Robotic Manipulation in Dynamic Scenarios via Bounding Box-Based Hindsight Goal Generation

Zhenshan Bing, Erick Álvarez, Long Cheng*, Fabrice O. Morin, Rui Li, Xiaojie Su, Kai Huang, and Alois Knoll, *Senior Member, IEEE*

***Abstract*—By relabeling past experience with heuristic or curriculum goals, state-of-the-art reinforcement learning (RL) algorithms such as hindsight experience replay (HER), hindsight goal generation (HGG), and graph-based hindsight goal generation (G-HGG) have been able to solve challenging robotic manipulation tasks in multi-goal settings with sparse rewards. HGG outperforms HER in challenging tasks in which goals are difficult to explore by learning from a curriculum, in which intermediate goals are selected based on the Euclidean distance to target goals. G-HGG enhances HGG by selecting intermediate goals from a precomputed graph representation of the environment, which enables its applicability in an environment with stationary obstacles. However, G-HGG is not applicable to manipulation tasks with dynamic obstacles, since its graph representation is only valid in static scenarios and fails to provide any correct information to guide the exploration. In this paper, we propose bounding box-based hindsight goal generation (Bbox-HGG), an extension of G-HGG selecting hindsight goals with the help of image observations of the environment, which make it applicable to tasks with dynamic obstacles. We evaluate Bbox-HGG on four challenging manipulation tasks, where significant enhancements in both sample efficiency and overall success rate are shown over state-of-the-art algorithms. The videos can be viewed at https://videoviewsite.wixsite.com/bbhgg.**

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

I. Introduction

THE study of deep reinforcement learning (RL) has enabled robots to perform many complex tasks [30], such as organizing books on a bookshelf [25], inserting a peg in a hole [27], achieving autonomous navigation of vehicles [1] and aerial drones [4]. The principle of RL is to learn an optimal policy through interactions with the environment by the agents. These interactions provide agents with rewards, which is the only mechanism through which agents can learn how to complete the tasks successfully. However, in most complex robotic tasks, where a concrete representation of efficient or even admissible behavior is unknown, it is extremely challenging and time-consuming to design an adequate task-tailored reward, thereby making this strategy impractical for wide robotic applications of RL.

Favorably, most tasks have clear success and failure conditions, which can be used to define a binary reward signal that indicates the task completion. This kind of binary reward is also known as a sparse reward and is easy to derive from the task definition with minimum effort. However, RL algorithms supporting sparse rewards usually suffer from bad learning efficiency, since they can only deliver shallow and insufficient information during training. To overcome this issue, Andrychowicz et al. [2] proposed the algorithm hindsight experience replay (HER), which improves the success of off-policy RL algorithms in multi-goal RL problems with sparse rewards. The concept of HER is to use previous experiences collected by the agent to define hindsight goals that are easy to learn at first, and then continue with more difficult goals. While HER has been proven to work efficiently in environments where goals can be easily reached through random explorations, it fails in environments where the goal distributions are far away from the initial states and hard to reach only by random exploration and the heuristic choice of hindsight goals from achieved states.

To tackle this problem, Ren et al. [23] propose hindsight goal generation (HGG), which uses hindsight goals as an implicit curriculum to guide the exploration towards intermediate goals that are easy to achieve in the short term and promising to lead to target goals in the long term. Despite HGG being successful at solving tasks with distant goals, it fails to solve tasks with obstacles in which its distance mechanism cannot be computed with the Euclidean metric. Our previous work, graph-based hindsight goal generation (G-HGG) [3], overcomes this problem by selecting hindsight goals based on the shortest distances in a precomputed obstacle-avoiding graph, which is a discrete representation of the environment. G-HGG shows outstanding performance in complex manipulation tasks with static obstacles compared with HER or HGG, but it assumes that the knowledge of the obstacles' dimensions and positions are known and constant, so that it can precompute the graph-based distance before the training. However, in most tasks, the obstacles' locations are not always known to the robot and they may change their positions dynamically, which makes HGG or G-HGG not applicable to such a task.

A common approach to obtain the information of a dynamic scenario at every time step is to use image observations, which are usually easy to obtain and can capture a great range of

Z. Bing, E. Álvarez, F. Morin, and A. Knoll are with the Department of Informatics, Technical University of Munich, Germany. E-mail: {bing, morinf, knoll}@in.tum.de, ga74kob@mytum.de.

L. Cheng is with the School of Computer Science, Wenzhou University, China. Email: longcheng.work@outlook.com.

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

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

*Corresponding author: Long Cheng. Email: longcheng.work@outlook.com

task-relevant features. Therefore, in this paper, we introduce bounding box-based hindsight goal generation (Bbox-HGG), an extension of G-HGG, to solve complex robotic manipulation tasks in dynamic scenarios with the help of image observations. We state that such tasks cannot be solved by state-of-the-art sparse-reward RL algorithms. We formulate our Bbox-HGG algorithm as *bounding box creation*, *obstacle-avoiding graph construction* as pretraining steps, and *state extension and multi-objective sparse rewards* as critical training steps. To make Bbox-HGG applicable to environments with dynamic obstacles, we first use the bounding box information extracted from image observations to create a graph-based representation of the environment to eliminate static obstacles from goal spaces. We second utilize the bounding box information to extend the observation states that can provide essential information about the dynamic obstacles to learn desired behaviors. We third propose a multi-objective sparse reward to penalize behaviors that will lead to any collision. Last, we design four new challenging robotic manipulation tasks, which contain both static and dynamic obstacles, to compare the performance of Bbox-HGG, G-HGG, and HGG.

Our main contribution to the literature is a sparse-reward algorithm that can solve complex manipulation tasks with dynamic obstacles using image observations. Specifically, first, we propose a self-supervised mechanism to train a bounding box encoder (BboxEncoder) to recognize the bounding boxes of objects from image observations. This BboxEncoder offers a practical way to extract object information from an unknown dynamic environment, which can be further used for training goal-conditioned RL agents. Second, we propose a mechanism to estimate the dimensions and locations of an obstacle to automate the creation of the obstacle-avoiding graph and its respective graph-based distances, which can be used to generate intermediate goals like HGG and G-HGG. Third, we propose a multi-objective sparse reward to penalize the agent for colliding with obstacles, which only requires minimum engineering to be adaptive to different environments. Last, experiment results demonstrate that Bbox-HGG provides a significant enhancement in both sample efficiency and overall success rate over G-HGG and HGG.

II. Related Work

Since our work consists of different modules, namely, robotic manipulation with sparse-reward-based RL, image-based RL, and object recognition, we briefly discuss the related work from these three main topics.

A. Manipulation with Sparse-Reward RL

Robotic manipulation, as a kind of challenging tasks, has been widely used to examine the performance of many different RL approaches, such as those based on experience relabeling [2], [36], intrinsic motivation [35], [6], and guided exploration and exploitation [9], [23]. The basic idea behind these approaches is to improve the exploration efficiency, which is crucial in sparse-reward RL settings, where effective exploration is extremely difficult in manipulation tasks due to the sparsity of the goal space and the uninformative sparse rewards. Experience relabeling-based approaches, represented by HER [2], leverage the notion that some uninformative data for one task is likely a rich source of information for another task. Some other hindsight experiences relabelling approaches further investigated how to improve data efficiency. For example, Li et al. [14] proposed a general multi-task hindsight relabelling approach from the perspective of inverse RL. Eysenbach et al. [8] further demonstrated that hindsight relabeling is inverse RL by maximizing the entropy and derived the theoretical optimum form of hindsight relabelling approaches. Intrinsic motivation-based approaches are inspired by the self-consciousness concept which encourages the agent to explore by providing an internal motivation. For example, in [29], the tactile information from the gripper is used to construct the intrinsic reward to encourage the interaction between the gripper and objects in the environment. Guided exploration extends the experience relabeling approaches by creating an implicit curriculum of hindsight goals to lead the exploration towards target goals. The guidance metric can be calculated using Euclidean distance [23], [9] or other customized distances [3].

B. Manipulation with Image-Based RL

To solve more complex tasks in which dynamic obstacles are involved, using images as state representations is an appealing idea since they are easy to get and contain plentiful information about the environment, regardless of whether it is static or dynamic. Since images are high-dimensional and less intuitive for the agent to directly learn, researchers have studied different methods to abstract low-dimensional representations from images to learn complex behaviors. Nasiriany et al. proposed latent embeddings for abstracted planning (LEAP [17]), which encodes images using a variational autoencoder (VAE) as a latent observation for learning a goal-conditioned RL policy. LEAP also trained a temporal difference model [21] to calculate a value function to predict if a goal is reachable, and therefore used the value function as a planner to select suitable intermediate goals sampled from the VAE for downstream tasks. Hafner et al. developed a model-based RL approach to work with images, which used a recurrent space model [11] to learn the dynamics of the environment in a latent space. They also utilized a VAE to encode and decode images and the recurrent network received sequences of latent representations and actions to predict the dynamic model of the task. Similar work such as curious object-based search agent (COBRA [31]) also integrated model-based RL with image observations and learned the dynamics of the environment from the latent representations.

