Learning to Plan & Schedule with Reinforcement-Learned Bimanual Robot Skills

Weikang Wan[1,2]* **Fabio Ramos**[1] **Xuning Yang**[1] **Caelan Garrett**[1]
[1]NVIDIA [2]University of California San Diego
{weikangw,ftozetoramos,xuningy,cgarrett}@nvidia.com

Abstract: Long-horizon contact-rich bimanual manipulation presents a significant challenge, requiring complex coordination involving a mixture of parallel execution and sequential collaboration between arms. In this paper, we introduce a hierarchical framework that frames this challenge as an integrated skill planning & *scheduling* problem, going beyond purely sequential decision-making to support simultaneous skill invocation. Our approach is built upon a library of single-arm and bimanual primitive skills, each trained using Reinforcement Learning (RL) in GPU-accelerated simulation. We then train a Transformer on a dataset of skill compositions to act as a high-level planner & scheduler, simultaneously predicting the discrete schedule of skills as well as their continuous parameters. We demonstrate that our method achieves higher success rates on complex, contact-rich tasks than end-to-end RL approaches and produces more efficient, coordinated behaviors than traditional sequential-only planners.

1 Introduction

Enabling robots to perform long-horizon, contact-rich manipulation is a longstanding goal in robotics, with bimanual systems offering the potential for human-like dexterity [1, 2]. A primary challenge in this domain lies in programming two arms to act in harmony to accomplish a complex goal. Such coordination demands a flexible control strategy that combines parallel, serial, and collaborative execution of skills.

Many previous approaches model manipulation as a sequential decision-making process, where a policy selects a single action or skill at each step [3, 4, 5]. This formulation, however, creates an inherent bottleneck for bimanual tasks, as it fails to capture opportunities for simultaneous execution and can lead to inefficient, underutilized behaviors. In this paper, we argue that the high-level decision problem is better framed as an integrated planning and scheduling problem, where the goal is to assign tasks to both arms, where at some points in time the arms act independently and at other points they act collaboratively.

To this end, we propose a hierarchical framework built upon a library of reinforcement-learned primitive skills. Our Transformer-based high-level policy functions as a skill scheduler, generating plans that specify the discrete skills and their continuous parameters for both arms. We highlight three contributions of this work:

- We propose a novel approach for bimanual manipulation that learns a library of arm-specific primitive skills using Reinforcement Learning (RL) and then through integrated skill planning & *scheduling* combines these in serial and parallel over time.
- We show how a Transformer-based scheduling policy can be trained to generate bimanual schedules that specify both discrete skills and their continuous parameters.
- We demonstrate through experiments that our approach achieves significantly higher success rates and efficiency compared to end-to-end and sequential planning baselines.

*Work done while interning at NVIDIA Research.

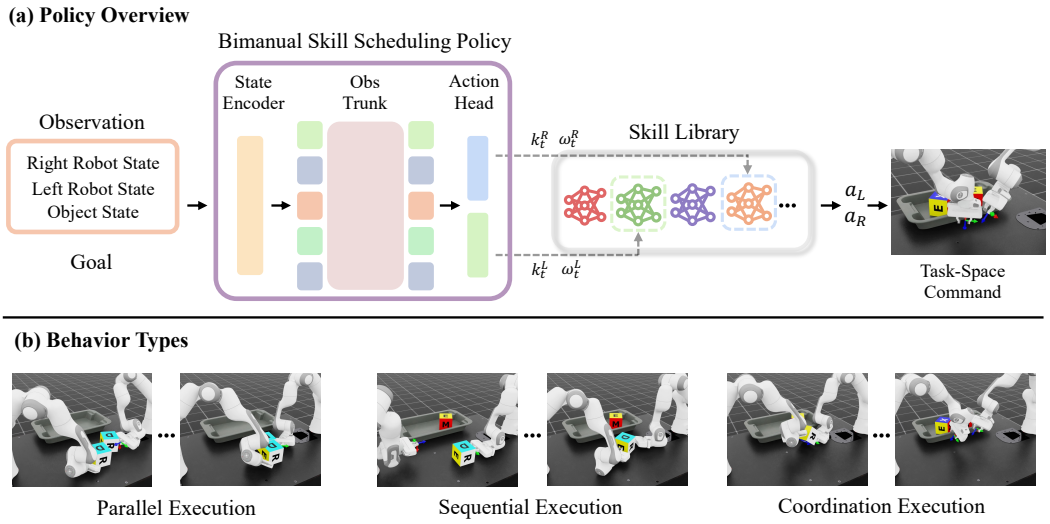

Figure 1: **(a) Policy Overview.** In our hierarchical framework, the high-level policy selects skills by predicting the skill index k^R, k^L and specifying the skill parameters ω^R, ω^L for both arms. **(b) Behavior Types.** Our method exhibits diverse bimanual behaviors in long-horizon tasks.

