Expansive Latent Planning for Sparse Reward Offline Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Sampling-based motion planning algorithms excel at searching global solution paths in geometrically complex settings. However, classical approaches, such as RRT, are difficult to scale beyond low-dimensional search spaces and rely on privileged knowledge e.g. about collision detection and underlying state distances. In this work, we take a step towards the integration of sampling-based planning into the reinforcement learning framework to solve sparse-reward control tasks from high-dimensional inputs. Our method, called VELAP, determines sequences of waypoints through sampling-based exploration in a learned state embedding. Unlike other sampling-based techniques, we iteratively expand a tree-based memory of visited latent areas, which is leveraged to explore a larger portion of the latent space for a given contingent of search iterations. We demonstrate state-of-the-art results in learning control from offline data in the context vision-based manipulation under sparse reward feedback. Our method extends the set of available planning tools in model-based reinforcement learning to include a latent planner that globally searches for a feasible paths, rather than being bound to a fixed prediction horizon.

Keywords: Reinforcement Learning, Planning, Robot Manipulation

1 Introduction

The acquisition of complex motor skills from raw sensory observations presents one of the main goals of robot learning. Reinforcement learning (RL) [1] provides a generic framework to obtain such decision-making policies through the interaction with an environment. Model-based RL methods [2] have recently gained much attention due to benefits in terms of sample-efficiency and robustness in long-horizon scenarios. To address the issue of short-sighted decisions, model-based agents are often combined with planning algorithms. However, effective planning with high-dimensional inputs, such as video data, is often challenging due to the increased complexity of the search space and the difficulty in generating accurate long-term predictions. Consequently, a growing body of research has explored the utilization of representation learning to simplify the decision-making problem by mapping it to an abstract and lower-dimensional latent state space [3, 4, 5, 6, 7].

The model-based reinforcement learning (RL) literature has investigated various planning methods in latent spaces, encompassing zero-order shooting-based approaches such as the Cross-Entropy Method (CEM) [8, 3] and Model-Predictive Path Integral (MPPI) [9, 10, 7], first-order gradient-based optimization [11, 4], and more recently, trajectory collocation using second-order solvers [6]. Despite this methodological diversity, the majority of existing tools primarily facilitate local optimization within a fixed prediction horizon. Even with guidance from value heuristics, such as the one proposed in [7], local minima may still impede progress, particularly when estimating the optimal value function is difficult due to sparse reward feedback or limited training data. In this paper, we argue that long-horizon planning in learned latent state spaces can pose significant geometric challenges, necessitating methods that strive for global exploration of the solution space.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

The limitations observed in existing methods raise the need for more sophisticated planning strategies that can seamlessly integrate with learned state and dynamics models. Sampling-based motion planning [12], provides a diverse range of algorithms for finding global paths between states in continuous and geometrically-complex environments. Recent works by [5, 13, 14] have proposed modifications of sampling-based planners for planning in latent spaces. However, these approaches either rely on expert data or are not directly applicable to reward-based learning settings. This paper explores the integration of sampling-based planning techniques into learned latent spaces, providing new avenues for model-based reinforcement learning. Specifically, we focus on the challenging scenario of offline RL [15], which is characterized by the amplified effects of value approximation errors [16]. Further, it allows us to better study the performance of planning in isolation by disentangling training and data collection. We introduce Value-guided Expansive Latent Planning (`VELAP`), which combines a sampling-based planning module with a suitable state embedding. Similar to [14], our search tree serves as a state memory and it used to guide the exploration towards undiscovered areas within the data support. Moreover, we leverage value heuristics obtained through temporal difference learning to accelerate the discovery of high reward states. We present a comprehensive benchmark evaluation focusing on vision-based control. For this purpose, we adapt the robot manipulator control environments from the meta-world benchmark suite [17]. Our experiments reveal that `VELAP` surpasses existing approaches by a significant margin in terms of episode success rate. We attribute this performance gain to it's ability to overcome local value optima through global exploration, in contrast to the prevalent approach of optimizing over a fixed horizons.

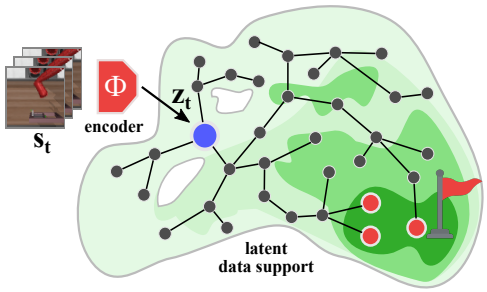


Figure 1: Our method grows a search tree in the latent space to globally explore reward-maximizing paths (blue:start, red:goal nodes, green: estimated values).

