








Explicit-Implicit Subgoal Planning for Long-Horizon Tasks with Sparse Reward

Fangyuan Wang , Anqing Duan , Peng Zhou , Shengzeng Huo , Guodong Guo , *Senior Member, IEEE*,
Chenguang Yang , *Fellow, IEEE*, and David Navarro-Alarcon , *Senior Member, IEEE*

Abstract—The challenges inherent in long-horizon tasks in robotics persist due to the typical inefficient exploration and sparse rewards in traditional reinforcement learning approaches. To address these challenges, we have developed a novel algorithm, termed Explicit-Implicit Subgoal Planning (EISP), designed to tackle long-horizon tasks through a divide-and-conquer approach. We utilize two primary criteria, feasibility and optimality, to ensure the quality of the generated subgoals. EISP consists of three components: a hybrid subgoal generator, a hindsight sampler, and a value selector. The hybrid subgoal generator uses an explicit model to infer subgoals and an implicit model to predict the final goal, inspired by way of human thinking that infers subgoals by using the current state and final goal as well as reason about the final goal conditioned on the current state and given subgoals. Additionally, the hindsight sampler selects valid subgoals from an offline dataset to enhance the feasibility of the generated subgoals. While the value selector utilizes the value function in reinforcement learning to filter the optimal subgoals from subgoal candidates. To validate our method, we conduct four long-horizon tasks in both simulation and the real world. The obtained quantitative and qualitative data indicate that our approach achieves promising performance compared to other baseline methods. These experimental results can be seen on the website <https://sites.google.com/view/vaesi>.

Note to Practitioners—This paper addresses the persistent challenges in executing long-horizon tasks in robotics, which are hindered by inefficient exploration and sparse rewards in traditional reinforcement learning methods. EISP overcomes these obstacles through the integration of a hybrid subgoal generator, a hindsight sampler, and a value selector, ensuring that the generated subgoals are both feasible and optimal, thereby facilitating the completion of the entire task by addressing each subgoal. Although preliminary results are promising, several challenges must be resolved in future research. Firstly, the automatic determination of the number of subgoals during training presents a challenge that may impact the low-level policy's efficacy in achieving each subgoal. Secondly, the proposed method relies on an offline dataset that lacks successful trajectories from the initial state to the final state, complicating the exploration of subgoals towards the final goal. Finally, the scalability across multiple tasks is relatively limited due to the time-intensive nature

of data collection for each task. Future research will focus on optimizing the allocation of timesteps to subgoals, enhancing exploration efficiency, and improving scalability across a broader range of tasks without extensive data collection.

Index Terms—Motion control; Learning control systems; Manipulator motion-planning; Motion-planning.

I. INTRODUCTION

HUMAN daily activities often involve performing long-horizon tasks, which are characterized by hierarchical structures and multiple distinct types of actions [1], [2]. These tasks require reasoning about a sequence of actions and their associated control signals over a long period of time [3]–[5], necessitating the successful execution of all subtasks [6]. Recent advances in robotic planning have demonstrated the effectiveness of robots in performing complex tasks through imitation and reinforcement learning [6]–[8]. However, those methods are typically limited to completing either short-term tasks with intensive rewards or long-horizon tasks guided by a series of instructions provided by humans [9]. Empowering robots with the ability to autonomously reason and plan long-horizon tasks could assist or even replace humans in dangerous or tedious tasks, further expanding the frontiers of robotic automation.

Current efforts to tackle long-horizon tasks involve imitating expert trajectories [4], [10], [11], increasing goal-oriented exploration [5], [12]–[14] and employing divide-and-conquer tactics [2], [3], [15], [16]. Imitation learning extracts key points from expert demonstrations to generate effective strategies for subgoal creation. However, this approach often suffers from the time-consuming nature of demonstration collection and the sub-optimality of expert strategy. While enhancing the exploration efficiency of robots is essential for the discovery of effective and feasible subgoals, goal exploration is generally confined to neighboring states of the explored region. Given the conspicuous lack of experience with rarely explored state spaces, a significant gap remains in the efficiency requisite for more generalized reinforcement learning applications [17]. Most existing methods [3], [15] that decompose complex tasks into simpler ones, cannot simultaneously guarantee the feasibility and optimality of the generated subgoal sequence and frequently operate under the strong assumption that the state space and the goal space are identical. Thus, developing a robot capable of solving an extensive range of combinatorial decision problems continues to pose a long-standing challenge.

To alleviate these challenges, we propose Explicit-Implicit Subgoal Planning (EISP) that solves the long horizon problem

This work is supported in part by the Research Grants Council (RGC) of Hong Kong under C4042-23GF, and in part by the PolyU-EIT Collaborative PhD Training Programme under application number 220724983. *Corresponding author: David Navarro-Alarcon.*

F. Wang, A. Duan, S. Huo and D. Navarro-Alarcon are with the Dept. of Mechanical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong. (e-mail: fangyuan.wang@connect.polyu.hk, anqing.duan@polyu.edu.hk, kyle-sz.huo@connect.polyu.hk, dna@ieee.org)

P. Zhou is with the Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong. (e-mail: jeffzhou@hku.hk)

F. Wang, G. Guo are with the Ningbo Institute of Digital Twin, Eastern Institute of Technology, China. (e-mail: fywang@eitech.edu.cn, gduo@eitech.edu.cn)

C. Yang is with the Department of Computer Science, University of Liverpool, Liverpool, UK. (e-mail: Chenguang.Yang@liverpool.ac.uk)

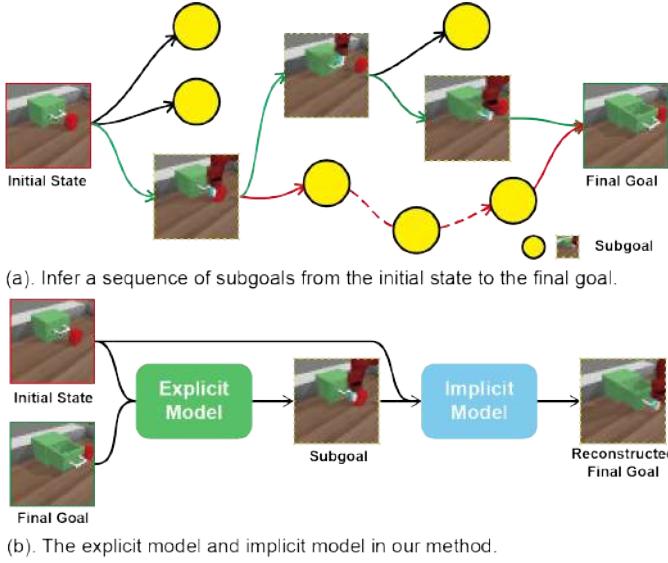


Fig. 1. (a) EISP infers multiple subgoals at different task stages. Green and red lines represent possible subgoal sequences. (b) We leverage the explicit model to infer subgoals and the implicit model to predict the final goal.

by inferring easy-to-achieve subgoals and then accomplishing each subgoal sequentially. Fig.1 (a) illustrates the potential subgoals that can be generated at different stages from the initial state to the final goal. The subgoal sequences connected by the green line and the red line represent possible approaches to accomplishing the long-horizon task. Because the subgoal sequence linked by the red line may require more time to complete the task, we consider the subgoal sequence connected by the green line is more optimal. Our study mainly focuses on the subgoal generation process (without delving into the underlying subgoal approach process). Inspired by the human approach to solving long-horizon tasks by breaking them down into smaller and manageable components, it is intuitive to employ an explicit model that takes as input the current state and final goal and outputs the subgoals, as demonstrated in previous studies [3], [15]. However, a critical aspect often overlooked is humans' ability to predict long-term outcomes by identifying short-term subgoals based on the current state. As depicted in Fig.1 (b), we formulate two models: one for inferring subgoals based on the current state and the final goal, and the other for reconstructing the final goal using the current state and the given subgoals. The former model explicitly generates the subgoal, whereas the latter provides a guarantee on the worst-case for the log-likelihood of the subgoal distribution. As suggested in [18], we anticipate that adding the implicit model may outperform using only an explicit model when accomplishing subgoal generation for long-horizon tasks.

The contributions of this study are outlined as follows:

- 1) We introduce a **Hybrid Subgoal Generator**, which employs both explicit and implicit models to generate accurate subgoals by maximizing the evidence lower bound of the final goal.
- 2) We quantify the quality of inferred subgoals by establishing two criteria: feasibility and optimality. We

employ a **Hindsight Sampler** to facilitate the feasibility of the generated subgoals and a **Value Selector** to facilitate the selection of optimal subgoals.

- 3) We conduct a series of simulations and real-world experiments to evaluate the performance of our algorithm. The results demonstrate that our method outperforms other approaches under comparable environmental conditions.

The rest of this paper is organized as follows: Sec. II reviews the state-of-the-art; Sec. III introduces the preliminaries; Sec. IV formulates the problem; Sec. V describes the proposed methodology; Sec. VI presents the experimental results; Sec. VII gives conclusions and future work.