C. Unsupervised Object Discovery

The capability to discover objects from visual observations is important for robotics, and therefore there are many studies that investigate unsupervised object discovery [34]. The Multi-Object Network (MONet) [5] was proposed to learn to decompose and represent complex scenarios into semantic components in an unsupervised manner by providing attention masks and reconstructing regions of images. Another

unsupervised object discovery method is scene representation via spatial attention and decomposition (SPACE [15]), which uses probabilistic inference to model images and generate factorized object representations. Similar to MONet and SPACE, Nash et al. proposed multi-entity variational autoencoder (MVAE [16]) to discover objects, in which the encoder returns a grid of latent representations, selects N representations with the highest KL divergence, and reconstructs them. Greff et al. developed the Iterative Object Decomposition Inference Network (IODINE) [10], which encodes an image with K latent variables and iteratively refines them to reconstruct the objects from the image correctly. These methods approach the unsupervised object discovery problem by learning latent variables that are independent for each object and can represent the appearance and position of objects.

III. Preliminaries

A. Multi-Object Network (MONet)

MONet is an algorithm that can decompose and represent challenging image scenes by jointly training an attention network and a VAE in an unsupervised manner [5]. The attention network ψ is used to create K different masks $m_1, ..., m_K \in \mathbb{R}^{H\times W\times 1}$ that divide the input image $\Lambda \in \mathbb{R}^{H\times W\times C}$ into regions so that each region only contains one single object including the background. H, W, and C are the height, width, and channel of an image. MONet concatenates each mask with the image and passes this information to the VAE, which reconstructs all the masks $\hat{m}_1, ..., \hat{m}_K \in \mathbb{R}^{H\times W\times 1}$ and parts of the image $\hat{\Lambda}_1, ..., \hat{\Lambda}_K \in \mathbb{R}^{H\times W\times C}$. The latent variables $z_1, ..., z_K$ contain the information to represent the image Λ. Since the latent space encodes the features of an object, the encoder ϕ has a posterior distribution $q_\phi(z_k|\Lambda, m_k)$ and the decoder θ the prior $p_\theta(\Lambda|z_k)$. Then, the network models the distribution $p(c|\{m_k\})$ that shows that some component c of the image is represented in the k^{th} slot. The corresponding posterior and prior are $q_\psi(c|\Lambda)$ and $p_\theta(c|\{z_k\})$. This network is trained with the following loss:

$$\begin{aligned}\mathcal{L}(\theta,\phi,\psi,\Lambda) = & -\log\sum_{k=1}^{K} m_k p_\theta(\Lambda|z_k) \\ & +\beta D_{KL}\left(\prod_{k=1}^{K} q_\phi(z_k|\Lambda,m_k)\|p(z)\right) \\ & +\gamma D_{KL}\left(q_\psi(c|\Lambda)\|p_\theta(c|\{z_k\})\right),\end{aligned} \tag{1}$$

where β and γ are hyperparameters to balance each component of the loss function. D_{KL} is the KL-divergence.

B. Hindsight Experience Replay (HER)

Hindsight Experience Replay (HER [2]) is a simple yet effective RL algorithm designed for goal-oriented tasks with sparse rewards, in which an agent usually fails to learn efficiently. This is because the uninformative sparse reward can only provide very shallow information about the task and the sparsity of the goal space makes the exploration even more challenging during training. To improve the learning efficiency, HER relabels the hindsight experience by leveraging the notion that some uninformative data for one goal is likely a rich source of information for another goal. In a multi-goal RL task with sparse rewards, HER assumes that every goal g corresponds to a predicate $f_g : \mathcal{S} \to \{0, 1\}$. A goal is considered as reached, once the agent achieves any state s that satisfies $f_g(s) = 1$. A sparse reward function is defined as $r_g(s, a) = -[f_g(s) = 0]$, meaning that the agent constantly receives negative rewards as long as it has not reached the goal. Only when the goal is reached, can zero reward be observed. In HER, every transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ is not only stored with the original goal g used for the episode, but also with a subset of other goals (hindsight goals) g' as $(s_t||g', a_t, r_t, s_{t+1}||g')$. Therefore, when replaying the resulting transitions $(s_t||g', a_t, r_t, s_{t+1}||g')$, the agent is more likely to encounter informative rewards. HER can be interpreted as an implicit curriculum, which first concentrates on intermediate goals that are easy to reach, and then moving on to more difficult goals that are closer to the target goals.

C. Hindsight Goal Generation (HGG)

HGG [23] extends HER to tasks with distant goal distributions that are far away from the initial state distribution and cannot be solved by heuristic exploration. These target goals $\mathcal{G}_T$ belong to a goal space $\mathcal{G}$ and the initial states $\mathcal{S}_0$ belong to the state space $\mathcal{S}$. The distribution $\mathcal{T}^* : \mathcal{G} \times \mathcal{S} \to \mathbb{R}$ determines how they are sampled. Instead of optimizing V^π with the difficult target goal - initial state distribution $\mathcal{T}^*$, which carries the risk of being too far from the known goals, HGG tries to optimize with a set of intermediate goals sampled from $\mathcal{T}$. On the one hand, the goals contained in $\mathcal{T}$ should be easy to reach, which requires a high $V^\pi(\mathcal{T})$. On the other hand, goals in $\mathcal{T}$ should be close enough to $\mathcal{T}^*$ to be challenging for the agent. This trade-off can be formalized as

$$\max_{\mathcal{T},\pi} V^\pi(\mathcal{T}) - L \cdot \mathcal{D}(\mathcal{T}^*, \mathcal{T}). \tag{2}$$

The Lipschitz constant L is treated as a hyper-parameter. In practice, to select these goals, HGG first approximates $\mathcal{T}^*$ by taking K samples from $\mathcal{T}^*$ and storing them in $\hat{\mathcal{T}}^*$. Then, for an initial state and goal $(\hat{s}_0^i, \hat{g}^i) \in \hat{\mathcal{T}}^*$, HGG selects a trajectory $\tau = \{s_t\}_{t=1}^T$ that minimizes the following function:

$$\begin{aligned}w(\hat{s}_0^i, \hat{g}^i, \tau) := & \, c\|m(\hat{s}_0^i) - m(s_0)\| \\ & + \min_{s_t\in\tau}\left(\|\hat{g}^i - m(s_t)\| - \frac{1}{L}V^\pi((s_0||m(s_t))\right).\end{aligned} \tag{3}$$

$m(\cdot)$ is a state abstraction to map from the state space to the goal space. $c > 0$ provides a trade-off between 1) the distance between target goals and 2) the distance between the goal representation of the initial states. 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 minimized (3). More formally,

$$g^i := \arg\min_{s_t\in\tau}\left(\|\hat{g}^i - m(s_t)\| - \frac{1}{L}V^\pi((s_0\|m(s_t))\right). \tag{4}$$

D. Graph-Based Hindsight Goal Generation (G-HGG)

G-HGG [3] identifies that the Euclidean distance metric $\|\cdot\|$ used in (3) and (4) is not applicable in environments with

obstacles, where it is not an accurate distance metric. And this leads to non-optimal or even incorrect intermediate goals for the agent to solve the task. G-HGG proposes replacing the Euclidean distance in HGG with a graph-based distance extracted from an obstacle-free graph $G = (V, E)$. The graph G serves as a discrete representation of the accessible goal space in the environment. Thus, one must define $\mathcal{G}_A \subset \mathcal{G}$ where $\mathcal{G}_A$ represents all the accessible goals of the environment. This implies that if some goal from the goal space $g \in \mathcal{G}$ lies inside an obstacle, then it is excluded from the accessible goal space. Additionally, a set of vertices V and weighted edges E are defined to discretize the accessible goal space. The Dijkstra's algorithm [7] is used to calculate the shortest path distance between each node pair and store them in a table. With this table, it is possible to create a metric d_G that maps any two points $g_1, g_2 \in \mathcal{G}$ to the closest discretized coordinates and read the distance from the table. d_G is used to replace the term $\|\hat{g}^i - m(s_t)\|$ in (3) and (4) with $d_G(\hat{g}^i - m(s_t))$.

IV. Problem Statement

In this paper, we focus on learning manipulation skills in dynamic environments via RL with sparse rewards, in which the localization information of each object is unknown to the RL agent. This constraint makes state-of-the-art RL algorithms inapplicable to such an environment that shares the following characteristics:

- An internal state space $S_{int} \subset \mathbb{R}^l$, $l \in \mathbb{N}$. It contains the internal information of a robotic arm, such as the joint positions and angular velocities.
- An external state space $S_{ext} \subset \mathbb{R}^3$. It is an image observation Λ that captures the information of the environment.
- A multidimensional state space S, which is the concatenation of S_{int} and S_{ext}.
- An action space $A \in \mathbb{R}^3$ that controls the position of the end effector.
- An initial state distribution $S_0 : S \to [0, 1]$.
- A goal space $\mathcal{G} \subset \mathbb{R}^2$. A goal is defined as a point on one 2D plane in the environment.
- A target goal distribution $\mathcal{G}_T \subset \mathcal{G} \to [0, 1]$.
- A goal predicate $f_g : \mathcal{S} \to \{0, 1\}, g \in \mathcal{G}$ to determine if a state is under the distance threshold δ_g to a goal:

$$f_g(s) := \begin{cases} 1, & \text{if } \|m(s) - g\| \leq \delta_g \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

- A sparse reward function $r_g : S \times A \to \mathbb{R}$ defined as:

$$r_g(s) := \begin{cases} 0, & \text{if } f_g(s) = 1 \\ -1, & \text{otherwise.} \end{cases} \tag{6}$$

- Obstacles $\{o_1, o_2, ...\}$ that can be either static or dynamic. In our environments, we consider obstacles located on a table; therefore, if an obstacle o_i is dynamic, it performs a linear motion with a velocity $v_i \in \mathbb{R}^2$ and two limit positions that o_i can reach periodically.