2 Related Work

Learning to control from visual input is becoming increasingly popular due to the wide availability of inexpensive sensors and the generic representational format of images. A considerable number of work uses deep generative models [18] to generate future images and plan actions via model predictive control [19, 20, 21, 22, 23]. These methods allow visual inspection of paths by humans, but this is accompanied by the difficulty of generating and evaluating high-resolution video sequences of many future steps. In this work, we instead follow the latent planning paradigm, which bypasses the need to synthesize high-dimensional samples and enables farsighted search with lower computational costs. Several works learn image distance metrics using unsupervised learning [24, 5] or RL [25, 26] to build an environment map that can be searched to generate visual paths. Map-based approaches have shown impressive results for navigation in static environments, but often do not scale well to complex settings with object interaction.

The work in [3] presents a model-based RL agent which uses a recurrent latent state space model to map image observations to a lower-dimensional space. Time-discrete state-action trajectories are optimized in the latent space using the Cross-Entropy Method (CEM) [8] with a learned dynamics model. In a follow-up work, [4] uses a differentiable latent planner to efficiently learn behaviors by propagating analytic value gradients back in time. Similarly, [11, 27] implement a differentiable planner that optimizes latent trajectories via gradient descent. [28] trains a goal-conditioned RL agent which generates future subgoal states through trajectory optimization within the latent space of a Variational Autoencoder (VAE) [29]. [6] introduces the concept of collocation to model-based RL for visual robot manipulation and optimizes state-action sequences directly with a second-order

optimizer. [10] solve dexterous manipulation tasks through latent trajectory optimization using a reward-weighted adaptation of MPPI [9]. Similarly, the method in [7] uses temporal difference values estimates as cost within an MPPI-based latent policy. The line of work in [30, 31, 32] combines model-based RL with Monte Carlo Tree Search (MCTS) [33] for long-horizon decision-making in vision-based tasks such as playing Atari games. While MCTS is mainly designed for discrete settings, [34] presents an adaptation for continuous action spaces where the search and the policy improvement are based on action sampling.

To address the offline RL setting, [35] modifies the model-based agent in [10]. Their approach ensures adherence to the data support by sampling actions from an imitation learning policy, thus addressing the issue of out-of-distribution actions [16]. [36] designed a hierarchical agent for sparse reward manipulation tasks. A VAE-based manager policy predicts subgoal states which a worker policy must achieve within a fixed contingent of steps. In [37], a RRT-like [38] latent planner is trained from high-dimensional offline data. Compared to ours, their method relies on collision checking data and is not designed for the more general reward-based setting. [13] leverages play data obtained from a human operator to train a task-conditioned policy which guides a tree search in a learned latent space. Our method is most related to the one recently presented in [14]. The authors introduce a sampling-based latent planner similar to the classical Expansive Space Trees (EST) algorithm [39]. Nevertheless, notable differences arise in terms of the problem types we tackle, resulting in the adoption of distinct sets of tools. While their approach is confined to goal-reaching navigation tasks, our method accommodates more general task specifications by leveraging sparse rewards. To accomplish this, we employ value-based reinforcement learning to (a) jointly optimize representations for planning alongside the control policy, (b) integrate learned heuristics for node and action selection during planning, and (c) identify suitable goals based on value estimates. These advancements significantly enhance the capabilities and versatility of our method, surpassing the scope of the previous work.

3 Preliminaries

MDPs and Offline RL A Markov decision process (MDP) is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, $\mathcal{P}(s'|s,a)$ are state dynamics, $r(s,a)$ is a scalar reward function, and γ is a discount factor. The goal of reinforcement learning [1] (RL) is to find a policy $\pi(a|s)$ that maximizes the expected discounted future reward $R[\tau]$ over all trajectories τ given an initial state distribution p_0 and induced by π, i.e., to optimize $\mathbb{E}_\pi[R[\tau]]$. The problem of offline RL [15] arises when training from a fixed dataset $\mathcal{D}$ consisting of trajectories generated by a behavioral policy π_β. Due to the limited coverage of $\mathcal{D}$ across the state-action space, effectively addressing the adverse consequences of poor approximations outside the data support becomes crucial in the development of offline RL methods [16].

Hindsight data relabeling Relabeling data has emerged as a popular technique in goal-conditioned off-policy RL [40, 41, 42, 43, 44] for the purpose of enhancing training efficiency. The underlying idea behind hindsight relabeling is to transform unsuccessful trajectories into successful ones by retrospectively modifying their goals [40]. This approach extends to offline trajectory datasets, where relabeling can automatically generate experiences for learning state-reaching behaviors [45, 46]. Specifically, failed transitions can be relabeled by designating the subsequent state as the desired goal and adjusting the corresponding reward accordingly. A connection between hindsight relabeling and contrastive learning was recently discussed in [47].

Sampling-based motion planning Sampling-based motion planners [12] compute feasible paths between points in a robot configuration space. At the core, these methods explore and construct a graphical representation of the continuous search space. The rapidly-exploring random tree (RRT) [38] is a widely use single-query planner, particularly suitable for scenarios with varying environments. It incrementally expands a tree structure by alternately sampling collision-free states from the robot's configuration space and connecting them to the nearest neighbor in the tree. Once a node

reaches the vicinity of the goal, a solution path is determined by backtracing to the root of the tree. Instead of sampling states from the configuration space, the expansive space trees (EST) planner [39] generates new states by expanding existing tree nodes using random actions.