II. RELATED WORK

Deep reinforcement learning (DRL) has proven effective in seeking optimal solutions for specific tasks through the utilization of well-designed reward functions. Building on the recent successes of DRL, researchers have expanded this technique to address long-horizon tasks with varying objectives, known as goal-conditioned reinforcement learning (GCRL). Current methodologies for tackling the long-horizon GCRL problem predominantly involve imitating expert trajectories, enhancing goal-oriented exploration, and decomposing complex tasks into smaller subtasks.

A. Imitation Learning

Recent advancements [4], [11] in imitation learning have demonstrated the capacity of humans to identify and learn critical points from expert demonstrations or trajectories to accomplish the final goal. Jin et al. [19] proposed a method enabling robots to learn an objective function from a few sparsely demonstrated keyframes to solve long-horizon tasks. Joey et al. [20] were the first to utilize representation learning with an instruction prediction loss, operating at a high level of abstraction, to accelerate imitation learning outcomes. Furthermore, Huang et al. [4] trained reinforcement learning (RL)-based agents to solve long-horizon tasks with a few expert demonstrations collected from virtual reality. However, the trajectories generated by human experts cannot cover the entire state space, leading to trained policies that generalize poorly to unseen tasks. Additionally, strategies trained via imitation learning may not be optimal, as the demonstration data are likely suboptimal, and consequently, the trained strategies are unlikely to surpass the demonstration policy. Our approach diverges from relying on expert demonstrations and instead employs reinforcement learning (RL) to explore and discover the optimal policy through trial and error.

B. Exploration Efficiency

Increasing the exploration efficiency of robots is crucial for solving long-horizon tasks. Numerous studies have endeavored to enhance exploration efficiency by enriching the experience replay buffer to incorporate more unknown states. Works such as RIG [21] and MEGA [13], prioritize exploring low-density or sparse regions, with the intent of maximizing the likelihood of visiting fewer states, which is equivalent to maximizing the

entropy of the desired goal distribution. Warde-Farley et al. [22], on the other hand, aim to learn goal-conditional policies and reward functions by maximizing the mutual information between the achieved and final goals. [12] leverages Large Language Models (LLMs) as assistive intrinsic rewards, guiding the agent to explore the environment more effectively. In general, most methods expand the exploration space but continue to rely on uniformly distributed actions and random action noise, which limits subgoal generation to neighboring states near the explored region. And LLMs-based exploration cannot always guarantee the feasibility of generated subgoals. Our work improves the exploration efficiency by utilizing an iterative updating approach to explore the subgoal space and ensure the feasibility of the generated subgoals.

C. Subgoal Orientated Goal-conditioned RL

Drawing inspiration from the human tendency to decompose long-horizon tasks into smaller ones, some researchers are exploring the generation of subgoals by selecting them from historical experiences [3], [23] or by training subgoal generation policies [15], [24]. The experience replay buffer, rich in valuable historical data, facilitates the extraction of valid states as subgoals in a simple and efficient manner, while also ensuring their reachability. However, those methods [15], [24] rely solely on the initial state to predict the future subgoals, lacking the capability to effectively explore the subgoal space and identify the optimal subgoal towards the final goal due to the experience replay buffer being filled with low-quality trajectories.

Recently, new image editing diffusion models such as SuSIE [25] and SkillDiffuser [26] have been introduced for subgoal generation. Despite their powerful capabilities, these models rely heavily on pretrained large diffusion models and robot demonstration video data, which restrict them to image-based environments. Additionally, the substantial computational resources required for these large models present a significant challenge for deployment in real-world robotic tasks. Some subgoal generation methods based on LLMs, such as [2], [16], have been proposed to generate text-based subgoals for low-level policy execution. However, these methods generally operate under the strong assumption that text-based states and goals can be obtained so that can be input to LLMs.

Contrary to previous work that infers subgoals exclusively using an explicit model, we restrict our policy search by employing an implicit model and the prior distribution implicitly represented in the offline dataset. Our work also adopts a divide-and-conquer manner to solve long-horizon tasks by utilizing the hierarchical architecture [27], [28], where the top-level focuses on reasoning and decision-making to generate subgoals, and the low-level interacts with the environment to achieve the subgoals sequentially.

III. PRELIMINARIES

A. Semi-MDP

We formulate the long-horizon task as a Semi-Markov Decision Process (Semi-MDP) [29]. More precisely, we extend the general MDP by adding option policy ψ and options \mathcal{O} .

The Semi-MDP is denoted by $\langle \mathcal{S}, \mathcal{A}, r, \rho_0, \rho_g, \mathcal{T}, \gamma, \mathcal{G}, \mathcal{O}, \psi \rangle$, where \mathcal{S} is state space containing all possible state, \mathcal{A} is action space and r is the reward function. We use ρ_0 and ρ_g to denote the distribution of initial state and final goal, respectively. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function and γ is the discount factor. \mathcal{G} is the goal space from which final goals and subgoals can be sampled. $\mathcal{O} : \langle \mathcal{I}, \pi, \mathcal{E} \rangle$ is the option consisting of three components, where $\mathcal{I} \in \mathcal{S}$ is the initial state space, $\mathcal{E} \in \mathcal{S}$ is the terminal state space, $\pi : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{A}$ is the specific policy for current option, respectively.

We denote the achieved goal at time step t as ag_t . Typically, $ag_t = \phi(s_t)$, where $\phi : \mathcal{S} \rightarrow \mathcal{G}$ represents a known and traceable mapping that defines goal representation [30]. The subgoal is often identified as the achieved goal at the terminal state of an option. $\psi : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{G}$ is the subgoal generation policy mapping the state and final goal to subgoal for inferring a sequential option $\{o_1, o_2, \dots, o_n\}$ where each $o_i \in \mathcal{O}$. And for each option o_i , the policy π is utilized when the agent encounters the initial state $s_0^i \in \mathcal{I}$ and terminates in $s_e^i \in \mathcal{E}$.

In sparse reward goal-conditioned RL, the reward function r is a binary signal indicating whether the current goal is achieved, as expressed by,

$$r(s_t, a_t, g) = \begin{cases} 0 & ||\phi(s_t) - g||_2 \leq \epsilon, \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

where ϵ is a small threshold indicating whether the goal is considered to be reached, and s_t, a_t are the state and action at time step t and g is the final goal. Denote that the initial state s_0 of the whole trajectory is the initial state of option o_1 , and the final state is the terminal state of option o_n . And the terminal state s_e^i of option o_i is the initial state s_0^{i+1} of next option o_{i+1} . All options are connected in a head-to-tail manner to form the entire trajectory.

B. Soft Actor-Critic

Soft Actor-Critic (SAC) [31] is an off-policy reinforcement learning algorithm that attempts to maximize the expected return and increase the policy's entropy. The underlying idea is that increasing entropy leads to more exploration and effectively prevents the policy from reaching a local optimum.

Let $\tau = \{s_0, a_0, s_1, a_1, \dots\}$ be the trajectory starting from initial state s_0 . We use $\rho_\tau = \rho_0 \prod_t \pi(a_t|s_t, g) \mathcal{T}(s_{t+1}|s_t, a_t)$ to denote the distribution of the τ induced by the policy π .

In contrast to the standard maximum expected reward used in traditional reinforcement learning, the objective of action policy π aims to maximize its entropy at each visited state by augmenting with an entropy term,

$$J(\pi) = \mathbb{E}_{g \sim \rho_g, \tau \sim \rho_\tau} \left[\sum_t \gamma^t r(s_t, a_t, g) + \alpha \mathcal{H}(\pi(\cdot|s_t, g)) \right] \quad (2)$$

where $\mathcal{H}(\pi(s_t, g))$ is the entropy of the policy π , and $\alpha > 0$ is the temperature parameter that determines the importance of the entropy term with respect to the reward and is used to control the stochasticity of the optimal policy.

Thus, we can obtain the optimal action policy π^* by:

$$\pi^* = \arg \max_{\pi} J \quad (3)$$

C. Hindsight Experience Replay

Hindsight Experience Replay (HER) [32] is an algorithm that improves data efficiency by relabeling data in the replay buffer. It builds on the basis that failed trajectories may still achieve other unexpected states, which could be helpful for learning a policy.

It mitigates the challenge of sparse reward settings by constructing imagined goals in a simple heuristic way to leverage previous replays. During the training process, the HER relabeled the desired goal of this trajectory by specific sampling strategies, such as *future* and *final*, which take the future achieved goal and the final achieved goal as the desired goal, respectively.

