Complex Robotic Manipulation via Graph-Based Hindsight Goal Generation

Zhenshan Bing, Matthias Brucker, Fabrice O. Morin, Rui Li, Xiaojie Su, *Senior Member, IEEE,*
Kai Huang*, and Alois Knoll, *Senior Member, IEEE*

***Abstract*—Reinforcement learning algorithms such as hindsight experience replay (HER) and hindsight goal generation (HGG) have been able to solve challenging robotic manipulation tasks in multi-goal settings with sparse rewards. HER achieves its training success through hindsight replays of past experience with heuristic goals, but under-performs in challenging tasks in which goals are difficult to explore. HGG enhances HER by selecting intermediate goals that are easy to achieve in the short term and promising to lead to target goals in the long term. This guided exploration makes HGG applicable to tasks in which target goals are far away from the object's initial position. However, the vanilla HGG is not applicable to manipulation tasks with obstacles because the euclidean metric used for HGG is not an accurate distance metric in such environment. Although with the guidance of a hand-crafted distance grid, grid-based HGG can solve manipulation tasks with obstacles, a more feasible method that can solve such tasks automatically is still on demand. In this paper, we propose graph-based hindsight goal generation (G-HGG), an extension of HGG selecting hindsight goals based on shortest distances in an obstacle-avoiding graph, which is a discrete representation of the environment. We evaluated G-HGG on four challenging manipulation tasks with obstacles, where significant enhancements in both sample efficiency and overall success rate are shown over HGG and HER. Videos can be viewed at https://videoviewsite.wixsite.com/ghgg.**

***Index Terms*—Reinforcement learning, hindsight experience replay, robotic arm manipulation, path planning.**

I. Introduction

In recent years, deep reinforcement learning (RL) [18] has made significant progress, with RL concepts being successfully applied to decision-making problems in robotics, which include, but not limited to, navigation tasks [40], helicopter control [24], hitting a baseball [27], door opening [5], screwing a cap onto a bottle [19], object manipulation [1], and many other control tasks [39], [17], [41], [36], [21], [42]. To train a meaningful policy for such tasks, neural networks are used as function approximators for learning a value function to optimize a long-term expected return. Estimation of the expected return is based on a reward function, which is highly associated with the tasks and therefore must be thoroughly shaped for policy optimization.

In most real-world applications, where a concrete representation of efficient or even admissible behavior is unknown, the design of a reward function is challenging and time-consuming, thereby hindering the wide applications of RL. Consequently, algorithms that can support learning from sparse rewards are highly desirable, e.g., a binary signal indicating successful task completion; since sparse rewards are easy to derive from the task definition without further engineering. Andrychowicz et al. [1] introduced an algorithm called "hindsight experience replay (HER)", which improves the success of off-policy RL algorithms in multi-goal RL problems with sparse rewards. The concept behind HER is to first learn with hand-crafted heuristic intermediate goals that are easy to achieve, and then continue with more difficult goals. Precisely, HER constructs hindsight goals from previously achieved states, replays known trajectories with these hindsight goals, and trains the goal-dependent value function based on the results. While HER has proven to work efficiently in environments where goals can be easily reached through random explorations [28], it fails to reach goals that are far away from initial states and hard to reach. Due to random explorations and the heuristic choice of hindsight goals from achieved states, hindsight goals keep being distributed around the initial state, far away from the target goals, which will never be reached since no positive reward signal is obtained.

Hindsight goal generation (HGG) [30] tackles the aforementioned problem by using intermediate hindsight goals as an implicit curriculum to guide exploration towards target goals. HGG aims at choosing hindsight goals that are both easy to achieve and challenging enough to help the function approximator learn how to achieve the target goals eventually. In HGG, the choice of hindsight goals is based on two criteria: the current value function (as much knowledge as possible about how to reach the hindsight goals) and the Wasserstein distance between the target goal distribution and the distribution of potential hindsight goals (goals as close as possible to the target goal distribution). The resulting Wasserstein-Barycenter problem is discretely solved using the euclidean distance metric between two goals sampled from the potential hindsight goal distribution and the target goal distribution. While HGG demonstrates higher sample efficiency than HER in environments where the euclidean metric is applicable, it fails in environments with obstacles without the guidance of a hand-crafted distance grid, where the

Z. Bing, M. Brucker, F. Morin, and A. Knoll are with the Department of Informatics, Technical University of Munich, Germany. E-mail: {bing, morinf, knoll}@in.tum.de, matthias.brucker@tum.de.

R. Li and X. Su are with the School of Automation, Chongqing University, China. Email: rui.li@cqu.edu.cn, suxiaojie@cqu.edu.cn

K. Huang is with the School of Data and Computer Science, Sun Yat-sen University, China.

*Corresponding author: Kai Huang. Email: huangk36@mail.sysu.edu.cn

shortest obstacle-avoiding distance between two goals cannot be computed with the euclidean metric.

In this paper, we propose graph-based hindsight goal generation (G-HGG), an extension of HGG, to solve complex robotic manipulation tasks with obstacles within the framework of sparse-reward RL. We state the complex robotic object manipulation tasks that can not be solved by state-of-the-art sparse-reward RL algorithms. We formulate our algorithm G-HGG as *graph construction*, *shorted distance computation* as pretraining steps. We then integrate *stopping condition* to the overall G-HGG algorithm extended on the basis of HGG. To make G-HGG applicable to environments with obstacles, the euclidean metric is replaced by a graph-based distance metric: the environment's goal space is represented by a graph consisting of discrete goals as vertices and obstacle-avoiding connections between the vertices as edges. The distance between two goals are then approximated by the shortest path on the graph between the two vertices that are closest to the two goals. We design four new challenging robotic object manipulation tasks, which contain obstacles with different difficulties. And we compare the performances of G-HGG, HGG, and HER by running them in that four challenging manipulation tasks.

Our main contribution to the literature is an algorithm that bridges graph-based planning and sparse-reward RL for solving complex robotic manipulation tasks. Specifically, first, we provides a graph-based distance metric for generating accessible hindsight goals that avoid obstacles and guide the explorations at the same time. This graph-based concept offers a way to quickly build up reachable 3D spaces that can be further used for training goal-conditioned RL agents. Second, with this graph-based distance metric, we proposed a novel algorithm G-HGG as an extension of HGG. The G-HGG can successfully solve complex object manipulation tasks in environments with obstacles. Third, we also introduce a stopping mechanism to terminate the hindsight goal exploration when those goals are close enough to target goals. This stopping mechanism often leads to faster learning and higher successful rates. Finally, by comparing the performances of G-HGG on four challenging object manipulation environments, we demonstrate that G-HGG provides a significant enhancement in both sample efficiency and overall success rate over HGG and HER. With ablation studies, we also show the robustness of G-HGG in terms of discretization density.

We first consider our work as a direct evidence that sparse reward RL can be used to solve complex manipulation tasks with obstacles, which is significant since it is difficult to design a proper dense reward for most of the real-world manipulation tasks and most of the tasks have obstacles in their scenarios. Secondly, we consider G-HGG as an alternative approach to design distance representations in manipulation tasks. This may inspire more methods to design even more sophisticated distance metric representation to solve more challenging tasks, such as those with dynamically moving obstacles. Third, from the perspective of implementing RL tasks in the real world, G-HGG also provides a practical solution to solve complex robotic manipulation tasks in the real world with the consideration that 3D distances can be acquired in such a fixed robotic arm scenario.

II. Related Work

Informative and effective explorations are essential for solving goal-conditioned RL tasks with sparse rewards. An amount of research has emerged and we briefly discuss them from four main ideas.

A. Prioritized Experience Replay

One major drawback of HER has been its inefficient random replay of experience. Research has shown that prioritized sampling of transitions from the replay buffer significantly increases sample efficiency. Prioritized sampling can be based on the TD-error [33], reward-weighted entropy [43], transition energy [44], and density of achieved goals [45]. Curriculum-Based Experience Replay (CHER) introduced by [11] adaptively prioritizes replay buffer entries according to their diversity with respect to other replay goals and their proximity to target goals. The diversity demonstrates the curiosity of an agent to explore the environment. The proximity based on euclidean distance represents how close are these goals close to the desired goal. However, CHER is not applicable to tasks with obstacles that can mislead the euclidean distance.

B. Demonstrations for Improved Exploration

Exploration is another challenging problem in sparse-reward RL algorithms such as HER. In scenarios with a high-dimensional action space and a large task horizon (e.g. far-away goals), finding a non-negative reward can be very difficult. Nair et al. use demonstrations to tackle this issue and learn challenging multi-step tasks such as object stacking [23]. More concretely, their approach combines multi-goal RL with imitation learning by introducing a demonstration replay buffer and a behavioral cloning loss.

C. Curriculum Learning

Another way to tackle the exploration problem in sparse-reward multi-goal RL is curriculum learning (CL) [31], which presents problems in a favorable order, the so-called curriculum. One CL approach to improve exploration is augmenting the sparse learning problem with basic, easy-to-learn auxiliary tasks [31], [9]. In the absence of extrinsic motivation due to the sparsity of external rewards, intrinsic motivation can be used to create a curriculum for improved exploration [34], [25], [14], [26], [37], [6], [3]. Another way of constructing a meaningful curriculum is to predict high-reward states and generate goals close to these meaningful states [16], [12], [13], [11]. Florensa et al. [12] have recently proposed a curriculum learning algorithm, in which a generative adversarial network is trained to produce goals of intermediate difficulty (GOID), which leverages from Least-Squares GAN [22]. The GOID approach first labels a set of goals based on the appropriate level of difficulty for current policy of the agent. Secondly, by using these labelled goals, GOID trains a generator to output new goals in the appropriate level of difficulty. Finally, GOID trains the agent using these generated intermediate goals. Ren et al. [30] have combined GOID with HER and compared its training performance to their own HGG algorithm on a FetchPush task. HGG demonstrated better sample efficiency than GOID+HER, which showed only minimal improvement over HER. Inverse curriculum generation ([13]) shows that curriculum learning also works reversely, when the agent gradually learns to reach the goal from a curriculum of start states chosen to be increasingly far from the goal. PCHID creates the curriculum by following the idea of dynamic programming with k-step inverse dynamics learning [38]. Different from HER, this method can be incorporated with both on-policy and off-policy RL algorithms. However, it suffers from being problematic when the dynamics is too complex to be approximated, such as a task with extensive interactions with an object.

