# PALMER: Perception-Action Loop with Memory Reorganization for Planning

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

To achieve full autonomy in a priori unknown real-world scenarios, agents should be able to operate without assuming any auxiliary instrumentation in the environment, act from high-dimensional sensory input, learn from past experience to adapt and improve, and be capable of long horizon reasoning. Classical planning algorithms are proficient at addressing the last requirement, while deep learning methods can provide the necessary flexibility to address the others. To get the best of both worlds, recent work has proposed goal-reaching methods that combine local policies and reachability estimates obtained through reinforcement learning with global graph search algorithms for long-horizon planning. However, the prevailing methods are still quite brittle, as false predictions from learning-based components can often severely debilitate downstream planning. To address this, we introduce a general-purpose planning algorithm called PALMER that creates a tight feedback loop between reinforcement learning, representation learning, sampling-based motion planning, and a non-parametric memory. Our approach augments previous work in two main ways: i) empirically, it is substantially more robust to false predictions, ii) conceptually, it defines new abstractions that can extend this line of work from goal-reaching problems to general RL problems with arbitrary reward functions, and can also offer compatibility with existing motion planning pipelines.

## 1 Introduction

Animals and humans operate on high-dimensional stimuli to achieve diverse and ever-changing goals necessary to fulfill the ecological requirements of survival [1, 2, 3, 4, 5]. Experiential learning through trial-and-error plays a fundamental role in this [6, 7, 8, 9, 10, 5], yet even in simplest environments a brute-force approach to trial-and-error is intractable. This in turn motivates mechanisms for retrospective reasoning: remembering relevant segments of past experience, restitching them in new counterfactual ways to form hypotheses, and executing them as part of a targeted search strategy. This can accelerate the trial-and-error process compared to a uniform search method, because the same behavior can remain valid for different goals and in different contexts due to the compositional structure of real-world environments. What functions can serve as building blocks for such a memory reorganisation mechanism to retrieve and reconnect bits of experience, and how can they be integrated into the current landscape of optimal control, reinforcement learning (RL), and robot motion planning?

The concept of dynamic programming is directly related to this discussion, as it greatly reduces the computational cost of trial-and-error by employing the principle of optimality [11] to prune away redundant computation. For graph search algorithms like A* [12], the incremental computation of optimal cost-to-come (i.e., value) functions by recombining past cost computations can loosely be

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

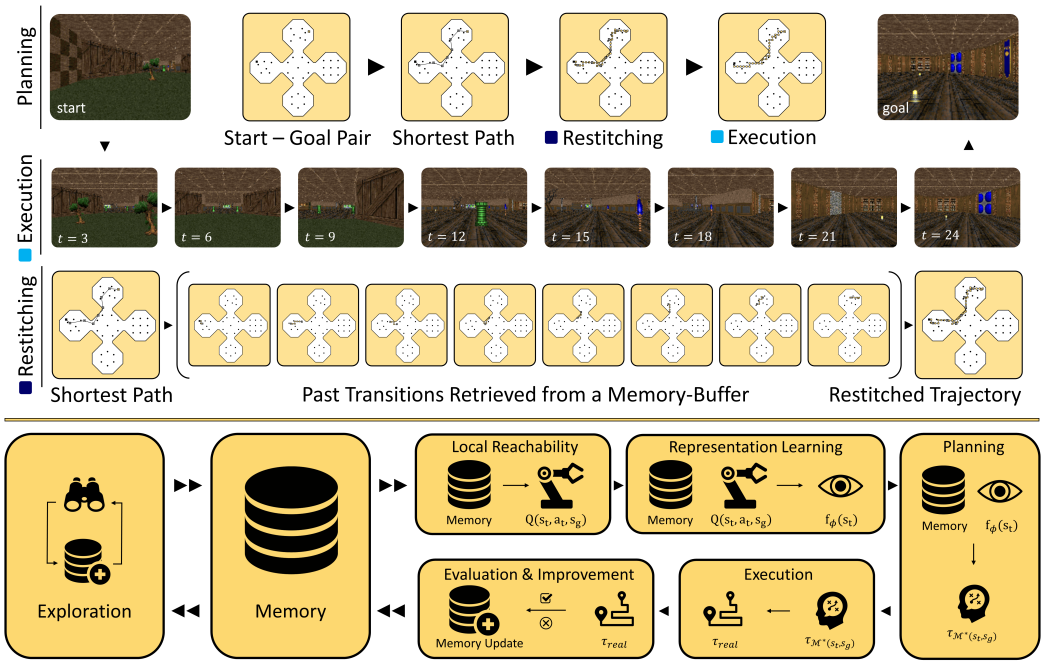


Figure 1: **Top**: Given a start-goal state pair, PALMER creates a trajectory that connects them by restitching the endpoints of past transitions in a memory-buffer. This is enabled by a perception model $f_\phi$ that can identify close-by states, and results in robust long-horizon planning. **Bottom**: To achieve this : i) it uses any *exploration* process to populate a memory-buffer, ii) uses off-policy Q-learning to obtain *local reachability* estimates, iii) uses them for *representation learning* to train $f_\phi$, iv) uses $f_\phi$ to *plan* over the memory-buffer, v) *executes* these plans, vi) *evaluates* the resulting transitions and inserts them back into the memory-buffer to *improve* its contents.

interpreted as a memory reorganization mechanism. One way to directly apply such explicit-search methods [13] to continuous environments is through employing discretization and interpolation based **approximate-overlap criteria** for state and action spaces [14, 15, 16, 17]. An alternative perspective adopted by approximate value iteration methods [18, 19, 20] is to instead perform the interpolation directly in value-space through function approximation [21, 22]. Broadly speaking, such algorithms implicitly **search through and stitch together** past transitions by virtue of minimizing Bellman error to recombine and propagate value estimates [23, 24]; therefore implementing a class of implicit-search methods identical in spirit to how graph search algorithms operate [13]. Recent work [25, 26, 27] combines these explicit-search [28, 12] and implicit trajectory-stitching [23, 24, 29] perspectives together, by setting edges between states using a distance regression model to build a planning graph, performing shortest path computations over it using graph search, and following these shortest paths with a learning-based local policy. Our paper builds upon this line of work.