IV. PROBLEM FORMULATION

Given an initial state $s_0 \in S$ and desired final goal $g \in \mathcal{G}$, our objective is to infer a sequence of feasible and optimal options $\{o_1, o_2, \dots, o_n\}$ by using subgoal generation policy ψ , where $o_i = \langle s_0^i, \pi, s_e^i \rangle, i \in [1, n]$. As the terminating state s_e^i becomes the initial state for the subsequent option, the option inference problem can be simplified to a terminal state inference problem. In other words, we aim to infer a sequence of feasible and optimal subgoals $\{\hat{g}_1, \hat{g}_2, \dots, \hat{g}_{n-1}\}$, where $\hat{g}_i = \phi(s_e^i)$ for $i = [1, n-1]$. For each subgoal \hat{g}_i , we will generate actions by the low-level policy π , i.e., $a = \pi(s_0^i, \hat{g}_i)$.

Definition IV.1 (Feasibility). The sequence of subgoals $\hat{g}_1, \hat{g}_2, \dots, \hat{g}_{n-1}$ is feasible if the robot can approach the subgoal \hat{g}_i from the previous one \hat{g}_{i-1} .

Within the subgoal space \mathcal{G} , the majority of subgoals generated by the subgoal generation policy are unfeasible, either due to spatial distance or the strict dependency order of subtasks inherent in long-horizon tasks [33]. An subgoal generation policy ψ is anticipated to generate a subgoal sequence $\hat{g}_1, \hat{g}_2, \dots, \hat{g}_{n-1}$, such that the transition probability from previous subgoal \hat{g}_{i-1} to current subgoal \hat{g}_i exceeds 0. Here, the transition probability is denoted as $\rho(s_0^i) \prod_{s=s_0^i, a \sim \pi} [\pi(s, \hat{g}_i) \mathcal{T}(s, a)]$ and $\rho(s_0^i)$ is the probability of state s_0^i . More details can be found in Supplementary section 1.

Definition IV.2 (Optimality). We call a sequence of subgoals $\hat{g}_1^*, \hat{g}_2^*, \dots, \hat{g}_{n-1}^*$ optimal only if the robot can achieve the maximum accumulated reward of all options to finish the task.

Resolving long-horizon tasks can be distilled down to the challenge of optimization over a sequence of subgoals for the goal-conditioned policy. This optimization over subgoals can be viewed as high-level planning, wherein the optimizer identifies waypoints to accomplish each subgoal [15]. We aim to find an optimal subgoal generation policy ψ^* to maximize the total accumulated reward of all options,

$$\psi^* = \arg \max_{\psi} \sum_{i=1}^n J_{o_i} \quad (4)$$

where J_{o_i} is the objective of option o_i , denoted as

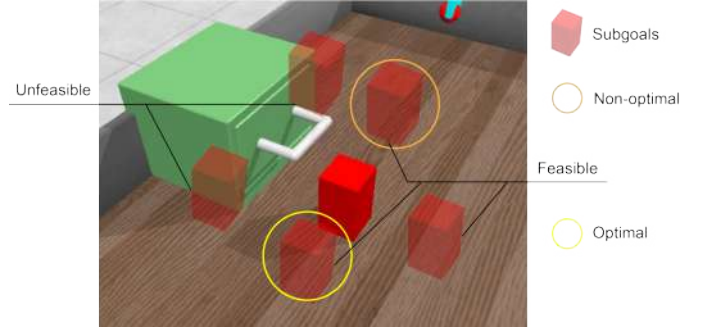


Fig. 2. Examples of feasible and optimal subgoals in the OpenDrawer environment are depicted.

$$J_{o_i}(\psi, \pi) = \mathbb{E}_{\hat{g}_i \sim \psi, a_t \sim \pi} \left[\sum_t \gamma^t r(s_t^i, a_t, \hat{g}_i) + \alpha \mathcal{H}(\pi(\cdot | s_t^i, \hat{g}_i)) \right] \quad (5)$$

We employ the OpenDrawer environment as a case study to elucidate the concept of feasible and optimal subgoals. As illustrated in Fig. 2, the ultimate objective is to open the drawer by manipulating the handle. However, a red obstacle block is positioned in front of the drawer, necessitating that the robotic arm first push the block away. This task encompasses two distinct skills: pushing the block and opening the drawer. The subgoal generation policy infers subgoals within the subgoal space, which can sometimes interfere with the drawer, rendering them unfeasible for the robot. Considering the spatial relationship between the block and the drawer, the subgoal encircled by the yellow line is deemed optimal as it enables the robot to move the block away more efficiently.

V. METHODOLOGY

As illustrated in Fig. 3 (b), our EISP algorithm comprises three components: the Hybrid Subgoal Generator, the Hindsight Sampler, and the Value Selector. The Hybrid Subgoal Generator explores the subgoal space and generates subgoals that incorporate additional features of the final goal, thereby enriching the offline dataset with trajectories containing more detailed subgoal information. The Hindsight Sampler and the Value Selector work in conjunction to produce more feasible and optimal subgoals, which in turn enhances the effectiveness of the Hybrid Subgoal Generator's exploration. The entire framework is trained jointly in an iterative manner, as shown in Fig. 3 (a), facilitating a balance between exploring the subgoal space and ensuring the feasibility and optimality of the subgoals.

A. Hybrid Subgoal Generator

Given an initial state s_0 and a final goal g , we expect the robot to infer a series of subgoals $\{\hat{g}_1, \hat{g}_2, \dots, \hat{g}_{n-1}\}$ to guide it to the final goal g . Generally, most current subgoal generators [3], [15], [23] employ explicit feed-forward models that maps subgoals directly from the state and the final goal. These methods are inspired by the human approach to solving long-horizon tasks, wherein humans generate subgoals

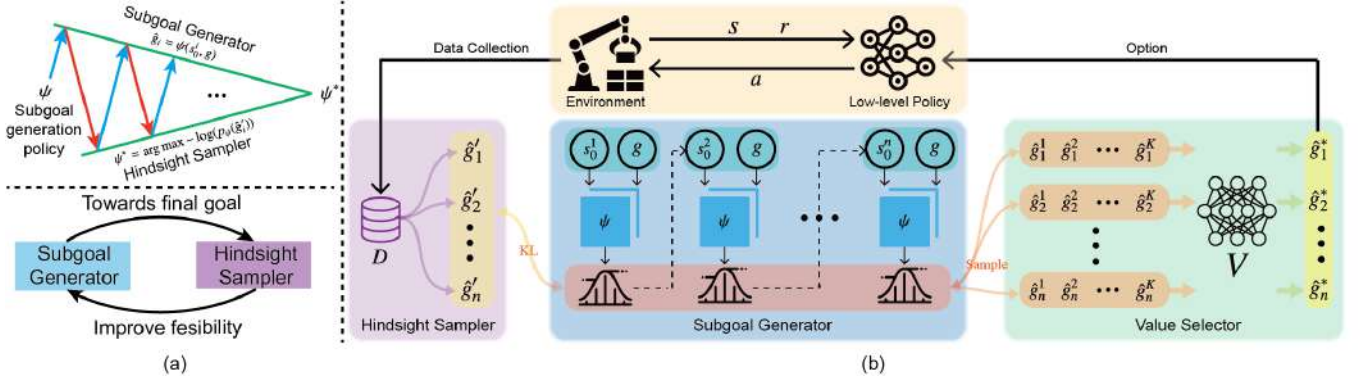


Fig. 3. (a). The iterative updating of the Subgoal Generator and the Hindsight Sampler. (b). The Explicit-Implicit Subgoal Planning (EISP) algorithm consists of three main components: a Hybrid Subgoal Generator, a Hindsight Sampler, and a Value Selector. The subgoal generator takes as input the current state and the desired goal and outputs subgoals to accomplish the long-horizon task. The Value Selector and the Hindsight Sampler are utilized to ensure that the subgoals are optimal and feasible, respectively.

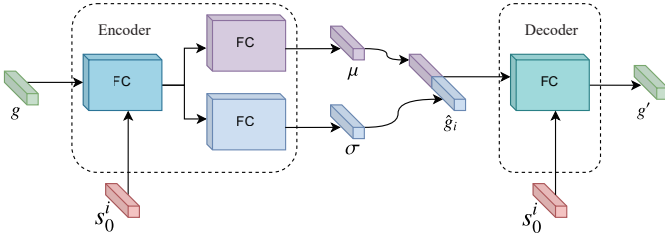


Fig. 4. Details of our proposed subgoal inference method. It inherits the variational autoencoder structure, where the encoder generates the subgoals by taking the current state and the final goal as inputs, and the decoder generates a reconstructed final goal conditioned on the current state and the subgoal.

to decompose long-horizon tasks into smaller ones based on the final goal and the current state. However, one critical aspect often overlooked is that humans can also predict long-term outcomes using short-term subgoals conditioned on the current state. Current research [18] in behavior cloning has demonstrated that implicit models possess a superior capacity to learn long-horizon tasks more effectively than their explicit counterparts. This evidence incites us to employ the implicit model in subgoal generators, which is anticipated to enhance the ability to solve long-horizon tasks significantly.