As introduced and explained in prior work, the task of designing RL algorithms that can learn manipulation skills in dynamic scenarios via sparse rewards is challenging and remains unsolved. The reasons are listed as follows.

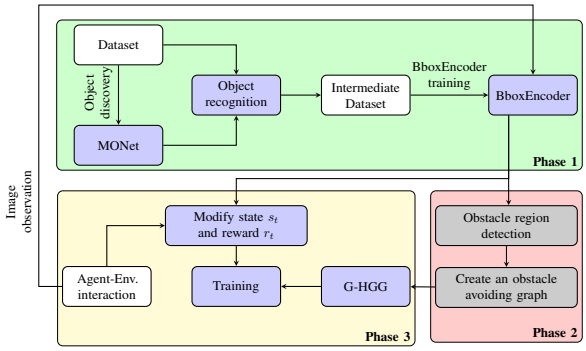


Fig. 1: Overview of the phases of our solution and their correlation. The BboxEncoder from the first phase is used in the second phase and the third phase. The graph created in the second phase is used in the third phase.

- Certain approaches, like HER [2] and EBP [36], only achieve their success through hindsight replays of past experience with heuristic goals. These methods are not able to learn long distant goals.
- Some guided exploration approaches can learn distant goals but are only applicable to scenarios with no obstacles or static obstacles since they require environmental information beforehand (CHER [9], HGG [23], G-HGG [3]). Therefore, these methods are not applicable to environments with dynamic obstacles.
- Prior studies assume complete knowledge of the localization information of static obstacles, which is either unknown or difficult to be obtained in the real world.

To tackle this problem, we consider using image observations as a way to acquire the localization information of the environment and then propose a method that can learn manipulation skills in dynamic scenarios via RL with a sparse-reward setup. In particular, our method aims to achieve the following goals. First, our method should be able to derive the object information from a dynamic environment using image observations, such as identifying and locating the manipulatable object and obstacles. Second, based on the derived information, our method should create a representation of the environment that can be used to generate accessible hindsight goals for the agent. Third, our method should learn a policy by only using sparse rewards, which allows the agent to reach distant goals and yet prevent collisions in the environment with minimal engineering effort.

V. Methodology

In this section, we first present the overview of our algorithm Bbox-HGG. Then, we give a detailed explanation of each component of Bbox-HGG, namely, the training of our BboxEncoder for object recognition, the adaption of G-HGG to the dynamic environment, and the extension of the state space and sparse reward function. Finally, we summarize Bbox-HGG with its pseudocode.

A. Overview

To tackle the limitations of prior works, our method first aims to acquire the localization information of each object in the

environment from image observations. Second, similar to the idea of G-HGG, our method also creates an obstacle-free graph, but with the environmental information obtained from the first step instead of using prior knowledge. Last, we introduce two additional mechanisms, namely, the extended observation and multi-objective sparse reward, to allow our policy to solve manipulation tasks in environments with dynamic obstacles. The overall architecture of our algorithm is shown in Figure 1. The algorithm is briefly explained in three phases as follows.

- In the first phase, we design a bounding box encoder (BboxEncoder) that can recognize the bounding box of each object in the environment via image observations. We use an indexing mechanism to differentiate the manipulatable object from obstacles.
- In the second phase, we estimate the regions that each obstacle will constantly occupy through the episode and create a graph avoiding regions that are constantly occupied by obstacles so that G-HGG can be applied to generate proper intermediate goals for training.
- In the third phase, we extend the observation space with the localization information of the bounding boxes. We also design a multi-objective sparse reward to penalize any collisions with the obstacles.

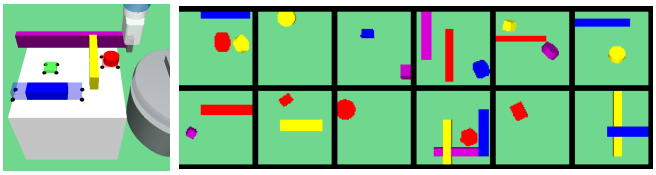
The first two phases are performed before the training of the RL agent and the third phase takes place during the training.

B. Bounding Box Encoder (BboxEncoder)

The objective of our BboxEncoder is to create a representation of the environment that can take images as the input and output the position and dimension information of the bounding box of each object inside. The BboxEncoder is trained with an MONet (See Section III-A) and an image dataset with objects rendered at random locations, which allows it to work in different environments. The MONet is also trained using the dataset in an unsupervised manner. The training pipeline of our BboxEncoder is summarized as follows.

- First, to train MONet to discover objects, we create a dataset with raw images that are randomly rendered from different environments.
- Second, with this raw dataset, we train MONet to discover each bounding box of the object via an unsupervised manner. It should be noted that the MONet can only create a mask for each object from the raw images, rather than generating the bounding box that we need.
- Third, we train our BboxEncoder to detect all the bounding boxes in a self-supervised manner using the dataset and the masks generated by MONet.

1) Dataset: Our tasks are performed in MuJoCo [28] environments based on the standard robotic manipulation benchmark from OpenAI Gym [20]. All our environments are manipulation tasks featuring a Fetch robot with a gripper that pushes a puck in an environment with dynamic obstacles (See Figure 2a). To model different scenarios, the dataset is created by rendering $38,400$ images of instantiated objects with different shapes, sizes, or colors. A more detailed description of the generated dataset can be found in Appendix A-A and some sampled images are visualized in Figure 2b. It should be noted that, during the dataset generation and the RL-training, we set the table and robot arm invisible when rendering an image to facilitate the identification of objects relevant for the task. In practice, it is possible to position the camera in a place where the robot arm does not occlude other objects or use multiple cameras to obtain the full information.



(a) Demo scene. (b) Top-view sampled images from the dataset.

Fig. 2: Visualization of the environment and the image dataset.

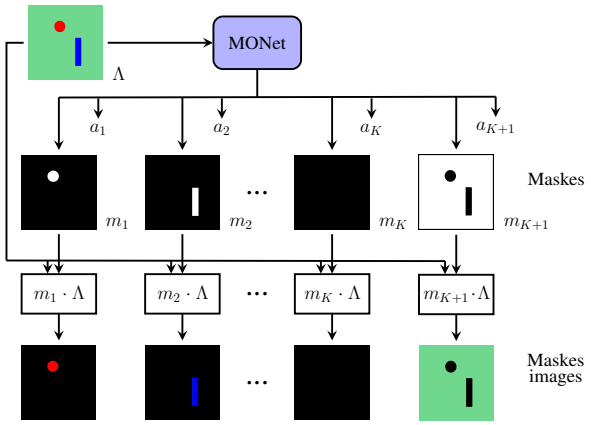


Fig. 3: The overview of MONet. This network receives an image Λ and creates masks (black and white images) $m_1, m_2, ..., m_{K+1}$ and latent vector representations $a_1, a_2, ..., a_K, a_{K+1}$ for the objects. The bottom row shows the masked images that are the multiplication of the image with each mask.

2) Object Discovery: In this paper, we use MONet to discover objects from image observations so that we can further train our BboxEncoder. Compared with other object discovery methods, MONet has a determined amount of slots for encoding different objects and one object is always assigned to the same slot. MONet also creates one mask for each slot, with which we can create a masked image that only contains one object. In our implementation, MONet is initialized with $K+1$ slots. The $K+1^{th}$ slot is designated to model the background, while other slots model the foreground objects. We train MONet with the dataset created from Section V-B1. And with a well-trained MONet, it can take an image Λ as the input and output latent vectors $a_1, a_2, ..., a_{K+1}$ and masks $m_1, m_2, ..., m_{K+1}$. It is important to note that Λ can have fewer than K objects and therefore some variables and masks will represent no object. Figure 3 shows the architecture of MONet, the masks it creates, and the masked images.