All above mentioned curriculum learning algorithms have demonstrated to improve exploration in sparse multi-goal RL. However, they share one significant drawback: exploration is unguided. Consider a scenario with a very large goal space, where target goals are far from the goal representation of the initial states. Even when following a curriculum based on auxiliary tasks, novelty, information gain, previous rewards, or training success, the agent has to gradually explore the entire goal space until a target goal is reached for the first time. When target goals are "hidden", e.g. by obstacles, this unguided exploration process takes an unreasonably long time.

D. Representation Learning

Representation learning [4] is a promising framework to solve goal-conditioned RL tasks, in which representative abstractions are interpreted from high-dimensional observations to low-dimensional latent states. With the representations as a planner, model-free RL algorithms are able to perform control tasks. Some work showed that learned representations can be used to solve navigation and goal-reaching tasks for mobile agents. Robotic manipulations tasks were implemented but limited to 2D space because most representations were learned from images [35], [15], [10].

III. Preliminaries

A. Goal-Conditioned RL

In goal-conditioned RL, an agent interacts with its environment to reach some goals, which can be modeled as a goal-conditioned Markov decision process (MDP) with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a goal space $\mathcal{G}$, a probabilistic transition function $P : S \times \mathcal{A} \to \mathcal{S}$, a reward function $r_g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor γ. At every time step t, the agent's behavior a_t is defined by a probabilistic policy $\pi(s_t||g)$, given by the current state s_t and the goal g (we use $||$ as a symbol for concatenation into $\mathcal{S} \times \mathcal{G}$). The task is to find an optimal policy that can maximize the expected curriculum reward starting from the initial state sampled from the initial state distribution $s \in S_0$, which is defined as

$$V^{\pi}(s||g) = \mathbb{E}_{s_0 = s \sim S_0,\, a_t \sim \pi(s_t||g),\, s_{t+1} \sim P(s_t, a_t)} \big[\sum_{t=0}^{\infty} \gamma^t r_g(s_t, a_t) \big] \tag{1}$$

B. Hindsight goal generation

HGG [30] is an extension of HER [1] to scenarios in which the target goal distribution differs a lot from the initial state distribution. The idea behind HGG is to guide exploration by choosing suitable intermediate goals. On the one hand, such intermediate goals should not be too far from goals that the agent knows how to reach, allowing the neural networks to generalize from this experience. On the other hand, they should not too close similar to already known goals that the agent does not learn anything new. Such intermediate goals should be properly explored to allow the network to generalize from this experience and learn new behaviors as well. HGG can be briefly explained as follows. A value function of a policy π for a specific goal g is assumed to have some generalizability to another goal g' close to g [2], [20]. This assumption is mathematically characterized via Lipschitz continuity as

$$|V^{\pi}(s||g) - V^{\pi}(s'||g')| \le L \cdot d(s||g, s'||g'), \tag{2}$$

where $d(s||g, s'||g')$ is a metric defined by

$$d((s||g), (s'||g')) = c||m(s) - m(s')||_2 + ||g - g'||_2. \tag{3}$$

$m(\cdot)$ is a state abstraction to map from the state space to the goal space. The hyper-parameter $c > 0$ provides a trade-off between 1) the distance between target goals and 2) the distance between the goal representation of the initial states. Assuming the generalizability condition (2) holds for two distributions $s_0^{(1)}||g^{(1)} \sim \mathcal{T}^{(1)}$ and $s_0^{(2)}||g^{(2)} \sim \mathcal{T}^{(2)}$, Ren et al. [30] demonstrated that

$$V^{\pi}(\mathcal{T}^{(2)}) \ge V^{\pi}(\mathcal{T}^{(1)}) - L \cdot \mathcal{D}(\mathcal{T}^{(1)}, \mathcal{T}^{(2)}), \tag{4}$$

where $\mathcal{D}(\cdot,\cdot)$ is the Wasserstein distance [32] based on $d(\cdot,\cdot)$, defined as

$$\mathcal{D}(\mathcal{T}^{(1)}, \mathcal{T}^{(2)}) := \inf_{\mu \in \Gamma(\mathcal{T}^{(1)}, \mathcal{T}^{(2)})} \left(\mathbb{E}_{\mu} \left[d(s_0^{(1)}||g^{(1)}, s_0^{(2)}||g^{(2)}) \right] \right). \tag{5}$$

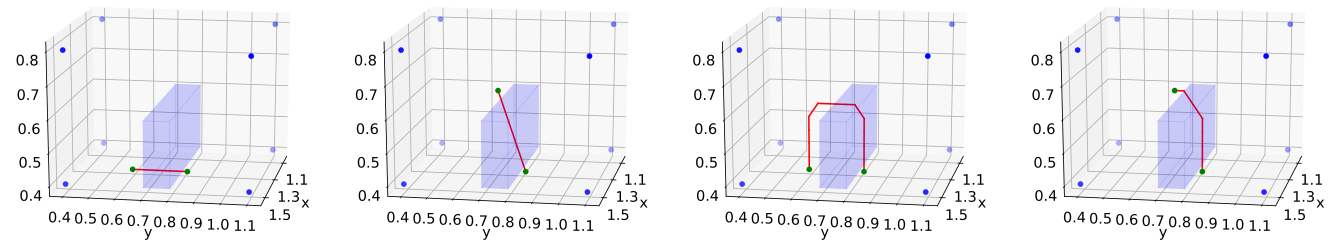
$\Gamma(\mathcal{T}^{(1)}, \mathcal{T}^{(2)})$ denotes the collection of all joint distributions $\mu(s_0^{(1)}||g^{(1)}, s_0^{(2)}||g^{(2)})$, the marginal probabilities of which are $\mathcal{T}^{(1)}$ and $\mathcal{T}^{(2)}$. With $\mathcal{T}^*$ denoting the joint distribution over initial state s_0 and goal g, the agent tries to find a policy π maximizing the expectation of the discounted cumulative reward based on the state value function defined in (1):

$$V^{\pi}(\mathcal{T}^*) := \mathbb{E}_{s_0||g \sim \mathcal{T}^*} \Big[V^{\pi}(s_0||g) \Big]. \tag{6}$$

From (4), it can be derived that optimizing this expected cumulative reward defined in (6) can be relaxed into the surrogate problem of finding

$$\max_{\mathcal{T}, \pi} V^{\pi}(\mathcal{T}) - L \cdot \mathcal{D}(\mathcal{T}, \mathcal{T}^*). \tag{7}$$

Joint optimization of π and $\mathcal{T}$ (7) is solved in a two-stage iterative algorithm. First, standard policy optimization maximizes the value function V^{π} based on experience generated from the intermediate task set $\mathcal{T}$. Second, the intermediate task set $\mathcal{T}$ is optimized while the policy π is kept constant. The second step is a variant of the Wasserstein Barycenter problem with the value function as a bias term for each initial state-goal pair, which can be solved as a bipartite matching problem [8]. For



(a) Case A with euclidean norm distance: Shortest path goes through the obstacle, $d^A_{L2} = 0.210$

(b) Case B with euclidean norm distance: Shortest path goes though the obstacle, $d^B_{L2} = 0.283$

(c) Case A with alternative distance: Shortest path goes around the obstacle, $d^A_{alt} = 0.566$

(d) Case B with alternative distance: Shortest path goes around the obstacle, $d^B_{alt} = 0.301$

Fig. 1: Shortest distances between two points (green dots) in 3D space with obstacles.

this to work, $\mathcal{T}^*$ is approximated by by K initial state-goal pairs sampled from it, resulting in $\hat{\mathcal{T}}^* = \{(\hat{s}^i_0, \hat{g}^i)\}^K_{i=1}$. Now, for every $(\hat{s}^i_0, \hat{g}^i) \in \hat{\mathcal{T}}^*$, a trajectory $\tau^i = \{s^i_t\}$ from the replay buffer is found that minimizes

$$w((\hat{s}^i_0, \hat{g}^i), \tau^i) := c||m(\hat{s}^i_0) - m(s^i_0)||_2 + \min_t \left(||\hat{g}^i - m(s^i_t)||_2 - \frac{1}{L} V^\pi(s^i_0||m(s^i_t))\right). \tag{8}$$

The Lipschitz constant L is treated as a hyper-parameter. These K trajectories τ^i minimize the sum

$$\sum_{(\hat{s}^i_0, \hat{g}^i) \in \hat{\mathcal{T}}^*} w((\hat{s}^i_0, \hat{g}^i), \tau^i) \tag{9}$$

Finally, from each of the K selected trajectories τ^i, the hindsight goal g^i is selected from the state $s^i_t \in \tau^i$, that minimized (8). More formally,

$$g^i = m\Bigg(\underset{s^i_t \in \tau_i}{\arg\min} \left(||\hat{g}^i - m(s^i_t)||_2 - \frac{1}{L} V^\pi(s^i_0||m(s^i_t))\right)\Bigg). \tag{10}$$

Combined with the idea of HER [1] and replacing (s_0, g) with $(\hat{s}^i_0, g^i)$, the generated hindsight transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ can be then stored in the replay buffer for training the policy. To this end, HGG is able to generate a curriculum of meaningful hindsight goals rather than hand-crafted heuristic goals from HER, guiding exploration towards target goals.

IV. Methodology

By guiding the agent to collect more valuable hinsight goals, HGG greatly extends the application scenarios in object manipulation tasks, in which the hindsight goal distribution differs significantly from the initial goal position. However, the hindsight goals generated from HGG is based on measuring the Wasserstein disitance between potential hindsight goals and the target goal distribution, which will fail in obstacle environments where euclidean distance is not applicable without the guidance of a hand-crafted distance grid.