Based on these insights, we propose the Hybrid Subgoal Generator, fundamentally structured as a Conditional Variational Autoencoder (CVAE) [34]. Within this framework, the encoder ψ serves as an explicit model that can infer subgoals based on the current state and the final goal, while the decoder ψ' functions as an implicit model, designed to reconstruct the final goal using the current state and the given subgoals, as depicted in Fig. 4. After preprocessing the state s and the final goal g through several fully connected layers, the encoder computes the mean μ and the standard deviation σ via two fully connected networks. The subgoal is then sampled from the normal distribution modeled using these calculated values of μ and σ . The decoder takes the subgoal \hat{g}_i as input and conditioned on the current state s_0^i to generate the reconstructed final goal g' .

Unlike the original CVAE, where the latent space serves as a low-dimensional mapping from the inputs, we employ the

CVAE structure but interpret the latent space as the subgoal space. This approach distinguishes our work from previous studies [15], which also use the CVAE structure to generate subgoals but focus on reconstructing the subgoal using a learned latent representation of the transitions. By modeling the conditional subgoal generation policy ψ , which takes as inputs the initial state s_0^i of option i and the final goal g , we transition the subgoal generation approach from the initial planner (infers subgoals only at the initial state) to an incremental Planner (infers subgoals based on current state). That is, the subgoals can be inferred incrementally by:

$$\hat{g}_i = \psi(s_0^i, g), \quad \text{for } i = 1, \dots, n-1 \quad (6)$$

where $s_0^1 = s_0$. To ensure the robustness of the inferred subgoals and prevent the agent from lingering at one subgoal, we impose a time limit of T . The subgoal generation policy ψ deduces a new subgoal either when the agent reaches the current subgoal or when the allocated time step T for the current subgoal expires.

We denote the implicit model as $\psi'(g'|s_0^i, \hat{g}_i)$, which is designed to accurately reconstruct the final goal from the subgoal. The objective is to maximize the log probability of the final goal, $\log p_{\psi'}(g)$, by maximizing its evidence lower bound (ELBO) [35], formulated as:

$$\mathbb{E}_{p_\psi(\hat{g}_i|g, s_0^i)} \left[\log \frac{p_{\psi'}(g, \hat{g}_i, s_0^i)}{p_\psi(\hat{g}_i|g, s_0^i)} \right] \quad (7)$$

Thus, the objective of the hybrid subgoal generator \mathcal{L}_{HB} can be defined as follows:

$$\mathbb{E}_{p_\psi(\hat{g}_i|g, s_0^i)} [\log p_{\psi'}(g|\hat{g}_i, s_0^i)] - D_{KL}[p_\psi(\hat{g}_i|s_0^i, g) \| p(\hat{g}_i)] \quad (8)$$

where $\mathbb{E}_{p_\psi(\hat{g}_i|g, s_0^i)} [\log p_{\psi'}(g|\hat{g}_i, s_0^i)]$ is the reconstruction loss, ensures that the predicated subgoals involving features of the final goal, guiding ψ to explore the subgoal space toward the final goal. And $p(\hat{g}_i)$ is the prior distribution over the subgoals, typically a standard normal distribution $\mathcal{N}(0, I)$. $D_{KL}[p_\psi(\hat{g}_i|s_0^i, g) \| p(\hat{g}_i)]$ denotes the Kullback-Leibler (KL) Divergence [35] between the prior distribution $p(\hat{g}_i)$ and the distribution of the generated subgoals by ψ . A detailed

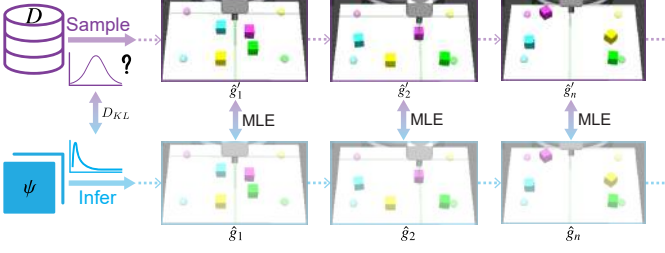


Fig. 5. Hindsight Sampler to guide the generator to produce feasible subgoals.

derivation can be found in Supplementary section 2. We will introduce an improved prior distribution for the subgoals in the following section.

B. Hindsight Sampler for Feasible Subgoals

The prior distribution of the subgoal \hat{g}_i in the CVAE component is denoted as $p(\hat{g}_i)$, as shown in Eq.12. It represents the initial assumptions about the subgoal without any data observations. By applying a better prior, it is possible to ensure the feasibility of the subgoal generator by minimizing the KL divergence between $p(\hat{g}_i)$ and $p_\psi(\hat{g}|s_0^i, g_i)$. However, the true distribution of the valid subgoals is not available in practice. The most common approach is to use a fixed standard Gaussian distribution as the prior [36] which is favored for its simplicity and computational efficiency but imposes limitations on the model's expressiveness and generative capabilities. Inspired by the VampPrior approach [37], which learns a more flexible prior distribution based on posterior data, we propose a method to derive a more practical subgoal distribution that surpasses the standard Gaussian distribution. Since empirically, the subgoal distribution within the offline dataset \mathcal{D} can be approximately regarded as the valid subgoal distribution, we utilize this offline dataset to sample valid subgoals and thereby derive a more accurate prior distribution.

Considering the quality of the offline dataset, which lacks successful trajectories from the initial state to the final goal, we introduce the Hindsight Sampler. This approach is derived from the Hindsight Experience Replay (HER) buffer [32], which relabels the future achieved goal in failed trajectories as the final goal, and then recalculate the rewards for all states within that trajectory. This revised trajectory is used to train the subgoal generation policy.

As shown in Fig. 5, during the training period, we first sample the trajectory τ_D with total time step T_τ from the offline dataset \mathcal{D} , and take the final achieved goal of this trajectory as the desired goal g of the current trajectory. We then select waypoints from the trajectory as subgoals at intervals of T_τ/n , denoted as $\{\hat{g}'_1, \hat{g}'_2, \dots, \hat{g}'_{n-1}\}$. We utilize the maximum likelihood estimation (MLE) [38], which is a method estimating the parameters of an assumed probability distribution given some observed data, to maximize the log-likelihood of the valid subgoals sampled from the offline dataset under the unknown subgoal generation policy ψ . The objective of the Hindsight Sampler \mathcal{L}_{HS} is as follows:

$$\mathcal{L}_{HS} = -\log(p_\psi(\hat{g}'_i)) \quad (9)$$

Algorithm 1: Collect Rollout Trajectories

Input: Subgoal generation policy ψ , low-level policy π , state value function V , time limit T , distance function $Dist$, initial state s_0 , final goal g

Output: \mathcal{D}

```

1  $s_0 \leftarrow$  initial state
2  $\hat{g}^* \leftarrow g$ 
3 forall  $t \leftarrow [0, 1, 2, \dots]$  do
4    $ag_t \leftarrow \phi(s_t)$ 
   /* Generate and select optimal subgoal */
5   if  $t \bmod T = 0$  or  $Dist(ag_t, \hat{g}^*) \leq \epsilon$  then
6     Sample  $K$  subgoal candidates  $\{\hat{g}^1, \hat{g}^2, \dots, \hat{g}^K\}$ 
       by using  $\psi$ 
7     Select optimal subgoal  $\hat{g}^*$  from candidates by
       using (11)
   /* Set subgoal as the final goal if they
       are too close */
8     if  $Dist(\hat{g}^*, g) \leq \epsilon$  then
9        $\hat{g}^* \leftarrow g$ 
10   $a_t \leftarrow \pi(s_t, \hat{g}^*)$ 
11  Execute action  $a_t$  and obtain next state  $s_{t+1}$  and
       reward  $r_t$ 
12  Store  $(s_t, a_t, r_t, s_{t+1}, \hat{g}^*)$  in experience replay
       buffer  $\mathcal{D}$ 

```

where $p_\psi(\hat{g}'_i)$ is the probability of sampled subgoals under the distribution generated by the hybrid subgoal generator.