4 Value-guided Expansive Latent Trees

In this section, we detail the elements that comprise `VELAP`, our proposed offline RL planning agent.

Problem definition We are interested in solving sparse reward continuous control tasks from high-dimensional inputs. For this purpose, we choose the example of visual control for a state space $\mathcal{S}$ and action space $\mathcal{A} = \mathbb{R}^{d_{\text{action}}}$. $\mathcal{S} = \mathbb{R}^{W \times H \times C \times N}$ describes sequential image data where W is the image width, H the height, C the channel dimension and N the number of frames. Note that we employ the MDP formulation, hence assume that states $s \in \mathcal{S}$ are informative to predict the distribution of future states. A sparse binary reward $r : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$ is designed to provide a positive value only upon successful completion of the task. To train our model, we are provided with an offline dataset $\mathcal{D}$ consisting of recorded transitions obtained from a sub-optimal policy.

Components To tackle the specified problem, we propose a model-based reinforcement learning agent that incorporates a tree-based search, inspired by ESTs [39], within a learned representation space. Our preference for EST over an RRT-based approach [38] is driven by the reasoning that the expansion step in ESTs eliminates the need for a global state sampler. It should be noted that learning such generative models can be challenging, as they require high-fidelity predictions to prevent negative assessment of out-of-distribution samples. Our approach involves several key components outlined in Eq. 1. The encoder ϕ maps input states to latent encodings, while the dynamics model h predicts future latent states based on actions, serving as a tool for expanding the search tree during planning. A local policy π^l is trained to navigate between neighboring states in the tree. The global policy π^g determines optimal actions with respect to our MDP task. During planning, we will use Q^l to derive a distance proxy between states and Q^g to estimate the remaining number of steps to the goal. Among various actor-critic offline RL methods available, we select TD3-BC [48] due to its robustness and ease of implementation. To improve the predictions of Q^l and measure value uncertainty, we employ an ensemble of n_{ens} Q-heads $\{Q^l_1, ..Q^l_{n_{\text{ens}}}\}$ similar to [49] (see App. B). For the following, we use k to denote the k-th ensemble memeber and define $Q^{i,j}_{\text{min}} := \min\{Q^l_k(z_i, z_j, \pi^l(z_i, z_j))\}_{k=1}^{n_{\text{ens}}}$ as the minimum and $Q^{i,j}_{\text{std}} := \text{std}(\{Q^l_k(z_i, z_j, \pi^l(z_i, z_j))\}_{k=1}^{n_{\text{ens}}})$ as the standard deviation of the ensemble predictions at a particular step. Finally, a conditional generative model g is used to facilitate sampling actions from the state-conditioned action distribution.

$$
\begin{aligned}
\text{State encoder:}\ & \phi : \mathcal{S} \rightarrow \mathcal{Z} \\
\text{Dynamics:}\ & h : \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{Z} \\
\text{Action model:}\ & g : \mathcal{Z} \times \mathbb{R}^m \rightarrow \mathcal{A} \\
\text{Local policy:}\ & \pi^l : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathcal{A} \quad Q^l : \mathcal{Z} \times \mathcal{Z} \times \mathcal{A} \rightarrow \mathbb{R} \\
\text{Global policy:}\ & \pi^g : \mathcal{Z} \rightarrow \mathcal{A} \quad Q^g : \mathcal{Z} \times \mathcal{A} \rightarrow \mathbb{R}
\end{aligned}
\tag{1}
$$

Alignment of representation and planner To compute feasible paths in $\mathcal{Z}$, our state representations must favor the approximation of long-horizon dynamics. In particular, our model should not only predict the next state with high accuracy but provide useful future waypoint predictions over several time steps. At the same time, our goal is to solve the MDP, hence a good representation should also learn relevant features which ease optimization of the control behaviors π^g and π^l. Existing work on model-based RL often use surrogate metrics for model learning (e.g. mean-squared error on dynamics or pixel reconstruction loss) which do not enforce compliance with the actual control performance. In the literature this misalignment between the environment model and planner [50] has been shown to hamper the performance of the controller. To overcome this issue, we train a state encoder ϕ through joint optimization with the latent dynamics, local and global RL policies, leading to the overall training objective $\mathcal{L}_{\text{model}}$ (Eq. 2). In this regard, $\mathcal{L}_{Q^l_k}$ describes the temporal difference (TD) value loss for training Q^l_k, $\mathcal{L}_{Q^g}$ the TD loss for training Q^g and $\mathcal{L}_h$ the

loss function for the dynamics model h. The hyperparameters c_0 and c_1 represent weighting factors.