In this section, we first provide the definition of the concrete problem of object manipulation in environments with obstacles, where HGG reaches its limits. We then introduce G-HGG as a solution, which is an extension of HGG using a graph-based distance measure.

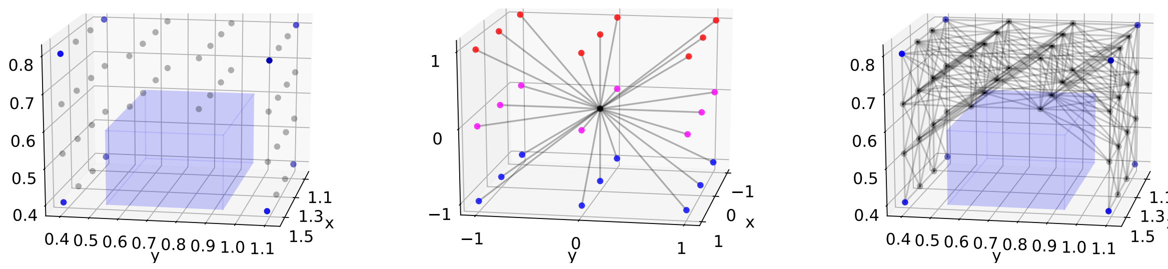
A. Problem Statement

In this paper, we focus on solving robotic object manipulation tasks with sparse rewards, where the goal is to move an object to a certain point in 3D space with a robotic gripper arm. Such scenarios show the following characteristics:

- A multidimensional action space $\mathcal{A} \subset \mathbb{R}^m, m \in \mathbb{N}$.
- A multidimensional state space $\mathcal{S} \subset \mathbb{R}^n, n \in \mathbb{N}$ with $n \geq 3$.
- An initial state distribution $\mathcal{S}_0 : \mathcal{S} \to [0, 1]$. Initial states should be starting conditions that are accessible for the robot to start the task.
- A 3-dimensional goal space $\mathcal{G} \subset \mathbb{R}^3$. A goal is defined as a point in 3D space.
- A target goal distribution $\mathcal{G}_T : \mathcal{G} \to [0, 1]$. Depending on this target goal distribution, goals $g \sim \mathcal{G}_T$ can be close to or far from the initial state's goal representation $m(s_0), s_0 \sim \mathcal{S}_0$.
- A goal predicate $f_g : \mathcal{S} \to \{0, 1\}, g \in \mathcal{G}$ where
$$f_g(s) = \begin{cases} 1, & \text{if } ||g - m(s)|| \leq \delta_g \\ 0, & \text{otherwise} \end{cases} \tag{11}$$
with mapping $m : \mathcal{S} \to \mathcal{G}$ s.t. $f_{m(s)}(s) = 1 \forall s \in \mathcal{S}$ and δ_g as a distance threshold.
- A sparse reward function $r_g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defined as $r_g(s, a) = -[f_g(s) = 0]$, where $g \in \mathcal{G}$.

Even though the general principle of HGG is applicable, there is one major limitation. Solving the Wasserstein Barycenter problem in HGG consists of the computation of the distance between a goal $g \in \mathcal{G}$ and the goal representation $m(s)$ of state $s \in \mathcal{S}$. However, in 3D space with obstacles, the euclidean metric used in HGG is generally not applicable.

To illustrate this point, we consider an example, where the goal is to pick up an object with a robotic arm, lift it over an obstacle (blue box), and place it at the target goal. Let us consider two cases: A (Figures 1a and 1c) and B (Figures 1b and 1d), where goals g_1 and g_2 are marked in green. In Figures 1a and 1b, the euclidean metric calculates the distance, $d_{euc} = ||g_2 - g_1||$, which is marked with red lines. If the task



(a) Creating vertices. (b) Creating outgoing edges. (c) Creating graph.

Fig. 2: Graph construction in the demo obstacle environment.

is completed in an environment without obstacles, this is an accurate distance metric, and the task can be solved using HGG. In environments with obstacles, however, the object cannot go through these obstacles, which means the euclidean metric is not a suitable distance metric. Then, a suitable shortest-path based distance measure similar to the ones in Figures 1c and 1d is required in such a scenario.

The problem of finding the shortest path between two points without intersecting an obstacle is known as the euclidean shortest path problem. In this paper, we propose an algorithm called graph-based hindsight goal generation (G-HGG), in which approximate shortest distances from discrete points in 3D space are pre-computed and used to compare distances between potential hindsight goals.

B. G-HGG

G-HGG is an extension of HGG to environments with obstacles, where the euclidean metric is not applicable as a distance metric. Hence, we reformulate (8) and (10), replacing the euclidean metric with the graph-based distance metric d_G .

With the new formulation, for every $(\hat{s}_0^i, \hat{g}^i) \in \hat{\mathcal{T}}^*$, we can find a trajectory from the replay buffer $\tau^i = \{s_t^i\} \in R$ that minimizes

$$w((\hat{s}_0^i, \hat{g}^i), \tau^i) := c||m(\hat{s}_0^i) - m(s_0^i)||_2 + \min_t \left(d_G(\hat{g}^i - m(s_t^i)) - \frac{1}{L} V^\pi(s_0^i || m(s_t^i)) \right), \tag{12}$$

where all these variables have the same meanings as in (8). Altogether, these K trajectories τ^i minimize the sum

$$\sum_{(\hat{s}_0^i, \hat{g}^i) \in \hat{\mathcal{T}}^*} w((\hat{s}_0^i, \hat{g}^i), \tau^i). \tag{13}$$

Finally, from each of the K selected trajectories τ^i, the hindsight goal g^i is selected from the state $s_t^i \in \tau^i$, that minimizes (12):

$$g^i = m\left(\underset{s_t^i \in \tau_i}{\arg\min} \left(d_G(\hat{g}^i - m(s_t^i)) - \frac{1}{L} V^\pi(s_0^i || m(s_t^i)) \right) \right). \tag{14}$$

The distance metric d_G is based on shortest paths in a graph with a discrete representation of the goal space as vertices. The computation of shortest path distances is done pre-training and consists of creating a graph representing the environment and pre-computing shortest distances among vertices.

1) Graph Construction: Let us consider an unbounded goal space $\mathcal{G}$ with an infinite number of goals; let us further define a continuous but bounded subset of the goal space, the accessible goal space $\mathcal{G}_A \subset \mathcal{G}$. $\mathcal{G}_A$ contains all potential goals that the object can reach. We then establish a representation of the accessible goal space $\mathcal{G}_A$ with an undirected graph. A graph $G = (P, E)$ consists of a set of vertices P with a set of weighted edges E, and an assigned weight w.

$$E \subset \{(p_1, p_2, w) \mid (p_1, p_2) \in P^2, p_1 \neq p_2, w \in \mathbb{R}\}, \tag{15}$$

where p_1 and p_2 are two possible vertices. In environments with obstacles, goals $g_{obs} \in \mathcal{G}$ lying within an obstacle that are blocked from being reached are not elements of the accessible goal space $g_{obs} \notin \mathcal{G}_A$. Since $\mathcal{G}_A$ is bounded, it can be enclosed in a parallelepipedic bounding box defined by values x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, $z_{max} \in \mathbb{R}$, describing the span of this box in each coordinate direction. We then use this box to generate a finite set of vertices $\hat{P}$ spatially arranged as an orthorhombic lattice. $\hat{P}$ is defined by the total number of vertices $n = n_x \cdot n_y \cdot n_z$, with $n_x, n_y, n_z \in \mathbb{N}$ in each coordinate direction of $\mathcal{G}_A$, or alternatively by the distance between two adjacent grid-points in each coordinate direction given by

$$\Delta_x = \frac{x_{max} - x_{min}}{n_x - 1}, \ \Delta_y = \frac{y_{max} - y_{min}}{n_y - 1}, \ \Delta_z = \frac{z_{max} - z_{min}}{n_z - 1}. \tag{16}$$

Finally, we define the set of vertices as $P = \hat{P} \cap \mathcal{G}_A$, where

$$\hat{P} := \{(x_{min} + \Delta_x \cdot i, \ y_{min} + \Delta_y \cdot j, \ z_{min} + \Delta_z \cdot k) \mid i \in [0, n_x - 1], j \in [0, n_y - 1], k \in [0, n_z - 1]\}. \tag{17}$$

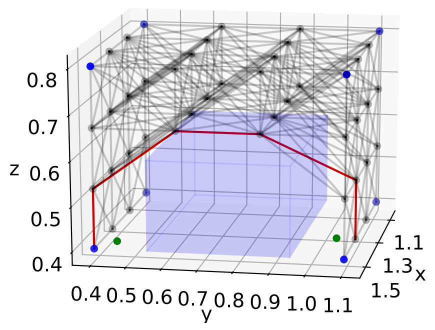
A set of vertices in a demo environment with an obstacle is illustrated in Figure 2a, $\mathcal{G}_A$ is defined as the cuboid space marked by the blue balls, in which the obstacle is depicted by the blue box. $n_x = n_y = n_z = 4$. All the vertices are evenly distributed in $\mathcal{G}_A$ and no one lies inside the obstacle.

As a next step, we connect two adjacent vertices $p_1 = (\hat{x}_1, \hat{y}_1, \hat{z}_1) \in P$, $p_2 = (\hat{x}_2, \hat{y}_2, \hat{z}_2) \in P$ with an edge of weight w considering the following:

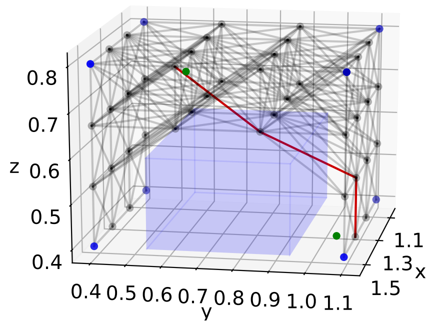
$$(p_1, p_2, w) \in E \iff |\hat{x}_2 - \hat{x}_1| \leq \Delta_x \text{ and } |\hat{y}_2 - \hat{y}_1| \leq \Delta_y \text{ and } |\hat{z}_2 - \hat{z}_1| \leq \Delta_z, \tag{18}$$

with $w := \sqrt{(\hat{x}_2 - \hat{x}_1)^2 + (\hat{y}_2 - \hat{y}_1)^2 + \cdot(\hat{z}_2 - \hat{z}_1)^2}$.