C. Value Selector for Optimal Subgoals

The Universal Value Function Approximator (UVFA) plays a critical role in goal-conditioned RL. It functions as a mapping from the goal-conditioned state space to a nonlinear value function approximator [39]. We denote the state value function as V . In the context of goal-conditioned RL, it is typically utilized to assess the current state under the condition of the goal. The value function V of SAC in goal-conditioned RL involves the extra entropy bonuses from every time step utilized to assess the state, denoted as follows:

$$V(s, g) = \mathbb{E}_{g \sim \rho_g, \tau \sim \rho_\tau} \left[\sum_t \gamma^t r(s_t, a_t, g) + \alpha \mathcal{H}(\pi(\cdot|s_t, g)) \right] \quad (10)$$

where $s_0 = s$. In our algorithm, V serves a dual purpose: guides the updates of low-level policy and selects optimal subgoals for the high-level planner. To generate subgoal \hat{g} at current state s_t , conditioned on the final goal g , we initially sample K candidates $\{\hat{g}^1, \hat{g}^2, \dots, \hat{g}^k, \dots, \hat{g}^K\}$ using the Hybrid Subgoal Generator, where $\hat{g}^k \sim \psi(s_t, g)$. As shown in Fig. 3, the candidates are then ranked using the state value function V , where higher state values represent better subgoals. Hence, the subgoal state corresponding to the highest value will be selected as the optimal subgoal \hat{g}^* for the action policy, that is,

$$\hat{g}^* = \arg \max_{\hat{g}^k} \{V(s, \hat{g}^k), k \in [1, K]\} \quad (11)$$

Algorithm 2: EISP

Input: \mathcal{D}
Output: ψ^*, ψ'^*

- 1 Sample mini-batch $\mathcal{B} \leftarrow \{(s_t, a_t, r_t, s_{t+1}, \hat{g}^*)\}_{t=1}^N \sim \mathcal{D}$
 - 2 Sample subgoals \hat{g}' by using **Hindsight Sampler**
 - 3 Calculate $\mathcal{L}_{HY} \leftarrow (8)$, $\mathcal{L}_{HS} \leftarrow (9)$
 - 4 Update subgoal generation policy by
 $\psi^*, \psi'^* = \arg \min_{\psi, \psi'} \mathcal{L}$
 - 5 Update low-level policy π
-

The optimal subgoal \hat{g}^* is subsequently utilized to guide the low-level action policy. The value function may not be optimal during the initial stages of training as the low-level policy may not be sufficiently trained. However, as the training of the value function progresses and improves, the selected subgoals also tend towards optimality.

D. Integrate with RL

Algorithm 1 provides the process of collecting rollout data to the experience replay buffer during the training of the subgoal generation policy. The distance function $Dist(g_1, g_2)$ represents the Euclidean distance between two goals g_1 and g_2 , which can be mathematically expressed as $Dist(g_1, g_2) = \sqrt{\sum_{f=1}^M (g_{1f} - g_{2f})^2}$, where M denotes the dimension of the goal space. During training, we first infer potential subgoal candidates through the Hybrid Subgoal Generator. Then, the optimal subgoal is selected using the Value Selector. By substituting the final goal with the selected subgoal, we obtain trajectories of transitions $(s_t, a_t, r_t, s_{t+1}, \hat{g}^*)$ and store them in the replay buffer \mathcal{D} .

The training process of the EISP is shown in Algorithm 2. Prior to EISP training, we utilize RL algorithms to pretrain the low-level action policy on an offline dataset containing short demonstrations of primitive interactions, such as picking, placing, opening, and closing actions. During EISP training, we incorporate new data collected by Algorithm 1 to train the high-level subgoal generator and fine-tune the low-level action policy. This is achieved by first sampling a mini-batch \mathcal{B} from \mathcal{D} , consisting of $N - 1$ transitions, where $t \in [1, N]$. For each transition, the final goal is relabeled by the future achieved goal, and valid subgoals \hat{g}' will be sampled by using the Hindsight Sampler. The objective of the EISP is formulated as:

$$\mathcal{L} = \mathcal{L}_{HY} + \beta \mathcal{L}_{HS} \quad (12)$$

where β is the hyper parameter. The explicit policy ψ and implicit policy ψ' can be updated by using stochastic gradient descent with \mathcal{L} . The proposed algorithm is entirely compatible with other off-policy algorithms, owing to its exclusive focus on high-level planning.

VI. EXPERIMENTAL RESULTS

A. Tasks

We evaluate EISP in four tasks, Stack, Push, OpenDrawer, and Store, both in simulation and real world, as illustrated

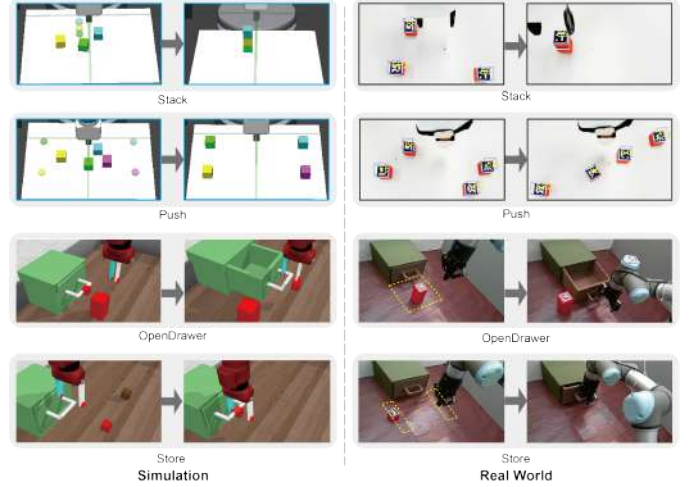


Fig. 6. Four manipulation tasks, i.e., Stack, Push, OpenDrawer, and Store. For the Stack and Push tasks, we label the initial states and the desired goals as solid cubes and transparent spheres. For the OpenDrawer and Store tasks, we use yellow dashed rectangles to represent the initial position of the blocks.

in Fig. 6. All simulation environments are created using the Gymnasium Robotics [40] Framework. The dimensional of the state space \mathcal{S} , action space \mathcal{A} , goal space \mathcal{G} , and the episode length L are provided in Table I. These tasks are considered as long horizon tasks, as they require robots to generate a sequence of actions and corresponding control signals over a long period of time. In our settings, we define the time step as the interval at which control signals are applied. The length of the total time steps required to complete each task serves as a measure of the time scale. The control signal refers to the action comprising the relative position of the end effector and the gripper open/close signal.

Stack The block stacking task entails picking up three blocks and sequentially placing them to the target. Long-horizon planning is challenging because the entire task must be completed without disrupting previously accomplished subtasks [41].

Push4 This task integrates four block-pushing tasks from the D4RL benchmark suite [42]. The robot is required to push four objects to their corresponding target. The complexity of the task increases when certain blocks obstruct others.

OpenDrawer The robot needs to open the drawer by first removing the block and then pulling the handle. In the image-based environment, the observation space consists of a $3 \times 48 \times 48$ image. The task is challenging due to its incorporation of multiple skills (pushing the block and pulling the handle) and task dependencies.

Store This task requires the robot to first open the drawer, then pick and place the block into the drawer, and finally close the drawer. This process involves multiple skills, such as pushing and picking. Similar to the OpenDrawer task, experiments are conducted in both state-based and image-based environments.

B. Implementation Details

Script policies for pretraining. The SAC algorithm serves as the foundational RL method for training the low-level action policy. Initially, the action policy is pretrained using an

TABLE I
PARAMETERS SETTING.

| Env | Stack | Push4 | OpenDrawer (State) | Store (State) |
|---------------|----------------|----------------|-----------------------|------------------|
| \mathcal{S} | 55 | 70 | 42 | 42 |
| \mathcal{G} | 9 | 12 | 6 | 6 |
| \mathcal{A} | 4 | 4 | 4 | 4 |
| L | 300 | 400 | 300 | 1000 |
| \mathcal{D} | 1e5 | 1e6 | 1e6 | 1e6 |
| N | 1024 | 1024 | 1024 | 1024 |
| SAC lr | $3e-3$ | $3e-4$ | $3e-4$ | $3e-3$ |
| π | 256×2 | 256×3 | 512×4 | 512×4 |
| α | 0.01 | 0.01 | 0.01 | 0.01 |
| γ | 0.99 | 0.99 | 0.99 | 0.99 |
| EISP lr | $1e-5$ | $1e-5$ | $1e-5$ | $1e-5$ |
| T | 30 | 30 | 50 | 50 |
| n | 4 | 4 | 6 | 6 |
| ψ | 256×2 | 256×2 | 512×4 | 512×4 |
| β | $1e-2$ | $1e-2$ | $1e-3$ | $1e-3$ |

offline dataset comprising primitive actions, such as picking and placing, among others. These demonstrations are relatively easy to obtain because they consist of short-term actions gathered through scripted policies trained using RL algorithms. **Network details.** We implement two probabilistic neural networks using one layer of fully connected network mapping from the encoder output to the subgoal space. The decoder has the opposite structure to the encoder. It takes the encoder-generated subgoals and the current state as inputs and outputs the reconstructed desired goal. We use a fixed temperature parameter $\alpha = 0.01$ for all tasks. Table I presents the specifications of various parameters, including the size of the replay buffer \mathcal{D} and the mini-batch \mathcal{B} , the network structure, and the learning rates of SAC and EISP. The weight β used in (12) refer to Table I. Both the option and action strategies use the Adam optimizer to update the network parameters. For image-based observations, we employ a vector quantized variational autoencoder (VQ-VAE) [43] to extract features, which are subsequently used as input for subgoal generation. Detailed training procedures and results of the VQ-VAE can be found in Supplementary Sections 3 and 4.