2 Related Work

Task and Motion Planning. Task and Motion Planning (TAMP) [6, 7, 8, 9, 10, 11] algorithms plan sequences of high-level actions and corresponding low-level motion plans, typically for single-robot scenarios. Multi-robot manipulation and TAMP [12, 13, 14] extend these principles to plan for multiple robots or arms and introduce opportunities for parallelism, but the jump to multiple robots and introduces significant complexity in coordination, collision avoidance, and task scheduling. Classical TAMP approaches rely on predefined models and constraints, which can be difficult to engineer for tasks requiring non-prehensile manipulation. Several recent learning for TAMP approaches relax this assumption by replacing handcrafted model components with learned ones [15, 16, 17, 5]. Others use imitation learning to plan [18, 19] and implicitly learn a planning model from demonstrations. Our approach also uses imitation learning for high-level reasoning but predicts bimanual behaviors, scheduling which skills should be used and when, both in serial and in parallel.

Hierarchical Modeling in Robotic Manipulation. Hierarchical manipulation methods [20, 21] build complex manipulation behaviors from a set of learned primitive control policies. Recent works use imitation [22, 23, 24] or reinforcement learning [25, 26, 27] to acquire these primitive policies, adding flexibility while benefiting from the temporal abstraction of primitives. However, these methods are often limited to single-arm scenarios and confined to prehensile manipulation primitives. In contrast, our method employs a set of versatile parameterized contact-rich single-arm and bimanual primitives for bimanual manipulation tasks.

3 Method

An overview of our pipeline is shown in Fig. 1. We first define the problem formulation in Sec. 3.1. We then introduce the details of the individual low-level skill training and the high-level bimanual skill scheduling policy in Sec. 3.2 and Sec. 3.3.

3.1 Problem Formulation

We formulate the long-horizon bimanual manipulation task as a two-level hierarchical decision problem. We first define the low-level MDP for learning individual skills and then describe the high-level MDP for scheduling these skills to achieve a final task goal $g \in \mathcal{G}$. Each primitive skill k from a finite library $\mathcal{K}$ is learned as a low-level policy $\pi_k^{\mathrm{L}}(a_t \mid s_t, \omega_t)$. This policy solves a short-horizon,

goal-conditioned Markov Decision Process (MDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}_k, \gamma, \Omega \rangle$. Here, $s_t \in \mathcal{S}$ represents the state at time t, $a_t \in \mathcal{A}$ is the low-level action, $\mathcal{T}$ is the state transition function, and $\gamma \in [0, 1)$ is the discount factor. The goal for this MDP is a skill parameter $\omega_t \in \Omega$ (e.g., a target object pose), and the policy is rewarded by a dense reward function $\mathcal{R}_k(s_t, a_t, \omega_t)$ for making progress towards this subgoal ω_t. Building on the skill library, the high-level problem is to select the skill k_t and its parameters ω_t to achieve the overall task goal g. This is governed by a high-level scheduling policy $\pi^{\mathrm{H}}(k_t, \omega_t \mid s_t, g)$. This high-level MDP is related to a semi-MDP [25], where each low-level skill acts as a temporally-extended action, similar to that of an option [28]. Each high-level "action" (k_t, ω_t) is executed by the corresponding low-level policy $\pi^{\mathrm{L}}_{k_t}$ for multiple timesteps until termination. The final, factorized policy for the complete task is thus: $\pi(a_t \mid s_t, g) = \pi^{\mathrm{H}}(k_t, \omega_t \mid s_t, g)\ \pi^{\mathrm{L}}_{k_t}(a_t \mid s_t, \omega_t)$.

3.2 Individual Low-Level Skills

We train a set of primitive single-arm and bimanual skills $\pi^{\mathrm{L}}_k(a_t \mid s_t, \omega_t)$ using goal-conditioned reinforcement learning. Each low-level skill policy takes as input the current state s_t and its parameters ω_t. For a single arm, s_t is comprised of the arm's proprioception (including joint positions, velocities, and end-effector pose), other arm's end-effector pose, and the object state. The parameters ω_t encode task-specific information such as the object goal pose. The policy outputs a target end-effector pose action a_t, which is then converted to joint torques via an Operational Space Controller (OSC) [29].