In environments with obstacles, it is important to make sure



(a) Case A: both goals $g_1, g_2 \in \mathcal{G}_A$. Therefore, $d_g(g_1, g_2) = \hat{d}_G(\nu(g_1), \nu(g_2)) = 1.085$



(b) Case B: both goals $g_1, g_2 \in \mathcal{G}_A$. Therefore, $d_G(g_1, g_2) = \hat{d}_G(\nu(g_1), \nu(g_2)) = 0.718$

Fig. 3: Graph-based distances and shortest paths (red lines) between two goals (green dots).

that no edge in the graph cuts through an obstacle. It should be noted that, for convex obstacles, we simply replace them with a minimal axis-aligned cuboid that can enclose the obstacle. We then remove this cuboid from $\mathcal{G}_A$; for a non-convex obstacle, we decompose it into several convex obstacles and then replace each of them with a minimal axis-aligned cuboid that can enclose the convex obstacle. Consequently, we require every convex sub-obstacle to be detected by at least one potential vertex $v \in \hat{P}$ but $v \notin \mathcal{G}_A$. Let us define the space of one of the cuboid with its edges α, β, γ. Thus, the graph must satisfy the graph density criterion (19) for every convex sub-obstacle, i.e., continuous set goals not included in $\mathcal{G}_A$:

$$\Delta_x < \alpha_{min}^{obs};\ \Delta_y < \beta_{min}^{obs};\ \Delta_z < \gamma_{min}^{obs}, \tag{19}$$

where α_{min}^{obs}, β_{min}^{obs}, $\gamma_{min}^{obs} \in \mathbb{R}$, describing the infimum length of edges of all the convex sub-obstacle. When vertices are connected according to (18), every vertex is connected to at most 26 adjacent vertices (as illustrated in Figure 2b). However, an edge only exists as long as the adjacent vertex is contained in $\mathcal{G}_A$, thus, in case of an environment with obstacles, there is never an edge cutting through an obstacle as long as the graph density criterion (19) is satisfied. Figure 2c illustrates the final graph consisting of edges and vertices in our demo environment.

2) Shorted Distance Computation: Considering the created graph that represents the environment, we can employ a shortest path algorithm such as Dijkstra's algorithm [7] to calculate shortest paths and shortest distances $\hat{d}_G$ between every possible pair of vertices $(p_1, p_2) = ((\hat{x}_1, \hat{y}_1, \hat{z}_1), (\hat{x}_2, \hat{y}_2, \hat{z}_2)) \in P^2$ in a graph $G = (P, E)$. All possible combinations of the resulting shortest distance function $\hat{d}_G$ can be efficiently pre-computed with Dijkstra and stored in an $n \times n$ table, where n denotes the number of vertices in P. An ablation study of n can be found in the Section VI.

Given two goals $g_1 = (x_1, y_1, z_1) \in \mathcal{G}$, $g_2 = (x_2, y_2, z_2) \in \mathcal{G}$ and a graph $G = (P, E)$ with representing the approximate goal space $\mathcal{G}_A \subset \mathcal{G}$ with x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}, Δ_x, Δ_y, and Δ_z, the graph-based distance $d : \mathcal{G}^2 \to \mathbb{R}$ is defined such that

$$d_G(g_1, g_2) = \begin{cases} \hat{d}_G\Big(\nu(g_1), \nu(g_2)\Big), & \text{if } g_1 \in \mathcal{G}_A \wedge g_2 \in \mathcal{G}_A \\ \infty, & \text{otherwise} \end{cases} \tag{20}$$

where $\nu : \mathcal{G}_A \to P$ maps goals in $\mathcal{G}_A$ to the closest vertex in P:

$$\nu(g) = \nu(x, y, z) = (\hat{x}, \hat{y}, \hat{z}) = \Big(x_{min} + \Delta_x \cdot \Big\lfloor \frac{x - x_{min}}{\Delta_x} \Big\rceil, \\ y_{min} + \Delta_y \cdot \Big\lfloor \frac{y - y_{min}}{\Delta_y} \Big\rceil,\ z_{min} + \Delta_x \cdot \Big\lfloor \frac{z - z_{min}}{\Delta_z} \Big\rceil\Big) \tag{21}$$

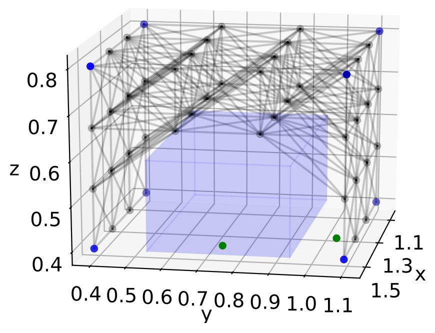
$\lfloor a \rceil$ rounds any $a \in \mathbb{R}$ to the closest integer value. Figure 3 illustrates a visualization of graph-based goal distance computation between goals $g_1 \in \mathcal{G}_A$, $g_2 \in \mathcal{G}_A$. In both Cases A (Figure 3a) and B (Figure 3b), both goals are contained in the accessible goal space $\mathcal{G}_A$. Therefore, the graph-based distance is approximated by the shortest distance between the two vertices $\nu(g_1)$ and $\nu(g_2)$ that are closest to the goals g_1 and g_2, respectively. Mathematically, $d(g_1, g_2) = \hat{d}_G(\nu(g_1), \nu(g_2))$.

In Figure 4, however, goal $g_1 \in \mathcal{G}$ is not contained in the accessible goal space $g_1 \notin \mathcal{G}_A$. In case C (Figure 4a), this is due to the location of g_1 inside the obstacle (blue box), whereas in case D, g_1 is located outside the defined boundaries of $\mathcal{G}_A$ (blue balls). In both cases, no shortest path can be computed and the graph-based distance is set to $d(g_1, g_2) = \infty$. It should be noted that both G-HGG and HGG use the Euclidean distance as a measure of the closeness and no extra information is used by G-HGG.

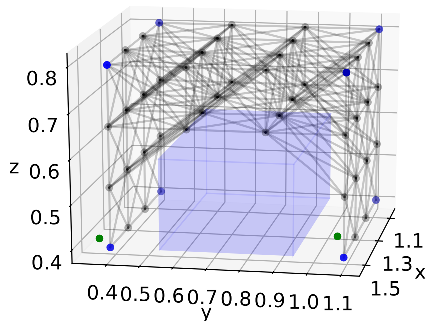
3) Algorithm: The overall G-HGG algorithm is provided as Algorithm 1. The main differences between HGG and G-HGG are in steps 2, 3 and 6. Step 2 (graph construction) and step 3 (shortest distance computation) are performed pre-training. The pre-computed table of shortest distances $\hat{d}_G$ is used in step 6. Similar to HGG, G-HGG is complementary and therefore compatible to other improvements of HER such as energy-based prioritization (EBP [44]), which focus on optimizing the replay process.

C. Stop Condition

The basic idea of HGG / G-HGG is to guide exploration by means of hindsight goals. This is especially useful at the beginning of training, where hindsight replay goals (derived from previously achieved states) are far away from target goals. When training progresses and a certain fraction $\delta_{stop} \in [0, 1]$ of the hindsight goal candidates for exploration are very close to sampled target goals $g \sim \mathcal{G}_T$, stopping HGG and continuing



(a) Case C: goal $g_1 \notin \mathcal{G}_A$ since it lies inside the obstacle. Therefore, $d(g_1, g_2) = \infty$.



(b) Case D: goal $g_1 \notin \mathcal{G}_A$ since it lies outside of $\mathcal{G}_A$ (cuboid space marked by the blue balls). Therefore, $d(g_1, g_2) = \infty$.

Fig. 4: Graph-based distances between two goals (green) $g_1 \notin \mathcal{G}_A$, $g_2 \in \mathcal{G}_A$ in demo obstacle environment. Shortest paths do not exist.

training with HER often leads to faster learning and higher success rates. This is because HGG and G-HGG select goals based on an inexact distance measure and will never consider some better goals that yield a slightly longer distance. When a certain exactness is reached, it is more efficient to train on the real generated goals for the final stage of fine tuning. Moreover, G-HGG is computationally expensive and therefore less effective when the hindsight goal candidates are close enough to the target goals.

To decide when to stop HGG / G-HGG, we can perform the HGG / G-HGG Stop Condition Check (Algorithm 2) after step 7 of G-HGG (Algorithm 1), respectively. Even though HGG and G-HGG perform reasonably well without stopping ($\delta_{stop} = 1$) and switching to HER based on the stop condition check, our findings show that a suitable choice of δ_{stop} can increase training performance whilst reducing computation time significantly. Results can be found in Section VI.

