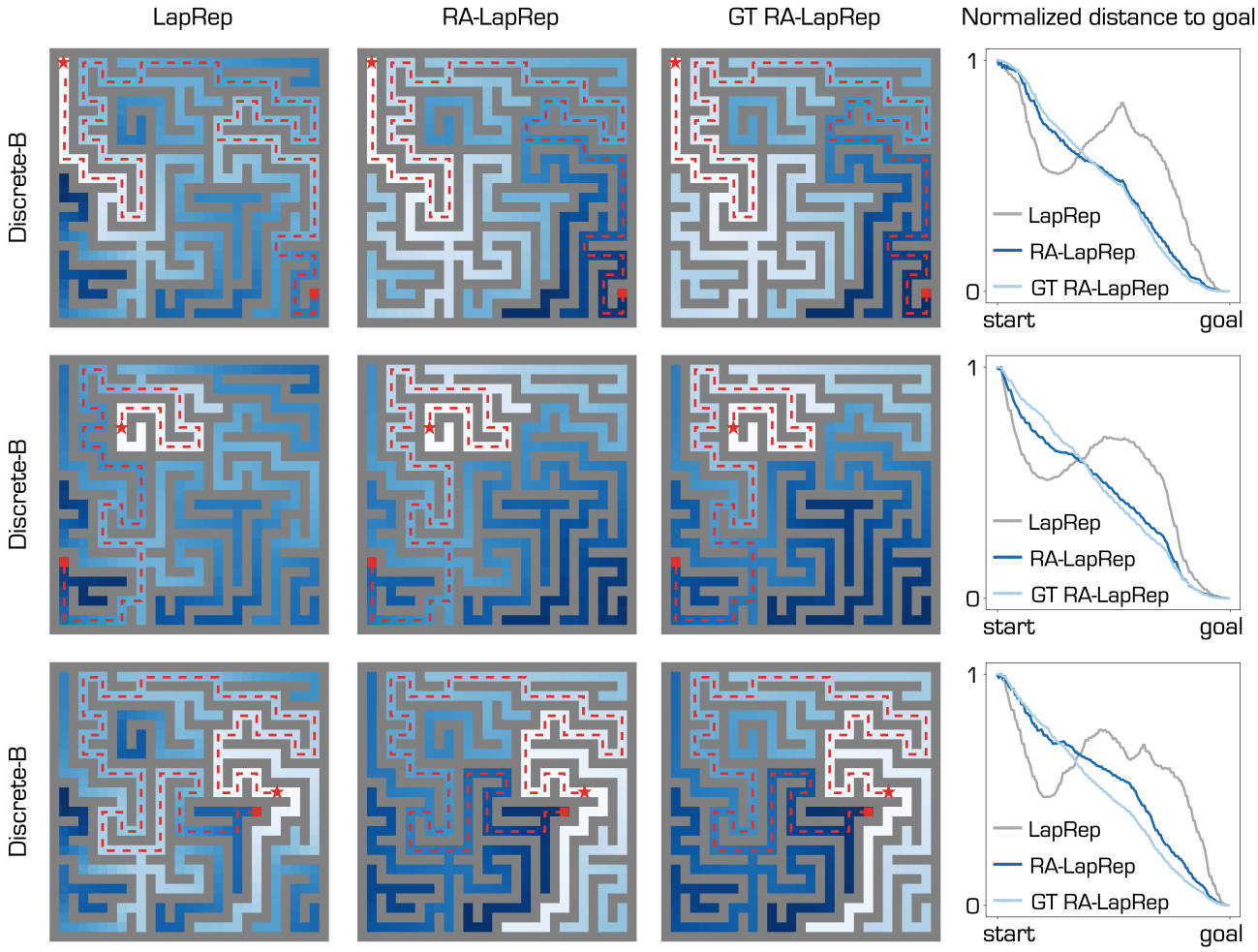

Figure 21. **Left 3 columns**: Visualization of the Euclidean distance between all states and the goals `Discrete-B` environment, when the representations are learned from top-view image observations. For each environment, two additional trajectories (different from the one in the main paper) are shown in red. **Right**: Normalized distance values for states in the trajectories.

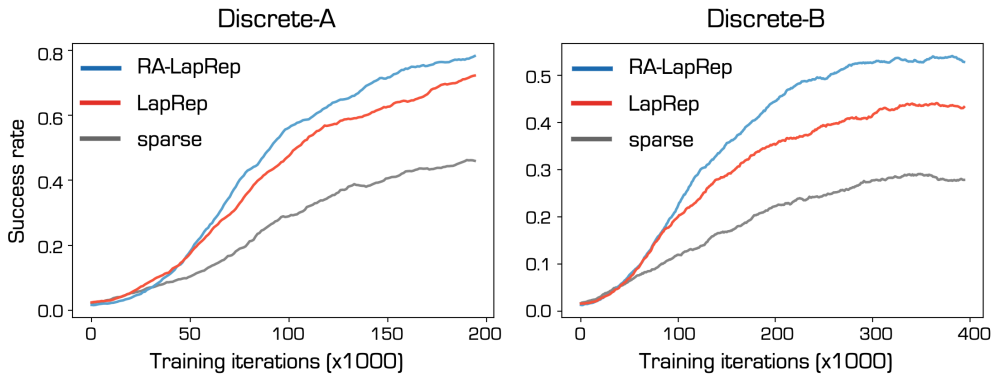

Figure 22. Reward shaping results in goal-reaching tasks using high-dimensional image input.