The low-level primitive skills are trained with dense rewards that combine a contact reward, task-relevant goal reaching reward, a binary success bonus, and an energy penalty. We use a unified template $r_t = r_{\text{contact}} + r_{\text{goal}} + r_{\text{suc}} - c_{\text{energy}}$, where the task success reward $r_{\text{suc}} = \mathbf{1}_{\text{suc}}$ is given when the pose of the object is within 0.05 m and 0.1 radians of the target pose. We add dense shaping $r_{\text{contact}} = \alpha_c(1 - \tanh(d_c/\sigma_c))$ and $r_{\text{goal}} = r_{\text{pos}} + r_{\text{rot}} = \alpha_p(1 - \tanh(d_p/\sigma_p)) + \alpha_r(1 - \tanh(d_r/\sigma_r))$, where d_c is the right end-effector–to–object distance, d_p is the object–to–goal position error, and d_r is the object–to–goal rotation error; $\alpha_c, \alpha_p, \alpha_r > 0$ are scaling coefficients and σ_c=0.2, σ_p=0.05, σ_r=10 are shaping scales; c_{energy} denotes an energy penalty: $c_{\text{energy}} = c_e \sum_{i=1}^{J} \tau_i\, \dot{q}_i$, where $c_e \in \mathbb{R}^+$ is a scaling coefficient, and τ_i and $\dot{q}_i$ are the joint torque and velocity of the ith joint.

In this work, we focus on five primitive skills: single-arm pushing, single-arm rotating, bimanual rotating, bimanual pushing, and bimanual pick-and-place. These skills were selected by human modelers to cover the fundamental capabilities required by our tasks. For all skills, the goal is specified by an object target position, an orientation, or full pose. Our approach is not limited to these skills and can in principle be applied to more than two robots.

3.3 Bimanual Skill Scheduling Policy

After the primitive skills are trained, they can be composed to complete difficult long-horizon tasks. We learn the high-level bimanual skill scheduling policy π^{H} through behavior cloning [30]. The goal is to train a policy that mimics the output of an expert planner that has access to privileged information used to generate the problem. The learned policy $\pi^{\mathrm{H}}(\omega_t, k_t \mid s_t, g)$ takes in the current observation and produces two types of outputs: a discrete skill k_t for each arm and the continuous skill parameters ω_t for the selected skill, which in this work, is a goal object pose.

Given the task goal g and current state information s_t, the high-level skill scheduling policy decides which skill and which ω_t to execute for each arm. To understand the task progress, π^{H} perceives the current object state, the task goal, and the proprioception states of both arms. The skill scheduling policy utilizes a transformer-based architecture that takes a sequence of recent observations as input, which aims to improve the temporal consistency of the policy's predictions. As an example, for the task of placing two tabletop objects into a grey bin in Fig. 2, the skill scheduling policy first calls pushing skills for both arms at the same time to move each object toward a suitable central

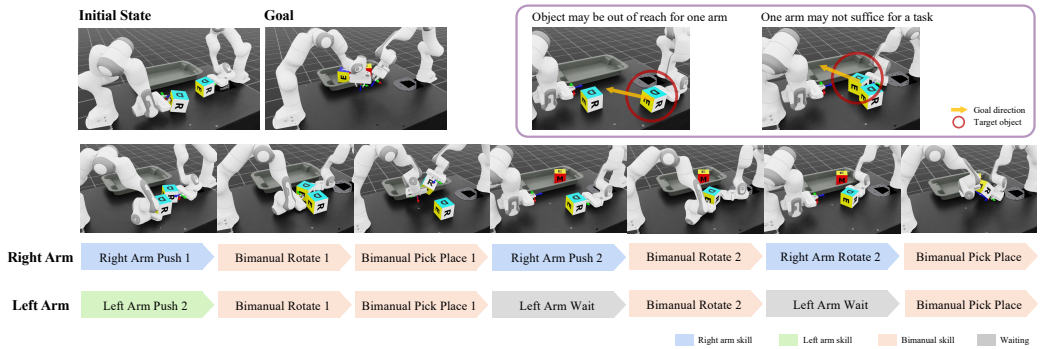

Figure 2: **Evaluation Visualization.** The long-horizon rearranging task presents diverse scenarios, such as objects being out of reach or requiring bimanual effort. Our policy handles these by planning the division of labor and collaboration between arms and correctly sequencing the necessary skills.

position, and then sequentially invokes bimanual rotating and bimanual pick-and-place skills on the two objects with appropriate skill parameters to complete the task.