## 2 Perception-Action Loop with Memory Reorganization[1]

*Contribution:* We propose two key ideas: **i)** retrieving trajectories of actually observed transitions from a memory-buffer, **ii)** restitching the endpoints of these transitions in new ways by using a perceptual **approximate-overlap criterion**. We show that these two mechanisms can be used to redefine the main subroutines shared by most classical sampling-based motion planning algorithms [30, 13], resulting in a class of retrospective planning methods that can be used to **search through and stitch together** contents of a memory-buffer to create roadmaps in hindsight (as depicted in Fig.1). Our approach operates entirely on offline, unlabeled data [23, 31, 32], and can therefore be flexibly combined with any auxiliary data collection method. Our experiments present a proof-of-concept by implementing an image-based navigation policy in simulation, using an offline memory-buffer populated with uniform random-walk exploration data.

[1]Most sub-sections have a corresponding section in the supplementary elaborating on implementation details.

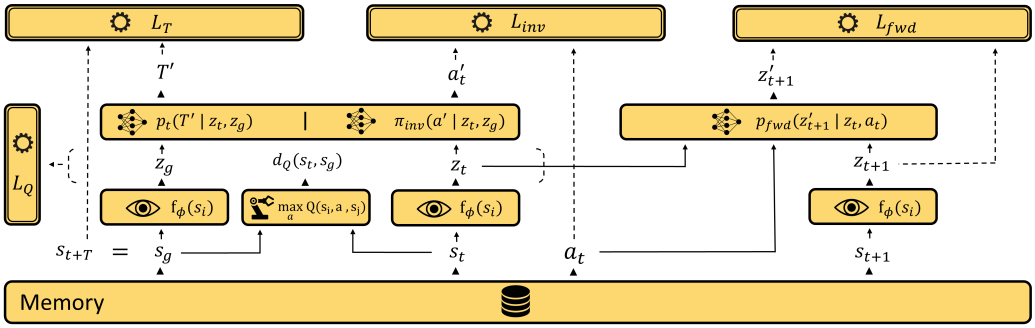


Figure 2: An overview of the functions, inputs, and losses used in our method (see Sec.2.2 for details). We aim to train a perceptual encoder $f_\phi$ with two properties: i) representations of two states should be close if they were observed to be easily reachable from each other, therefore implementing a form of *visual memory*, ii) representation of a state should capture a minimal sufficient statistic to inform an agent when and how it can reach nearby states, therefore implementing a form of *visual feature extraction*.

*Nomenclature:* An environment is represented as a tuple $\langle \mathcal{S}, \mathcal{A}, p_{env} \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, and $p_{env}(s'|s,a)$ is the markovian transition dynamics. A trajectory $\tau \in \mathcal{T}$ is any sequence of states and actions. $\tau_{i,s,0}$ ; $\tau_{i,s,-1}$ ; $\tau_{i,s,j}$ denote the first, last, and $j$'th states in $\tau_i$ respectively. The length of a trajectory in terms of timesteps is denoted as $len(\tau)$, and concatenation of two trajectories is denoted as $\tau_{cat} = \tau_1 \circ \tau_2$. We assume an additive reward function $\mathcal{R} : \mathcal{T} \to \mathbb{R}$ where $\mathcal{R}(\tau_1 \circ \tau_2) = \mathcal{R}(\tau_1) + \mathcal{R}(\tau_2)$ holds. We use the words reward and cost interchangeably in the text, but our equations use the convention of minimizing costs $-\mathcal{R}(\tau)$. We call a current/goal state pair $(s_c, s_g) \in \mathcal{S} \times \mathcal{S}$ an interaction proposal with the environment (i.e., start from the current state and reach the goal state). We call a finite set of trajectories $\mathcal{M} = \{\tau_i\}$ a memory-buffer.

### 2.1 Perceptual Representations that Capture Local Reachability and Functional Similarity

A key component of our framework is a perceptual encoder $f_\phi(s) : \mathcal{S} \to \mathbb{R}^d$ that maps states into a representation space where the L2 distance $d_\phi(s_t, s_g) := \|f_\phi(s_t) - f_\phi(s_g)\|$ captures local reachability. We say two states approximately overlap if $d_\phi(s_t, s_g) \leq d_p$ holds, where $d_p$ is a hyper-parameter. An interaction proposal $(s_t, s_g)$ is represented by concatenating the representations of its individual states as $f_\phi(s_t, s_g) := concat(f_\phi(s_t), f_\phi(s_g))$, with a corresponding L2 distance $d_\phi[(s_{c1}, s_{g1}), (s_{c2}, s_{g2})] := \|f_\phi(s_{c1}, s_{g1}) - f_\phi(s_{c2}, s_{g2})\|$ that should capture functional similarity. We employ a slight abuse of notation where $f_\phi$ and $d_\phi$ are used with both states and interaction proposals as input. What we mean by capturing functional similarity is that **if two interaction proposals can be achieved through similar actions $\tau_a$ and with similar cost $-\mathcal{R}(\tau)$, then the distance $d_\phi$ between their perceptual representations should be small**. More rigorously, we can define the following symmetric distances between states and between interaction proposals:

$$d_Q(s_1, s_2) := -\, max(\, V(s_1, s_2), V(s_2, s_1)\,) \tag{1}$$

$$d_Q[(s_{c1}, s_{g1}), (s_{c2}, s_{g2})] := -\, max(\, d_Q(s_{c1}, s_{c2}), d_Q(s_{g1}, s_{g2})\,), \tag{2}$$

where $V(s_i, s_j) := max_a\, Q(s_i, a, s_j)$, and $Q(s_i, a, s_j)$ is a goal-conditioned Q-function [33, 34]. Equation 1 is the two-way consistency criterion proposed in [26], while eq.2 upper-bounds the additional cost that can be incurred by an optimal policy executing the interaction proposal $(s_{c1}, s_{g1})$ in place of $(s_{c2}, s_{g2})$ and vice versa. What we want from $f_\phi(s)$ is for $d_\phi[(s_{c1}, s_{g1}), (s_{c2}, s_{g2})]$ and $d_Q[(s_{c1}, s_{g1}), (s_{c2}, s_{g2})]$ to correlate (i.e., be related by a monotonic transformation).