$$\mathcal{L}_{\text{model}} = \frac{1}{n_{\text{ens}}} \sum_k \mathcal{L}_{Q_k^l} + c_0 \cdot \mathcal{L}_{Q^g} + c_1 \cdot \mathcal{L}_h \tag{2}$$

$$\mathcal{L}_{Q_k^l} = \mathop{\mathbb{E}}_{\mathcal{D}'}[(Q_k^l(z_t, z^g, a_t) - (r_t + \gamma Q_{\text{min}}^{t+1,g}))^2] \tag{3}$$

$$\mathcal{L}_{Q^g} = \mathop{\mathbb{E}}_{\mathcal{D}}[(Q^g(z_t, a_t) - (r_t + \gamma Q^g(z_{t+1}, \pi^g(z_{t+1}))))^2] \tag{4}$$

In accordance to the standard TD3-BC training objective, we simultaneously optimize the corresponding policies π^l and π^g in $\mathcal{L}_{\pi^l}$ and $\mathcal{L}_{\pi^g}$ (Eq. 5). Note that this step is done by alternating between optimizing $\mathcal{L}_{\text{model}}$ and policy improvement while the encoder parameters are kept fixed during the policy update [1].

$$\begin{aligned} \mathcal{L}_{\pi^l} &= \mathop{\mathbb{E}}_{\mathcal{D}'}[-Q_{\text{min}}^{t,g}] + c_2 \cdot \mathop{\mathbb{E}}_{\mathcal{D}'}[(\pi^l(z_t, z^g) - a_t)^2] \\ \mathcal{L}_{\pi^g} &= \mathop{\mathbb{E}}_{\mathcal{D}}[-Q^g(z_t, \pi^g(z_t))] + c_3 \cdot \mathop{\mathbb{E}}_{\mathcal{D}}[(\pi^g(z_t) - a_t)^2] \end{aligned} \tag{5}$$

To provide data for training π^l and Q_k^l, we synthesize a dataset of state-reaching experiences $\mathcal{D}'$ by relabeling the transitions in $\mathcal{D}$. More specifically, we employ hindsight goal relabeling [45, 46] to sample goals $s^g \in \mathcal{S}$ and use a binary reward to indicate success (see App. B). For training the dynamics model, we choose a contrastive loss objective (CPC) [52] similar to [14]. In practice, we found this approach to work better in maintaining accurate long-term predictions compared to a standard mean-squared error objective (App. E.3).

Exploration strategy We formulate our RL decision-making task as a geometric planning problem and seek a planner that efficiently explores the latent space searching for high-valued states. Similar to [14], we follow the concept of ESTs [39] and iteratively expand the current set of nodes through action sampling. The tree $\mathcal{T}=(\mathcal{V}, \mathcal{E})$ can be seen as a growing memory of latent nodes $\mathcal{V} \subset \mathcal{Z}$ and transitions $\mathcal{E} \subset \mathcal{Z} \times \mathcal{Z}$. The core mechanism behind our expansion strategy is summarized in Alg. 1. We first initialize $\mathcal{T} = (\mathcal{V}=\{z_{\text{init}}\}, \mathcal{E}=\emptyset)$ where $z_{\text{init}} \in \mathcal{Z}$ is the latent encoding of the current state $s_{\text{init}} \in \mathcal{S}$ obtained from ϕ. For n_{iter} steps, a node z_{expand} is drawn using a categorical distribution P_{node} defined over $\mathcal{V}$. Starting from z_{expand}, the dynamics h rolls out a short n_{sim}-step state sequence given actions drawn from our generative model g (or π^g). Since Q_k^l estimates the return for trying to reach a node under sparse rewards, a temporal distance proxy is given by $\log_\gamma Q_k^l$. To account for value approximation errors [16], we will use the minimum value among the ensembles predictions to obtain a conservative distance estimate. After every n_{sim}-step expansion with h, we determine if the transition from z_{exp} to z_{new} is feasible by checking if $Q_{\text{min}}^{\text{exp,new}}$ is above a threshold $\tau_{\text{discard}}^{\text{low}}$. If it lies below this threshold, we discard z_{new}. Secondly, we also reject it if the corresponding value of $Q_{\text{std}}^{\text{exp,new}}$ is above a threshold $\tau_{\text{discard}}^{\text{std}}$. The purpose of this second rejection step is filter states in which the epistemic uncertainty, i.e. model uncertainty, is high and thereby avoid the evaluation of high-uncertainty areas, for example outside the support of the latent data distribution. Lastly, we also want to determine if the newly generated node is sufficiently novel