3.4 Data Generation.

To train the skill scheduling policy, we first generate a dataset of expert demonstrations. Specifically, we implement a custom demonstration generator that has access to privileged problem information. This generator programmatically constructs multiple successful skill sequences, each being an ordered composition of low-level skills. For each sequence, it samples skill parameters ω and executes the full plan, retaining only successful rollouts. During data generation, we additionally define a single-arm *waiting* skill that keeps one arm stationary. For example, for the bimanual rotating skill we sample the object goal rotation within a specified range. This data generation method yields diverse successful sequence data, covering different ways of bimanual division of labor, asynchronous execution, and collaboration.

3.5 Policy Learning.

Once we have collected the required skill-sequence data, we train a Transformer-based bimanual skill scheduling policy $\pi^{\mathrm{H}}(\omega_t, k_t \mid s_t, g)$. The training supervision for π^{H} comes from the skill-index labels and skill parameters for both arms produced by the data-generation pipeline. We use a cross-entropy loss for skill-index prediction and an Mean Squared Error (MSE) loss for skill-parameter regression; the objective is $\mathcal{L} = \mathbb{E}_t\big[\sum_{u\in\{L,R\}} \mathrm{CE}(k^*_{t,u}, \hat{k}_{t,u}) + \lambda_\omega \left\|\omega^*_{t,u} - \hat{\omega}_{t,u}\right\|_2^2\big]$, where λ_ω balances the two terms.

4 Experiments

We evaluate our approach with simulated experiments to answer the following questions: **Q1.** Does our hierarchical framework achieve higher success rates on long-horizon bimanual manipulation tasks compared to flat end-to-end approaches? **Q2.** Can our bimanual skill scheduling policy effectively generate valid schedules that sequence both discrete skills and their continuous parameters for two arms simultaneously? **Q3.** Does our planing & scheduling formulation lead to more efficient and coordinated bimanual behaviors?

4.1 Experiments Setup

We evaluate our framework on a long-horizon, contact-rich manipulation task with goal that: "*the objects are in the bin*". This task involves placing one or two bulky objects into a bin, where the objects are too large for a single arm to grasp, lift, or reorient alone, which necessitates non-prehensile manipulation. Furthermore, an object's initial position may be out of reach for one arm. Conse-

quently, this task demands a dynamic combination of strategies: from synchronized coordination for collaborative lifting to independent, asynchronous execution for repositioning. The variability in initial object layouts requires the policy to plan and schedule different skill sequences for each scenario. We build this environment and conduct all simulation experiments in IsaacLab [31, 32]. To compare different approaches, we use three metrics: task success rate (SR) and task completion progress (CP), which is defined as the percentage of optimally-ordered task stages completed, and episode duration (ED) to measure completion efficiency.

4.2 Quantitative Results

Baselines. We compare our proposed framework against several representative baselines and ablations. We consider the following baselines: 1) RL-scratch is vanilla PPO [33] algorithm which learns the entire task from scratch. 2) Hierarchical RL (HRL) [26] is a hierarchical approach where a high-level policy selects the pre-trained low-level skills using RL. 3) Sequential-only planning [15] is a baseline that can only select one skill at each decision step, thus enforcing sequential, non-parallel execution. 4) Ours (Single Arm Ablation) is an ablation of our method constrained to use only a single arm. For all hierarchical methods, we use the same library of pre-trained low-level skills.

Method	**Rearrange One Object**			**Rearrange Two Objects**		
	SR (%) ↑	CP (%) ↑	ED (s) ↓	SR (%) ↑	CP (%) ↑	ED (s) ↓
RL-Scratch	0.0 ± 0.0	26.4 ± 2.2	10.0 ± 0.0	0.0 ± 0.0	12.3 ± 1.7	20.0 ± 0.0
Hierarchical RL	20.0 ± 5.5	42.2 ± 6.1	9.1 ± 0.3	7.4 ± 3.2	19.8 ± 2.0	19.3 ± 0.3
Sequential Planning	48.2 ± 3.0	$\mathbf{66.5 \pm 3.7}$	7.6 ± 0.2	32.4 ± 3.2	47.9 ± 3.2	17.1 ± 0.2
Ours (Single Arm)	0.0 ± 0.0	32.1 ± 2.5	10.0 ± 0.0	0.0 ± 0.0	14.0 ± 1.6	20.0 ± 0.0
Ours	$\mathbf{51.3 \pm 2.0}$	65.2 ± 3.8	$\mathbf{6.4 \pm 0.2}$	$\mathbf{38.7 \pm 2.4}$	$\mathbf{56.1 \pm 4.8}$	$\mathbf{14.3 \pm 0.2}$

Table 1: Comparison of our method against baselines. We report the mean and standard deviation of the Success Rate (SR), Completion Progress (CP), and Episode Duration (ED) over 100 rollouts.

