Hierarchies of Planning and Reinforcement Learning for Robot Navigation

Wöhlke, J.; Schmitt, F.; van Hoof, H.

Link to publication

Hierarchies of Planning and Reinforcement Learning for Robot Navigation

Jan Wöhlke[1,2], Felix Schmitt[1], and Herke van Hoof[2]

Abstract— Solving robotic navigation tasks via reinforcement learning (RL) is challenging due to their sparse reward and long decision horizon nature. However, in many navigation tasks, high-level (HL) task representations, like a rough floor plan, are available. Previous work has demonstrated efficient learning by hierarchal approaches consisting of path planning in the HL representation and using sub-goals derived from the plan to guide the RL policy in the source task. However, these approaches usually neglect the complex dynamics and sub-optimal sub-goal-reaching capabilities of the robot during planning. This work overcomes these limitations by proposing a novel hierarchical framework that utilizes a trainable planning policy for the HL representation. Thereby robot capabilities and environment conditions can be learned utilizing collected rollout data. We specifically introduce a planning policy based on value iteration with a learned transition model (VI-RL). In simulated robotic navigation tasks, VI-RL results in consistent strong improvement over vanilla RL, is on par with vanilla hierarchal RL on single layouts but more broadly applicable to multiple layouts, and is on par with trainable HL path planning baselines except for a parking task with difficult non-holonomic dynamics where it shows marked improvements.

I. INTRODUCTION

In recent years, applying reinforcement learning (RL) to sequential decision making problems, has led to some remarkable successes like playing games with super-human performance [25], [28] and mastering robotic manipulation [16] or locomotion [26] tasks. However, application of RL to robotic navigation tasks like avoiding obstacles and adapting the motion to environmental influences, in order to reach a goal, is still challenging. This is due to the long decision horizon and the naturally sparse reward that is only received for successful goal-reaching. Frequently used proxy rewards such as negative Euclidean distance to the goal can result in undesired side effects, like trapping the robot at obstacles. Therefore, several orthogonal concepts have been suggested to improve data-efficiency in sparse reward RL, including curriculum learning [30], [31], [29], advanced exploration [32], [34], or intrinsic motivation [33], [35]. Our approach is related to hierarchical RL (HRL) [22], [11], [23], [24] tackling long decision horizons by using a hierarchal policy.

In these "vanilla" HRL approaches, abstraction is mainly achieved temporally by choosing different time scales for the hierarchy levels. Furthermore, the literature mainly focused on solving single instance tasks, like a specific environment layout [11]. In real world robotic navigation tasks, the
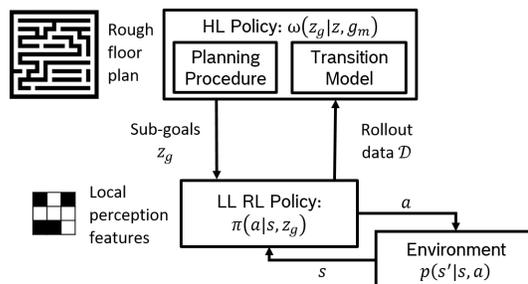

Fig. 1: Our Hierarchical Planning and RL Framework

environment layout or goal location might be different each time. At the same time, we often have additional information such as a rough floor plan with the goal location available. Using sampling-based planners [8], [9] to obtain waypoints led to impressive real world results [10] but a simulator that accurately represents the full navigation state space needs to be queried during planning.

While the task might be very complex in the original, continuous "low-level" (LL) state and action space, including the full robot and environment dynamics, we observe that the task is much easier in an abstract "high-level" (HL) floor plan representation where we can easily determine positional sub-goals. In these cases, previous work [40], [45] has demonstrated efficient learning by hierarchal approaches consisting of planning in the abstract HL representation and guiding the RL agent in the source task by sub-goals derived from the plan. However, they did not adapt the HL planning to the LL policy capabilities. In robotic navigation tasks, it is not realistic to assume that the robot can always reach a proposed sub-goal since the given HL representation might neglect important information about the robot or environmental dynamics as well as the current capabilities of the LL policy that affects the sub-goal-reaching.

Formalizing the assumptions about continuous state and action space robotic navigation tasks given a rough floor plan of the environment, we derive a novel hierarchical framework (see Fig. 1) that overcomes these limitations by utilizing a trainable reactive policy for planning in the HL representation instead of purely conducting shortest path planning. This has the benefit that failures of the LL policy to reach the sub-goals are considered by adapting the HL planning utilizing gathered rollout data. Differentiable planning modules approximating value iteration (VI) [1], [5] suggest simple integration with RL but have limitations with respect to stability with longer planning horizons [1] or properties of the assumed transition model [5]. Therefore,

[1]Bosch Center for Artificial Intelligence, Renningen, Germany.   JanGuenter.Woehlke@de.bosch.com, Felix.Schmitt@de.bosch.com

[2]UvA-Bosch Delta Lab, University of Amsterdam, Amsterdam, Netherlands. h.c.vanhoof@uva.nl

we additionally present a HL planning policy based on exact VI featuring a learned transition model.

The key contributions of our work are:

- Proposing a novel HRL framework derived from stated assumptions regarding domain knowledge combining planning in a HL state space with sub-goal guided RL in the original state space. This framework encompasses both the previous shortest path HL planning approaches as well as novel, trainable HL policies.
- Presenting a novel framework instance employing value iteration in the HL state space using a transition model learned from data gathered during LL RL (VI-RL).
- A detailed analysis in which scenarios and specific implementation learned HL policies provide benefits over pure shortest path plans.