Algorithm 1 Node sampling and tree expansion

Given: z_{init}, n_{iter}, n_{sim}, $\tau_{\text{discard}}^{\text{neigh}}$, $\tau_{\text{discard}}^{\text{std}}$ g, h, π^g, Q^g, Q_k^l, π^l
Initialize: $\mathcal{V} \leftarrow \{z_{\text{init}}\}$, $\mathcal{E} \leftarrow \emptyset$
for n_{iter} steps **do**
  Sample node z_{exp} from $\mathcal{V}$ given $P_{\text{node}}(\mathcal{V})$
  $z_{\text{new}} \leftarrow z_{\text{exp}}$
  Simulate forward using dynamics for n_{sim} steps
  for n_{sim} steps **do**
    Sample action $a \sim g(.|z_{\text{new}})$ (or $a = \pi^g(z_{\text{new}})$)
    $z_{\text{new}} \leftarrow h(z_{\text{new}}, a)$
  end for
  Reject node if too close to existing one in the tree or
  if the value uncertainty is too high
  if $Q_{\text{min}}^{\text{exp,new}} > \tau_{\text{discard}}^{\text{low}}$ and $Q_{\text{std}}^{\text{exp,new}} < \tau_{\text{discard}}^{\text{std}}$ **then**
    if $\max\{Q_{\text{min}}^{i,\text{new}} | z_i \in \mathcal{V}\} < \tau_{\text{discard}}^{\text{high}}$ **then**
      Add new node to tree
      $V \leftarrow V \cup \{z_{\text{new}}\}$; $E \leftarrow E \cup \{z_{\text{exp}\rightarrow\text{new}}\}$
    end if
  end if
end for

[1] We optimize the state representation during the critic update instead of the policy improvement step as motivated by the empirical analysis in [51].

from the existing ones $\mathcal{T}$ and discard it otherwise. We found this step to be necessary to keep computation at a moderate level by sparsifying the tree. For that, we discard z_{new} if $\max\{Q^{i,\text{new}}_{\text{min}} | z_i \in \mathcal{V}\}$ is above a threshold $\tau^{\text{high}}_{\text{discard}}$. In other words, we find the closest neighbor z_{neigh} in the tree and reject z_{new} if there already exists a node which can transition to it within few steps. If z_{new} passes the previous stages, it is added to $\mathcal{T}$, i.e. $\mathcal{V} \leftarrow \mathcal{V} \cup \{z_{\text{new}}\}$ and $\mathcal{E} \leftarrow \mathcal{E} \cup \{z_{\text{exp}\rightarrow\text{new}}\}$. For more details on the expansion step see App. B.5.

Node sampling So far we haven't defined the node sampling distribution P_{node}. To achieve fast and task-oriented exploration, we combine two sampling heuristics based on (a) the inverse number of neighbors around each node and (b) the state-action value of Q^g. (a) leads to quick exploration of undiscovered latent states, while (b) drives the planner towards high-reward areas. For both parts, we use exponential weighting as shown in Eq. 6 and 7. In this regard, n_i^{neigh} corresponds the number of incoming neighbors for a node ($\mathcal{V}^{\text{neigh}}_{\rightarrow \text{i}}$). We compose P_{node} by sampling according to P_{sparse} with probability p^{sparse} and from P_{value} with p^{sparse} (otherwise random uniform).

$$P_{\text{sparse}}(z_i) = \frac{e^{-n_i^{\text{neigh}}/T_{\text{sparsity}}}}{\sum_{z_j \in \mathcal{V}} e^{-n_j^{\text{neigh}}/T_{\text{sparsity}}}} \quad (6)$$

$$P_{\text{value}}(z_i) = \frac{e^{Q_i^g/T_{\text{value}}}}{\sum_{z_j \in \mathcal{V}} e^{Q_j^g/T_{\text{value}}}} \quad (7)$$

Action sampling The model g mimics the state-dependent action distribution in the data and is represented by a conditional VAE [29]. Sampling actions from g reduces the evaluation of undesired state-actions pairs which were not observed in $\mathcal{D}$. To help our planner discover task-relevant areas quicker, we further predict actions with π^g probability p^{policy}.

Planning objective and control We presented a planner which builds a sparse tree representation in the latent space while being guided by value and sparsity heuristics. To pick the best path, we must define an objective that ranks all paths in $\mathcal{T}$. In practice, we first identify G, the set of goal nodes based on thresholding the value predicted by Q^g with τ_{goal}. Among the nodes in G, we then choose the path g^* which is associated with the minial path length computed based on using Q^l_{min} as a distance proxy between subsequent states (Eq. 8). If $G = \emptyset$, we simply pick the one in that leads to the highest-valued state (based on Q^g).

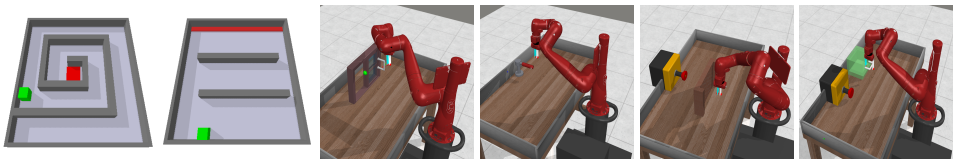
$$g^* = \arg\min_{g \in \mathcal{G}} c(g) \qquad \text{with} \qquad c(g) = \sum_{(i,j) \in \mathcal{E}_g} \log_\gamma Q^{i,j}_{\text{min}} \quad (8)$$

To use our planner in a control setting, we embed it into a model-predictive control loop (appendix: Alg. 2). The controller queries our planner every n_{replan} steps and uses the local policy π^l to navigate between nodes in the planned sequences of latent states. If close enough, the controller switches to the next waypoint, which we determine by checking the value of $Q^{i,j}_{\text{min}}$ against a threshold τ_{wp}.

5 Experiments

5.1 Empirical evaluation in simulation

Baselines To assess the effectiveness of `VELAP`, we measured its performance against the following baselines. *Behavioral cloning* (**BC**), a simple but often successful method that imitates the behavioral policy π_β using a supervised learning objective. We evaluated a second version of this approach only trained on the subset of successful trajectories ($\mathcal{D}^*$). **TD3-BC** [48], an adaptation of the Twin Delayed DDPG algorithm [53] which circumvents the effect value overestimation by adding an imitation objective to the policy update. **IQL** [54], a state-of-the-art offline RL baseline. **MPPI** [9] provides the base planning algorithm in various model-based RL methods ([10, 7]). We consider an implementation for the offline learning setup which uses TD3-BC within the cost update during optimization. **MBOP** [35], a model-based agent which uses an imitation learning policy to bias the action sampling in MPPI. **IRIS** [36], an offline RL method particularly designed for sparse reward settings. It uses a hierarchical decomposition of the policy for which a manager predicts feasible subgoals given future candidate states (n-step horizon) sampled from a generative model (cVAE) which a worker policy must achieve. We also examine **IRIS (multi-step)**, where the set of subgoals



(a) SpiralMaze (b) ObstacleMaze (c) WindowClose (d) FaucetClose (e) ButtonWall (f) DrawerButton

Figure 2: Evaluation environments adapted from meta-world robotics benchmark.

Table 1: Success rates (%) on test scenarios with unseen object variations (except *SprialMaze*).

METHOD	BC	BC ($\mathcal{D}^*$)	TD3-BC	IQL	MPPI	MBOP	IRIS	IRIS (MULTI-STEP)	VELAP
SPIRAL MAZE	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	15 ± 31	$\mathbf{94 \pm 3}$
OBSTACLE MAZE	0 ± 0	15 ± 6	35 ± 22	6 ± 3	83 ± 11	40 ± 25	50 ± 25	62 ± 14	$\mathbf{97 \pm 2}$
WINDOW	0 ± 0	34 ± 11	16 ± 8	9 ± 4	70 ± 7	23 ± 4	69 ± 3	43 ± 20	$\mathbf{78 \pm 4}$
FAUCET	0 ± 0	36 ± 6	13 ± 7	8 ± 4	41 ± 7	33 ± 2	10 ± 2	3 ± 1	$\mathbf{51 \pm 12}$
BUTTONWALL	0 ± 0	0 ± 0	2 ± 2	0 ± 0	9 ± 10	0 ± 0	35 ± 5	8 ± 8	$\mathbf{76 \pm 9}$
DRAWERBUTTON	0 ± 0	0 ± 0	0 ± 0	1 ± 0	0 ± 0	0 ± 0	5 ± 3	0 ± 0	$\mathbf{11 \pm 3}$

is generated by shooting a future state sequence using the cVAE. To establish a fair comparison and disentangle the effects of the representation and planner, we use the same representations, dynamics models across all methods. Further details are provided in App. D.

Tasks We consider the simulated visuomotor control tasks depicted in Fig. 2. In *SpiralMaze*, the velocity of a block robot is controlled in order to navigate from the outermost point of a maze to the inner region. This task was designed to require farsighted planning, as the temporal distance to the goal is approximately 150 steps. In the *ObstacleMaze* environment, the block robot must travel to the opposite wall of the room while two random positioned obstacles appear in the center of the workspace. Additionally, we evaluate the *WindowClose*, *FaucetClose* environments from the meta-world [17] benchmark. As done in [6], we use sparse binary rewards and render images from a static camera. Moreover, we propose two new settings *ButtonWall* and *DrawerButton*. In *ButtonWall*, the robot must first navigate around a wall (varying position) and then press a button. To solve the task *DrawerButton*, the agent must first close a drawer and then press a button (both randomly initialized). Our training data $\mathcal{D}$ consists of random trajectory data and a small number of noisy expert demonstrations. We use a latent space of size 32 and RGB images of resolution 64×64. For details on the environments, data collection and hyperparameters see App. B.

Results An overview of the numberical benchmark evaluation is given in Table 1. `VELAP` consistently outperform the baselines across all environment in terms of average episode success rate. The improvements are particularly visible in tasks which require far-reaching planning such as *SpiralMaze* and *ButtonWall*. These results support that our tree-based exploration strategy is indeed effective at solving sparse-reward RL problems. Fig. 3 illustrates a 2D embedding of a latent path for the *SpiralMaze* task computed with `VELAP`. It suggests that our method explores global solutions and identifies one which reaches through the entire maze. Fig. 4 demonstrates similar capabilities for the *ObstacleEnv* task. Further, it supports that our representation accommodates for the different locations of obstacles, creating latent spaces that mirror the topology of the underlying state space. Ablations results concerning the embedding and dynamics model are provided in App. E.