Results and analysis. Table 1 provides comprehensive results for all methods on two task variations: 1) rearranging one object and 2) rearranging two objects into a bin. It answers **Q1** and **Q2** by showing that our method outperforms the RL-scratch baseline, achieving a 45% higher Success Rate (SR) and 41% higher Task Completion Progress (CP). Additionally, we visualize the bimanual skill schedule during evaluation in Fig. 2, showing that our bimanual skill scheduling policy not only selects the correct single-arm or bimanual skills in different situations, but also efficiently sequences skills for both arms to complete the task. We answer **Q3** by comparing our method with Sequential Planning, which results in a 16% reduction in Episode Duration (ED) and highlights our method's ability to more efficiently plan and schedule for both arms simultaneously. Policy rollout videos are available at `https://www.youtube.com/watch?v=3DI_yTml3J4`.

5 Conclusion

In this work, we introduced a hierarchical framework that addresses long-horizon bimanual manipulation by formulating it as an integrated skill planning & scheduling problem. Our method utilizes a Transformer-based policy to generate coordinated plans, simultaneously selecting discrete skills and regressing their continuous parameters. Experimental results validate that our approach achieves much higher success rates than end-to-end RL and produces more efficient, parallelized behaviors than planners restricted to sequential actions.

References

[1] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous systems*, 60(10):1340–1353, 2012.

[2] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin. Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups. *IEEE transactions on automation science and engineering*, 18(3):888–901, 2021.

[3] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

[4] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7477–7484. IEEE, 2022.

[5] B. Jiang, Y. Wu, W. Zhou, C. Paxton, and D. Held. Hacman++: Spatially-grounded motion primitives for manipulation. *arXiv preprint arXiv:2407.08585*, 2024.

[6] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[7] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE international conference on robotics and automation*, pages 1470–1477. IEEE, 2011.

[8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and systems*, volume 12, page 00052. Ann Arbor, MI, USA, 2016.

[9] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[10] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020.

[11] W. Shen, C. Garrett, N. Kumar, A. Goyal, T. Hermans, L. P. Kaelbling, T. Lozano-Pérez, and F. Ramos. Differentiable gpu-parallelized task and motion planning. 2024.

[12] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 945–952. IEEE, 1994.

[13] K. Harada, T. Tsuji, and J.-P. Laumond. A manipulation motion planner for dual-arm industrial manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 928–934. IEEE, 2014.

[14] P. Huang, R. Liu, C. Liu, and J. Li. Apex-mr: Multi-robot asynchronous planning and execution for cooperative assembly. *arXiv preprint arXiv:2503.15836*, 2025.

[15] A. Simeonov, Y. Du, B. Kim, F. Hogan, J. Tenenbaum, P. Agrawal, and A. Rodriguez. A long horizon planning framework for manipulating rigid pointcloud objects. In *Conference on Robot Learning*, pages 1582–1601. PMLR, 2021.

[16] J. Liang, X. Cheng, and O. Kroemer. Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation. *arXiv preprint arXiv:2206.12728*, 2022.

[17] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6541–6548. Ieee, 2021.

[18] Z. Yang, C. R. Garrett, T. Lozano-Perez, L. Kaelbling, and D. Fox. Sequence-Based Plan Feasibility Prediction for Efficient Task and Motion Planning. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.061.

[19] M. Dalal, A. Mandlekar, C. R. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=QNPuJZyhFE.

[20] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6-7):866–894, 2021.

[21] B. Hedegaard, Z. Yang, Y. Wei, A. Jaafar, S. Tellex, G. Konidaris, and N. Shah. Beyond task and motion planning: Hierarchical robot planning with general-purpose policies. *arXiv preprint arXiv:2504.17901*, 2025.

[22] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE international conference on robotics and automation*, pages 763–768. IEEE, 2009.

[23] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[24] A. Mandlekar, C. R. Garrett, D. Xu, and D. Fox. Human-in-the-loop task and motion planning for imitation learning. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=G_FEL3OkiR.

[25] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[26] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[27] Z. Zhou, A. Garg, D. Fox, C. R. Garrett, and A. Mandlekar. SPIRE: Synergistic planning, imitation, and reinforcement learning for long-horizon manipulation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=cvUXoou8iz.

[28] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5*, pages 212–223. Springer, 2002.

[29] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[30] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[31] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[32] NVIDIA. Isaac Sim. URL https://github.com/isaac-sim/IsaacSim.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.