Number of subgoals for each task. The number of subgoals used in the Hindsight Sampler is specified in Table I. The number of subgoals is determined by the timesteps allocated for completing a subgoal and the entire task. For instance, the expected total timesteps for the Store is approximately 300, which comprises at least four subgoals: opening the drawer, picking up the block, placing the block, and closing the drawer. To ensure that all subgoals are simple, the timesteps of a subgoal should be less than 60. We allocate 50 timesteps for each subgoal in practice.

C. Qualitative Analysis

This section demonstrates the feasibility and optimality of inferred subgoals through the qualitative results.

Feasibility Fig. 7 shows the subgoals generated by EISP in four state-based tasks mentioned above. We mark the inferred subgoals as transparent cubes in the Stack and Push task and solid small cubes in the OpenDrawer and Store

task. For each task, we show the sequence of subgoals in different stages, starting from the initial state to the final goal. Each task involves different skills; for example, the Stack task uses Pick and Place skills iteratively to reach the final goal, whereas the Store task requires four skills to finish the entire task. Besides, we also infer subgoals for image-based environments, specifically OpenDrawer and Store, with the results shown in Fig. 8. For these tasks, low-dimensional features extracted from image observations serve as input, and the inferred subgoals are reconstructed using a pretrained VQ-VAE decoder. These reconstructed subgoals are depicted in Fig. 8 with yellow dashed rectangles.

It should be noted that many inferred subgoals are still noisy, indicating that certain subgoals may be unreachable within a given time step. This phenomenon is expected, as these subgoals typically provide the direction to aim for rather than the precise position the robot should achieve. The robot can ultimately achieve the final goal as this direction is progressively updated. Furthermore, EISP demonstrates robustness against non-optimal or unfeasible subgoals that may emerge during strategy execution, as it infers subgoals based on the current observation. For instance, the third subgoal generated in the Stack task does not seem optimal since it is not directly on the path to the final goal. However, it still provides directional guidance for the robot. As time progresses and the time budget for the current option expires, the next subgoal will offer more accurate information for the robot to achieve its objective.

Optimality To assess the optimality of generated subgoals, we visualize the distribution of subgoals in the high-dimensional subgoal space. For simplicity, the results of later experiments with OpenDrawer and Store were conducted in image-based environments. We compute the V values of 1000 randomly sampled subgoals from the subgoal space of each task. The t-SNE [44] algorithm is then employed to reduce the high-dimensional data into a two-dimensional format. The results are visualized in Fig. 9. Subgoals with lower V values are marked with lighter hues, indicating they may not be the optimal choice for the current state. In contrast, subgoals with higher V values are shown with darker hues, suggesting they would be more beneficial for the task. Subgoal candidates obtained from the trained subgoal strategy are also plotted as green crosses in the figure. The plot reveals that the subgoals inferred by the subgoal generator are mainly distributed within the space of higher V values. Initially, these subgoals are not located in regions with high V values, but as the training goes on and the dataset is updated with high-quality data, they are gradually shifted to regions with high V values.

D. Quantitative Analysis

Additionally, we perform qualitative experiments to demonstrate the superiority of our proposed methods. We conduct two ablation experiments, with the Hindsight Sampler and the Value Selector components being excluded respectively, to isolate their individual contributions to the overall performance. We also conduct comparative experiments against current state-of-the-art subgoal-generator algorithms, namely **RIS** (Reinforcement learning with Imaged subgoals) [23], **HP**

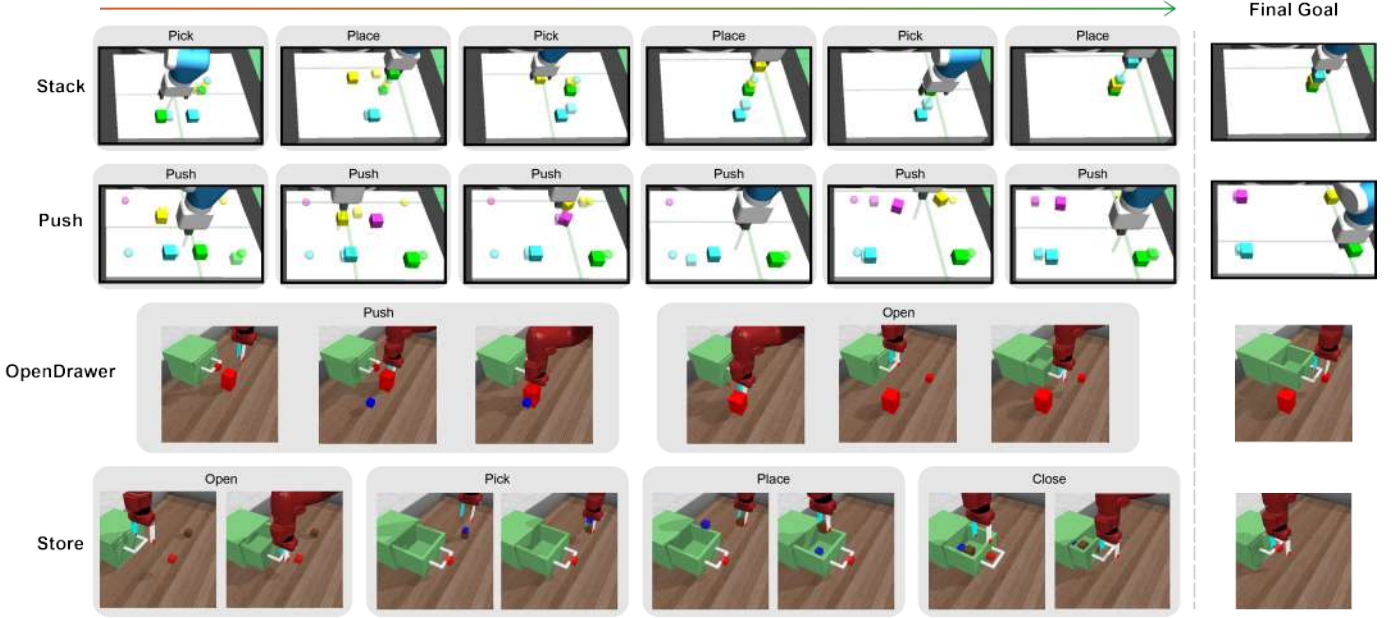


Fig. 7. Subgoal sequences generated by subgoal generation policy ψ from the initial state to desired goal on Stack, Push, OpenDrawer and Store. The transparent cubes in Stack and Push tasks, as well as the red and blue cubes in OpenDrawer and Store tasks, represent subgoals generated by EISP.

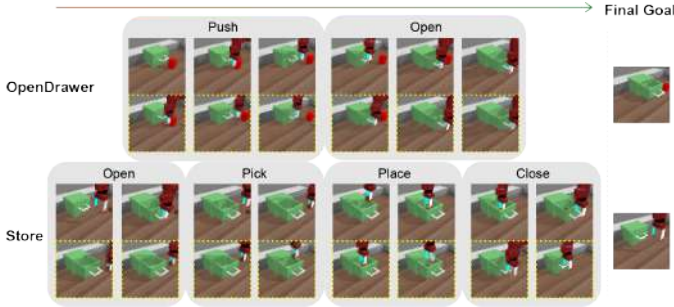


Fig. 8. Subgoal sequences generated by subgoal generation policy ψ from the initial state to desired goal on image-based environments. The figures with dashed rectangles (below) are the subgoals in the current observation (above).

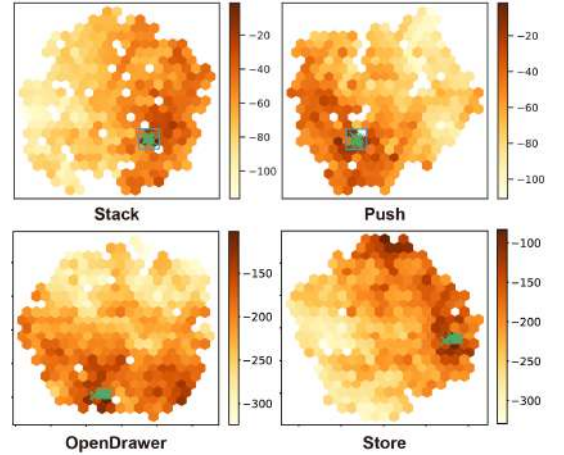


Fig. 9. A heatmap of high-dimensional subgoal space visualized by the t-SNE algorithm. The colors are granted according to the magnitude of the V value. Darker colors indicate better subgoals. It is shown that the distributions of the subgoals (marked in green crosses) generated by the subgoal generation policy share a high V value.

(Hindsight Planner) [3], and **LEAP** (Latent Embeddings for Abstracted Planning) [24].

- **RIS** aims to identify subgoals that are maximally distant from the initial state and final goal.
- **HP** employs the LSTM [45] architecture to sequentially generate subgoals by continuously integrating previously generated subgoals.
- **LEAP** learns the goal-conditioned policy predicated on the latent embedding of original complex observations.