5.2 Physical robot experiments

To test our method under realistic conditions, we designed two manipulation tasks using a low-cost robot arm (Fig. 5). In the first, the robot must push a sponge into the marked region. In the second, the robot is holding a rope and must navigate around a cylindrical object and then place it held end of the rope into the colored region. We directly control the endeffector position and orientation

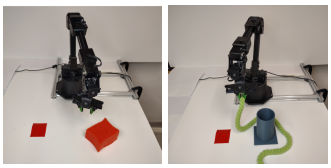


Figure 5: Physical manipulation of a sponge (left) and rope (right).

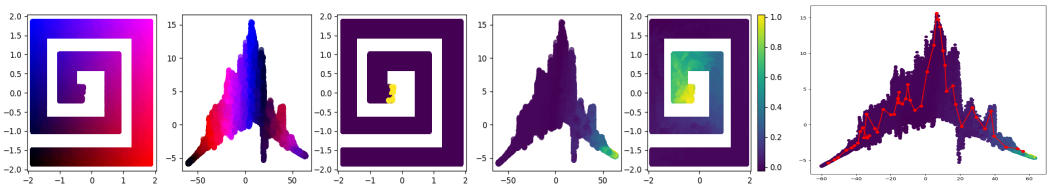


Figure 3: *SpiralMaze* (a) xy coordinates of block robot (b) Isomap embeddings of latent space (c) reward for xy-coordinates (d) estimated Q-values (e) predicted rewards(f) planned latent path computed with `VELAP`

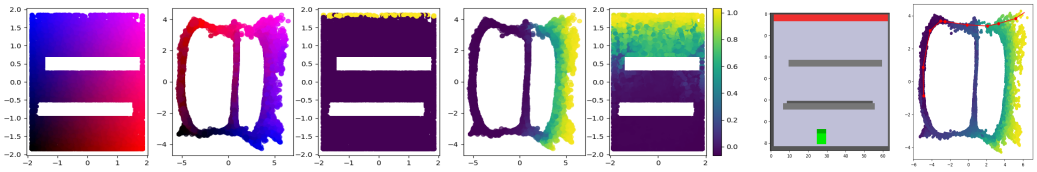


Figure 4: *ObstacleMaze* (a) xy coordinates of block robot (b) Isomap embeddings of latent space (c) reward for xy-coordinates (d) estimatd Q-values (e) predicted rewards (f) planned latent path computed with `VELAP`

resulting in a 4-dimensional action space. Both tasks are challenging in terms of perception and control as the initial configuration of the objects are randomized. Further details on the both tasks and data collection are given (App. E.1). The results of a comparison with BC, BC ($\mathcal{D}^*$) and IRIS are presented in Table 11 (App. E.1). Our method surpasses the performance of its baselines achieving $14/20$ successful episode on the sponge manipulation and $8/20$ on the rope manipulation task. Videos demonstrations can be found at https://sites.google.com/view/velap-corl/home.

6 Limitations and Future Directions

Our method lays the groundwork for future enhancements. While `VELAP` was designed for offline RL, it can be adapted to the online setting by interleaving data collecting and learning [3, 6]. It could also enhance sample efficiency by improving policy and critic updates through planning. [55, 4, 30]. Currently, `VELAP` uses deterministic dynamics and encoder models, limiting it to fully-observable MDPs. By incorporating probabilistic transition models and state filtering approaches (similar to [3, 56]), it can be extended to partially observable settings. While our method outperforms existing baselines, it still struggles with complex tasks which we account to the remaining difficulty in estimating accurate latent dynamics for long-horizon planning. Presently, our planner selects paths by minimizing the distance to high-reward areas. To further minimize the effects of model inaccuracies, the planning strategy could incorporate uncertainty propagation and assessment in the tree branches. Notably, we discovered that in more complex tasks, such as *DrawerButton* and the rope manipulation scenario, failures often occurred when the agent moved to the final goal region without completing the required subtask, like closing the drawer or maneuvering the rope around the pole. We believe that improved handling of uncertainties, along with risk-aware reward measures, could potentially lessen the planner's greedy exploitation of model errors in the future. In this context, classical belief space planning [57] provide another potential improvement avenue.

7 Conclusion

We presented `VELAP`, a model-based planning agent for sparse rewards offline RL. Unlike most planners in model-based RL, we propose a tree-based exploration algorithm similar to the type of sampling-based planners in robot motion planning. An empirical comparison on visual control tasks demonstrated significant improvements of our method over existing baselines. We hope that our results will stimulate further research on the integration of sampling-based planning, representation learning and model-based RL.

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[2] T. M. Moerland, J. Broekens, and C. M. Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[3] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565, 09–15 Jun 2019.

[4] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[5] K. Liu, T. Kurutach, C. Tung, P. Abbeel, and A. Tamar. Hallucinative topological memory for zero-shot visual planning. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6259–6270. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/liu20h.html.