Notably, for Algorithm 2, the mapping m is not required to be invertible as long as m^{-1} is not computed explicitly. This is the case for goal predicates f_g, which can be easily computed without even using m:

$$f_g(s) = f_g(m^{-1}(g')) = \begin{cases} 1, & \text{if } \parallel g - m(m^{-1}(g')) \parallel = \\ & \parallel g - g' \parallel \leq \delta_g \\ 0, & \text{otherwise} \end{cases} . \tag{22}$$

Algorithm 1 Graph-Based Hindsight Goal Generation (G-HGG)

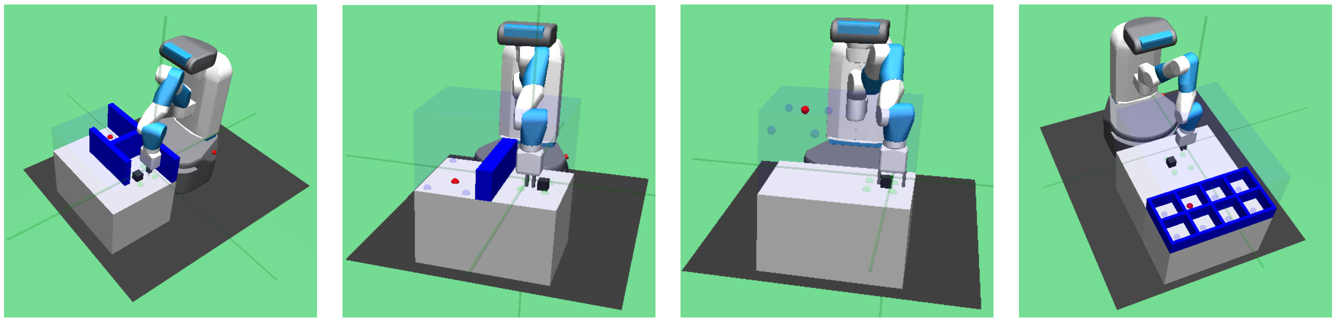
```
1:  Given: off-policy algorithm $\mathbb{A}$, sampling strategy $\mathbb{S}$, reward function $r_g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
2:  Construct a graph $G = (V, E)$ as a discrete representation of $\mathcal{G}$    ▷ section IV-B1
3:  Pre-compute shortest distances $\hat{d}_G$ between every pair of vertices $(p_1, p_2) \in P^2$ with Dijkstra
4:  Initialize $\mathbb{A}$ and replay buffer $R$
5:  for $iteration$ do
6:      Construct a set of $M$ intermediate tasks $\{(\hat{s}_0^i, g^i)\}_{i=1}^M$:    ▷ HGG
        • Sample target tasks $\{(\hat{s}_0^i, \hat{g}^i)\}_{i=1}^K \sim \mathcal{T}^*$
        • Find $K$ distinct trajectories $\{\tau^i\}_{i=1}^K$ that together minimize (13)    ▷ weighted bipartite matching, based on $d_G \approx \hat{d}_G$
        • Find $M$ intermediate tasks $(\hat{s}_0^i, g^i)$ by selecting intermediate goal $g^i$ from each $\tau^i$ according to (14)    ▷ based on $d_G \approx \hat{d}_G$
7:      for $episode = 1, M$ do
8:          $(s_0, g) \leftarrow (\hat{s}_0^i, g^i)$    ▷ hindsight goal-oriented exploration
9:          for $t = 0, T-1$ do
10:             Sample an action $a_t$ using the policy from $\mathbb{A}$ with noise: $a_t \leftarrow \pi(s_t||g) + \mathcal{N}_t$
11:             Execute the action $a_t$ and observe a new state $s_{t+1}$
12:         for $t = 0, T-1$ do
13:             $r_t := r_g(s_t, a_t)$; Store transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ in $R$;    ▷ experience replay
14:             Sample a set of additional goals for replay $G := \mathbb{S}(current\, episode)$
15:             for $g' \in G$ do
16:                 $r' := r_{g'}(s_t, a_t)$; Store the transition $(s_t||g', a_t, r', s_{t+1}||g')$ in $R$;    ▷ HER
17:     for $t = 1, N$ do
18:         Sample a minibatch $B$ from the replay buffer $R$    ▷ HER or EBP
19:         Perform one step of optimization using $\mathbb{A}$ and minibatch $B$ using DDPG
```

V. Experiments

In this experiments section, we show the performance of G-HGG compared to HGG, HER and GOID-HER on four different MuJoCo environments, which are modified versions of the Fetch gripper environments provided by [28]. Three of the environments contain obstacles, which makes the underlying object manipulation task especially challenging to solve.

A. Environments

To demonstrate the advantages of G-HGG over HGG, HER, and GOID-HER, we create new experimental environments based on the standard robotic manipulation environments from OpenAI Gym [28]. All our environments are MuJoCo environments featuring a modeled Fetch robot with a gripper.



(a) FetchPushLabyrinth (b) FetchPickObstacle (c) FetchPickNoObstacle (d) FetchPickAndThrow

Fig. 5: Robotic manipulation environments.

Algorithm 2 HGG / G-HGG Stop Condition Check

```
1: Given:
     • A set of sampled target tasks $\{(\hat{s}_0^i, \hat{g}^i)\}_{i=1}^K \sim \mathcal{T}^*$
     • A set of intermediate tasks $\{(\hat{s}_0^j, g^j)\}_{j=1}^M$ selected by HGG / G-HGG
     • A stop condition parameter $\delta_{stop} \in [0, 1]$
     • A goal predicate $f_g$ similar to (11) with a traceable mapping $m : \mathcal{S} \to \mathcal{G}$
2: $c \leftarrow 0$
3: for $j = 1, M$ do
4:     $b \leftarrow 0$
5:     for $i = 1, K$ do
6:         if $f_{\hat{g}^i}(m^{-1}(g^j)) == 1$ then
7:             $b \leftarrow 1$
8:     $c += b$
9: if $\frac{c}{M} \geq \delta_{stop}$ then
10:    Stop condition is satisfied $\rightarrow$ Continue with HER.
11: else
12:    Stop condition is not satisfied $\rightarrow$ Continue with HGG / G-HGG.
```

They are inspired by the environments provided by [28] and thus adopt the their control strategy:

- The state space $\mathcal{S}$ contains joint positions and joint velocities for all joints of Fetch, the position of the end effector, as well as the object's position and orientation.
- Depending on whether gripper control is enabled or disabled, the action space is three- or four-dimensional. An action consists of the end effector's position for the next time step (three coordinates), in case of enabled gripper control, the gripper's opening control parameter is added as a forth component.
- The end effector's position of the Fetch robot is controlled via inverse kinematics and motion capturing. While the orientation of the end effector is fixed, the coordinates of its position at a certain time step can be explicitly enforced via input coordinates.

*1) **FetchPushLabyrinth**:* (Figure 5a): the goal is to push the object (black cube) from its initial position around the blue obstacles (labyrinth) to a goal. The gripper remains permanently closed and gripper control for picking up the object is disabled, leading to a three-dimensional action space. The accessible goal space $\mathcal{G}_A$ is visualized by the semi-transparent blue box of length 0.5 m, width 0.7 m, and height 0.2 m, excluding the blue obstacle labyrinth. Note that it is crucial to define $\mathcal{G}_A$ properly: if the accessible goal space contained space above the labyrinth, the shortest path would require lifting the object over the labyrinth and not pushing it around the obstacles. Since the gripper is permanently closed, such paths are impossible to achieve in this environment and G-HGG would fail.

*2) **FetchPickObstacle**:* (Figure 5b): the goal is to pick up the cube from its initial position, lift it over the obstacle, and place it at a goal position. Gripper control is enabled, the gripper can be symmetrically opened and closed by a single actuator, leading to a 4-D action space. The accessible goal space $\mathcal{G}_A$ is visualized by the semi-transparent blue box of length 0.5 m, width 0.7 m, and height 0.4 m, excluding the blue obstacle.

*3) **FetchPickNoObstacle**:* (Figure 5c): the goal is to pick up the cube from its initial position, lift it up, and place it at a goal position. No obstacle is present in this scenario, but the target goals are located in the air. Gripper control is enabled, the gripper can be symmetrically opened and closed by a single actuator, leading to a 4-D action space. The accessible goal space $\mathcal{G}_A$ is visualized by the semi-transparent blue box of length 0.5, width 0.7, and height 0.4. This environment is specially designed to compare the performances of different algorithms in an environment without obstacles.

*4) **FetchPickAndThrow**:* (Figure 5d): the goal is to pick up the cube from its initial position, lift it up, and throw it into one of the eight boxes (obstacles). The gripper can be symmetrically opened and closed by a single actuator, leading to a 4-D action space. The accessible goal space $\mathcal{G}_A$ is visualized by the semi-transparent blue box of length 0.85, width 0.7, and height 0.4 excluding the blue walls of the eight obstacle boxes.

B. Results

We tested G-HGG on the four environments to compare its performance to HGG, HER, and GOID-HER. The performance of HER is also examined, since both G-HGG and HGG use HER as a benchmark. Since we know from [44] and [30] that EBP [44] significantly enhances the performance of both HER and HGG, we used EBP in all our trainings of HER, HGG, and G-HGG. The results clearly show that G-HGG outperforms HGG by far in environments with obstacles, both in terms of sample efficiency and maximum success rate. In the environment without obstacles, the performance of G-HGG was still comparable to HGG in terms of sample efficiency and maximum success rate. It should be noted that the sample efficiency means that G-HGG can learn more quickly than HGG or HER using the same amount of iterations, which is different from computation time.

Figure 6 shows success rates of G-HGG, HGG, HER, and GOID-HER (median and interquartile range of five training runs each), plotted over training iterations in four environments. The most remarkable results can be observed in the FetchPushLabyrinth environment. While HER, HGG, and GOID-HER display no success over 400 iterations, G-HGG reaches a success rate of 80% after 170 iterations, increasing to over 90% after 300 iterations. By comparing a sample of hindsight goals from iterations 20, 40, 60, and 80 (Figure 7), it becomes obvious that G-HGG outperforms HGG by far. While the graph-based distance metric used in G-HGG leads to a choice of hindsight goals guiding the agent around the obstacle towards the target goals, HGG repeatedly uses goals that are closest to the target goals with respect to the euclidean metric. Since this euclidean metric based shortest path is blocked by an obstacle, the agent gets stuck trying to reach the hindsight goals (pushing the object against the obstacle), never achieving more promising goals, and thus, never reaching the target goals. HER or COID-HER can not solve the task.

In the FetchPickObstacle environment, G-HGG outperforms HGG in terms of sample efficiency, due to graph-based distances supporting goals that avoid the obstacle. Since the euclidean distance metric is at least partly (in x and y direction) valid, HGG eventually achieves a notable success rate as well. However, the plots show that there is a large variance within the HGG trainings, emphasizing the disadvantage of random exploration in HGG with non-meaningful hindsight goals over guided exploration in G-HGG. GOID-HER only shows a success rate around 40% and HER cannot solve the task.