### 2.2 Combining Representation Learning with Goal-conditioned Reinforcement Learning

To train such a perceptual encoder $f_\phi$, we propose a model (depicted in Fig.2) that combines the following standard components: **i)** $z = f_\phi(s)$, projecting a given image into a latent vector; **ii)** $p_{fwd}(z'_{t+1} \mid z_t, a_t)$, modelling the transition distribution induced by $p_{env}(s'|s,a)$ over the latent space $z = f_\phi(s)$, as discussed in [35, 36]; **iii)** $\pi_{inv}(a'_t \mid z_t, z_g)$, defining a distribution of actions to achieve a given interaction proposal, as discussed in [35, 36, 27]; **iv)** $p_t(T' \mid z_t, z_g)$, modelling

the distribution of timesteps necessary to achieve a given interaction proposal, as discussed in [37]; **v)** $Q(s_t, a_t, s_g)$, a Q-value function that provides local reachability estimates between pairs of states, as discussed in [25, 33]. Following [25, 33], we train $Q(s_t, a_t, s_g)$ using a reward function $r(s_t, a, s_{t+1}, s_g) = -\mathbb{1}_{s_{t+1} \neq s_g}$. This reward function means $max_a Q(s_t, a, s_g) = -len(\tau_{sp})$ holds for the optimal policy, where $\tau_{sp}$ is the shortest-time path that connects $(s_t, s_g)$ (see [25] for a further discussion). After training $Q(s_t, a_t, s_g)$ in isolation, we freeze its parameters and use it to define a contrastive loss function [38] $L_Q$ to optimize $f_\phi$. To train the remaining components, we randomly sample a transition $(s_t, a_t, s_{t+1})$ and a time difference $T$, and set the goal state as $s_g := s_{t+T}$, as in hindsight goal relabelling [39, 33]. We then minimize the following losses:

- $L_Q(s_t, s_g) = l_{hinge}(d_\phi(s_t, s_g) - d_p)\ \mathbb{1}_{d_Q(s_t,s_g) \leq c_Q} + l_{hinge}(d_p - d_\phi(s_t, s_g))\ \mathbb{1}_{d_Q(s_t,s_g) \geq c_Q}$ where $l_{hinge}$ is the hinge loss [40]. This contrastive loss dictates that perceptual representations should be close together (i.e., $d_\phi(s_t, s_g) \leq d_p$ holds) if and only if two states are close to each other in terms of reachability (i.e., $d_Q(s_t, s_g) \leq c_Q$ holds), where $d_p$ and $c_Q$ are hyperparameters.
- $L_T(T', T)$, $L_{inv}(a'_t, a_t)$, and $L_{fwd}(z'_{t+1}, z_{t+1})$ are MSE and cross-entropy losses [36, 37]. $L_T$ and $L_{inv}$ dictate that perceptual representations should capture enough information to know when and how an agent can reach from one state to another, while $L_{fwd}$ dictates that they should capture only a minimal-sufficient statistic for doing so ([36] presents a more elaborate discussion).

### 2.3 Perceptual Memory Retrieval (PMR)

Given any interaction proposal $(s_c, s_g)$ as a query, we use the components from Sec.2.2 to retrieve the highest scoring trajectory from $\mathcal{M}$ whose first-last states overlap with the query pair according to our approximate-overlap criterion from Sec.2.1. This solves the following optimization problem:

$$\tau_{\mathcal{M}(s_c,s_g)} := \underset{\tau \in \mathcal{M}}{\arg\min} -\mathcal{R}(\tau), \tag{3}$$

$$\text{s.t. } d_\phi[(\tau_{s,0}, \tau_{s,-1}), (s_c, s_g)] \leq d_p, \tag{4}$$

$$\|len(\tau) - \mathbb{E}_{T \sim p_t(T\ |f_\phi(s_c), f_\phi(s_g))}[T]\| \leq \eta, \tag{5}$$

where $\mathcal{R}$ can be any general reward function (for navigation, we use $-\mathcal{R}(\tau) = len(\tau)$ as in [25]). Verbally, eq.3 searches for the lowest cost trajectory $\tau_{\mathcal{M}(s_c,s_g)} \in \mathcal{M}$ that satisfies two constraints: i) $(\tau_{s,0}, \tau_{s,-1})$ approximately-overlap with $(s_c, s_g)$ according to $d_\phi$ (cons.4), ii) its actually observed distance metric $len(\tau)$ matches the distance estimate $p_t$ between $(s_c, s_g)$ (cons.5).

### 2.4 Trajectory Restitching Through 'Retrospective Planning'

Sampling-based geometric motion-planning algorithms such as RRT [41, 42], RRT* [30], and PRM [43, 44] connect a set of 3D points sampled from obstacle-free space with line segments in order to iteratively build a roadmap (i.e., planning graph). We instead reimagine them as memory search mechanisms by altering their graph building subroutines so that whenever an edge is created, a trajectory is retrieved through PMR (eq.3) and stored in that edge. Our new definitions for these subroutines directly mirror the original subroutines given in the authoritative monograph of [30]:
*1) Sampling:* Sampling originally returns a point from obstacle free space. We instead return a state $s_c$ from $\mathcal{M}$ using any distribution (e.g., uniform, or based on visitation-counts).
*2) Lines and Their Cost:* The equivalent of drawing a line segment in our framework is retrieving a trajectory $\tau_{\mathcal{M}(s_c,s_g)}$, and its length and cost are $len(\tau_{\mathcal{M}(s_c,s_g)})$ and $-\mathcal{R}(\tau_{\mathcal{M}(s_c,s_g)})$ respectively.
*3) Nearest State and Neighborhood Queries:* Given a query point $s_i$, these subroutines return the closest point or a neighborhood of points within a distance, among a set of vertices $V = \{s_j\}$. We preserve these definitions, and only replace the metric from euclidean distance to $len(\tau_{\mathcal{M}(s_c,s_g)})$.

$$Nearest(V, s_g) := \underset{s_c \in V}{\arg\min}\ len(\tau_{\mathcal{M}(s_c,s_g)})$$

$$Near(V, s_g, r) := \{s_c \in V : len(\tau_{\mathcal{M}(s_c,s_g)}) \leq r\}$$

*4) Collision Tests:* Collision tests originally prevent the sampling and line drawing subroutines from intersecting obstacles. Since we are planning in retrospect, any such undesirable event can be handled during PMR by the cost function (i.e., if $\tau$ has such an event, this reflects on $-\mathcal{R}(\tau)$).

**Algorithm 1** R-PRM (Roadmap Construction)

**Input:** $f_\phi, \mathcal{M}$
$V \leftarrow \{SampleFree_i\}_{i=1,\ldots,num_vertices}$; $E \leftarrow \emptyset$ ▷ Initialize vertices and edges
**for each** $s_i \in V$ **do**
  $U \leftarrow Near(V, s_i, r) \setminus \{s_i\}$
  **for each** $s_j \in U$ **do** ▷ Place PMR trajectories in edges
    $E \leftarrow E \cup \{(s_i, s_j) : \tau_{edge} = \tau_{\mathcal{M}(s_i,s_j)},\ d_{edge} = -\mathcal{R}(\tau_{\mathcal{M}(s_i,s_j)})\}$
**return** $G = (V, E)$

**Algorithm 2** R-PRM (Trajectory Restitching Given the Constructed Roadmap)

**Input:** $s_c, s_g, G = (V, E), \mathcal{R}(\tau), f_\phi, \mathcal{M}$
**for each** $s_i \in V$ **do** ▷ Insert $s_c$ and $s_g$ into the PRM graph
  **if** $len(\tau_{\mathcal{M}(s_c,s_i)}) \leq r$ **then** ▷ Place PMR trajectories in edges
    $E \leftarrow E \cup \{(s_c, s_i) : \tau_{edge} = \tau_{\mathcal{M}(s_c,s_i)},\ d_{edge} = -\mathcal{R}(\tau_{\mathcal{M}(s_c,s_i)})\}$
  **if** $len(\tau_{\mathcal{M}(s_i,s_g)}) \leq r$ **then**
    $E \leftarrow E \cup \{(s_i, s_g) : \tau_{edge} = \tau_{\mathcal{M}(s_i,s_g)},\ d_{edge} = -\mathcal{R}(\tau_{\mathcal{M}(s_i,s_g)})\}$

$\tau_{stitched} \leftarrow \emptyset$
$\{s_j\} \leftarrow ShortestPath(s_c, s_g, G, \mathcal{R}(\tau))$ ▷ Trajectory stitching by dynamic programming
**for** $0 < i < |\{s_j\}|$ **do** ▷ Concatenate PMR trajectories along the shortest path
  $\tau_{stitched} \leftarrow \tau_{stitched} \circ \tau_{\mathcal{M}(s_{i-1},s_i)}$
**return** $\tau_{\mathcal{M}^*(s_c,s_g)} = \tau_{stitched}$

Using these subroutines directly in-place of their originals, we reimplement retrospective equivalents of PRM, RRT, and RRT*, which we call R-PRM, R-RRT, R-RRT*. We denote the resulting restitched trajectory as $\tau_{\mathcal{M}^*(s_c,s_g)}$. Algorithms 1, 2 describe R-PRM as an example, and the supplementary contains descriptions for R-RRT, R-RRT*. We emphasize that for any general reward function $\mathcal{R}$, as the number of sampled vertices in R-PRM and R-RRT* increases, $\mathcal{R}(\tau_{\mathcal{M}^*(s_c,s_g)})$ gets optimized through dynamic programming (i.e., by minimizing the Bellman error between vertices of the roadmap $G$), therefore employing the same mechanism as sampling-based geometric planning [30].

### 2.5 PALMER: Closing the Perception-Action Loop to Optimize Memory Contents

We iteratively form and execute $\tau_{\mathcal{M}^*(s_c,s_g)}$, and whenever execution is successful, we insert the resulting transitions back into $\mathcal{M}$ to optimize its contents through trial-and-error. This creates the following perception-action loop: **i)** $\mathcal{M}$ with optimized contents is used to train a more accurate $Q(s_t, a, s_g)$, **ii)** a more accurate $Q(s_t, a, s_g)$ creates a more accurate approximate-overlap criterion $d_\phi$, **iii)** a better $d_\phi$ generates better $\tau_{\mathcal{M}^*(s_c,s_g)}$, **iv)** better $\tau_{\mathcal{M}^*(s_c,s_g)}$ result in higher frequencies of successful execution to further optimize $\mathcal{M}$, therefore looping back to step i (see the supplementary for an algorithmic description). This is essentially a generalized policy-iteration process [22], where generating and executing $\tau_{\mathcal{M}^*(s_c,s_g)}$ is equivalent to policy-evaluation, and putting only successfully executed trajectories back into $\mathcal{M}$ and then updating $Q$ and $f_\phi$ is equivalent to policy improvement.

## 3 Related work

*Self-supervised goal reaching:* Our approach is closely related to goal-reaching methods that combine learning-based distance-regression with graph search, particularly Semi-parametric Topological Memory (SPTM) [27] and Search on the Replay Buffer (SoRB) [25], which we compare to in our experiments. The key difference of our approach is that it uses transitions that are **actually observed**. This brings three main benefits. First is robustness; local reachability estimates are susceptible to overestimation when evaluated between pairs of states that are far apart or unreachable, since they rarely occur together and are therefore essentially out of distribution for the distance regression model (see [45] for a discussion). This creates 'hallucinated' shortcuts in the graph that corrupt shortest

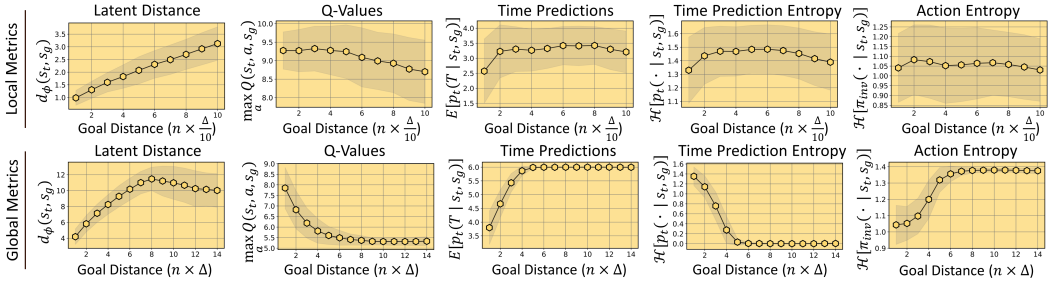


Figure 3: Distances in the representation space of $f_\phi$ correlate with reachability, compared with other suitable metrics from Sec.2.2 such as Q-values, average time-predictions, and the entropies of time-prediction and action distributions (the latter increases with goal distance as there are larger numbers of ways to reach the goal state).

path queries [25, 26]. To address this, [27] employs temporally consistent localization and adaptive waypoint selection, while [25] employs distributional Q-learning and an ensemble of Q-functions. In our approach, eq.3 naturally adresses this problem, since it requires an actual short trajectory in the dataset approximately connecting two states before marking them as close. If there is no such trajectory and yet two states are close, our method can still connect them during the trajectory restitching phase. The second benefit of our approach is that it can optimize general reward functions, since it decouples the reachability metric $len(\tau)$ (used in nearest neigbor queries and as a threshold to create edges) from the downstream task reward $\mathcal{R}(\tau)$ (used to set edge distances), unlike previous work. The third benefit of our approach is that it creates dense trajectories rather than sparse subgoal sequences. Therefore, any information stored by the agent during exploration can later be searched, stitched, and returned as part of $\tau_{\mathcal{M}^*(s_c,s_g)}$. This in turn makes all of this information available for further use by any downstream processing pipeline. For example: unorganized depth images, odometry, and proprioception information originally logged by a hand-engineered exploration policy [46] can be restitched and used to initialize a trajectory-optimization method [47], whose results can be combined with a local reactive motion controller using Riemannian Motion Policies [48].

*Image-Based Navigation:* [49, 50, 51] present learning-based navigation systems that incrementally build roadmaps through online operation. Our approach has two main differences: **i)** it builds a roadmap entirely using raw offline data, therefore allowing applications like multi-robot learning without additional loop-closure mechanisms to fuse graphs from multiple agents, **ii)** our approach can optimize general reward functions, therefore it is not limited to navigation.

*Robot Motion Planning:* A common approach to motion planning is to first run a sampling-based planning algorithm [13, 30], and then refine the result through trajectory optimization [52, 53, 54] to satisfy constraints [55, 17, 56]. An important bottleneck is that sampling-based planning algorithms require a precomputed map of the environment, and our approach extends such algorithms in a way that relaxes this requirement by replacing a precomputed map with raw exploration experience.

*SLAM and Geometric Maps:* SLAM based methods [57] can autonomously construct high-fidelity geometric maps [58, 59], therefore alleviating the bottleneck of precomputing environment maps. The downside of such approaches is that they can abstract away useful physical and semantic affordances. For example, a purely geometric map cannot plan a path through a traversable field of tall-grass, while our approach can learn and utilize such affordances as long as they are represented in past experiences (i.e., if there exists transitions in $\mathcal{M}$ that traverse through tall grass).

# 4 Experiments[2]

*Setup:* Our navigation experiments are performed in ViZDoom [60]. Following previous work [25, 26], states solely consist of four images that form a panorama (i.e., $4 \times 3 \times 160 \times 120$ dimensions), and actions move the agent North/South/East/West by a fixed distance $\Delta$. The map contains many long-thin column-like obstructions (shown as dots in visualizations), as we found that image-based navigation policies are prone to getting stuck in such obstacles (an observation also made by [25]).

[2] All experiments have a corresponding section in the supplementary providing further implementation details.

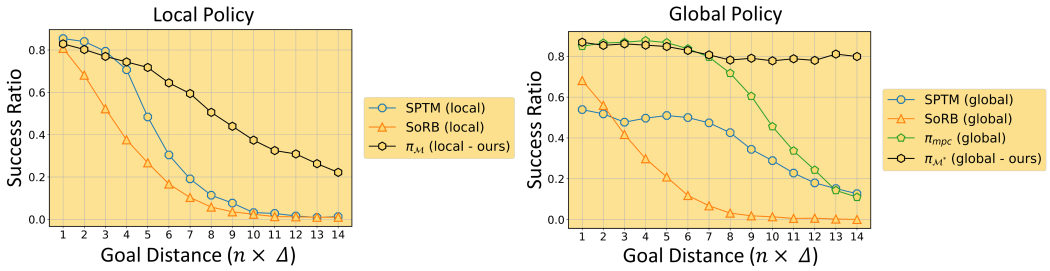


Figure 4: Comparisons of our local policy $\pi_{\mathcal{M}}$ and global policy $\pi_{\mathcal{M}^*}$ with SPTM and SoRB. $\pi_{\mathcal{M}}$ performs well because it avoids getting stuck (as such events are filtered by eq.3), while $\pi_{\mathcal{M}^*}$ performs well because it builds robust roadmaps without hallucinated shortcuts; therefore avoiding the main failure modes of the baselines.

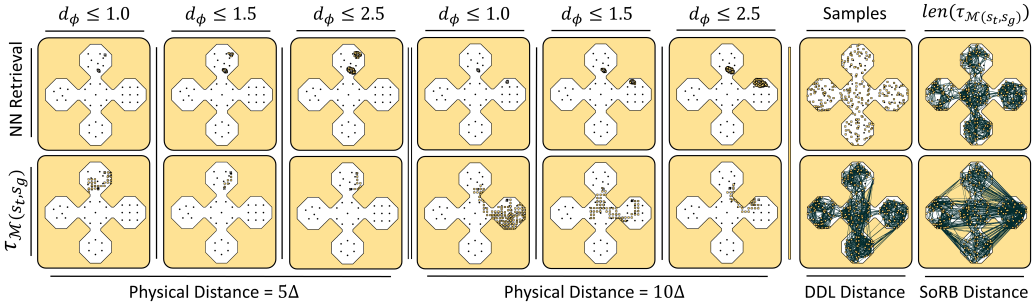


Figure 5: At the core of PALMER is a process called *perceptual memory retrieval* (PMR). Given a query pair of current-goal states, PMR searches the memory-buffer to retrieve the highest scoring trajectory $\tau_{\mathcal{M}(s_t,s_g)}$ whose first and last states are close to the query pair according to the approximate-overlap criterion $d_\phi$. **Left, Middle**: Visualizations of $\tau_{\mathcal{M}(s_t,s_g)}$ retrieved using PMR and nearest-neighbor states retrieved using $d_\phi$. **Right**: Setting edges of a roadmap using $len(\tau_{\mathcal{M}(s_t,s_g)})$, compared with distance estimates used in SORB and DDL [37]. We found that distance estimates from baselines are prone to setting false edges that cross map boundaries.

*Validating Perceptual Representations:* Fig.3 shows that $d_\phi(s_t, s_g)$ obtained from our model captures a suitable notion of local reachability, compared with other potential metrics. Fig.5 in turn shows that retrieving nearest-neighbor states from $\mathcal{M}$ using $d_\phi$ (i.e., NN retrieval) returns physically close states.

*Validating Perceptual Memory Retrieval (PMR):* Fig.5 shows visualizations of trajectories retrieved with PMR. We implement a retrieval policy $\pi_{\mathcal{M}}$ that computes $\tau_{\mathcal{M}(s_t,s_g)}$ through eq.3 at each timestep $t$ and executes $argmax_a\ Q(s_t, a, \tau_{\mathcal{M}(s_t,s_g),s,1})$, therefore forming a model predictive control (MPC) loop. We evaluate $\pi_{\mathcal{M}}$ in an image-based navigation task where start/goal images are sampled randomly to have an euclidean distance $n \times \Delta$ in between, and a trial is considered successful if the agent can get within $\Delta$ proximity of the goal position within $4 \times n$ time-steps. We use the local policies from SORB [25] and SPTM [27] as baselines. Fig.4 shows the results. The main mode of failure for both SPTM and SORB local policies is that they get stuck in column-like structures. $\pi_{\mathcal{M}}$ avoids this, since eq.3 retrieves collision free $\tau_{\mathcal{M}(s_t,s_g)}$.

*Robust Distances:* PMR also helps avoid hallucinations in local distance regression. Fig.5 illustrates this point by setting edges between sampled states by thresholding $len(\tau_{\mathcal{M}(s_t,s_g)})$, where methods of [25, 37] are used as baselines. It can be seen that edges set by $len(\tau_{\mathcal{M}(s_c,s_g)})$ are more robust.

*Retrospective Planning Algorithms:* Fig.6 shows visualizations of planning graphs and $\tau_{\mathcal{M}^*(s_c,s_g)}$ produced by R-PRM, R-RRT, and R-RRT*. It can be seen that R-PRM doesn't contain any hallucinated edges, while R-RRT and R-RRT* maintain the visual characteristics of their classical counterparts (i.e., R-RRT has jagged branches with uniform coverage, while R-RRT* has straight branches shooting out from the root). We implement an MPC policy $\pi_{\mathcal{M}^*}$ that replans at each timestep $t$ using Algorithm 2 to return $\tau_{\mathcal{M}^*(s_t,s_g)}$, and executes $argmax_a\ Q(s_t, a, \tau_{\mathcal{M}^*(s_t,s_g),s,1})$. We again use SORB [25] and SPTM [27] as baselines.[3] Fig.4 shows the results. In addition to the local policy

[3]For a comparison without confounders, we train SoRB with DDQN [61] rather than distributional Q-learning [62], and we do not employ temporally consistent localization for SPTM, as such fixes are equally applicable to our method and orthogonal to the discussion. The supplementary provides further elaboration.

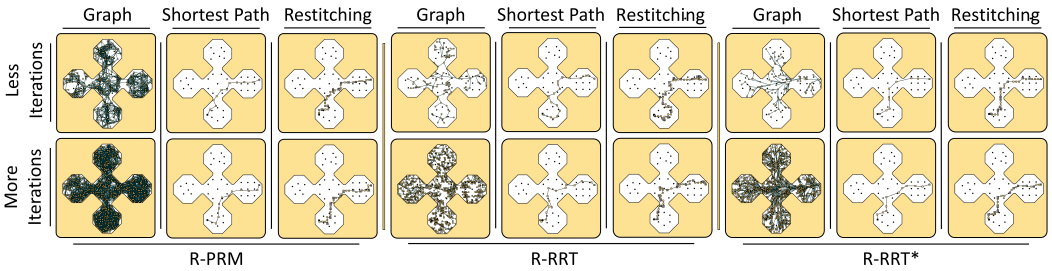


Figure 6: We repurpose conventional sampling-based planning algorithms by altering their graph building subroutines so that whenever an edge is created, a trajectory is retrieved through PMR and stored in that edge (a trajectory stitching mechanism that we call *memory reorganization*). We visualize results of trajectory stitching and planning graphs produced by our memory reorganization algorithms R-PRM, R-RRT, R-RRT*.

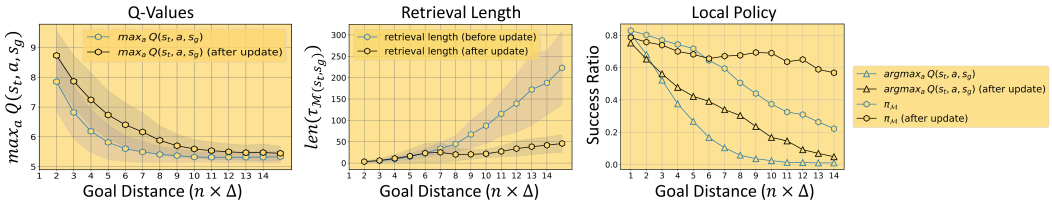


Figure 7: In PALMER, a policy has three groups of parameters: $Q(s_t, a_t, s_g)$, $f_\phi$, and the contents of $\mathcal{M}$. Iteratively forming plans through memory reorganization and executing them creates a feedback loop between these components, where: i) *actions inform perception* during the training of $f_\phi$, ii) *perception facilitates actions* through the formation roadmaps, and iii) *memory serves as the medium* for this reciprocal interaction. As a result, trajectories produced by explicit planning are gradually internalized as implicit behavior encoded in the model parameters. This leads to: Q-values propagating further into distant goals (**Left**), memory contents getting closer to optimal (**Middle**), and performances of local policies showing significant improvement (**Right**).

getting stuck, a new mode of failure for both baselines is that false distance estimates throw-off graph search by setting hallucinated shortcuts. $\pi_{\mathcal{M}^*}$ avoids this hallucination issue thanks to cons.4, 5 in eq.3. An additional baseline is $\pi_{mpc}$, which extends the SPTM local policy by using $p_{fwd}$ and $p_t$ from Sec.2.2 to implement an MPC loop with n-step look-ahead. $\pi_{mpc}$ avoids the pitfall of getting stuck in columns thanks to n-step lookahead, but still isn't sufficient for global navigation as the accuracy of simulated rollouts from $p_{fwd}$ decreases with the number of time-steps.

*Closing the Loop with PALMER:* We optimize the contents of $\mathcal{M}$ by iteratively generating and executing $\tau_{\mathcal{M}^*(s_c,s_g)}$. We then retrain all model components only on the resulting new data that is equal in size to the initial unoptimized $\mathcal{M}$. Fig.7 shows the results. When $\pi_{\mathcal{M}}$, and $argmax_a Q(a_t, a, s_g)$ are used as policies, their success ratio increases significantly if they are trained on the optimized $\mathcal{M}$. Q-value estimates trained on the optimized $\mathcal{M}$ also **propagate better to goals further away**. The scaling of $len(\tau_{\mathcal{M}(s_c,s_g)})$ with goal-distance changes from an exponential trend to an approximately linear one, due to the inclusion of transitions from successfully executed $\tau_{\mathcal{M}^*(s_c,s_g)}$.

*Additional Experiments in Habitat and Generalization:* To further validate our method, we ran experiments in the Habitat simulator [63]. As shown in Fig.8, we find that our method can allow image-based navigation in this new domain with significantly different visuals and layouts (i.e., real-world apartments), action space (i.e., turn-left, turn-right, go-forward), and state space (i.e., single $256 \times 256$ RGB images with 150 FOV). Perhaps more surprisingly, we find that training $f_\phi$ on exploration data from a *single* apartment generalizes substantially well to any unseen apartment, which directly allows perceptual memory retrieval and retrospective planning when provided with a corresponding memory-buffer. The supplementary contains further evaluations and visualizations.

## 5 Discussion and Future Directions

*When would PMR create a false shortcut:* This can only happen if three things simultaneously go wrong: i) $p_t$ hallucinates for the pair $(s_c, s_g)$ (cons.5), ii) $d_\phi$ hallucinates for each of $s_c$ and $s_g$ (cons.4), iii) these three independent hallucinations happen in such a way that the actual $len(\tau_{\mathcal{M}(s_c,s_g)})$ is

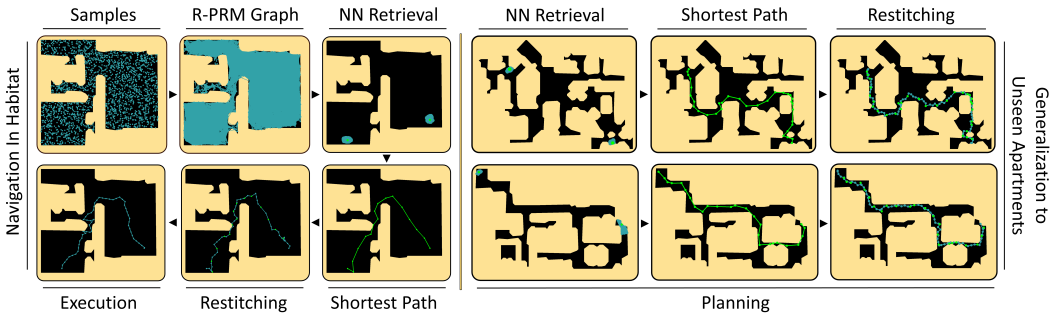


Figure 8: Left: Our method is compatible with different state and action spaces, as demonstrated in the Habitat simulation environment. Right: We found that training the perception model $f_\phi$ on a *single* apartment generalizes sufficiently well to allow perceptual memory retrieval and retrospective planning in any unseen apartment.

consistent with the false $p_t$ estimate (cons.5). We never observed this to happen in our experiments.
*Is PALMER less expressive than standard deep Q-learning:* Two important premises of deep Q-learning [64, 17] are: i) minimizing Bellman error through temporal-difference (TD) updates can restitch observed transitions in new optimal ways [23, 24], ii) a neural network can learn to extrapolate Q-values to unobserved but close-by states in high-dimensional spaces (e.g. images) [18]. Both arguments are equally valid for our approach, since it can: i) restitch transitions at **arbitrary resolutions** (i.e., anywhere from one-step transitions to multi-step trajectories) by virtue of sampling-based planning, ii) group together close-by states through $d_\phi$. Therefore, PALMER is an RL algorithm that: **i)** optimizes Bellman error through sampling-based optimal planning rather than gradient-based TD-updates [18], **ii)** performs extrapolation between states using a perceptual-backbone $f_\phi$ rather than a deep Q-network, and **iii)** replaces the greedy-policy $argmax_a\ Q(s_t, a, s_g)$ and value estimate $max_a\ Q(s_t, a, s_g)$ with $argmax_a\ Q(s_t, a, \tau_{\mathcal{M}^*(s_t,s_g),s,1})$ and $\mathcal{R}(\tau_{\mathcal{M}^*(s_c,s_g)})$ respectively. The key benefits of these alterations come into play when $s_t$ and $s_g$ are far apart, and these benefits are: **i)** the checks in cons. 4, 5 that **prevent hallucinations** in $Q(s_t, a, s_g)$, **ii) global propagation of value estimates** by virtue of employing sampling-based planning methods, which are known to be particularly proficient at searching high-dimensional state spaces across long-horizons [56, 13].
*Combining PALMER with standard deep Q-learning:* Our approach can also be flexibly combined with any traditional Q-learning method [18, 65, 20], by using our retrospective-planning algorithms (Sec.2.4) as experience replay methods [66]. This alternative approach stitches together $\tau_{\mathcal{M}^*(s_c,s_g)}$ during training, and perform backwards TD-updates over this trajectory starting from $s_g = \tau_{\mathcal{M}^*(s_c,s_g),s,-1}$ and ending at $s_c = \tau_{\mathcal{M}^*(s_c,s_g),s,0}$. As suggested by Fig.7, this can allow value estimates $Q(s_t, a, s_g)$ to propagate more globally. Our proof-of-concept experiments identify this as a promising direction, and we leave a further extensive evaluation to future work.