We evaluate EISP and other baselines on four long-horizon tasks. The success rates and expected returns of the trained policies are illustrated in Fig. 10 and Table II. For each task, the highest returns are marked in bold, while the second highest returns are underlined. The results demonstrate that the EISP, which employs iterative joint training of all modules, achieves the highest success rates and expected returns.

The strategy without the Subgoal Generator, labeled **EISP w/o SG**, is implemented by removing the reconstruction loss. This modification results in a comparatively lower return than

the original EISP due to the absence of exploration information about the final goal, which is crucial for the Subgoal Generator to create subgoals that guide the agent towards the final goal. The strategy without the Hindsight Sampler, labeled **EISP w/o HS**, encounters difficulties in making the subgoals generated feasible, resulting in low success rates across all four environments. Conversely, the strategy lacking the Value Selector module, denoted as **EISP w/o VS**, still manages to achieve a measure of success (for instance, approximately a 60% success rate on the Stack task). This suggests that the Value Selector module primarily functions to facilitate the training of the subgoal generator, enabling the strategy to converge more rapidly and attain a higher success rate.

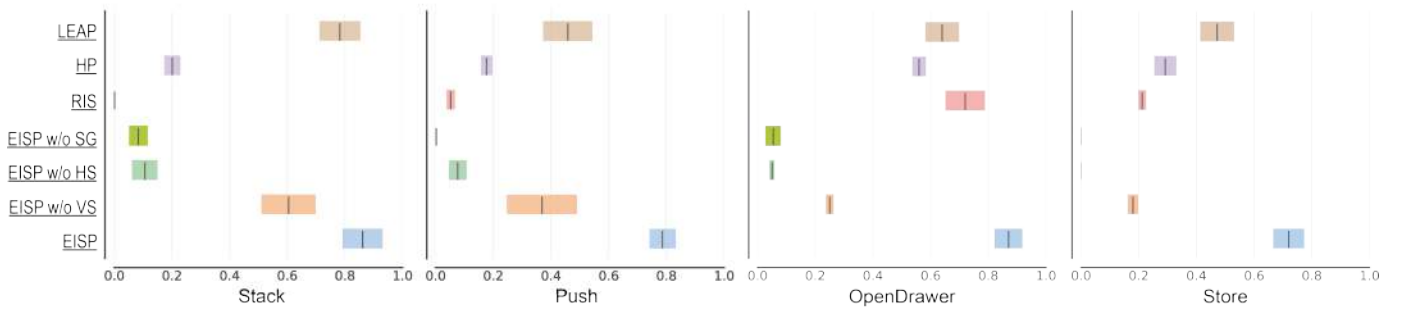


Fig. 10. The success rate of EISP and other baselines. The performance of EISP is higher than others in all four environments.

The **LEAP** and **HP** methods rely solely on the initial state to infer subgoals, which proves insufficient for tasks with varying desired goals. These methods lack robustness for complex tasks in most cases, as variations in the desired goal lead to less accurate subgoals, resulting in task failure. Furthermore, **RIS** samples subgoals randomly from the experience buffer can result in inaccurate subgoals, hindering the generation of optimal lower-level action policies. For detailed information on episodic returns throughout the entire training procedure, you can refer to Supplementary section 5.

TABLE II

TABLE OF THE EXPECTED RETURNS FOR DIFFERENT TRAINED POLICIES AFTER TESTING FIVE TIMES WITH DIFFERENT SEEDS. HIGHER RETURNS INDICATE BETTER PERFORMANCE.

| Policy | Stack | Push | OpenDrawer | Store |
|--------------------|---------------|----------------|----------------|----------------|
| RIS | -269.46 | -270.27 | -256.374 | -865.453 |
| HP | -283.46 | -311.96 | -272.97 | -844.58 |
| LEAP | -159.80 | -245.29 | -266.34 | -639.97 |
| EISP w/o SG | -290.3 | -340.32 | -292.56 | -987.73 |
| EISP w/o HS | -284.67 | -344.58 | -283.75 | -990.43 |
| EISP w/o VS | -201.33 | -313.15 | -284.72 | -775.61 |
| EISP (Ours) | -89.29 | -228.89 | -246.91 | -581.96 |

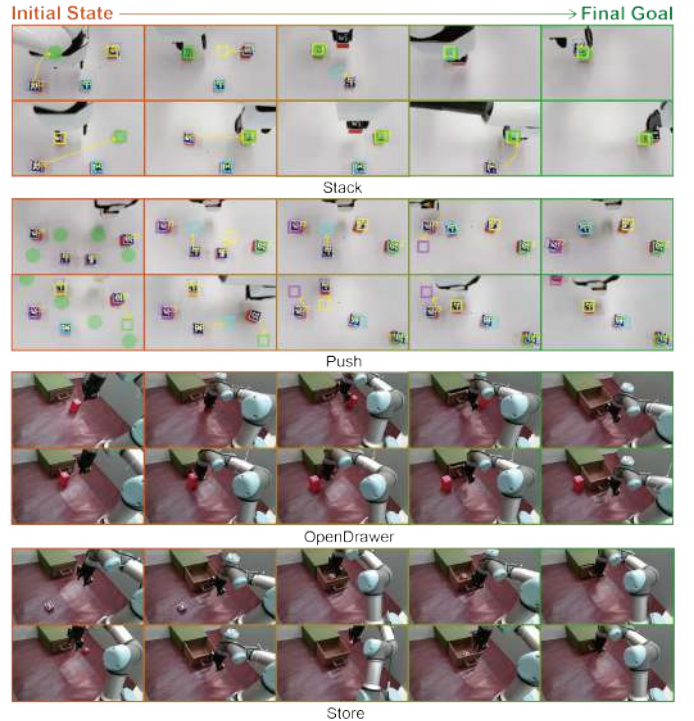


Fig. 12. The trajectories generated by the four manipulation tasks when executed in a real environment are presented. For the Stack and Push tasks, green spherical shadows represent the final goals, and transparent squares denote the subgoals generated during execution.



Fig. 11. Environments setup. Left: Tiago robotic arm. Right: UR3 robotic arm.

E. Real World Demonstrations

To demonstrate the adaptability of our algorithms in the real world, we employ the Tiago++ robotic arm [46] to execute Stack and Push tasks, and the UR3 robotic arm for OpenDrawer and Store tasks. The setups of these four environments are the same as in the simulation. Our experiments

utilize Pinocchio [47] for both motion planning and inverse kinematics. As depicted in Fig. 11, red cubes are manipulated by the robotic arm to complete the assigned task, with the number of red blocks used in Stack and Push being 3 and 4, respectively. We mount a calibrated Intel RealSense D435i RGB-D camera on the top of the table to facilitate top-view observation. We also fix a soft beam to the end of the gripper to mitigate potential collisions between the gripper, the table, and the blocks during the execution of the Push task.

The strategy employed to guide the robot is identical to that used in the simulation. To gather environmental observations, we utilize Aruco markers for the detection of each object's and gripper's position. Fig. 12 illustrates the subgoals inferred from the initial state to the desired goals for four manipulation

tasks. The desired goals are denoted as green spherical shades and the subgoals as transparent squares. We also conducted robustness tests and failed case studies, with the results presented in Supplementary sections 6 and 7.

VII. CONCLUSION

In conclusion, we propose Explicit-Implicit Subgoal Planning (EISP), an algorithm that leverages an explicit model to produce subgoals and an implicit model to provide guarantees on the worst-case log-likelihood of the subgoal distribution. While the efficacy of the proposed algorithm is substantiated by both qualitative and quantitative results, it is constrained by the predetermined number of subgoals during training, limited exploration of the subgoal space towards the final goal, and poor scalability across multiple tasks. Future work will concentrate on addressing these limitations.