FetchPickNoObstacle is an environment where G-HGG has no advantage over HGG. As no obstacle is present, the euclidean metric is applicable to compare distances, allowing HGG to perform well. Since the euclidean metric used in HGG is more exact than the graph-based distances in G-HGG, it is not surprising that HGG yields slightly better training results. However, the similarity of the curves shows that G-HGG's performance is comparable to HGG in terms of sample efficiency and success rate even in environments without obstacles. Overall, we assume that G-HGG is generally applicable to all environments where HGG yields good training results. GOID-HER can solve the task as well but with a poor sample efficiency compared to G-HGG or HGG. HER still fails to reach the target goals. It should be noted that FetchPickNoObstacle is more difficult than the similar task FetchPickAndPlace-v0 from HER. In FetchPickNoObstacle, the mean distance from the initial position to the target goals is 0.5 m, and this distance is only 0.15 m in FetchPickAndPlace-v0.

FetchPickAndThrow is a difficult task, since target goals are not uniformly sampled from a continuous target goal distribution, but from a discrete set of eight goals. G-HGG yields better results than HGG in terms of success rate after 600 iterations and is clearly more sample efficient. Both HGG and G-HGG cannot achieve success rates above 60% due to the difficulty of the task, involving picking the object, lifting it up, and dropping it while giving it a well-dosed push in the desired direction. Since the final policy trained to achieve the task results from guided, but still random exploration, it is no surprise that the agent cannot develop a perfect throwing motion from sparse rewards.

VI. Discussion

We provide an ablation study on the stop condition parameter δ_{stop} and the number of vertices n_x, n_y, n_z.

A. *Stop Condition Parameter*

Figure 8 illustrates training success rates for different values of the stop condition parameter δ_{stop} for both HGG and G-HGG in environments FetchPushLabyrinth and FetchPickAndThrow. In alignment with our main results, both plots show that G-HGG outperforms HGG significantly, regardless of the value of δ_{stop}. Furthermore, G-HGG proves to be robust to changes in δ_{stop}. In FetchPushLabyrinth (Figure 8a), G-HGG performs best with $\delta_{stop} = 0.3$, but other choices of δ_{stop} only show a slight degradation in overall training success and sample efficiency. In FetchPickObstacle (Figure 8b), G-HGG performs best with $\delta_{stop} = 0.6$. In FetchPickNoObstacle, $\delta_{stop} = 0.3$ gives the best performance since it is closest to HGG. In FetchPickAndThrow (Figure 8d), $\delta_{stop} > 0.5$ is the best choice. From the results, we can find that $\delta_{stop} = 0.3$ performs best in three scenarios except the FetchPickObstacle, since this task requires more accurate guidance to the target goal positions. For the other three tasks, the learning process can be accelerated by stopping the graph-based guidance once the agent figures out how to get around the obstacle.

Despite G-HGG being generally robust to changes in δ_{stop}, we recommend rough parameter tuning on the stop condition parameter. As a rule of thumb, values around $\delta_{stop} = 0.3$ usually yield good success rates and come with the benefit of considerably reduced computation time. Notably, all values $\delta_{stop} > 0.5$ yield the same G-HGG training behavior in FetchPickandThrow, since the stop criterion is never satisfied for these values. This is also the case for $\delta_{stop} > 0.5$ in HGG runs of FetchPickAndThrow and for $\delta_{stop} > 0.1$ in HGG runs of FetchPushLabyrinth.

B. *Number of Vertices*

Figure 9 shows the training success rates for different numbers of vertices n in all four environments. All plots show that G-HGG is robust to changes in the number of vertices and demonstrates reasonable performance in all chosen configurations.

Notably, the number of vertices n cannot be directly specified but depends on the choice of n_x, n_y, and n_z. The same holds for Δ_x, Δ_y, and Δ_z, all of which are directly derived from n_x, n_y, and n_z, respectively. An overview of the parameters

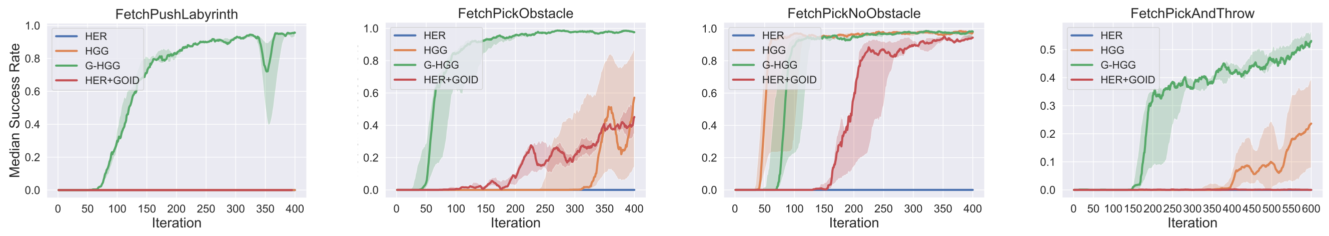


Fig. 6: Median test success rate (line) and interquartile range (shaded) of G-HGG, HGG, HER, and GOID-HER.

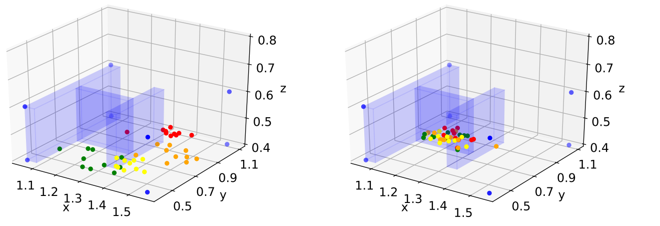


Fig. 7: Hindsight goals in FetchPushLabyrinth after 20 (red), 40 (orange), 60 (yellow) and 80 (green) episodes of G-HGG (Left) / HGG (Right). One iteration contains $M = 50$ episodes.

used in this ablation study is provided in Table I. Since the task FetchPickNoObstacle has no obstacles, we mainly discuss the results from the other three environments. The blue lines in the plots correspond to the minimal number of vertices required to satisfy the vertex density criterion (19) of the respective environment. $\delta_g = 0.05$ in our environments and can be perceived as a kind of critical length of the environment. We can find from all the results that the number of vertices n has an impact on the sample efficiency but less impact on the final successful rate. Even for the minimum number of vertices, G-HGG still manages to solve the task and the success rate is comparable to the other tests. The green lines were computed with parameter choices from our main results (See Figure 6). The orange lines correspond to minimal configurations following a rule of thumb that we recommend for choosing the appropriate number of vertices: n_x, n_y, n_z should be chosen such that 1) the vertex density criterion is met and 2) such that $\Delta_x, \Delta_y, \Delta_z < \delta_g$, where δ_g is the distance threshold for considering a target goal. We would suggest that a dense graph generally leads to better performances compared to a sparse graph, but at the same time it will lead to computation burden due to the calculation of the distances in the graph.

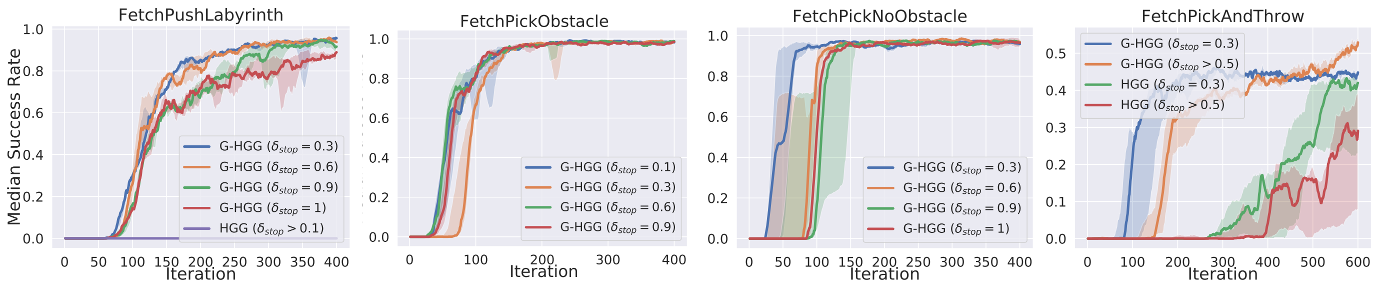
TABLE I: G-HGG parameters for ablation study on n in FetchPushLabyrinth and FetchPickObstacle environment. Note that $\delta_{stop} = 0.3$ and $\delta_g = 0.05$ in all runs.

Environment	n	n_x	n_y	n_z	Δ_x	Δ_y	Δ_z
FetchPushLabyrinth	532	14	19	2	0.038	0.039	0.200
FetchPushLabyrinth	1330	14	19	5	0.038	0.039	0.050
FetchPushLabyrinth	10571	31	31	11	0.017	0.023	0.020
FetchPickObstacle	120	3	10	4	0.250	0.077	0.133
FetchPickObstacle	1485	11	15	9	0.050	0.050	0.050
FetchPickObstacle	10571	31	31	11	0.017	0.023	0.040
FetchPickNoObstacle	120	3	10	4	0.250	0.077	0.133
FetchPickNoObstacle	1485	11	15	9	0.050	0.050	0.050
FetchPickNoObstacle	10571	31	31	11	0.017	0.023	0.040
FetchPickAndThrow	11616	44	44	6	0.020	0.0016	0.080
FetchPickAndThrow	18207	51	51	7	0.0017	0.014	0.067
FetchPickAndThrow	73960	86	86	10	0.010	0.008	0.044

C. Improvement Based on HGG With Hand-Crafted Distance Metric

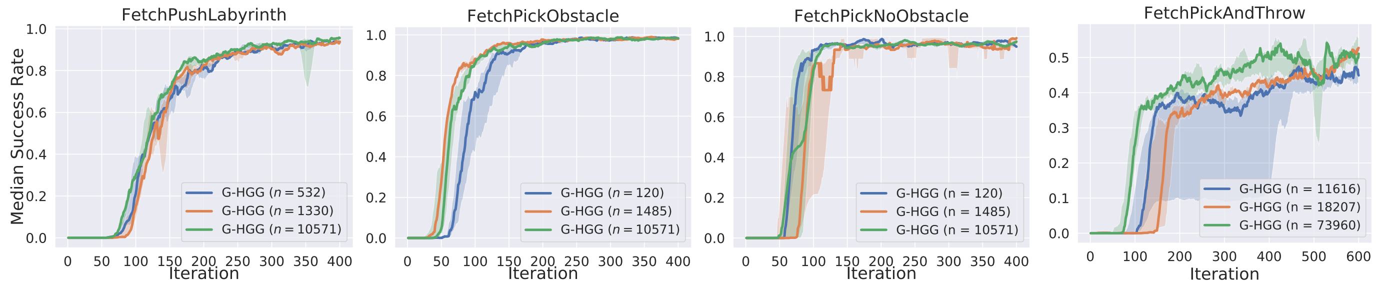
In the original paper of HGG, the authors proposed a hand-crafted distance metric for HGG to solve tasks with obstacles, which was called grid-distance HGG. The authors of HGG indicated that it would be a future direction to investigate ways to obtain or learn a good metric instead of using hand-crafted metrics such a polygonal line. Our G-HGG provides a general solution to find good metrics in complex manipulation tasks and further improve on the basis of HGG. Our work G-HGG further improved the performances from three aspects.