## 6 Conclusion and Limitations

We presented PALMER, an approach that brings together perception and action to search and restitch offline, unlabeled past experiences for planning. Our experiments show that PALMER can solve long-horizon goal-reaching tasks from continuous high-dimensional inputs. The biggest limitation of our current exposition as well as the most immediate direction for future work is to extend our experiments to more complex planning problems [67, 68], general RL benchmarks [23], and real-world robotic systems [69, 70, 71]. We believe that our memory-based planning perspective highlights a number of interesting questions for future research. First, which transitions should be kept in the memory-buffer $\mathcal{M}$, and which ones should be discarded? $\mathcal{M}$ cannot be infinitely expanded after deployment, and it is critical to distill away redundancies between stored experiences. Second, when the environment undergoes a change, which transitions in the memory-buffer remain valid and can still be used for planning, and which ones become invalid? A mechanism that can answer this question can allow quick and sample-efficient adaptation to environmental changes. Third, how can we extend $f_\phi$ to allow more abstract associations and functional equivariances between states? This can improve generalization by defining a more flexible notion of experience retrieval that can recycle past behavior in new contexts and for new tasks. We leave these questions to future work.

## References


[1] S. E. Palmer, *Vision science: Photons to phenomenology*. MIT press, 1999.

[2] J. J. Gibson, *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[3] J. K. O'regan and A. Noë, "A sensorimotor account of vision and visual consciousness," *Behavioral and brain sciences*, vol. 24, no. 5, pp. 939–973, 2001.

[4] J. D. Co-Reyes, S. Sanjeev, G. Berseth, A. Gupta, and S. Levine, "Ecological reinforcement learning," *arXiv preprint arXiv:2006.12478*, 2020.

[5] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, vol. 299, p. 103535, 2021.

[6] E. L. Thorndike, "Animal intelligence: An experimental study of the associative processes in animals." *The Psychological Review: Monograph Supplements*, vol. 2, no. 4, p. i, 1898.

[7] ——, "The law of effect," *The American journal of psychology*, vol. 39, no. 1/4, pp. 212–222, 1927.

[8] D. T. Campbell, "Perception as substitute trial and error." *Psychological review*, vol. 63, no. 5, p. 330, 1956.

[9] R. J. Herrnstein, "On the law of effect," *Journal of the experimental analysis of behavior*, vol. 13, no. 2, pp. 243–266, 1970.

[10] S. D. Whitehead and D. H. Ballard, "Learning to perceive and act by trial and error," *Machine Learning*, vol. 7, no. 1, pp. 45–83, 1991.

[11] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[12] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136

[13] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[14] A. W. Moore and C. G. Atkeson, "Memory-based reinforcement learning: Converging with less data and less real time," in *Robot Learning*. Springer, 1993, pp. 79–103.

[15] ——, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103–130, 1993.

[16] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine learning*, vol. 49, no. 2, pp. 291–323, 2002.

[17] P. Abbeel, *Advanced Robotics*, 2019. [Online]. Available: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[20] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[21] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific, 1996.

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

[23] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," 2020.

[24] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn *et al.*, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," *arXiv preprint arXiv:2104.07749*, 2021.

[25] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[26] S. Emmons, A. Jain, M. Laskin, T. Kurutach, P. Abbeel, and D. Pathak, "Sparse graphical memory for robust planning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5251–5262, 2020.

[27] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[29] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine, "Rvs: What is essential for offline rl via supervised learning?" *arXiv preprint arXiv:2112.10751*, 2021.

[30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[31] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *CoRR*, vol. abs/2005.01643, 2020. [Online]. Available: https://arxiv.org/abs/2005.01643

[32] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*. Springer, 2012, pp. 45–73.

[33] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*, vol. 2. Citeseer, 1993, pp. 1094–8.

[34] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*. PMLR, 2015, pp. 1312–1320.

[35] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," *Advances in neural information processing systems*, vol. 29, 2016.

[36] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[37] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine, "Dynamical distance learning for semi-supervised and unsupervised skill discovery," *arXiv preprint arXiv:1907.08225*, 2019.

[38] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[39] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[40] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[41] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning."

[42] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[43] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[44] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and automation*, vol. 14, no. 1, pp. 166–171, 1998.

[45] S. Tian, S. Nair, F. Ebert, S. Dasari, B. Eysenbach, C. Finn, and S. Levine, "Model-based visual planning with self-supervised functional distances," *arXiv preprint arXiv:2012.15373*, 2020.

[46] A. Nanavati, N. Walker, L. Taber, C. Mavrogiannis, L. Takayama, M. Cakmak, and S. Srinivasa, "Not all who wander are lost: A localization-free system for in-the-wild mobile robot deployments," in *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, 2022, pp. 422–431.

[47] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[48] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.

[49] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Recon: Rapid exploration for open-world navigation with latent goal models," *arXiv preprint arXiv:2104.05859*, 2021.

[50] D. Shah and S. Levine, "Viking: Vision-based kilometer-scale navigation with geographic hints," *arXiv preprint arXiv:2202.11271*, 2022.

[51] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 672–678.

[52] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[53] ——, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[54] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[55] R. Tedrake, *Underactuated Robotics*, 2022. [Online]. Available: http://underactuated.mit.edu

[56] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard, *Principles of robot motion: theory, algorithms, and implementations*, 2005.

[57] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[58] M. Bujanca, P. Gafton, S. Saeedi, A. Nisbet, B. Bodin, M. F. O'Boyle, A. J. Davison, P. H. Kelly, G. Riley, B. Lennox *et al.*, "Slambench 3.0: Systematic automated reproducible evaluation of slam systems for robot vision challenges and scene understanding," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6351–6358.

[59] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[60] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *2016 IEEE conference on computational intelligence and games (CIG)*. IEEE, 2016, pp. 1–8.

[61] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[62] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 449–458.

[63] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9339–9347.

[64] S. Levine, *Deep Reinforcement Learning*, 2022. [Online]. Available: http://rail.eecs.berkeley.edu/deeprlcourse/

[65] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[66] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[67] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[68] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

[69] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Neural autonomous navigation with riemannian motion policy," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8860–8866.

[70] C. Paxton, N. Ratliff, C. Eppner, and D. Fox, "Representing robot task plans as robust logical-dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5588–5595.

[71] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]