3) BboxEncoder: The BboxEncoder is designed to take an image Λ as the input and yield the axis-aligned bounding boxes surrounding those objects in the image. One bounding box is defined as a vector $b \in \mathbb{R}^4$, which consists of a position vector

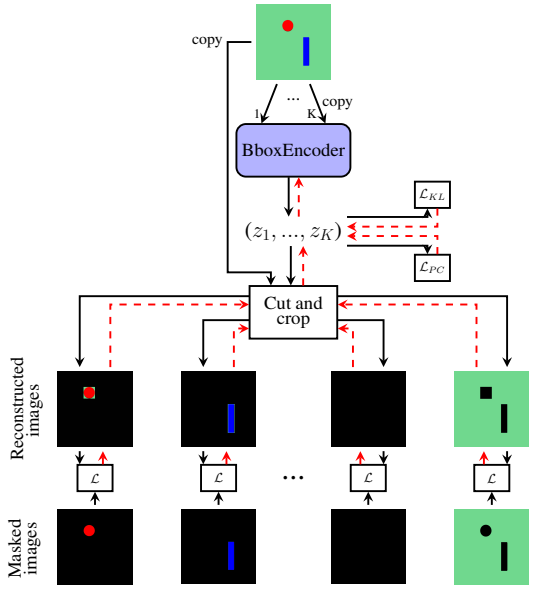


Fig. 4: Training pipeline of the BboxEncoder. It receives an image Λ as the input and the latent variables $z_1, z_2, ..., z_K$ contains the information of each bounding box. Cut and crop operations are used to reconstruct the images and the masked images from MONet are used to train the BboxEncoder in a self-supervised manner.

Fig. 5: Spatial transformations as cut and crop operations.

$p = (x, y)$ and a dimension vector $d = (w, h)$, where (x, y) is the center of the bounding box and (w, h) are half of the size of the width and height. Therefore, an image Λ with n objects contains the bounding boxes $b_1, b_2, ..., b_n, b_{n+1}$.

The overview of the training pipeline of the BboxEncoder is shown in Figure 4. During training, the BboxEncoder receives an image Λ as the input and outputs bounding boxes $(b_1, b_2, ..., b_K)$ to reconstruct the masked images $(m_1 \cdot \Lambda, ..., m_{K+1} \cdot \Lambda)$ by cut and crop operations on the image Λ. BboxEncoder uses the latent variable z_k to represent the k^{th} bounding box, namely, $z_k^p \in \mathbb{R}^2$ for positions, $z_k^d \in \mathbb{R}^2$ for the dimensions, and $z_k^{pres} \in \mathbb{R}$ for the presence of the bounding boxes. With z_k^p and z_k^d, we perform cut and crop operations by applying the spatial transformation with matrix $M = \begin{bmatrix} z^d(0) & 0 & z_k^p(0) \\ 0 & z_k^d(1) & z_k^p(1) \end{bmatrix}$, as illustrated in Figure 5. By applying the spatial transformation, we can get the reconstructed images, which are expected to approximate the masked images from MONet. It should be noted that the BboxEncoder also models empty bounding boxes since some masked images $m_k \cdot \Lambda$ are empty as explained above. To recognize such bounding boxes, we use variable z_k^{pres} as a probability to indicate the presence of an object in $m_k \cdot \Lambda$.

The architecture of the BboxEncoder comprises K convolutional blocks, one for each slot representing objects. Each of them receives Λ as the input and extracts the features for the corresponding object. These outputs are passed to three shared multi-layer perceptrons, which outputs K vectors containing the variables z_k^p, z_k^d, and z_k^{pres}. The architecture of the BboxEncoder is shown in Figure 6. With $z_k = (z_k^p, z_k^d, z_k^{pres})$, the loss used to train this network is:

$$\begin{aligned} \mathcal{L}(\Lambda) = \mathop{\mathbb{E}}_{q(\{z_k\}_{k=1}^K|\Lambda)} \Big[& -\log \sum_{k=1}^{K} p(\Lambda \cdot m_k | z_k) \\ & - \alpha \cdot \log p(\bar{\Lambda} | \{z_k\}_{k=1}^K) \\ & + \sum_{k=1}^{K} D_{KL}(q(z_k|\Lambda \cdot m_k) \| p(z_k)) \Big] + \beta \cdot \mathcal{L}_{PC} \end{aligned} \quad . \quad (7)$$

The first term corresponds to the probability that the bounding box of a slot contains the object extracted by the mask. The second term corresponds to the probability that we can reconstruct the background by removing all bounding boxes modeled by the latent variables. In this term, α is a binary coefficient. We use it to activate the second term after a certain number of training steps. $\overline{\Lambda}$ is the background image. Since we are reconstructing the images with cut and crop operations, this loss helps to create more accurate bounding boxes. The third term is the KL-divergence of the latent variables.

The last term is an additional loss called the perceptual cycle-consistency (PC) loss, which is inspired by [32] and we adapt it for our solution. $\mathcal{L}_{PC}$ is used to improve the disentanglement between object representations, which means that the appearance variation of one object is reflected only in the corresponding latent variable that models this object instead of other latent variables. The way to construct this loss is described as follows. We first create an augmented latent variable $\hat{z}_i, 0 \leq i \leq K$ by setting:

$$\hat{z}_i^p \sim \mathcal{U}(-1, 1), \ \hat{z}_i^d \sim \mathcal{U}(-0.02, 0.02) + z_i^d, \ \hat{z}_i^{pres} \leftarrow z_i^{pres}, \quad (8)$$

where $\mathcal{U}$ means the uniform distribution. Second, we create the augmented image $\overline{\Lambda}$ using $\hat{z}_i$ and the set $\{z_k\}_{k=1,k\neq i}^K$ by

$$\overline{\Lambda} \leftarrow reconstruct(\{z_k\}_{k=1,k\neq i}^K \cup \hat{z}_i). \quad (9)$$

We reconstruct a new image $\overline{\Lambda}$ by combining all the cut and cropped images into a new image. The variables $\{z_k\}_{k=1,k\neq i}^K$ and $\hat{z}_i$ determine the position and dimension of the corresponding cut and cropped images in the image $\overline{\Lambda}$. Third, we feed $\overline{\Lambda}$ to our BboxEncoder Φ and we can get $\{\overline{z}_k\}_{k=1}^K \leftarrow \Psi(\overline{\Lambda})$. Finally, the loss $\mathcal{L}_{PC}$ is calculated as the mean squared error (MSE) between the set of variables of the original image Λ and the reconstructed image $\overline{\Lambda}$ without considering the selected index i as

$$\mathcal{L}_{PC} \leftarrow MSE(\{\overline{z}_k\}_{k=1,k\neq i}^K, \{z_k\}_{k=1,k\neq i}^K) \quad (10)$$

In (7), we use α and β to activate the respective terms of the losses in later training steps, since the bounding boxes are not correct at the beginning.

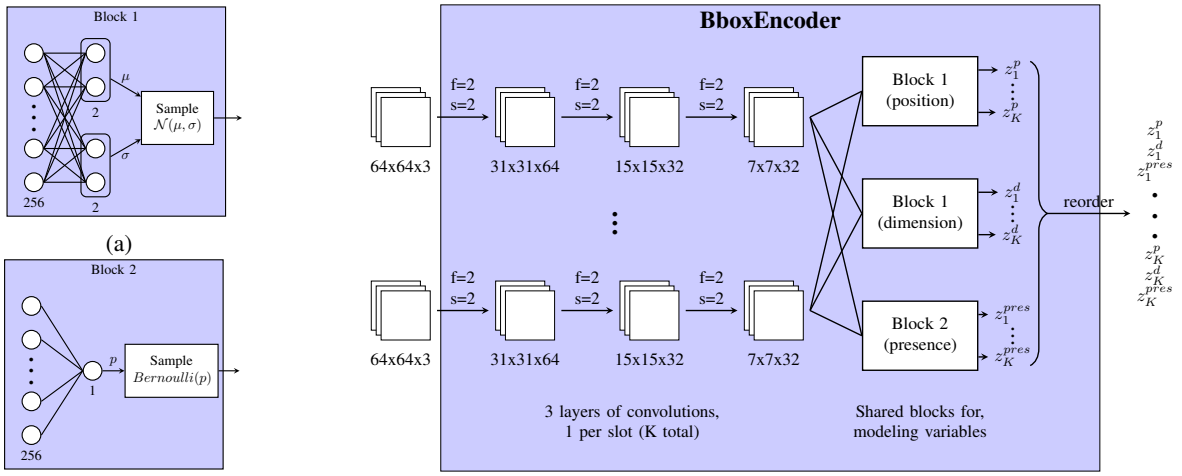


Fig. 6: Architecture of BboxEncoder. (a) Block 1 of Multi-layer perceptrons for sampling variational variables. (b) Block 2 of Multi-layer perceptrons for sampling variational variables. (c) Convolutional layers are on the left and connect to the block of Multi-layer perceptrons.

C. Adaption of G-HGG

In this section, we first provide a strategy to infer the index of each bounding box to differentiate the manipulatable object and obstacles. Second, we explain how to automatically create the graph representation of an environment with dynamic obstacles like G-HGG.

1) Object Index Inference: Despite the fact that the BboxEncoder can localize objects and extract their bounding boxes, it is not possible to automatically differentiate different types of bounding boxes, such as the manipulable object, an obstacle, or an empty bounding box. To formalize this, we first define an index set I for each representation of the object in an environment as $I = \{1, 2, ..., K\}$. Our first step is to identify a subset $I_{active} \subset I$ that represents all nonempty bounding boxes. Within I_{active}, the second step is to identify the index of the manipulable object I_m and the indexes for obstacles $I_{obstacle}$.