We will work out these contributions by investigating the following research hypothesis:

H.1: Hierarchically combining planning in an abstract HL representation with goal-conditioned RL significantly increases the data-efficiency over flat "vanilla" RL in continuous dynamics, sparse reward, long horizon tasks.

H.2: Learning a HL transition model reflecting the RL policy's capabilities is necessary to handle environmental influences, or complex, non-holonomic dynamics.

H.3: Transition models should consider both the HL state and action (chosen sub-goal); purely state-dependent models are insufficient to model non-holonomic constraints.

II. Related Work

First ideas to generate sub-goals and hierarchically combine policies date back more than two decades [20], [21], [19]. Recent HRL approaches like Option Critic [22], Feudal Networks (FuN) [23], or HiRO [11] make use of these ideas combining them with modern neural network policy architectures and RL algorithms: FuN and HiRO combine a HL policy that set sub-goals at a low frequency with a LL policy that receives intrinsic rewards for reaching these sub-goals. However, conventional HRL approaches do not consider available additional task knowledge.

Using sampling-based planners on a given map with reachability estimators learned from training experience to set sub-goals for local RL navigation policies were proposed in [8] (PRM) and [9] (RRT). However, an accurate simulator of the environment that allows for evaluating configurations in the original state space is necessary to execute the planning.

Several works combine planning on an abstract level with LL RL. They use different abstract HL representations like a symbolic representation [45], [47], an abstract representation based on a predicate logic [40], or skills learned in an unsupervised fashion [46]. The approaches presented in [40], [45] do not adapt their HL planning to the LL RL policy capabilities. In [46], the skills are not further updated in the target task and might therefore be bound to the environment and specific conditions under which they were discovered. While in [47] the HL planning is adapted, the approach was only demonstrated for discrete state and action spaces but not for complex continuous ones relevant for robotic navigation.

Instead of pre-defining the HL representation, several works suggest ways to generate them from data: [41] built an abstract representation of the task by connecting collected experience in the replay buffer to a graph using a learned reachability estimator. A series of sub-goals is then planned by searching the obtained graph for the shortest path to the goal. This requires that the goal state can be connected to the obtained experience. In navigation tasks with large layouts where the goals state can be far away from experienced states or novel layouts this is not necessary feasible. Other approaches for learning plannable representations [36], [37], [38], [39] struggle with similar challenges. Nevertheless, combining this orthogonal line of work with our hierarchical framework could be interesting future research.

Instead of learning a HL representation, various work in robotic navigation has focused on using an a-priori provided map. In [44] a base-level map is segmented to obtain a higher-level map representation for planning. Value iteration (VI) is carried out on different levels. In [42] sub-environments are dynamically constructed as series of spatial-based state abstractions such that a tessellation of the environment results. Gupta et al. [43] map first-person views to a top-down belief map of the world to apply a differentiable neural network planner on it. These works do not consider the continuous LL state space underlying robotic navigation problems and specifically for [44] do not learn the transition model for VI but assume an optimistic one.

It is, however, crucial to adapt the planning to the robot capabilities and the environmental dynamics, using rollout data, when combining sub-goal planning with RL. Here, differentiable planning modules [1], [6], [5], [7] address this issue. Requiring a discrete, ideally grid-like, environment representation, they enable learning value-function-based planning via backpropagation. The Value Iteration Network (VIN) [1] is a CNN architecture to approximate several recursions of VI planning. It has been generalized to arbitrary graphs [2] and multiple levels of hierarchical planning [3]. The instability issues, especially with respect to long planning horizons, due to the recurrent structure are addressed in [4] and [5]. MVProp [5] mitigates unstable iteration by moving all trainable parameters into purely state-dependent propagation factors (normalized to $[0, 1]$). This implicit assumption on the transition dynamics can harm the performance, for example in non-holonomic navigation tasks.

III. Problem Statement

In this work, we consider MDPs m sampled from a distribution of Markov Decision Processes (MDPs) $\mathcal{M}$. All MDPs m share the same state and action space $\mathcal{S}$ and $\mathcal{A}$, the same goal-based reward function $r(s, g_m)$ with a goal g_m sampled from a goal space $g_m \sim \mathcal{S}_{g,m} \subseteq \mathcal{S}$, and the same time horizon T as well as discount factor γ. Each MDP m has initial states s_0 sampled from an initial state space $s_0 \sim \mathcal{S}_{0,m} \subseteq \mathcal{S}$. Furthermore, each MDP has its own dynamics $\mathcal{P}_m$, but we assume that the dynamics are locally the same, i.e. the probability of a state transition is given by

the same function of MDP instance specific features values $\mathcal{P}_m\left(s'|a,s\right) = f\left(\phi_{s,m}, \phi_{s',m}, \phi_{a,m}\right)$.

The task is to find a goal-conditioned policy $\pi\left(a|s,g_m\right)$ that maximizes the expected returns under the distribution of MDPs, the goal and initial state distributions, and the dynamics:

$$\max_{\pi(a|s,g_m)} \mathbb{E}_{m\sim\mathcal{M},g_m\sim\mathcal{S}_{g,m},s_0\sim\mathcal{S}_{0,m},\mathcal{P}_m,\pi}\left[\sum_{t=0}^{T}\gamma^t r\left(s_t,g_m\right)\right] \tag{1}$$

We focus on sparse reward signals, which are common for robotic navigation problems. Therefore, the reward function is of the following form: $r\left(s,g_m\right) = \mathbb{I}_{d(s,g_m)\leq\epsilon} + c$ with $d\left(\cdot,\cdot\right)$ being a (weighted) distance function, ϵ the tolerance, and c a constant to potentially offset the reward signal.