[6] O. Rybkin, C. Zhu, A. Nagabandi, K. Daniilidis, I. Mordatch, and S. Levine. Model-based reinforcement learning via latent-space collocation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9190–9201. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/rybkin21b.html.

[7] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

[8] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer. Chapter 3 - the cross-entropy method for optimization. In C. Rao and V. Govindaraju, editors, *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pages 35–59. Elsevier, 2013. doi:https://doi.org/10.1016/B978-0-444-53859-8.00003-5. URL https://www.sciencedirect.com/science/article/pii/B9780444538598000035.

[9] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016. doi:10.1109/ICRA.2016.7487277.

[10] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.

[11] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4732–4741. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/srinivas18b.html.

[12] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[13] B. Ichter, P. Sermanet, and C. Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=yhy25u-DrjR.

[14] R. Gieselmann and F. T. Pokorny. Latent planning via expansive tree search. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=zSdz5scsnzU.

[15] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[16] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

[17] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf.

[20] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control. *arXiv e-prints*, art. arXiv:1812.00568, Dec. 2018.

[21] L. Yen-Chen, M. Bauza, and P. Isola. Experience-embedded visual foresight. In *Conference on Robot Learning*, 2019.

[22] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gzR2VKDH.

[23] S. Tian, S. Nair, F. Ebert, S. Dasari, B. Eysenbach, C. Finn, and S. Levine. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=UcoXdfrORC.

[24] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

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

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

[27] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn. Unsupervised visuomotor control through distributional planning networks. In *Proceedings of Robotics: Science and Systems*, Freiburgim-Breisgau, Germany, June 2019. doi:10.15607/RSS.2019.XV.020.

[28] S. Nasiriany, V. Pong, S. Lin, and S. Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/c8cc6e90ccbff44c9cee23611711cdc4-Paper.pdf.

[29] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2014.

[30] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, dec 2020. doi:10.1038/s41586-020-03051-4. URL https://doi.org/10.1038%2Fs41586-020-03051-4.

[31] J. Schrittwieser, T. K. Hubert, A. Mandhane, M. Barekatain, I. Antonoglou, and D. Silver. Online and offline reinforcement learning by planning with a learned model. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=HKtsGW-lNbw.

[32] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao. Mastering atari games with limited data. In *NeurIPS*, 2021.

[33] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. CG'06, page 72–83, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3540755373.

[34] T. Hubert, J. Schrittwieser, I. Antonoglou, M. Barekatain, S. Schmitt, and D. Silver. Learning and planning in complex action spaces. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4476–4486. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/hubert21a.html.

[35] A. Argenson and G. Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OMNB1G5xzd4.

[36] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420. IEEE, 2020.

[37] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094, 2018. doi:10.1109/ICRA.2018.8460730.

[38] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[39] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3, pages 2719–2726 vol.3, 1997. doi:10.1109/ROBOT.1997.619371.

[40] 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*, 30, 2017.

[41] B. Eysenbach, X. Geng, S. Levine, and R. R. Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in neural information processing systems*, 33:14783–14795, 2020.

[42] A. Levy, R. Platt, and K. Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

[43] T. Davchev, O. O. Sushkov, J.-B. Regli, S. Schaal, Y. Aytar, M. Wulfmeier, and J. Scholz. Wish you were here: Hindsight goal selection for long-horizon dexterous manipulation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=FKp8-pIRo3y.

[44] A. Li, L. Pinto, and P. Abbeel. Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767, 2020.

[45] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. C. Julian, C. Finn, and S. Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1518–1528. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/chebotar21a.html.

[46] J. Li, C. Tang, M. Tomizuka, and W. Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022. doi:10.1109/LRA.2022.3190100.

[47] B. Eysenbach, T. Zhang, S. Levine, and R. Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=vGQiU5sqUe3.

[48] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[49] G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Neural Information Processing Systems*, 2021.

[50] N. Lambert, B. Amos, O. Yadan, and R. Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

[51] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021.

[52] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[53] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[54] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[55] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi:10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

[56] N. Srivastava, W. Talbott, M. B. Lopez, S. Zhai, and J. Susskind. Robust robotic control from pixels using contrastive recurrent state-space models. *arXiv preprint arXiv:2112.01163*, 2021.

[57] V. Indelman, L. Carlone, and F. Dellaert. Towards planning in generalized belief space. In *International Symposium on Robotics Research (ISRR)*, December 2013.

[58] J. Bialkowski, S. Karaman, and E. Frazzoli. Massively parallelizing the rrt and the rrt. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3513–3518, 2011. doi:10.1109/IROS.2011.6095053.

[59] B. Ichter. *Massive Parallelism and Sampling Strategies for Robust and Real-Time Robotic Motion Planning*. PhD thesis, Stanford University, 2018.

[60] J. B. Tenenbaum, V. de Silva, and J. C. Langford. 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.

[61] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[62] M. Raisi, A. Noohian, L. McCutcheon, and S. Fallah. Value summation: A novel scoring function for mpc-based model-based reinforcement learning. *ArXiv*, abs/2209.08169, 2022.