To obtain these indexes, we need to sample two different types of rollout before training. In the first type, the agent does not perform any action, which means that the variation of one bounding box's position only comes from the dynamics of the environment, so that we can get I_{active}. In the second type, the agent is controlled to perform some random actions on the manipulable object. Therefore, we expect that, in the second type, the position of the manipulatable object varies more than that in the first type. Then, we can calculate the errors between the mean positions in both types and identify the index that corresponds to the manipulable object I_m.

From the sampled rollouts, we feed the image observations to the BboxEncoder. For the first type of rollout, we can get two sets $A^p = \{(z_1^p, ..., z_K^p)_j\}_{j=1}^J$ and $A^{pres} = \{(z_1^{pres}, ..., z_K^{pres})_j\}_{j=1}^J$, which correspond to the sets of the position and presence. For the second type, we can get one set $B^p = \{(z_1^p, ..., z_K^p)_j\}_{j=1}^J$. J is the total number of sampled images. With A^p, A^{pres}, and B^p, we can calculate the index I_m and $I_{obstacle}$ by following Algorithm 1. With these indexes, we can extract the position and dimension of each bounding box from the environment, namely, b_m and $b_i, i \in I_{obstacle}$.

Algorithm 1 Object Index Inference

Given:
- Set of position variables A^p and presence variables A^{pres} from rollouts with no movement from the agent
- Set of positions variables B^p from rollouts with random movement by the agent
- $I_{active} = \emptyset$ and $I_{obstacle} = \emptyset$ ▷ empty set: $\emptyset$

μ^{pres} = mean(A^{pres}) ▷ calculate mean per slot
for $i = 1, K$ **do**
  if $\mu_i^{pres} > \delta$ **then** ▷ e.g. δ=0.8
    $I_{active} \leftarrow I_{active} \cup \{i\}$
$\mu^{A,p}$ = mean(A^p)
$\mu^{B,p}$ = mean(B^p)
$d_i = (\|\mu_i^{A,pos} - \mu_i^{B,pos}\|_2), i \in I_{active}$
$m = \arg\max_{i \in I_{active}}(d_i)$
$I_{obstacle} \leftarrow I_{active} \backslash \{m\}$
$I_m = \{m\}$
Returns: $I_{obstacle}$, I_m

2) Distance Graph: In order to select intermediate goals that are reachable for the agent to reach, we need to create an obstacle-free graph as a representation of the environment where the space occupied by obstacles are removed. Detailed explanation about building the obstacle-free graph can be found in [3]. The space covered by the bounding box of each obstacle is considered as an obstacle. Therefore, we need to calculate the dimensions of the bounding box of each obstacle. In contrast to G-HGG that defines $\mathcal{G}_A$ using the localization information of an obstacle from the environment, we obtain this information from the proposed BboxEncoder. For the same reason, G-HGG is only applicable to stationary environments, while our method can be used in dynamic environments. Specifically, we first estimate an obstacle space $\mathcal{G}_{obstacle}$, so that $\forall g \in$

$\mathcal{G}_{obstacle} \implies g \notin \mathcal{G}_A$. Since the obstacles are constantly moving, $\mathcal{G}_{obstacle}$ also changes at every timestep. An ideal option would be to estimate a distance graph at each timestep, but creating such a graph is not practical due to the computation burden and the memory constraint. In this paper, we only consider a region that is constantly occupied by an obstacle as $\mathcal{G}_{obstacle}$. Then, we can locate such a region $\mathcal{G}_i$ by calculating the position boundaries $x_{i,min}, x_{i,max}, y_{i,min}, y_{i,max}$ of the i^{th} obstacle. Finally, we can get the overall $\mathcal{G}_{obstacle}$ as

$$\mathcal{G}_i := \{(x,y) \in \mathcal{G} | x_{i,min} \leq x \leq x_{i,max}, y_{i,min} \leq y \leq y_{i,max}\} \tag{11}$$

$$\mathcal{G}_{obstacle} := \bigcup_{i \in I_{obstacle}} \mathcal{G}_i \tag{12}$$

To estimate $x_{i,min}, x_{i,max}, y_{i,min}, y_{i,max}$, we consider the bounding box for i^{th} obstacle with its width w_i, height h_i and coordinates of the center that are collected during rollouts, namely, $X = \{x_{i,1}, x_{i,2}, ...\}$ and $Y = \{y_{i,1}, y_{i,2}, ...\}$. We calculate $x_{i,min} = max(X) - w_i, x_{i,max} = min(X) + w_i, y_{i,min} = max(Y) - h_i, y_{i,max} = min(Y) + h_i$. Here we estimate the region $\mathcal{G}_i$ that is covered by the obstacle through the whole episode. If $x_{i,min} > x_{i,max}$ or $y_{i,min} > y_{i,max}$, we set $\mathcal{G}_i = \emptyset$, since there is no region that the obstacle will always occupy.

D. Extension of States and Sparse Reward

In this subsection, we first present the extension of the observation state with the information of the bounding boxes. Second, we provide a multi-objective sparse reward so that the agent can learn better obstacle-avoiding behavior.

1) State Extension: To properly evade dynamic obstacles, the agent needs to observe the movement of obstacles at every timestep, and these observations can be obtained using our BboxEncoder. In this work, we extend the state representation with additional information as follows:

- positions of the manipulable object: (x_m^t, y_m^t)
- positions of i^{th} obstacle: (x_i^t, y_i^t)
- velocity $v_i^t = (v_{x_i}^t, v_{y_i}^t)$ of i^{th} obstacle:

$$v_{x_i}^t = \frac{x_i^t - x_i^{t-1}}{\Delta t},\ v_{y_i}^t = \frac{y_i^t - y_i^{t-1}}{\Delta t}$$

- coordinate angle between i^{th} obstacle and the manipulable object:

$$\alpha_i^t = \tan^{-1}(\frac{y_i^t - y_m^t}{x_i^t - x_m^t})$$

- minimal distance d_i^t between i^{th} obstacle and the object:

$$e_1 = |x_m^t - x_i^t| - w_m^t - w_i^t,\ e_2 = |y_m^t - y_i^t| - h_m^t - h_i^t$$

$$l = (max(0, e_1), max(0, e_2))$$

$$d_i^t = \|l\|_2$$

- distances $c_i^t = (c_{i,1}^t, c_{i,2}^t, c_{i,3}^t, c_{i,4}^t)$ to corners of i^{th} obstacle from the center of manipulable object:

$$c_{i,1}^t = \|(x_i^t - w_i^t - x_m^t, y_i^t - h_i^t - y_m^t)\|_2$$

$$c_{i,2}^t = \|(x_i^t + w_i^t - x_m^t, y_i^t - h_i^t - y_m^t)\|_2$$

$$c_{i,3}^t = \|(x_i^t - w_i^t - x_m^t, y_i^t + h_i^t - y_m^t)\|_2$$

$$c_{i,4}^t = \|(x_i^t + w_i^t - x_m^t, y_i^t + h_i^t - y_m^t)\|_2$$

. Finally, the extended state $\bar{s}_t$ is described as

$$\bar{s}_t = (\underbrace{s_t}_{\text{original state}}, \underbrace{x_m^t, y_m^t, x_1^t, y_1^t, ..., \underbrace{x_i^t, y_i^t, v_i^t, d_i^t, c_i^t \alpha_i^t}_{\text{measures for } i^{th} \text{ obstacle}}, ...}_{\text{measures for all obstacles}}). \tag{13}$$

The original state $s_t = S_{int}$ is described in Section IV.

2) Sparse Reward Modification: By observing the experiment, we notice that the agent can reach the target goal even after colliding with the obstacles in the process of completing the task, which should be punished since we expect the agent to finish the task in a collision-free manner. Prior work tackles this problem by designing a dense reward that is related to some measurements to the obstacle to achieve collision-free movement. However, designing an adequate task-tailored reward is challenging and time-consuming.

To balance the simplicity of using sparse rewards and the expected task-driven behavior, we propose a multi-objective sparse reward, which is a conditioned binary reward function with different magnitudes for each objective. Specifically, the multi-objective sparse reward is defined as

$$r_g(s) := \begin{cases} \eta, & \text{if collision} \\ 0, & \text{if } f_g(s) = 1 \\ -1, & \text{otherwise.} \end{cases} \tag{14}$$

The additional reward η is a constant value that is designed to punish the collision, which can be viewed as a hyperparameter. When the agent collides with any obstacle, it will be given a reward $\eta < -1$. While the agent fails to reach the goal without colliding with any obstacle, it will be given a reward -1. Only when the agent reaches the goal successfully, will it be rewarded with 0. Experiment results show that the smaller η is, the easier it is for the agent to avoid the obstacle. However, if η is too small, the agent may not learn anything, since it avoids proximity to the region where the obstacle moves, even if it is necessary to reach the goal. Examples and detailed illustrations are given in Section VI-C. The collision condition is triggered when $d_i^t \leq 0$.

E. Algorithm Overview

The overall Bbox-HGG algorithm is provided as Algorithm 2. The BboxEncoder is pre-trained with the image dataset and MONet. Line 1 implements the object index inference algorithm presented in Section V-C1. From line 3 to 10 G-HGG is implemented with the modifications presented in Section V-C2. Lines 15 and 18 implement the modification of the state representation explained in Section V-D1. The code of Bbox-HGG is available at here[1].