1) The biggest disadvantage of the grid-base distance in HGG is the fact that it is hand-crafted and thus not applicable for different tasks or different goals from the same task. G-HGG proposed a method to compute a graph-based distance automatically regardless of different goals in the same environment and also enable it to be applicable to different tasks. We give two examples to show this disadvantage of HGG. The first example is illustrated in Figure 10a. For the FetchPickObstacle task, if we use a hand-crafted distance metric as HGG did (orange solid line), we can solve the task when the goal appears at the first red point. But for a multi-goal setting, the goal can be generated at any place in the square area and if the goal is far away from the first goal (orange dash line), it will require a different distance metric. We take the FetchPickAndThrow as the second example. Since this task has two rows of targets, it is impossible to solve the multi-goal task with hand-crafted distance metrics. Thus, we can demonstrate the hand-crafted distance metric from HGG is not applicable in this task.
2) The grid-based distance was not accurate or valid when there was an obstacle in the scenario. A detailed example can be found in Figure 11. HGG calculated the distance between one initial point to the goal point with two parts, namely, the minimal distance from the initial point to the grid, and the distance to the goal point along the grid. Take P_1 as an example, the distance is the combination of d_1^1, d_2^1, d_3^1, and d_4^1, which is correct in this case. But for a point $P2$, the distance is the combination of d_1^2, d_2^2, d_3^2, and d_4^2, which is not the shortest distance to the goal. From the figure we can find that a better distance metric to the goal is the distance combination of $d^{2'}_1$ and $d^{2'}_2$.



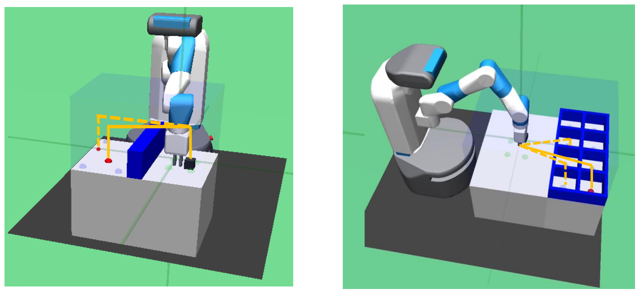
(a) FetchPushLabyrinth (b) FetchPushObstacle (c) FetchPushNoObstacle (d) FetchPickAndThrow

Fig. 8: Ablation study on δ_{stop} for G-HGG and HGG in all four environments.



(a) FetchPushLabyrinth (b) FetchPickObstacle (c) FetchPickNoObstacle (d) FetchPickAndThrow

Fig. 9: Ablation study on n for G-HGG in all four environments. A detailed overview on used parameters is provided in Table I.



(a) FetchPickObstacle. (b) FetchPickObstacle.

Fig. 10: Illustration of two hand-crafted distance metrics in the FetchPickObstacle task.

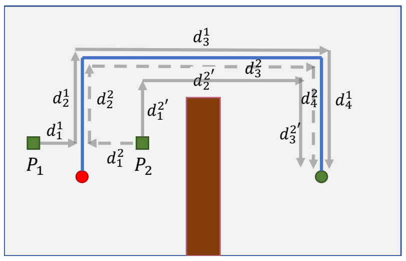


Fig. 11: Illustration of the hand-crafted distance metric from HGG paper.

3) Although it can be easily extended to 3D scenarios, the grid-based distance in HGG only provided a simple example in a 2-D scenario. While all the tasks with obstacles that G-HGG solves in this work are 3-D.

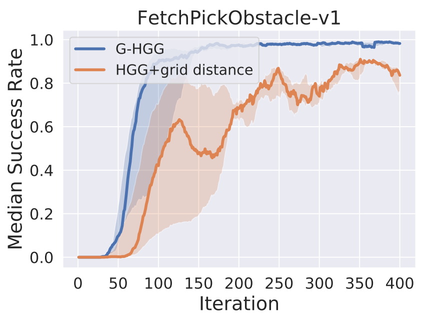


Fig. 12: Median test success rate of G-HGG and the HGG with the hand-crafted distance metric for the FetchPickObstacle task (threshold $\delta_g = 0.03$).

As discussed before, the grid distance used in HGG was hand-crafted, thus it will be impossible to design such a grid distance for every possible goal even in a simple scenario such as the FetchPickObstacle. Thus, we only compare the performances between G-HGG and HGG without the hand-crafted distance metric.

Although the grid-based HGG is not generally applicable for tasks with multi-goal setting and has to be created manually, we still present the results of HGG with a hand-crafted distance metric for the FetchPickObstacle task and compare it to G-HGG for illustration (See Figure 12). For this task, the default distance threshold δ_g is 0.05 and the performance of HGG with hand-crafted distance is shown in Figure 4 of the original HGG paper. The grid-distance based HGG can solve it with a success rate close to 100%. One of the important reasons that the grid-distance based HGG can solve the task is the high density of the target goals. Once the agent reaches real target goals, the algorithm will learn quickly. However, if the target goals are sparsely generated, this inaccurate grid-distance metric will have great difficulties to reach real target goals. In order to simulate a scenario that the target goals are more sparsely distributed, we make the region of target goals smaller by setting the distance threshold δ_g as 0.03. We find that, for this very specific case, grid-based HGG performs worse than G-HGG. With the guidance of the hand-crafted distance metric, grid-based HGG starts to learn slower than G-HGG and only ends at a success rate around 80%. This is because some of the target goals are still far from the hand-crafted distance metric and cannot be reached. However, with the guidance of the automatic graph-based distance metric, G-HGG is able to solve the task better.

VII. Conclusion

We proposed a novel automatic hindsight goal generation algorithm G-HGG on the basis of the HGG for robotic object manipulation in environments with obstacles, by which the selection of valuable hindsight goals is generated with a graph-based distance metric. We formulated our solution as a graph construction and shortest distance computation process as pre-training steps. Experiments on four different challenging object manipulation tasks demonstrated superior performance of G-HGG over HGG and HER in terms of both maximum success rate and sample efficiency.

Future research could concentrate on improvement, extension, and real-world deployment of G-HGG. First, it would be an important advance to bridge the gap between theory and practice by deploying a policy learned with G-HGG to a physical robot. Second, we are positive that G-HGG could as well be applied to more diverse tasks. One possible idea would be to extend G-HGG to more general goal spaces $\mathcal{G} \subset \mathbb{R}^m, m > 3$, for example a six-dimensional goal space in object manipulation including both position and rotation of the object. Last, it would be interesting to investigate how to further increase the performance of G-HGG in sparse-reward robotic object manipulation tasks in environments with obstacles.

Appendix A
Experiment Settings

A. Experiment Settings

For all environments presented in this work, the number of episodes per iteration of HER, HGG or G-HGG is $M = 50$, with $T = 100$ timesteps per episode. The distance threshold δ_g for successfully achieving a target goal is $\delta_g = 0.06$ in FetchPickAndThrow and $\delta_g = 0.05$ in all other environments. G-HGG parameters for each environment as defined in Section IV and used in Section V are provided in Table II.

B. Evaluation Details

- All curves in this work are plotted from five runs with random task initializations and seeds.
- Shaded regions indicate a 50% population around the median.

TABLE II: G-HGG parameters for different environments

Environment	n_x	n_y	n_z	δ_{stop}	n	Δ_x	Δ_y	Δ_z
FetchPushLabyrinth	31	31	11	0.3	10571	0.017	0.023	0.020
FetchPickObstacle	31	31	11	0.3	10571	0.017	0.023	0.040
FetchPickNoObstacle	31	31	11	0.3	10571	0.017	0.023	0.040
FetchPickAndThrow	51	51	7	0.9	18207	0.017	0.014	0.067

- All curves are plotted using the same parameters described in sections A-A and B-B, except the ablation section.

C. Overall Computation Time

In Table III, we list the average computation times of our experimental runs from Section V. The experiments were run on hardware described in B-A. Notably, computation time is highly dependent on hardware, implementation, and other factors. Therefore, the informative value of this table is limited to giving an impression of the order of magnitude and the relationship between computation times of different algorithms.

TABLE III: Computation times (in hours) of experimental runs of G-HGG, HGG, and HER, in different environments. Parameters are described in Sections V, Appendix A, and B.

Environment	HER	HGG	G-HGG
FetchPushLabyrinth	6.5	7	9
FetchPickObstacle	6.5	7	8
FetchPickNoObstacle	6.5	6.5	10
FetchPickAndThrow	10	11	29

Appendix B
Implementation Details

A. Hardware and Software

We carried out experiments on an 8-core machine with 16GB of RAM. We implemented G-HGG in Tensorflow (version 1.14) and ran it on Ubuntu 16.04 with Python 3.5.2. The implementation of G-HGG is based on the HGG implementation given by [30].

B. Hyper-Parameters

Hyper-parameters using DDPG and HER are kept the same as in the results of [30]. These parameters are equal to the ones used for benchmark results given by [1] and [29], except for the number of MPI workers and the buffer size. Hyper-parameters related to graph generation, the details on Data Processing, and training of G-HGG are documented in the github repository at https://videoviewsite.wixsite.com/ghgg.

Acknowledgment

This project/research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No.945539 (Human Brain Project SGA3).

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5049–5059, 2017.

[2] Kavosh Asadi, Dipendra Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning. *35th International Conference on Machine Learning, ICML 2018*, 1(1):419–435, 2018.

[3] A Aubret, L Matignon, and S Hassas. A survey on intrinsic motivation in reinforcement learning. 2019.

[4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[5] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3381–3388, 2017.

[6] Cédric Colas, Pierre Founder, Olivier Sigaud, Mohamed Chetouani, and Pierre Yves Oudeyer. CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 2372–2387, 2019.

[7] E.W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik 1*, 1959.

[8] Ran Duan and Hsin Hao Su. A scaling algorithm for maximum weight matching in bipartite graphs. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1413–1424, 2012.

[9] Manfred Eppe, Sven Magg, and Stefan Wermter. Curriculum goal masking for continuous deep reinforcement learning. *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics, ICDL-EpiRob 2019*, pages 183–188, 2019.

[10] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15220–15231, 2019.

[11] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhang Zhengyou. Curriculum-guided Hindsight Experience Replay. *NeurIPS 2019*, (3):1–12, 2019.

[12] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. *Proceedings of the 35th International Conference on Machine Learning, PMLR 80*, 2018.

[13] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse Curriculum Generation for Reinforcement Learning. *1st Conference on Robot Learning, CoRL 2017*, pages 1–14, 2017.

[14] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. 2017.

[15] Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies. *ArXiv*, abs/1811.07819, 2019.

[16] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–19, 2019.

[17] Z. Huang, X. Xu, H. He, J. Tan, and Z. Sun. Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(4):730–741, 2019.

[18] Bahare Kiumarsi, Kyriakos G Vamvoudakis, Hamidreza Modares, and Frank L Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, 29(6):2042–2062, 2017.

[19] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17:1–40, 2016.

[20] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–27, 2019.

[21] Mufti Mahmud, Mohammed Shamim Kaiser, Amir Hussain, and Stefano Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE transactions on neural networks and learning systems*, 29(6):2063–2079, 2018.

[22] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley. On the effectiveness of least squares generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12):2947–2960, 2019.

[23] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6292–6299, 2018.

[24] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, pages 363–372, 2006.

[25] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *34th International Conference on Machine Learning, ICML 2017*, 6:4261–4270, 2017.

[26] Alexandre Péré, Sebastien Forestier, Olivier Sigaud, and Pierre Yves Oudeyer. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–26, 2018.

[27] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[28] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. pages 1–16, feb 2018.

[29] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–18, 2018.

[30] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via Hindsight Goal Generation. *33rd Conference on Neural Information Processing Systems, NeurIPS 2019*, 2019.

[31] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van De Wiele, Volodymyr Mnih, Nicolas Heess, and Tobias Springenberg. Learning by playing - Solving sparse reward tasks from scratch. *35th International Conference on Machine Learning, ICML 2018*, 10(2017):6910–6919, 2018.

[32] Ludger Rüschendorf. The wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, 1985.

[33] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–21, 2016.

[34] Satinder Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems*, 2005.

[35] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4732–4741, 2018.

[36] Z. Sui, Z. Pu, J. Yi, and S. Wu. Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020.

[37] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, (i):1–16, 2018.

[38] Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. Policy continuation with hindsight inverse dynamics. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[39] Z. Wan, C. Jiang, M. Fahad, Z. Ni, Y. Guo, and H. He. Robot-assisted pedestrian regulation based on deep reinforcement learning. *IEEE Transactions on Cybernetics*, 50(4):1669–1682, 2020.

[40] Y. Wang, H. He, and C. Sun. Learning to navigate through complex dynamic environment with modular deep reinforcement learning. *IEEE Transactions on Games*, 10(4):400–412, 2018.

[41] X. Yang, H. He, and D. Liu. Event-triggered optimal neuro-controller design with reinforcement learning for unknown nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(9):1866–1878, 2019.

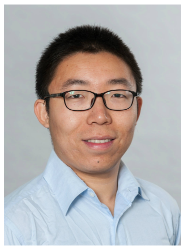
[42] Wei Zhang, Xuanyu He, Weizhi Lu, Hong Qiao, and Yibin Li. Feature aggregation with reinforcement learning for video-based person re-

identification. *IEEE transactions on neural networks and learning systems*, 30(12):3847–3852, 2019.

[43] Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 13022–13035, 2019.

[44] Rui Zhao and Volker Tresp. Energy-Based Hindsight Experience Prioritization. *2nd Conference on Robot Learning, CoRL 2018*, 2018.

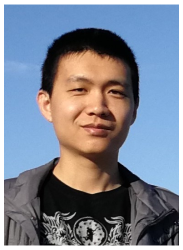
[45] Rui Zhao and Volker Tresp. Curiosity-Driven Experience Prioritization via Density Estimation. *32nd Conference on Neural Information Processing Systems, NIPS 2018*, 2019.



Zhenshan Bing received his doctorate degree in Computer Science from the Technical University of Munich, Germany, in 2019. He received his B.S degree in Mechanical Design Manufacturing and Automation from Harbin Institute of Technology, China, in 2013, and his M.Eng degree in Mechanical Engineering in 2015, at the same university. Dr. Bing is currently a postdoctroal researcher with Informatics 6, Technical University of Munich, Munich, Germany. His research investigates the bio-robot which is controlled by RL and its related applications.

Matthias Brucker Matthias Brucker received his B. Sc. Degree in Engineering Science at Technical University of Munich, Germany, in 2020. He is currently studying Robotics, Systems and Control at the Swiss Federal Institute of Technology in Zurich, Switzerland. His research interest is artificial intelligence on robotics implementations, especially with reinforcement learning algorithms.

Fabrice O. Morin received an engineering degree from the Ecole Nationale Supérieure des Mines de Nancy (Nancy, France) in 1999, a Master's Degree in Bioengineering from the University of Strathclyde (Glasgow, United Kingdom) in 2000, and a Ph.D. in Materials Science from the Japanese Advanced Institute of Science and Technology (Nomi-Shi, Japan) in 2004. After several post-docs at the University of Tokyo (Japan) and the IMS laboratory (University of Bordeaux, France), in 2008 he joined Tecnalia, a nonprofit RTO in San Sebastián (Spain), first as a senior researcher, then as a group leader. He worked on various projects in Neurotechnology and Biomaterials, funded both by public programs and private research contracts. Since 2017, he has worked as a scientific coordinator at the Technical University of Munich (Germany) where, in the framework of the Human Brain Project, he oversees the development of software tools for embodied simulation applied to Neuroscience and Artificial Intelligence.



Rui Li received the B.Eng degree in automation engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2013 and the Ph.D. degree from the Institute of Automation, Chinese Academy of Science (CASIA), Beijing, China in 2018, respectively. He visited Informatics 6, Technical University of Munich as a guest postdoc researcher in 2019. Currently he is a assitant researcher with School of Automation, Chongqing University. His research interests include intelligent robot system and high-precision assembly.

Xiaojie Su (SM'18) received the PhD degree in Control Theory and Control Engineering from Harbin Institute of Technology, China in 2013. He is currently a professor and the associate dean with the College of Automation, Chongqing University, Chongqing, China. He has published 2 research monographs and more than 50 research papers in international referred journals.

His current research interests include intelligent control systems, advanced control, and unmanned system control. He currently serves as an Associate Editor for a number of journals, including IEEE Systems Journal, IEEE Control Systems Letters, Information Sciences, and Signal Processing. He is also an Associate Editor for the Conference Editorial Board, IEEE Control Systems Society. He was named to the 2017, 2018 and 2019 Highly Cited Researchers list, Clarivate Analytics.

Kai Huang joined Sun Yat-Sen University as a Professor in 2015. He was appointed as the director of the Institute of Unmanned Systems of School of Data and Computer Science in 2016. He was a senior researcher in the Computer Science Department, the Technical University of Munich, Germany from 2012 to 2015 and a research group leader at fortiss GmbH in Munich, Germany, in 2011. He earned his Ph.D. degree at ETH Zurich, Switzerland, in 2010, his MSc from University of Leiden, the Netherlands, in 2005, and his BSc from Fudan University, China, in 1999. His research interests include techniques for the analysis, design, and optimization of embedded systems, particularly in the automotive and robotic domains. He was awarded the Program of Chinese Global Youth Experts 2014 and was granted the Chinese Government Award for Outstanding Self-Financed Students Abroad 2010. He was the recipient of Best Paper Awards ESTC 2017, ESTIMedia 2013, SAMOS 2009, Best Paper Candidate ROBIO 2017, ESTMedia 2009, and General Chairs' Recognition Award for Interactive Papers in CDC 2009. He has served as a member of the technical committee on Cybernetics for Cyber-Physical Systems of IEEE SMC Society since 2015.

Alois Knoll (Senior Member) received his diploma (M.Sc.) degree in Electrical/Communications Engineering from the University of Stuttgart, Germany, in 1985 and his Ph.D. (*summa cum laude*) in Computer Science from Technical University of Berlin, Germany, in 1988. He served on the faculty of the Computer Science department at TU Berlin until 1993. He joined the University of Bielefeld, Germany as a full professor and served as the director of the Technical Informatics research group until 2001. Since 2001, he has been a professor at the Department of Informatics, Technical University of Munich (TUM), Germany . He was also on the board of directors of the Central Institute of Medical Technology at TUM (IMETUM). From 2004 to 2006, he was Executive Director of the Institute of Computer Science at TUM. Between 2007 and 2009, he was a member of the EU's highest advisory board on information technology, ISTAG, the Information Society Technology Advisory Group, and a member of its subgroup on Future and Emerging Technologies (FET). His research interests include cognitive, medical robotics, multi-agent systems, data fusion, adaptive systems, multimedia information retrieval, model-driven development of embedded systems with applications to automotive software and electric transportation, as well as simulation systems for robotics and traffic.