Assumptions on Domain Knowledge: We will first explain at a robotic navigation task example the additional assumptions made to utilize a rough floor plan in our hierarchical framework. Afterwards, we will formally state them. As we will see later, these assumptions can often be fulfilled in practical applications and making use of them enables solving Eq. 1 more data-efficiently (see Sec. V).
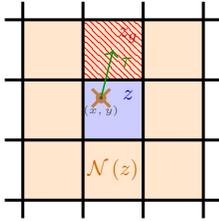

Fig. 2: Exemplary High-Level State Space Illustration

Imagine an ant-like robot navigating a continuous dynamics environment as depicted in Fig.2. Given a rough floor plan, we can obtain an abstract, finite HL state space $\mathcal{Z}$ by partitioning the map into the shown grid of cells. Localizing the robot in x, y coordinates, the corresponding grid cell (shown in blue) determines the current HL state z (*Asm.1*). In order to plan the next sub-goal, we need to have knowledge about local HL state neighborhood relations. In our example, all adjacent grid cells are the neighboring HL states $\mathcal{N}\left(z\right)$ (*Asm.2*). Given a sub-goal z_g in the HL state space (shaded in red), we need a way to condition our local LL sub-goal navigation policy acting in the original continuous state space on the HL sub-goal. Therefore a target vector τ (green) is calculated by taking the difference between the center of a grid cell and the robots' x, y coordinates such that $s+\tau\left(s,z\right)$ is the LL sub-goal (*Asm.3*). The HL state neighborhood $\mathcal{N}\left(z\right)$ must contain at least all directly reachable HL states. Else there is a risk that the only existing route to the goal is missed due to the state space abstraction (*Asm.4*). To take into account and learn the robots capabilities for navigating various layouts, we need to be able to model HL state transitions using local features. In case of our grid representation, integer values indicating the presence or absence of obstacles or the terrain type for the (eight)

neighboring grid cells $z' \in \mathcal{N}\left(z\right)$ can serve as such local map features $\psi_{z,m}$ (*Asm.5*).

In summary, we formally assume:

Asm.1: There exists a finite HL state space $\mathcal{Z}$ with known $z = f_{\mathcal{Z}}\left(s\right)$.

Asm.2: The neighbors $\mathcal{N}\left(z\right)$ of a HL state z are known.

Asm.3: There exists a function returning for a state s and a HL state z a target vector $\tau\left(s,z\right) = f_\tau\left(s,z\right)$ such that $f_{\mathcal{Z}}\left(s + \tau\left(s,z\right)\right) = z$.

Asm.4: $\mathcal{N}\left(z\right) \subset \mathcal{Z}$ contains at least all HL states directly reachable from z.

Asm.5: Local features ψ of any HL state are available. The probability of transitioning from z to $z' \in \mathcal{N}\left(z\right)$ under any policy conditioned on a goal $z_g = f_{\mathcal{Z}}\left(g\right)$ can be approximated: $p_{\mathcal{P}_m,\pi(a|s,g)}\left(z'|z,z_g\right) \approx g\left(\psi_{z,m}, \psi_{z',m}, \psi_{z_g,m}\right)$.

IV. HIERARCHICAL POLICY AND LEARNING FRAMEWORK

We introduce a novel policy and learning framework that efficiently solves Eq. 1 by utilizing the domain knowledge formalized in the assumptions stated above. It consists of:

- A *goal-conditioned HL policy* $\omega\left(z_g|z,g_m\right)$ that selects a HL sub-goal $z_g \in \mathcal{N}\left(z\right)$ from the set of neighbors. This policy generally operates at a lower frequency than the LL state space transitions: A new sub-goal z_g is selected if either the previous is reached or the number of time steps exceeds a threshold $T_{z_g} \ll T$.
- A *sub-goal-conditioned LL policy* $\pi\left(a|s, f_\tau\left(s,z_g\right) + s\right)$ pursuing the sub-goals $\tau(s,z_g)+s$ set by the HL policy, instead of the MDP goal state g_m. In order to generalize across sub-goals as well as MDPs m, we model this policy by a function of the target vector and state features $\pi\left(a|s, f_\tau\left(s,z_g\right) + s\right) = h\left(a, f_\tau\left(s,z_g\right), \psi_{z,m}\right)$.

We assume only some domain knowledge, like a rough floor plan, to be given. The complex continuous LL dynamics of the robot interacting with its environment are unknown.

In order to handle these unknowns, we introduce a hierarchical framework that allows training the HL sub-goal planning and the LL policy jointly using the same experience.

A. Learning the Low-Level (LL) Policy

During a rollout $\{(s_t, a_t, r_t, s_{t+1})_{t=1,...,T}\}$ in the environment, we track the corresponding HL states and actions $\{(z_t, z_{g,t}, z_{t+1})_{t=1,...,T}\}$ as well as the time steps when a new HL sub-goal $z_{g,i}$ is selected $\{(t_i)_{i=1,...,n}\}$. We train a LL sub-goal conditioned policy $\pi_\theta\left(a|s, f_\tau\left(s,z_g\right) + s\right)$ with parameters θ using the following intrinsic sub-goal-returns:

$$\hat{R}_i = \sum_{t=t_i}^{t_{i+1}} \gamma^{t-t_i}\mathbb{I}_{f_{\mathcal{Z}}(s_{t+1})=z_{g,t}}, \tag{2}$$

An optimal LL policy with respect to these intrinsic rewards results in *recursive optimality* [18]. This means that the LL policy is only "locally" optimal with respect to its sub-goal z_g. Even with an optimized HL policy the combined HL + LL policy can be sub-optimal with respect to Eq. 1 but will due to *Asm.2* not miss an existing path to the goal. However,

smaller "sub-episodes" of maximum length T_{z_g} improve the data-efficiency and enable generalization across layouts [18]. While we employ TRPO [14] for LL policy optimization, in our experiments, any other RL algorithm can be used as well.

B. Learning the High-Level (HL) Policy

Value iteration planning with explicit dynamics model (learning) [VI-RL]: The discrete nature and small size of the HL state space $\mathcal{Z}$ and the HL action space $\mathcal{Z}_g$ combined with the known neighbor relations render value iteration (VI) tractable. To apply VI, we derive the HL reward function $r_{\mathcal{Z}}(z) = \mathbb{I}_{z=f_{\mathcal{Z}}(g_m)} + c$ from the given goal-based reward function of the MDP. Furthermore, we need a model of the HL state transitions $p_{\pi_\theta}(z'|z, z_g)$ that takes into account the dynamics induced by the sub-goal-conditioned LL policy $\pi_\theta(a|s, f_\tau(s, z_g) + s)$. We consider two different approaches for obtaining such a model:

Optimistic Model [VI-RL OM]: Similar to [40], [44] this model optimistically assumes that a sub-goal z_g always results in a transition to the corresponding HL neighbor: $p_{\pi_\theta}(z'|z, z_g) \approx \mathbb{I}_{z'=z_g}$. Using this model results in pure shortest path planning in grid-world domains.

Learned Model [VI-RL]: Using the collected HL transition data $\{(z_{t_i}, z_{g,t_i}, z_{t_{i+1}})_{i=1,...,n}\}$, we can learn a state feature-based model $p_{\pi_\theta}(z'|z, z_g) \approx \hat{g}_\lambda(\psi_{z,m}, \psi_{z',m}, \psi_{z_g,m})$, with parameters λ, of the transitions. As we assumed these features to govern the HL dynamics, expressive models like deep neural networks should be able to closely approximate the dynamics. This has the benefit that the HL policy takes into account the LL policy capabilities. We train the model to maximize the sum of the log-likelihoods $\log \hat{g}_\lambda(\psi_{z,m}, \psi_{z',m}, \psi_{z_g,m})$ regarding the observed data.

Using the reward function $r_{\mathcal{Z}}$ and the HL dynamics model p_{π_θ}, we employ VI to obtain value functions $V(z), Q(z, z_g)$ that are used by the HL policy for sub-goal selection: $\omega(z_g|z, g_m) = \mathbb{I}_{z_g = \arg\max_{z' \in \mathcal{N}(z)} Q(z,z')}$. During training, we employ an epsilon-greedy-variant of this policy for rollouts and use a replay-buffer for de-correlating training data, which we empirically found to result in more robust training. See Alg. 1 for algorithmic implementation details of VI-RL.

Differentiable planning module with implicitly learned dynamics model (MVPROP-RL): In RL tasks with a discrete, grid-like state space representation, a differentiable recursive architecture like MVProp [5] can be used to approximate the value function. In our setting, we can therefore employ MVProp to approximate the HL value function $V_\pi(z)$ with a model $T \hat{V}_\lambda^K(z)$ with T being the time horizon. $\hat{V}_\lambda^K(z)$ is the result of K recursions where the value of neighboring states is propagated to state z based on a state-dependent propagation factor $p_\lambda(z)$ representing simplified transition dynamics. We train the model $T \hat{V}_\lambda^K(z)$ with collected data $\{(z_{t_i}, z_{g,t_i}, z_{t_{i+1}})_{i=1,...,n}\}$ using similar to [5] a TD-0 loss (but omitting the importance weighting, which we empirically found unnecessary): $l(z, z', \lambda) = \left(\left(r(z) + \left(1 - \mathbb{I}_{f_{\mathcal{Z}}(g_m)=z}\right) \gamma T \hat{V}_{\lambda_{\text{target}}}^K(z') \right) - T \hat{V}_\lambda^K(z) \right)^2$.

Algorithm 1: VI-RL with LL TRPO Policy

Input: Distribution of MDPs $\mathcal{M}$, domain knowledge ($\mathcal{Z}$, functions $f_{\mathcal{Z}}(s)$, $f_\tau(s, z)$, and $\mathcal{N}(z)$, and features $\psi_{z,m}$), HL transition model $\hat{g}_\lambda$, LL goal-conditioned policy π_θ, HL and LL experience buffers D_H, D_L, LL horizon T_{z_g}.

1 **for** $j = 1$ *to* j_{max} **do**
2   **while** $|D_L| < $ TRPO batch size **do**
3   *# Sample environment, start, and goal*
4       $m \sim \mathcal{M}, s_{0,m} \sim \mathcal{U}(\mathcal{S}_{0,m}), g_m \sim \mathcal{U}(\mathcal{S}_{g,m})$
5   *# Conduct value iteration (VI)*
6       $V(z), Q(z, z') = \text{VI}(\mathcal{Z}, \mathcal{N}(z), \hat{g}_\lambda, g_m)$
7   *# Obtain HL policy*
8       $\omega(z_g|z, g_m) = \mathbb{I}_{z_g = \arg\max_{z' \in \mathcal{N}(z)} Q(z,z')}$
9   *# Carry out RL episode*
10      $t = 0, i = 0 \; t_0 = 0, z_{t_0} = f_{\mathcal{Z}}(s_{0,m})$
11      $z_{g,t_0} \sim \text{epsilon-greedy}(\omega(z_{t_0}))$
12   **while** $t < T$ *and* $d(s_t, g_m) > \epsilon$ **do**
13         $a_t \sim \pi_\theta(s_t, f_\tau(s_t, z_{g,t_i}))$
14         $s_{t+1} \sim \mathcal{P}_m(s_t, a_t), r_t = \mathbb{I}_{f_{\mathcal{Z}}(s_{t+1})=z_{g,t_i}}$
15         $D_L \leftarrow \{s_t, a_t, r_t, s_{t+1}\}$
16   **if** $(f_{\mathcal{Z}}(s_t) = z_{g,t_i}) \vee (t - t_i = T_{z_g})$ **then**
17   *# Sub-goal reached*
18            $t_{i+1} = t, z_{t_{i+1}} = f_{\mathcal{Z}}(s_{t+1})$
19            $D_H \leftarrow \{z_{t_i}, z_{g,t_i}, z_{t_{i+1}}\}$
20            $z_{g,t_{i+1}} \sim \text{epsilon-greedy}(\omega(z_{t_{i+1}}))$
21            $i \leftarrow i + 1$
22   *# Update transition model using Adam [48]*
23      $\lambda_{j+1} \leftarrow \text{Adam}(\lambda_j, D_H)$
24   *# Update RL policy*
25    $\theta_{j+1} \leftarrow \text{TRPO}(\theta_j, D_L), D_L \leftarrow \emptyset$

We obtain the HL policy as $\omega(z_g|z, g_m) = \mathbb{I}_{z_g = \arg\max_{z' \in \mathcal{N}(z)} \hat{V}_\lambda^K(z')}$. Similar to VI-RL we employ an epsilon-greedy exploration and a replay-buffer.

V. EXPERIMENTAL EVALUATION

So far, we formalized assumptions for robotic navigation tasks given a rough floor plan and formulated a framework that aims at accelerating learning by utilizing this domain knowledge. In the following, we empirically investigate the research hypotheses stated in Sec. I.[1]

A. Environments

We consider three simulated continuous dynamics robotic navigation domains, depicted in Fig. 3, which demonstrates the broad feasibility of the assumptions (Sec. III). The use of simulation enables us to perform the number of training rollouts and independent trainings (seeds) required by RL.

Four Rooms: A continuous dynamics (x, y-position and -velocity) point mass robot needs to navigate through the

---

[1]Code is planned to be made available at https://github.com/boschresearch/Hierarchies-of-Planning-and-Reinforcement-Learning-for-Robot-Navigation is planned

(a) Four Rooms
$T = 100$, $T_{z_g} = 2$

(b) MuJoCo Ant Mazes
$T = 2000$, $T_{z_g} = 20$

(c) Vehicle Parking
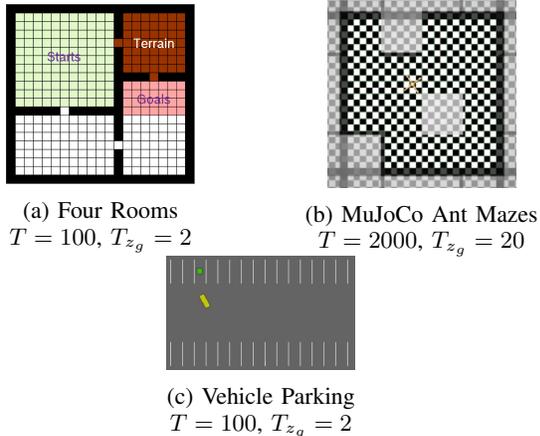$T = 100$, $T_{z_g} = 2$

Fig. 3: Simulation Environments Overview

environment in Fig. 3a. Starting positions are sampled in the top left, whereas goal locations are sampled in the upper part of the bottom right room. The continuous x, y-accelerations are the LL control inputs. The discrete HL state representation results from tiling the layout into the shown 21×21 tiles. The top-right room features terrain that uniformly slows down the robot motion. For the HL transition model of VI-RL, we train an MLP architecture. It receives HL state features, the terrain information (free, wall, terrain) for the eight neighbors, and an integer indicating the relative neighbor chosen as sub-goal, as inputs. A vector of transition probabilities for the eight neighbors is the output.

MuJoCo Ant Mazes: A MuJoCo [13] ant robot with complex continuous dynamics needs to navigate one of 25 randomly sampled 6×6 block environments with different obstacle structure. We obtain the HL state space by subdividing each block into four tiles (24×24 overall). The resolution is a hyper-parameter to be reasonably chosen considering robot dimensions and sub-goal horizon T_{z_g}.

Vehicle Parking [17]: A non-holonomic bicycle dynamics vehicle needs to park in a randomly selected slot, starting in the middle of the parking lot. Unlike the other domains, the robot orientation ϑ is part of the goal specification and an integral part of the state and action space. We obtain a 3D HL state space by tiling the x, y-position into 24×12 tiles and the orientation into eight $45°$ segments. With no obstacles, any HL position has identical (terrain) features. Therefore, we remove these from the input of the HL transition model of VI-RL. Instead, we augment the in- and output to account for the ϑ segments. For MVPROP(-RL), we use a 3D-CNN. The robot receives a sparse reward for successfully parking.

B. Baselines

Vanilla RL (BSL): A flat, non-hierarchical architecture, using the goal-conditioned RL policy with a target vector to the final MDP goal g_m instead of a sub-goal target vector.

RRT-Waypoints (RRT-WP): Inspired by [9], an RRT planner generates a path of waypoints from start to goal in 2D x, y-space using the map. The LL goal-conditioned RL policy is trained to reach the waypoint selected as sub-goal.

HiRO [11] is a state-of-the-art HRL approach. It uses a conventional MLP HL policy, trained via off-policy RL (TD3 [15]) and operating in the continuous state space like the LL policy. The LL policy is trained to follow target vectors generated (at the same frequency as in VI-/MVPROP-RL) by the HL. While in [11] both policy levels use distance-based reward shaping, we provide the HL policy with the environmental sparse rewards to ensure comparability with the other approaches and avoid trapping the robot at obstacles.

Aiming for a fair comparison, we use the same number of hierarchy levels (except BSL) and the same goal-conditioned LL TRPO [14] policy architecture for all approaches.

C. Simple Point-Mass Navigation

In order to investigate under which circumstances learning HL transition dynamics implicitly (MVPROP-RL) or explicitly (VI-RL) is necessary (see research hypothesis *H.2*), we conduct experiments in a simplistic environment with a focus on the relative effects of environmental conditions instead of absolute performance. We compare the goal-reaching probability across training for *Four Rooms* domain with (Fig. 4b) and without (Fig. 4a) the hard-to-traverse terrain in the top-right room. The distance-wise shortest paths from the starts to the goals traverse the top-right room. With the terrain, however, the optimal policy takes the "longer" routes through the bottom-left room. Looking at the results, vanilla RL (BSL) is not able to reach the goals (*H.1*). VI-RL and MVPROP-RL, like HIRO HER, achieve 100% goal-reaching. Using the capacious transition model necessary for the more difficult realistic tasks (e.g. Sec. V-D), VI-RL learns a bit slower. Comparing VI-RL with the learned transition model to VI-RL OM with the optimistic model, it turns out that the performance is similar without the terrain, whereas the goal-reaching of VI-RL OM drops to less than 20% with the terrain, because the episodes 'time out' while crossing the terrain. We witness a less severe performance drop of RRT-WP. Due to the sampling-based planning, by chance, a path through either the top-right or bottom-left room is found, resulting in 50% goal-reaching. In conclusion, these evaluations in the simplistic environment show that learning the HL transition dynamics is crucial for reacting to environmental conditions, like terrain-type, which is necessary in realistic set-ups (*H.2*).

D. Robotic Maze Navigation

In real application, a robot needs to be able to navigate a broad range of environment layouts. Therefore, we evaluate the algorithms on the *MuJoCo Ant Mazes* environment, featuring 25 maze layouts randomly picked for every rollout. We excluded HiRO as it was not designed to solve multiple task instances[2]. The results are depicted in Fig. 4c. VI-RL and MVPROP-RL perform similar, clearly outperforming the vanilla RL baseline BSL (see research hypothesis *H.1*). Surprisingly, using the optimistic model (OM) does not decrease the VI-RL performance, despite the very complex

---

[2]Additional experiments showed that replacing the HL MLP policy with a CNN does not successfully enable HiRO to handle multiple layouts.

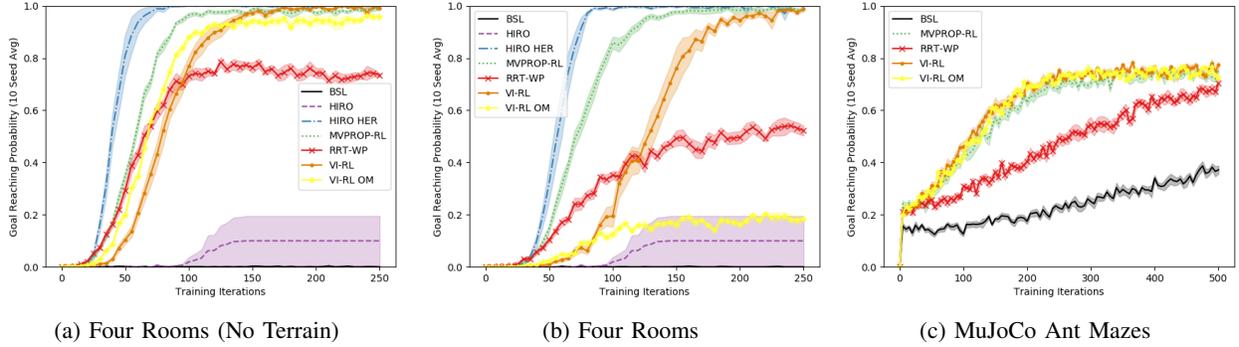(a) Four Rooms (No Terrain) (b) Four Rooms (c) MuJoCo Ant Mazes

Fig. 4: Maze Navigation; Line: Mean across 10 Seeds; Shaded Area: Standard Error

ant dynamics. It is apparently similarly difficult to move the ant in any specific direction throughout the training. Hence, purely planning the shortest path is sufficient in this static setting (*H.2*). In this light, the slower learning of RRT-WP is explained by the RRT only approximating the shortest path.

E. Non-Holonomic Vehicle Parking

As previously stated, learning *Vehicle Parking* adds the difficulty of non-holonomic dynamics to the learning task, involving a tight coupling of location and orientation. These non-holonomic constraints must be taken into account during HL planning, in order to master the task (see research hypothesis *H.3*). The results are depicted in Fig. 5. Apart from VI-RL, basically, all other approaches are not able to significantly increase the goal-reaching probability in reasonable time. VI-RL quickly picks up the pace and reaches over 80% goal-reaching probability. This task provides another yet very different example to the *Four Rooms* with terrain domain where learning the transition dynamics is crucial: The purely shortest path planning of VI-RL OM does not consider the non-holonomic constraints and therefore does not provide effective guidance (*H.2*). Furthermore, the experiment shows the limitations of the assumptions underlying the transition model of MVProp [5]: Only learning state-dependent propagation factors p that do not take into account to which specific neighboring HL state the robot shall transition is not sufficient. Without any obstacles in this environment, in principal, all HL states are traversable, however, specific transitions like going 'north' while the robot orientation is 'west' may be difficult to execute. In contrast, the learned HL transition model in VI-RL explicitly takes these non-holonomic constraints into account and hence is able to guide the LL policy to reach the goal (*H.3*).

VI. Conclusion

In this work, we formalized the domain knowledge typically available in robotic navigation tasks, like a rough floor plan of the environment, into assumptions. Based on these, we derived a novel framework combining planning in a high-level (HL) state space with sub-goal guided RL in the original continuous state space. Furthermore, we presented a
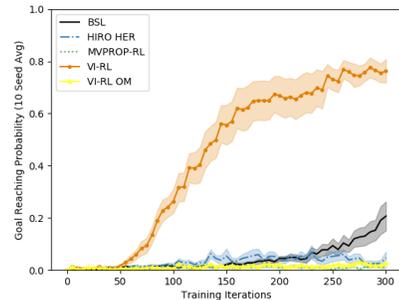

Fig. 5: Vehicle Parking; Line: Mean across 10 Seeds; Shaded Area: Standard Error

novel, specific instance of our framework building on value iteration planning in the HL state space using a transition model learned from data gathered during RL (VI-RL).

In several difficult sparse reward robotic navigation tasks, we examined several research hypotheses with the following outcomes: (*H.1*) Hierarchically combining planning and RL enables handling random task instances while achieving significant data-efficiency gains over non-hierarchical RL. (*H.2*) Comparing different HL planning approaches, we found that in static 2D navigation environments learning HL transition models is, surprisingly, often not necessary and pure shortest path planning performs equally well. However, learning the dynamics becomes crucial when the navigation needs to take into account environmental influences like the terrain type. (*H.3*) In the practically relevant non-holonomic vehicle parking task featuring a more complex HL state space, we uncovered limitations of the optimistic model as well as the purely state-dependent transition model of MVProp planning. Our novel algorithm VI-RL overcomes these limitations by employing a transition model accounting for the selected HL sub-goal state and significantly outperforms all baselines.

In future work, an interesting direction of research could be to also learn the transformation of the original state space into the HL state space instead of designing it based on the domain knowledge. This way, the presented approaches could be even more broadly applicable.

REFERENCES

[1] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, Value Iteration Networks, Advances in Neural Information Processing Systems, 2016, p.2154–2162.

[2] S. Niu, S. Chen, H. Guo, C. Targonski, M. Smith, and J. Kovačević, Generalized Value Iteration Networks: Life Beyond Lattices, AAAI, 2018.

[3] D. Schleich, T. Klamt, and S. Behnke, Value Iteration Networks on Multiple Levels of Abstraction, Robotics: Science and Systems, 2019.

[4] L. Lee, E. Parisotto, D. Chaplot, E. Xing, and R. Salakhutdinov, Gated Path Planning Networks, International Conference on Machine Learning, 2018, p.2947–2955.

[5] N. Nardelli, G. Synnaeve, Z. Lin, P. Kohli, P. Torr, and N. Usunier, Value Propagation Networks, International Conference on Learning Representations, 2019.

[6] J. Oh, S. Singh, and H. Lee, Value Prediction Network, Advances in Neural Information Processing Systems, 2017, p.6118–6128.

[7] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, Universal Planning Networks, International Conference on Machine Learning, 2018.

[8] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, PRM-RL: Long-Range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning, International Conference on Robotics and Automation, 2018, p.5113–5120.

[9] H. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, Rl-RRT: Kinodynamic Motion Planning via Learning Reachability Estimators from RL Policies, IEEE Robotics and Automation Letters, vol. 4, no. 4, 2019, p.4298–4305.

[10] A. Francis, A. Faust, H. Chiang, J. Hsu, J. Kew, M. Fiser, and T. Lee, Long-Range Indoor Navigation with PRM-RL, IEEE Transactions on Robotics, 2020.

[11] O. Nachum, S. Gu, H. Lee, and S. Levine, Data-Efficient Hierarchical Reinforcement Learning, Advances in Neural Information Processing Systems, 2018, p.3303–3313.

[12] O. Nachum, S. Gu, H. Lee, and S. Levine, Near-Optimal Representation Learning for Hierarchical Reinforcement Learning, arXiv preprint arXiv:1810.01257, 2018.

[13] E. Todorov, T. Erez, and Y. Tassa, Mujoco: A Physics Engine for Model-Based Control, International Conference on Intelligent Robots and Systems, 2012, p.5026–5033.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, Trust Region Policy Optimization, International Conference on Machine Learning, 2015, p.1889–1897.

[15] S. Fujimoto, H. van Hoof, and D. Meger, Addressing Function Approximation Error in Actor-Critic Methods, International Conference on Machine Learning, 2018, p.1582–1591.

[16] M. Andrychowicz, F. Wolski, A. Ray, J.Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, Hindsight Experience Replay, Advances in Neural Information Processing Systems, 2017, p.5048–5058.

[17] E. Leurent, An Environment for Autonomous Driving Decision-Making, GitHub, 2018.

[18] M. Ghavamzadeh and S.Mahadevan, Hierarchically Optimal Average Reward Reinforcement Learning, International Conference on Machine Learning, 2002, p.195–202.

[19] R. Sutton, D. Precup, and S. Singh, Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning, Artificial intelligence, Elsevier, vol. 112, no. 1-2, 1999, p.181–211.

[20] J. Schmidhuber, Learning to Generate Sub-Goals for Action Sequences, Artificial Neural Networks, 1991, p.967–972.

[21] P. Dayan and G. Hinton, Feudal Reinforcement Learning, Advances in Neural Information Processing Systems, 1993, p.271–278.

[22] P. Bacon, J. Harb, and D. Precup, The Option-Critic Architecture, AAAI, 2017.

[23] A. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, FeUdal Networks for Hierarchical Reinforcement Learning, International Conference on Machine Learning, 2017, p.3540–3549.

[24] A. Levy, G. Konidaris, R. Platt, and K. Saenko, Learning Multi-Level Hierarchies with Hindsight, International Conference on Learning Representations, 2019.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, and others, Human-Level Control through Deep Reinforcement Learning, Nature, Nature Publishing Group, vol. 518, no. 7540, 2015, p.529.

[26] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, High-Dimensional Continuous Control Using Generalized Advantage Estimation, International Conference on Learning Representations, 2016.

[27] S. Levine, C. Finn, T. Darrell, and P. Abbeel, End-to-End Training of Deep Visuomotor Policies, The Journal of Machine Learning Research, JMLR.org, vol. 17, no. 1, 2016, p.1334–1373.

[28] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and others, A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play, Science, American Association for the Advancement of Science, vol. 362, no. 6419, 2018, p.1140–1144.

[29] J. Wöhlke, F. Schmitt,and H. van Hoof, A Performance-Based Start State Curriculum Framework for Reinforcement Learning, Autonomous Agents and MultiAgent Systems, 2020, p.1503–1511.

[30] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, Reverse Curriculum Generation for Reinforcement Learning, Conference on Robot Learning, 2017, p.482–495.

[31] P. Klink, H. Abdulsamad, B. Belousov, and J. Peters, Self-Paced Contextual Reinforcement Learning, Conference on Robot Learning, 2019.

[32] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, Deep Exploration via Bootstrapped DQN, Advances in Neural Information Processing Systems, 2016, p.4026–4034.

[33] D. Pathak, P. Agrawal, A. Efros, and T. Darrell, Curiosity-Driven Exploration by Self-Supervised Prediction, Computer Vision and Pattern Recognition Workshops, 2017, p.16–17.

[34] Z. Hong, T. Shann, S. Su, Y. Chang, T. Fu, and C. Lee, Diversity-Driven Exploration Strategy for Deep Reinforcement Learning, Advances in Neural Information Processing Systems, 2018, p.10489–10500.

[35] N. Haber, D. Mrowca, S. Wang, L. Fei-Fei, and D. Yamins, Learning to Play With Intrinsically-Motivated, Self-Aware Agents, Advances in Neural Information Processing Systems, 2018, p.8388–8399.

[36] T. Kurutach, A. Tamar, G. Yang, S. Russell, and P. Abbeel, Learning Plannable Representations with Causal InfoGAN, Advances in Neural Information Processing Systems, 2018, p.8733–8744.

[37] D. Corneil, W. Gerstner, and J. Brea, Efficient Model-Based Deep Reinforcement Learning with Variational State Tabulation, International Conference on Machine Learning, 2018, p.1049–1058.

[38] T. Kipf, E. van der Pol, and M. Welling, Contrastive Learning of Structured World Models, International Conference on Learning Representations, 2019.

[39] E. van der Pol, T. Kipf, F. Oliehoek, and M. Welling, Plannable Approximations to MDP Homomorphisms: Equivariance under Actions, Autonomous Agents and MultiAgent Systems, 2020, p.1431–1439.

[40] M. Eppe, P. Nguyen, and S. Wermter, From Semantics to Execution: Integrating Action Planning with Reinforcement Learning for Robotic Tool Use, arXiv preprint arXiv:1905.09683, 2019.

[41] B. Eysenbach, R. Salakhutdinov, and S. Levine, Search on the Replay Buffer: Bridging Planning and Reinforcement Learning, Advances in Neural Information Processing Systems, 2019, p.15246–15257.

[42] A. Ma, M. Ouimet, and J. Corts, Hierarchical Reinforcement Learning via Dynamic Subspace Search for Multi-Agent Planning, Autonomous Robots, 44, 2020, p.485?503.

[43] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, Cognitive Mapping and Planning for Visual Navigation, IEEE Conference on Computer Vision and Pattern Recognition, 2017, p.2616–2625.

[44] Z. Zivkovic, B. Bakker, and B. Krose, Hierarchical Map Building and Planning Based on Graph Partitioning, IEEE International Conference on Robotics and Automation, 2006, p.803–809.

[45] K. Yamamoto, T. Onishi, and Y. Tsuruoka, Hierarchical Reinforcement Learning with Abductive Planning, arXiv preprint arXiv:1806.10792, 2018.

[46] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, Dynamics-Aware Unsupervised Discovery of Skills, arXiv preprint arXiv:1907.01657, 2019.

[47] D. Lyu, F. Yang, B. Liu, and S. Gustafson, SDRL: Interpretable and Data-Efficient Deep Reinforcement Learning Leveraging Symbolic Planning, AAAI, vol. 33, 2019, p.2970–2977.

[48] D. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980, 2014.