VI. Experiments

In our experiments, we compare the performance of Bbox-HGG against G-HGG and HGG on four challenging tasks.

[1]https://github.com/erick-alv/HGG-extended-master

Algorithm 2 BBOX-HGG

Given:
- Pretrained BboxEncoder: Ψ ▷ See Section V-B
- Policy and value networks: θ, ϕ

Sample transitions and encode them with Ψ to infer objects indices $I_{obstacle}$ and m ▷ See Algorithm 1
Sample new set of transitions to estimate $\mathcal{G}_{obstacle}$ and modify $\mathcal{G}_A$ ▷ modified G-HGG
Construct graph G=(V, E) and precompute distance d_G ▷ G-HGG
Initialise θ and ϕ
Initialise replay buffer $\mathcal{D}$
for iteration **do**
    Sample targets tasks $\{(\hat{s}_0^i, \hat{g}^i\}_{i=1}^M \sim \mathcal{T}^*$ and render goal images $\{g_{img}^i\}_{i=1}^M$
    Encode images Ψ, get coordinates with algorithm 1 and index m: $\{(z_m^p)^i\}_{i=1}^M$
    Find trajectories and goals g^i by optimizing (3); using d_G and coordinates obtained with algorithm 1 ▷ modified G-HGG
    for episode **do**
        $(s_0, g) \leftarrow (\hat{s}_0^i, g^i)$
        for t = 0, ..., T-1 **do**
            Get states s_t and image im_t
            Extend s_t to $\bar{s}_t$ ▷ See Section V-D
            $a_t \leftarrow \pi_\theta(\bar{s}_t \| g) + \mathcal{N}_t$
            Perform a_t, get s_{t+1} and render im_{t+1}
            Extend s_{t+1} to $\bar{s}_{t+1}$
            Store $(\bar{s}_t \| g, a_t, r_g(\bar{s}_{t+1}), \bar{s}_{t+1} \| g)$ in D
    for t=1, ..., N **do**
        Sample a minibatch B from $\mathcal{D}$ using HER
        Optimize θ and ϕ using DDPG with B

Moreover, to provide a ground truth, we also test our algorithm by replacing the bounding box information obtained from the BboxEncoder with the real bounding box information retrieved from the simulation.

A. Environments

To demonstrate the effectiveness and advantages of Bbox-HGG, we create four new experimental environments on top of the well-used benchmark environments developed by [19]. All our tasks are simulated in MuJoCo [28], in which a Fetch robot with a gripper is controlled to push a puck through environments with dynamic obstacles. These tasks share the following characteristics:

- The agent receives a state containing the joint positions and velocities of the robotic arm. This information is directly retrieved from the simulation. This state is additionally extended as described in Section V-D1.
- The objects in the environment are located on an 0.5 m square table. The images used for the BboxEncoder are captured from a camera located 2.1 m above the table. The camera's field of view is 15°.
- The robot is controlled by a three-dimensional vector describing the end effector's position. There is no control over the gripper since we only have pushing tasks.
- The accessible goal space $\mathcal{G}_A$ is defined by a 2D region on the table.

*1) **FetchMovingObstacle**:* (Figure 7a): In this environment, the robot must push the manipulatable object (red puck) from its initial position to the goal position (sample from the green region). The start position of the object is selected randomly inside the shaded red region. This environment only contains one obstacle (blue cube) that moves in the shaded-blue region along the x-axis. At the beginning of each episode, the obstacle's velocity is randomly sampled from an array, which begins at 1 m/s and ends at 1.5 m/s with an interval of 0.05 m/s.

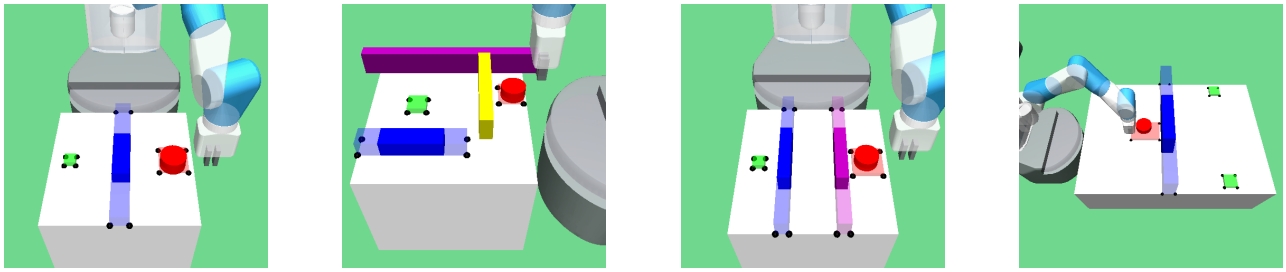
*2) **FetchMovingCom**:* (Figure 7b): The task in this environment is the same as FetchMovingObstacle and the respective areas are marked with the same color. In addition to one moving obstacle, there are another two static obstacles in this environment. The robot has to push the puck to pass through the free space between the yellow obstacle and the moving obstacle, which is not big enough for the puck to pass all the while and the agent must wait for the right moment when the space is big enough.

*3) **FetchDoubleObstacle**:* (Figure 7c): In this environment, the goal is to push the puck to a goal position passing through two dynamic obstacles that move in opposite directions to each other. The velocity of each obstacle is independently sampled from an array, which begins at 0.6 m/s and ends at 1.1 m/s with an interval of 0.0167 m/s.

*4) **FetchSlideObstacle**:* (Figure 7d): In this environment, the robot must slide the puck to two target goal regions. The simulation samples a goal randomly from one of these two regions at the beginning of each episode. The obstacle moves along the y-axis obstructing the puck from reaching the goal and forcing the agent to act at the right moment when it is not blocked. Detailed descriptions of these four environments are introduced in Appendix A-B.

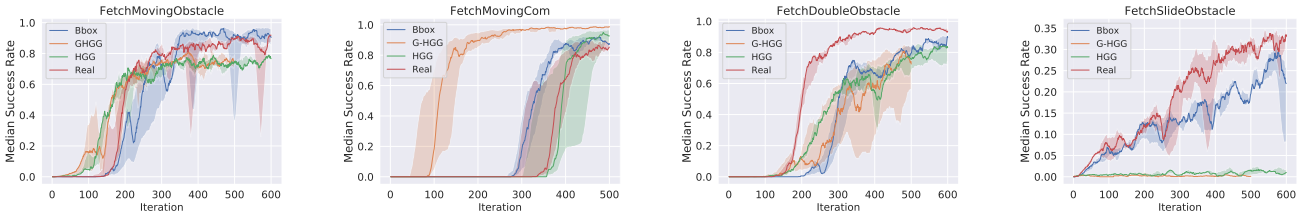
B. Results

We tested Bbox-HGG on the four environments to compare its performance with G-HGG, HGG, and the ground truth from two sets of results, namely, the training sample efficiency and the testing success rate of the best policy. It should be noted that HER is not examined in this paper, since G-HGG has demonstrated that HER can not solve manipulation tasks with obstacles [3]. To examine the training sample efficiency (Figure 8), we only compare the median success rate against the training iteration without considering no collision as a task completion condition, since we expect G-HGG or HGG would fail to finish the task without receiving any information about the dynamic obstacles. The testing success rate (Figure 9) is calculated by averaging the performance of the best policy from each algorithm in 100 episodes. It should be noted that the ground truth results (labeled as "Real") are generated by running Bbox-HGG with the true information about the bounding boxes from the environment instead of estimating them via



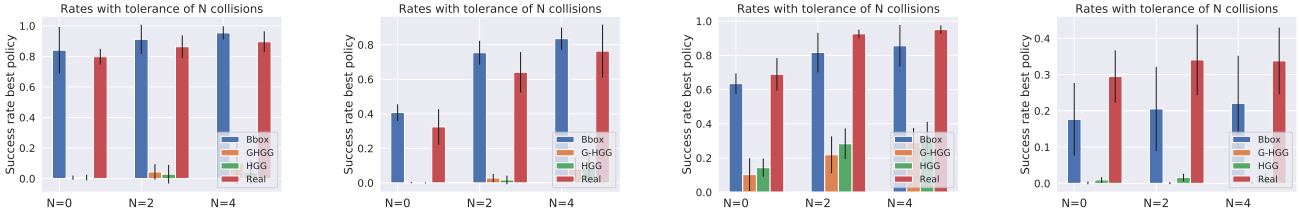
(a) FetchMovingObstacle (b) FetchMovingCom (c) FetchDoubleObstacle (d) FetchSlideObstacle

Fig. 7: Robotic manipulation environments.



(a) FetchMovingObstacle (b) FetchMovingCom (c) FetchDoubleObstacle (d) FetchSlideObstacle

Fig. 8: Learning progress of the median training success rate (line, **without considering any collision behavior**) and interquartile range (shaded) of Bbox-HGG, G-HGG, HGG, and the ground truth of Bbox-HGG (Real).



(a) FetchMovingObstacle (b) FetchMovingCom (c) FetchDoubleObstacle (d) FetchSlideObstacle

Fig. 9: Success rates of the collision avoidance testing of Bbox-HGG, G-HGG, HGG, and the ground truth of Bbox-HGG.

the BboxEncoder. These tests have a tolerance parameter $N \in \{0, 2, 4\}$ for the number of collisions that can be allowed per episode. If the number of collisions surpasses N, then the episode is terminated as a failure. The most remarkable results can be observed, in all four environments, Bbox-HGG can learn obstacle-avoiding behavior with a similar success rate across different numbers of tolerance parameters, while the other algorithms are not able to solve any of the tasks.

In the FetchMovingObstacle environment, as shown in Figure 9a, while G-HGG and HGG display an almost zero success rate with different collision tolerances, Bbox-HGG reaches a minimum average success rate of 80%, increasing to over 90% when $N = 2$. The error bars also indicate that Bbox-HGG has a robust performance and gets more accurate when the tolerance increases. As shown in Figure 8a, Bbox-HGG also exhibits the best training success rate and yet has as competitive a sample efficiency as HGG. Since the obstacle moves along the whole table and leads to $\mathcal{G}_{obstacle} = \emptyset$, our algorithm is still able to learn obstacle-avoiding behavior with the help of our extended state and multi-objective sparse reward, which makes the agent more cautious when passing the area of an obstacle. Surprisingly, we find that our Bbox-HGG achieves even better performance than the ground truth that uses the real information about the bounding boxes. We believe that, since the coordinates obtained from the Bbox-Encoder are not rigorously accurate, the agent learns to keep some security distances around the obstacle to guarantee its success.

In the FetchMovingCom environment, as shown in Figure 9b, Bbox-HGG achieves a success rate of 40% with zero tolerance of collision and increases to around 80% with $N = 2$, since the safe space is very tiny for the object to pass through. However, HGG or G-HGG still achieves almost zero success rates across different tolerances. As shown in Figure 8b, G-HGG has better sample efficiency than Bbox-HGG without considering the collision. This is because the obstacles are not challenging enough to prevent G-HGG finishing the task without considering any collision. On the other hand, Bbox-HGG is more cautious when passing through the obstacles. This result is also demonstrated by Figure 11a, where G-HGG is equivalent to G-HGG when the collision sparse reward $\eta = -1$. Similar to the results of FetchMovingObstacle, Bbox-HGG still exhibits slightly better performance than the ground truth approach in terms of success rate and sample efficiency.

The FetchDoubleObstacle is an environment extended on

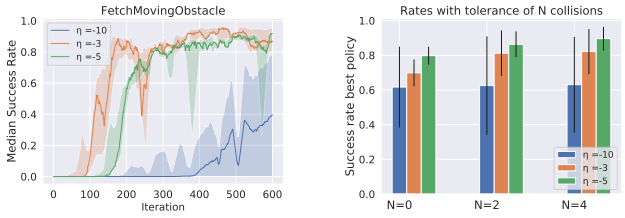
the basis of FetchMovingObstacle, in which two obstacles are moving in opposite directions with randomly sampled velocities. As shown in Figure 9c, Bbox-HGG is still able to solve the task with a success rate over 80% with a small tolerance ($N = 2$), which outperforms G-HGG and HGG. As shown in Figure 8c, Bbox-HGG exhibits slightly better sample efficiency than HGG and eventually peaks at around 90%. The ground truth still shows better performance than Bbox-HGG, since the narrow gap between the two obstacles requires more accurate representation information about the environment.

The FetchSlideObstacle task is challenging since the two target goal spaces are separated from each other, which makes it a multi-task RL environment that can not be perfectly solved by HGG or G-HGG even without the obstacle. Nevertheless, Bbox-HGG still yields better results than the other algorithms in terms of success rate and is clearly more sample efficient. As shown in Figure 9d, Bbox-HGG can achieve a success rate of 30%, while G-HGG and HGG display no success rates. As shown in Figure 8d, HGG suffers from a very bad sample efficiency since the object is very close to the obstacle and Bbox-HGG gradually increases its success rate during training. In this environment, the ground truth approach shows better performance since the object is too close to the obstacle, which requires accurate information to move forward.

C. Ablation Study

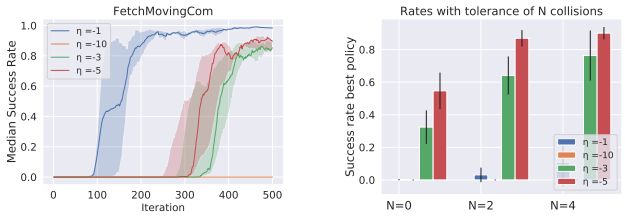
We first provide an ablation study on the collision reward η, which is manually designed in the multi-objective sparse reward as defined in (14). Figures 10 and 11 illustrate the training success rates and testing success rates with different collision tolerance parameters for Bbox-HGG in FetchMovingObstacle and FetchMovingCom. In line with our main results, Bbox-HGG can learn outstanding obstacle-avoiding skills in both scenarios with minimum effort on designing the multi-objective sparse reward. In both scenarios, we can find that a small η is helpful to achieve better testing success rates with different tolerance parameters (Figure 10b and Figure 11b), for example, $\eta = -5$ has better performance than $\eta = -3$. However, a low value $\eta = -10$ leads to worse success rates (FetchMovingObstacle) or even failure to solve the task at all (FetchMovingCom), since the excessive collision penalty makes the agent unable to move to target goals without being worried about colliding with any obstacles. For the same reason, we can observe that the sample efficiency increases with η consistently (Figure 10a and Figure 11a), where $\eta = -1$ exhibits the best sample efficiency while $\eta = -10$ yields the worst sample efficiency. Despite Bbox-HGG being generally robust to changes in η, we recommend rough parameter tuning on the collision reward for each scenario.

We second provide an ablation study on Bbox-HGG with and without the extension of states and sparse-reward. Due to the page limit, we choose the FetchDoubleObstacle environment as the example to show the performance. Figure 12 shows the training success rates and testing success rates with different configurations, namely, the full Bbox-HGG, Bbox-HGG without state extension, and Bbox-HGG without reward extension. Bbox-HGG achieves over 90% success rate during training



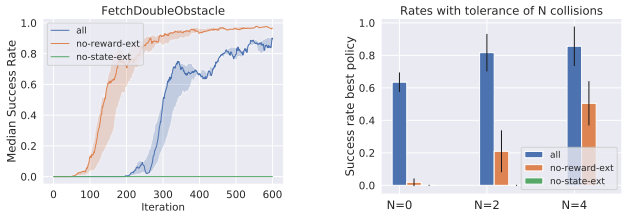
(a) Training success rate. (b) Testing success rate.

Fig. 10: Ablation study of η in FetchMovingObstcle.



(a) Training success rate. (b) Testing success rate.

Fig. 11: Ablation study of η in FetchMovingCom.



(a) Training success rate. (b) Testing success rate.

Fig. 12: Ablation study on Bbox-HGG without state extension or reward extension in FetchDoubleObstacle.

and 60% success rate during testing with a tolerance parameter $N = 0$. However, Bbox-HGG without state modification can not solve the task at all since it lacks the information of the obstacles. Bbox-HGG without reward extension is similar to G-HGG since it only receives a simple sparse reward. This configuration achieves better sample efficiency than Bbox-HGG during training and very bad testing success rate when considering the collision behavior. As explained before, the better sample efficiency achieved by Bbox-HGG without reward extension is because it does not consider the collision behavior, which can be demonstrated by the testing success rate. In Figure 12b, the success rate of the Bbox-HGG without reward extension drops significantly, while the full Bbox-HGG is able to perform well. In brief, the results demonstrate the state and reward extensions are important for Bbox-HGG.

D. Discussions and Limitations

Although we have extensively evaluated Bbox-HGG in simulations, our method is still applicable in a real-world setup. For instance, to generate the image dataset automatically in the real world, a camera can be mounted on the ceiling to capture images of the environment, where the robotic arm can be controlled to move the objects randomly for different

setups. This has been proven to be applicable by many works, such as [13], [22], [33]. To ensure a successful sim-real policy transfer, we can first increase the simulation fidelity and use domain randomization technology to improve the robustness of the policy learned in the simulation. Some similar ideas can be found in [24], [18], [26], [12]. There are a few limitations of the proposed method that can be studied for future work. For example, we only consider 2D information about the obstacles that the agent can move around, while there are 3D obstacles in the real world. Therefore, we also disable the gripper control of the agent, which can be further utilized to avoid obstacles.

VII. Conclusion

This work introduces a novel automatic hindsight goal generation algorithm, Bbox-HGG, which is an extension of HGG and G-HGG for challenging manipulation tasks in environments with both static and dynamic obstacles. Specifically, we first propose a BboxEncoder that can estimate the bounding box information for each object in an environment. According to the bounding box information, we second design an automatic way to create a valid graph for computing an obstacle-free graph, extend the observation state, and modify a multi-objective sparse reward. Experiments on four different challenging object manipulation tasks demonstrate the superior performance of Bbox-HGG over HGG and G-HGG in terms of both learning efficiency and maximum success rate. For future work, we aim to improve, extend, and deploy Bbox-HGG in real-world applications. For instance, first, it would be an important advance to bridge the gap between theory and practice by deploying a policy learned with Bbox-HGG to a physical robot. Second, we are positive that Bbox-HGG could as well be applied to more diverse tasks with even better object recognition solutions. Last, it would be very interesting to investigate how to solve similar tasks only using one-dimensional sparse rewards.

Appendix A
Experiment Settings

A. Dataset Generation

To create the images, an environment is set up in the following manner. The table is created with a box with center at $(1.3, 0.75, 0.2)^{\mathrm{T}}$ and dimensions $(0.5, 0.5, 0.4)$ for width, length, and height. The camera is positioned at $(1.3, 0.75, 2.5)^{\mathrm{T}}$, is pointing at $(1.3, 0.75, 0.2)^{\mathrm{T}}$, and has $15°$ as the field of view. When rendering an image, the table is set invisible. The number of instantiated objects is limited by $n_{max} = 4$. The number of rectangles n_{rect} in an image is selected from $\{0, 1, 2, 3\}$ with probabilities $(0.3, 0.2, 0.3, 0.2)$, the number of cylinders n_{cyl} from $\{0, ..., n_{max} - n_{rect}\}$ and the number of cubes n_{cub} from $\{0, ..., n_{max} - n_{rect} - n_{cyl}\}$ uniformly. Each of the objects is created with the factors listed in Table I. Values are selected uniformly from the given range. In addition, each object has a color from the set {red, green, purple, yellow}.

B. Environment Settings

For our experiments, the agent was trained during E epochs and C=20 cycles. Each iteration comprised $M = 50$ episodes of T time steps.

TABLE I: Factors of variation of objects dataset

Object type	Factor of variation	Range
Rectangle	Height	$\{0.02, ..., 0.035\}$
	Width	$\{0.02, ..., 0.04\}$
	Length	$\{0.08, ..., 0.25\}$
	Rotation z axis	$\{0°, 90°\}$
Cylinder	Height	$\{0.02, ..., 0.035\}$
	Radius	$\{0.035, ..., 0.08\}$
	Rotation x axis	$\{0°, ..., 180°\}$
	Rotation y axis	$\{0°, ..., 180°\}$
Cube	Length	$\{0.02, ..., 0.035\}$
	Rotation x axis	$\{0°, ..., 90°\}$
	Rotation y axis	$\{0°, ..., 90°\}$
	Rotation z axis	$\{0°, ..., 90°\}$

TABLE II: Parameters for environments

Environment	E	T
FetchMovingObstacle	30	100
FetchMovingCom	25	100
FetchDoubleObstacle	30	100
FetchSlideObstacle	30	50

In addition, for the proximity tolerance to a goal, we select $\delta_g = 0.05$. The regions to sample target goals and start position are in Table III. The configuration of the obstacles is listed in Table IV. The fourth column shows the axis where the obstacle moves, and the lower and upper limit in this axis. The fifth column indicates how many equally distant samples are taken from the velocity range; these are then used for the random selection of a speed at the beginning of an episode.

TABLE III: Start and goal regions

Environment	Region	Range x-axis	Range y-axis
FetchMoving-Obstacle	Target goal	$(1.28, 1.32)$	$(0.55, 0.59)$
	Start position	$(1.28, 1.32)$	$(0.91, 0.95)$
FetchMoving-Com	Target goal	$(1.37, 1.43)$	$(0.67, 0.73)$
	Start position	$(1.08, 1.12)$	$(0.63, 0.67)$
FetchDouble-Obstacle	Target goal	$(1.28, 1.32)$	$(0.55, 0.58)$
	Start position	$(1.28, 1.32)$	$(0.92, 0.96)$
FetchSlide-Obstacle	Target goal 1	$(1.57, 1.73)$	$(1.07, 1.17)$
	Target goal 2	$(1.57, 1.73)$	$(0.33, 0.43)$
	Start position	$(0.7, 0.8)$	$(0.09, 1.12)$

TABLE IV: Obstacles configuration

Environment	Obstacle	Dimensions (x, y, z)	Movement: axis limits	Velocity: n-samples range
FetchMoving-Obstacle	Blue obstacle	(0.09, 0.03, 0.025)	x-axis 1.05; 1.55	10 (1, 1.5)
FetchMoving-Com	Blue obstacle	(0.09, 0.03, 0.025)	x-axis 1.24; 1.55	20 (0.3, 0.5)
	Purple obstacle	(0.3, 0.02, 0.04)	none	none
	Yellow obstacle	(0.02, 0.15, 0.04)	none	none
FetchDouble-Obstacle	Blue obstacle	(0.1, 0.02 0.035)	x-axis 1.05; 1.55	30 (0.6, 1.1)
	Purple obstacle	(0.1, 0.02 0.035)	x-axis 1.05; 1.55	30 (0.6, 1.1)
FetchSlide-Obstacle	Blue obstacle	(0.04, 0.17, 0.1)	y-axis 0.31; 1.19	30 (0.9, 1.2)

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] Péter Almási, Róbert Moni, and Bálint Gyires-Tóth. Robust reinforcement learning-based autonomous driving agent for simulation and real world. *arXiv preprint arXiv:2009.11212*, 2020.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[3] Zhenshan Bing, Matthias Brucker, Fabrice O. Morin, Rui Li, Xiaojie Su, Kai Huang, and Alois Knoll. Complex robotic manipulation via graph-based hindsight goal generation. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021.

[4] Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. Learning visuomotor policies for aerial navigation using cross-modal representations. In *In submission*, March 2020.

[5] Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *CoRR*, abs/1901.11390, 2019.

[6] Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[8] Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *arXiv preprint arXiv:2002.11089*, 2020.

[9] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. 2019.

[10] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019.

[11] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[12] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[13] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[14] Alexander Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. *CoRR*, abs/2001.02407, 2020.

[16] C. Nash, S. Eslami, C. Burgess, I. Higgins, Daniel Zoran, T. Weber, and P. Battaglia. The multi-entity variational autoencoder. 2018.

[17] Soroush Nasiriany, Vitchyr H Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. *arXiv preprint arXiv:1911.08453*, 2019.

[18] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[19] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[20] 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.

[21] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

[22] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

[23] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation, 2019.

[24] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pages 262–270. PMLR, 2017.

[25] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[26] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[27] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. Learning robotic assembly from cad. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3524–3531. IEEE, 2018.

[28] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[29] Nikola Vulin, Sammy Christen, Stefan Stevšić, and Otmar Hilliges. Improved learning of robot manipulation tasks via tactile intrinsic motivation. *IEEE Robotics and Automation Letters*, 6(2):2194–2201, 2021.

[30] Saiwei Wang, Xin Jin, Shuai Mao, Athanasios V. Vasilakos, and Yang Tang. Model-free event-triggered optimal consensus control of multiple euler-lagrange systems via reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 8(1):246–258, 2021.

[31] Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

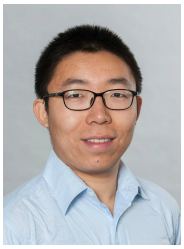
[32] Yanchao Yang, Yutong Chen, and Stefano Soatto. Learning to manipulate individual objects in an image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2020.

[33] Jingwei Zhang, Lei Tai, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4(2):1148–1155, 2019.

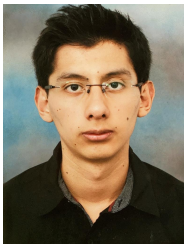
[34] Chaoqiang Zhao, Gary G. Yen, Qiyu Sun, Chongzhen Zhang, and Yang Tang. Masked gan for unsupervised depth and pose prediction with scale consistency. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2020.

[35] Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual information state intrinsic control. *arXiv preprint arXiv:2103.08107*, 2021.

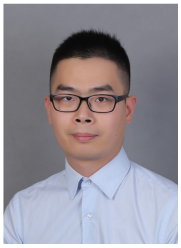
[36] Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, pages 113–122. PMLR, 2018.



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, Germany. His research investigates bio-robots which are controlled by artificial neural networks and related applications.

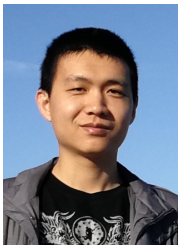


Erick Álvarez received his B. Sc. Degree in Games 2Engineering at Technical University of Munich, Germany, in 2021. His research interest is artificial intelligence on robotics implementations, especially with reinforcement learning algorithms.



Long Cheng joined Wenzhou University as an assistant professor in 2021. He received his doctorate degree in Computer Science from the Technical University of Munich, Germany, in 2018. He received the B.S. degree in mechanical design manufacturing and automation and theM.Eng. degree in mechanical engineering from the Harbin Institute of Technology, Harbin, China, in 2011 and 2013, respectively. His research interests include snake-like robots, artificial neural networks, and real-time systems.

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. There, 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.

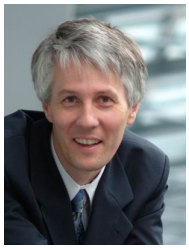


Rui Li received the B.Eng degree in automation engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2013 and the Ph.D. degree from the Institute of Automation, Chinese Academy of Science (CASIA), Beijing, China in 2018, respectively. He visited Informatics 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 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). In this capacity, he was actively involved in developing the concept of the EU's FET Flagship projects. His research interests include cognitive, medical and sensor-based 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.