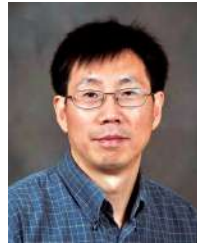
REFERENCES

- [1] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 239–252, 2022.
- [2] M. Dalal, T. Chiruvolu, D. S. Chaplot, and R. Salakhutdinov, "Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks," in *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.
- [3] Y. Lai, W. Wang, Y. Yang, J. Zhu, and M. Kuang, "Hindsight planner," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 690–698.
- [4] L. Huang, Z. Zhu, and Z. Zou, "To imitate or not to imitate: Boosting reinforcement learning-based construction robotic control for long-horizon tasks using virtual demonstrations," *Automation in Construction*, vol. 146, p. 104691, 2023.
- [5] H. Wang, H. Zhang, L. Li, Z. Kan, and Y. Song, "Task-driven reinforcement learning with action primitives for long-horizon manipulation skills," *IEEE Transactions on Cybernetics*, 2023.
- [6] S. Sohn, H. Woo, J. Choi, and H. Lee, "Meta reinforcement learning with autonomous inference of subtask dependencies," in *International Conference on Learning Representations*, 2020.
- [7] M. H. Lim, A. Zeng, B. Ichter, M. Bandari, E. Coumans, C. Tomlin, S. Schaal, and A. Faust, "Multi-task learning with sequence-conditioned transporter networks," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2489–2496.
- [8] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.
- [9] J. Borras, G. Alenya, and C. Torras, "A grasping-centered analysis for cloth manipulation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 924–936, 2020.
- [10] S. Paul, J. Vanbaars, and A. Roy-Chowdhury, "Learning from trajectories via subgoal discovery," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [11] N. Patton, K. Rahmani, M. Missula, J. Biswas, and I. Dillig, "Programming-by-demonstration for long-horizon robot tasks," *Proceedings of the ACM on Programming Languages*, vol. 8, no. POPL, pp. 512–545, 2024.
- [12] E. Triantafyllidis, F. Christianos, and Z. Li, "Intrinsic language-guided exploration for complex long-horizon robotic manipulation tasks," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 7493–7500.
- [13] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba, "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7750–7761.
- [14] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," in *International Joint Conferences on Artificial Intelligence*, 2022.
- [15] K. Fang, P. Yin, A. Nair, and S. Levine, "Planning to practice: Efficient online fine-tuning by composing goals in latent space," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4076–4083.
- [16] J. Zhang, L. Tang, Y. Song, Q. Meng, H. Qian, J. Shao, W. Song, S. Zhu, and J. Gu, "Fltrnn: Faithful long-horizon task planning for robotics with large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6680–6686.
- [17] L. Wu and K. Chen, "Goal exploration augmentation via pre-trained skills for sparse-reward long-horizon goal-conditioned reinforcement learning," *Machine Learning*, pp. 1–31, 2024.
- [18] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.
- [19] W. Jin, T. D. Murphey, D. Kulić, N. Ezer, and S. Mou, "Learning from sparse demonstrations," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 645–664, 2022.
- [20] J. Hejna, P. Abbeel, and L. Pinto, "Improving long-horizon imitation through instruction prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 7857–7865.
- [21] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," *Advances in neural information processing systems*, vol. 31, 2018.
- [22] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih, "Unsupervised control through non-parametric discriminative rewards," in *International Conference on Learning Representations*, 2018.
- [23] E. Chane-Sane, C. Schmid, and I. Laptev, "Goal-conditioned reinforcement learning with imagined subgoals," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1430–1440.
- [24] S. Nasiriany, V. Pong, S. Lin, and S. Levine, "Planning with goal-conditioned policies," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [25] K. Black, M. Nakamoto, P. Atreya, H. R. Walke, C. Finn, A. Kumar, and S. Levine, "Zero-shot robotic manipulation with pre-trained image-editing diffusion models," in *The Twelfth International Conference on Learning Representations*, 2023.
- [26] Z. Liang, Y. Mu, H. Ma, M. Tomizuka, M. Ding, and P. Luo, "Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16467–16476.
- [27] Y. Gao, J. Wu, X. Yang, and Z. Ji, "Efficient hierarchical reinforcement learning for mapless navigation with predictive neighbouring space scoring," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [28] P. Sermanet, T. Ding, J. Zhao, F. Xia, D. Dwibedi, K. Gopalakrishnan, C. Chan, G. Dulac-Arnold, S. Maddingeni, N. J. Joshi *et al.*, "Robovqa: Multimodal long-horizon reasoning for robotics," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 645–652.
- [29] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.
- [30] Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng, "Exploration via hindsight goal generation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
- [32] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.
- [33] S. Sohn, J. Oh, and H. Lee, "Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies," *Advances in neural information processing systems*, vol. 31, 2018.
- [34] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.
- [35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2013.
- [36] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.
- [37] J. Tomczak and M. Welling, "Vae with a vampprior," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1214–1223.

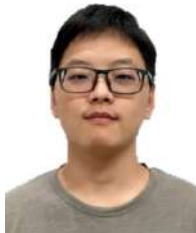
- [38] R. J. Rossi, *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.
- [39] T. Schaul, D. Horgan, K. Gregor, and D. Silver, “Universal value function approximators,” in *International conference on machine learning*. PMLR, 2015, pp. 1312–1320.
- [40] R. de Lazcano, K. Andreas, J. J. Tai, S. R. Lee, and J. Terry, “Gymnasium robotics,” 2023. [Online]. Available: <http://github.com/Farama-Foundation/Gymnasium-Robotics>
- [41] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Overcoming exploration in reinforcement learning with demonstrations,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.
- [42] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, “D4rl: Datasets for deep data-driven reinforcement learning,” in *International Conference on Learning Representations*, 2021.
- [43] A. Van Den Oord, O. Vinyals *et al.*, “Neural discrete representation learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [44] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [45] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [46] J. Pages, L. Marchionni, and F. Ferro, “Tiago: the modular robot that adapts to different research needs,” in *International workshop on robot modularity, IROS*, vol. 290, 2016.
- [47] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, “The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives,” in *IEEE International Symposium on System Integrations (SII)*, 2019.



Shengzeng Huo received the B.Eng. degree in vehicle engineering from the South China University of Technology, China, in 2019, the M.Sc. degree in mechanical engineering from The Hong Kong Polytechnic University, Hong Kong, in 2020, where he is currently pursuing the Ph.D. degree in the same discipline. In 2024, he was a research intern at Tencent Robotics X, China. His current research interests include bimanual manipulation, deformable object manipulation, and robot learning.



Guodong Guo (M’07-SM’07) received the B.E. degree in Automation from Tsinghua University, Beijing, China, the Ph.D. degree in Computer Science from the University of Wisconsin, Madison, WI, USA. He is now a Professor at Eastern Institute of Technology, and the Vice President of Ningbo Institute of Digital Twin, China. He is also affiliated with the Dept. of Computer Science and Electrical Engineering, West Virginia University, USA. His research interests include computer vision, biometrics, machine learning, and multimedia. He is an AE of several journals, including IEEE Trans. on Affective Computing. He received the North Carolina State Award for Excellence in Innovation in 2008, New Researcher of the Year (2010-2011), and Outstanding Researcher (2017-2018, 2013-2014) at CEMR, WVU. He was selected the “People’s Hero of the Week” by BSJB under Minority Media and Telecommunications Council (MMTC) in 2013. His papers were selected as “The Best of FG’13” and “The Best of FG’15”, respectively, and the “Best Paper Award” by the IEEE Biometrics Council in 2022.



Fangyuan Wang received the M.Sc. degree in software engineering from Zhejiang Sci-Tech University, China, in 2022. He is currently pursuing the Ph.D. degree in mechanical engineering at The Hong Kong Polytechnic University, Hong Kong, which is conducted in collaboration with the Eastern Institute of Technology, China. His research interests focus on embodied AI, reinforcement learning, multi-agent systems, and robotic manipulation.



Anqing Duan received his Ph.D. degree in robotics from the Italian Institute of Technology (IIT) and the University of Genoa (UniGe), Italy, in 2021. He was a research associate at The Hong Kong Polytechnic University (PolyU). He is currently a visiting assistant professor with the Robotics Department at Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI). His research interests include robot learning and control with a focus on human-centered and healthcare robotic applications.



Peng Zhou received his Ph.D. degree in robotics from PolyU, Hong Kong, in 2022. In 2021, he was a visiting Ph.D. student at KTH Royal Institute of Technology, Stockholm, Sweden. He is currently a Research Officer at the Centre for Transformative Garment Production and a Postdoctoral Research Fellow at The University of Hong Kong. His research interests include deformable object manipulation, robot reasoning and learning, and task and motion planning.



Chenguang Yang (Fellow, IEEE) received the B.Eng. degree in measurement and control from Northwestern Polytechnical University, Xian, China, in 2005, and the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010. He performed postdoctoral studies in human robotics at the Imperial College London, London, U.K from 2009 to 2010. He is Chair in Robotics with Department of Computer Science, University of Liverpool, UK. He was awarded UK EPSRC UKRI Innovation Fellowship and individual EU Marie Curie International Incoming Fellowship. As the lead author, he won the IEEE Transactions on Robotics Best Paper Award (2012) and IEEE Transactions on Neural Networks and Learning Systems Outstanding Paper Award (2022). He is the Corresponding Co-Chair of IEEE Technical Committee on Collaborative Automation for Flexible Manufacturing. His research interest lies in human robot interaction and intelligent system design.



David Navarro-Alarcon (Senior Member, IEEE) received his Ph.D. degree in mechanical and automation engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2014. From 2015 to 2017, he was a Research Assistant Professor at the CUHK T Stone Robotics Institute. Since 2017, he has been with The Hong Kong Polytechnic University (PolyU), Hong Kong, where he is currently an Associate Professor with the Department of Mechanical Engineering. His current research interests include perceptual robotics and control theory. Dr. Navarro-Alarcon currently serves as an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS.