ARCH: Hierarchical Hybrid Learning for Long-Horizon Contact-Rich Robotic Assembly

Jiankai Sun[1] **Aidan Curtis**[2] **Yang You**[1] **Yan Xu**[3] **Michael Koehle**[4] **Qianzhong Chen**[1]
Suning Huang[1] **Leonidas Guibas**[1] **Sachin Chitta**[4] **Mac Schwager**[1] **Hui Li**[4]
[1]Stanford University [2]MIT [3]University of Michigan [4]Autodesk Research
{jksun@,yangyou@,qchen23@,suning@,schwager@,guibas@cs.}stanford.edu
curtisa@mit.edu yxumich@umich.edu
{michael.koehle,sachin.chitta,hui.xylo.li}@autodesk.com

Abstract: Generalizable long-horizon robotic assembly requires reasoning at multiple levels of abstraction. While end-to-end imitation learning (IL) is a promising approach, it typically requires large amounts of expert demonstration data and often struggles to achieve the high precision demanded by assembly tasks. Reinforcement learning (RL) approaches, on the other hand, have shown some success in high-precision assembly, but suffer from sample inefficiency, which limits their effectiveness in long-horizon tasks. To address these challenges, we propose a hierarchical modular approach, named **A**daptive **R**obotic **C**ompositional **H**ierarchy (**ARCH**), which enables long-horizon, high-precision robotic assembly in contact-rich settings. **ARCH** employs a hierarchical planning framework, including a low-level primitive library of parameterized skills and a high-level policy. The low-level primitive library includes essential skills for assembly tasks, such as grasping and inserting. These primitives consist of both RL and model-based policies. The high-level policy, learned via IL from a handful of demonstrations, without the need for teleoperation, selects the appropriate primitive skills. We extensively evaluate our approach in simulation and on a real robotic manipulation platform. We show that **ARCH** generalizes well to unseen objects and outperforms baseline methods in terms of success rate and data efficiency. More details are available at: https://long-horizon-assembly.github.io.

Keywords: Robotic Assembly, Long-horizon Assembly, Hybrid Learning

1 Introduction

The manufacturing industry is increasingly turning to robotic systems for assembly tasks, driven by the need for greater precision, consistency, and efficiency [1, 2]. Nevertheless, long-horizon, contact-rich, and high-precision robotic assembly tasks continue to pose significant challenges in robotic automation, as these tasks demand sophisticated learning and object interaction capabilities that go beyond traditional approaches [3]. Current industrial robotic applications to assembly and manufacturing are largely engineered for a specific task and struggle with adapting to varied assembly scenarios and component variations. Ideally, such systems should be capable of handling high-precision contact-rich operations [4] and generalizing to long-horizon new scenarios from limited demonstrations [5] using multimodal inputs such as visual and force-torque feedback [6, 7].

Toward this goal, recent advances in learning-based solutions have started to tackle some of these problems. End-to-end imitation learning (IL) from human demonstrations has made encouraging progress [8, 9, 10], but usually requires a significant amount of expert demonstration data [11], which can be costly to collect, and often struggles with high-precision tasks. While reinforcement learning (RL) can drive specific assembly operations, such as high-precision insertion, it usually falls

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

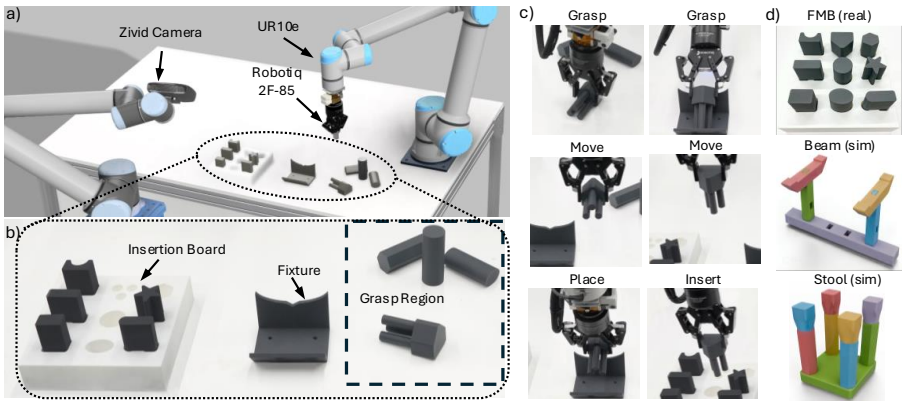


Figure 1: a) Workcell setup: a camera captures RGB-D images while a robot equipped with a force-torque sensor is used for the manipulation tasks. b) Task example: objects are randomly placed within the Grasp Region on the table. The robot must insert them into the correct receptacle in the Insertion Board, adjusting their orientation with the Fixture if necessary. c) Examples of the library of RL and model-based action primitives. d) Evaluation tasks: FMB Assembly, 5-part Beam Assembly, and 9-part Stool Assembly.

short on more complicated long-horizon tasks [12, 13, 14], where training faces challenges of sparse rewards and an exploding sampling space due to the larger long-horizon footprint. Alternatively, naively sequencing atomic actions or behaviors can lead to compounding errors, diminishing overall system performance [15].

To address these challenges, we propose a hierarchical approach for robotic assembly, especially targeted at long-horizon, contact-rich, high-precision settings. Our method adopts a hierarchical framework that consists of a library of parameterized low-level primitive skills (e.g., grasp, insert) and a high-level policy that sequences these primitives based on task context. To build robust and adaptable low-level primitives, we leverage both classical motion planning (MP) algorithms and RL policies. This hybrid MP-RL strategy enables the system to leverage prior knowledge for efficient execution of routine motions while remaining flexible enough to handle contact-rich and uncertain dynamics. The RL components are trained in simulation using domain randomization, enabling feasible transfer to real-world execution. At the high level, the policy operates over a compact action space, enabling efficient learning from a handful of human demonstrations via imitation learning, thereby alleviating the data collection burden common in end-to-end imitation learning pipelines.

In summary, our main contributions are as follows:

- We introduce **A**daptive **R**obotic **C**ompositional **H**ierarchy (**ARCH**), a hierarchical framework designed to tackle the challenging problem of *long-horizon robotic assembly*.
- We develop a hybrid low-level skill library that includes both MP algorithms and RL policies for efficient, adaptive, and high-precision behaviors in contact-rich assembly scenarios. Additionally, we design a high-level policy based on a Diffusion Transformer (DiT), trained via imitation learning with only a handful of demonstrations, to efficiently sequence and deploy these primitive skills.
- We evaluate our framework on three long-horizon robotic assembly tasks, each involving the assembly of 4 to 9 objects. Experiments in both simulation and the real world show that, despite being trained on a single object, our approach generalizes to unseen objects and outperforms baseline methods in terms of both success rate and data efficiency.

2 Related Work

2.1 Task and Motion Planning

Task and Motion Planning (TAMP) [16, 17, 18] is an effective approach for long-horizon manipulation problems as it can resolve temporally dependent constraints through hybrid symbolic-

continuous reasoning [19, 20]. These plans may involve regrasping [21], clearing obstructing objects [22], or moving to gather information [23]. Despite these abilities, such methods typically require hand-designed symbolic transition functions and continuous parameter samplers. Additionally, TAMP methods are computationally expensive, which limits their ability to handle failures in dynamic tasks. We argue that the symbolic transition functions and samplers can be replaced by a learnable high-level policy to address the issues mentioned above.

2.2 Learning for Long-Horizon Manipulation

There is a large body of research in learning for long-horizon manipulation tasks. Diffusion policies [8, 24, 25] have been shown to be a powerful tool for addressing complex manipulation tasks by leveraging their generative and multi-modal capabilities. However, they require a large amount of training data, i.e., human demonstrations, to learn effective end-to-end policies for complex, long-horizon tasks. Therefore, a significant amount of research has adopted the divide-and-conquer mindset and tackled the task by decomposing it into easier and reusable subtasks, instead of learning an entire task with a single policy.

Skill decomposition and chaining [26, 27, 28, 29, 30, 31] is a promising way to synthesize long-horizon and complex behaviors by sequentially chaining previously learned simpler skills through transition functions. Mishra et al. [32] trains individual skill diffusion models as action primitives and combine them at test time, when learned distributions of the skills are linearly chained to solve for a long-horizon goal during evaluation. Mao et al. [33] learns primitive skills from human demonstrations based on haptic feedback and segments a long-horizon task into a sequence of skills for dexterous manipulation. Chen et al. [34] introduces a bi-directional optimization framework that chains RL-trained sub-policies together using a transition feasibility function for long-horizon dexterous manipulation.

2.3 Hierarchical Modeling in Robotics

Hierarchical approaches typically offer policies at varying abstraction levels [35, 36, 37, 38, 39, 40]. Xian et al. [41] and Ma et al. [42] propose to learn a high-level policy from demonstrations to predict end-effector key poses and a low-level diffusion-based trajectory generator for connecting these key poses to achieve the final goal. MimicPlay [5] learns a high-level plan from human videos of manipulating objects and low-level visuomotor controls from teleoperated demonstrations on the real robot. MAPLE [43] enhances standard RL algorithms by incorporating a predefined library of behavioral primitives. The most relevant work to our approach uses hierarchical policy decomposition for a multi-stage cable routing task [44]. They define both scripted and imitation-learned low-level primitives, and train a high-level policy to select among them. In contrast, we incorporate both model-based and model-free low-level primitives, and learn a high-level policy that selects and instantiates parameterized primitives. Our RL policy for the contact-rich insertion primitive is trained in simulation and transferred to the real world via zero-shot sim-to-real deployment.

3 Method

In this work, we model the long-horizon robot assembly task as a parameterized-action Markov decision process (PAMDP) [45]. A PAMDP problem consists of the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ where $\mathcal{S}$ is the continuous state space, $\mathcal{A}$ is a set of discrete action primitives $\{a_1, a_2, ..., a_k\}$, each with m_a parameters $X_a \subseteq \mathbb{R}^{m_a}$, $P(s, a, s')$ is the probability of transitioning to state s' when taking action primitive a in state s, $R(s, a)$ is the reward from taking action primitive a in state s, and γ is the discount factor. Each parameterized action primitive a can be instantiated as either a model-based policy or an RL policy. Our goal in a PAMDP is to find a policy $\pi(a, x|s)$ that minimizes the discrepancy between the learned policy and the expert demonstrations that maximize reward under the PAMDP.

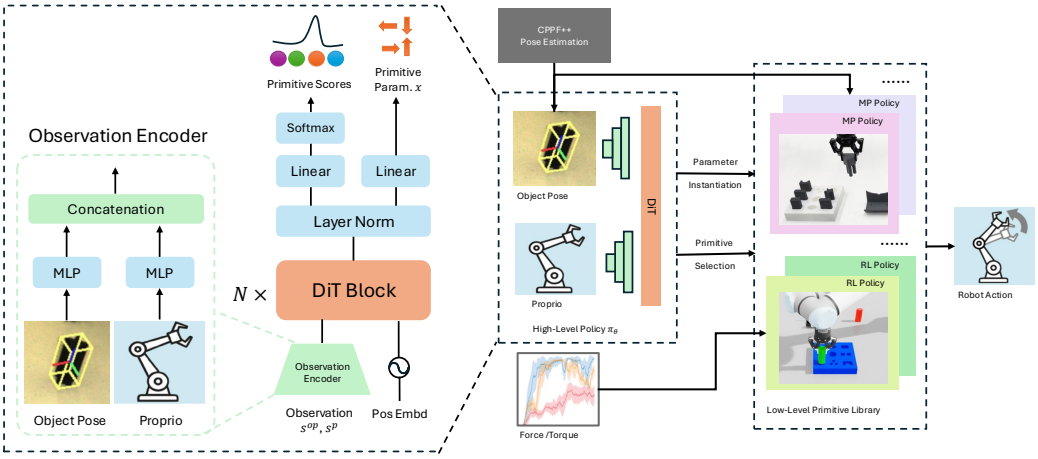


Figure 2: We propose a hierarchical framework for long-horizon robotic assembly. The high-level policy π_θ, trained via imitation learning, takes as input the object pose from a pose estimator and the robot's proprioceptive state, and outputs a categorical distribution over low-level primitives and their parameters. The selected primitive is then executed by a low-level policy, which is either reinforcement learning (RL)-based or motion planning (MP)-based. Specifically, we train RL policies in simulation for contact-rich skills, such as 'insert', while MP policies are used for primitives operating in free space, such as 'move'.

Solving a PAMDP involves not only selecting the best discrete primitive but also optimizing over the parameter space for each primitive at each state. In this work, we fix a set of primitives that are building blocks of manipulation for assembly tasks (See Section 3.1) and attempt to learn the high-level policy $\pi_\theta(a, x|s)$ as a neural network with parameters θ. This results in a hierarchical framework, shown in Figure 2. In this section, we describe our assumptions, the low-level primitive library, the high-level policy architecture, the data collection pipeline, and the pose estimation module.

We make the following assumptions in this work: (1) the CAD models of the assembly parts are known, as is common in industrial settings; (2) the base component of each assembly, as well as the fixture used for object reorientation, is fixed to the table. Under these assumptions, a single pose estimation step—combined with the known CAD models—is sufficient to determine the target insertion poses for each object with minimal error. Note that assumption (2) can be relaxed by performing continuous pose estimation, i.e., repeatedly estimating object poses throughout the task to account for potential shifts or disturbances.

3.1 Low-level Primitive Library: Modeling Basic Skills with Motion Planning and Reinforcement Learning

We focus on providing agents with a library of flexible primitives that act as foundational components for high-precision, contact-rich robotic assembly tasks. The hierarchical decision-making system operates independently of the specific implementations of these primitives, which may include either closed-loop, learning-based skills or model-based motion planners. Regardless of how they function internally, it is essential that the primitives are adaptable to varying behaviors, which is why we introduce parameters to customize each primitive. These parameters typically have clear semantics, such as an object index or a 6-DoF end-effector pose for a grasping primitive. While these primitives offer flexibility, we acknowledge that they are not exhaustive and can be extended.

`GRASP`(o, p_g): The robot moves its end-effector to pre-grasp pose $p_g \in \mathbf{SE}(3)$ and closes its gripper to grasp object o. A motion planner is used for its execution.

PLACE(o, p_p): The robot has object o in grasp and moves its end-effector to pre-place pose $p_p \in \mathbf{SE}(3)$, and opens its gripper to place object o. A motion planner is used for its execution.

MOVE(g_m): The robot moves its end-effector to pose $g_m \in \mathbf{SE}(3)$. A motion planner is used for its execution.

INSERT(o, g): The robot has object o in grasp and inserts it into its goal pose $g \in \mathbf{SE}(3)$. An RL policy $\pi_L(\phi)$ is employed for two primary reasons: first, insertion goal poses are subject to inaccuracies from pose estimation errors; second, insertion tasks inherently involve complex, contact-rich interactions between the object and its receptacle. We train an RL policy in simulation [46] using PPO [47]. Its observation space includes end-effector (EE), force-torque (FT), and EE pose relative to the goal pose; its action is EE velocity u_t; and its reward $\tilde{r}(s_t^p, u_t, g)$ is the negative distance between the current pose s_t^p and the goal pose g.

The objective of the goal-conditioned INSERT primitive is to reach the goal pose g that maximizes the expectation of the cumulative return, as Equation 1 shows:

$$\mathcal{J}(\phi) = \mathbb{E}\left[\sum_t \gamma^t \tilde{r}(s_t^p, u_t, g)\right]. \tag{1}$$

3.2 High-level Policy: Composing Primitives via Imitation

As shown in Figure 2, the high-level policy $\pi_\theta(a, x|s)$ takes pose information obtained from pose estimation (Section 3.4) for a given object o and robot proprioception as inputs s, to select the appropriate primitive a from the low-level primitive library and instantiates it with parameters x for low-level control. Parameters x contains the object index o and pose information obtained from pose estimation. We collect human demonstration data (Section 3.3) and train the high-level policy via imitation learning.

We modify the Diffusion Transformer (DiT) [48] architecture as the backbone of our high-level policy, considering that its strong sequential data handling capability is suitable for handling a history of previous states and actions. The DiT outputs softmax scores for each primitive and the continuous action parameter for the primitive with the highest score. These two functions are supervised using cross-entropy loss and MSE loss, respectively. We employ DiT blocks with adaptive LayerNorm-Zero conditioning. We use the Robomimic observation encoder [49] to extract features from object pose (s^{op}) and proprioceptive pose (s^p). The goal of imitation learning is to find a parameterized policy π_θ that can maximize the likelihood function based on the currently collected demonstration data $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{x})\}$:

$$\theta = \arg\max_\theta \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{x}\sim\mathcal{D}} \pi_\theta(\mathbf{a}, \mathbf{x}|\mathbf{s}). \tag{2}$$

3.3 Data Collection

As our low-level primitives are implemented with either RL policies or MP policies, no demonstration data is needed. In this section, we focus on the data collection process for the high-level IL policy. Our hierarchical framework abstracts away the motion details from the demonstrator and allows them to demonstrate by selecting primitives and parameters instead of teleoperation. More specifically, from the library of pre-defined action primitives (Section 3.1), a human demonstrator uses a keyboard to select the most appropriate primitive and its parameters to complete the assembly task at hand. For example, to initiate the GRASP primitive of object o, the demonstrator selects the primitive index and the object index. The pose estimation module (Section 3.4) estimates the object pose p_o and the continuous parameter, pre-grasp pose p_g, is calculated by adding a fixed offset from p_o. p_g is then passed to the motion planner to execute the primitive. To initiate the INSERT primitive of object o, the demonstrator selects the primitive index and the object index. As described in the assumptions, the insertion goal pose g for each object is known with a small margin of error. A pre-trained RL policy is used with g as its goal pose.

The dataset denoted as $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{x})\}$, consists of sensor observations $\mathbf{s}$, primitive indexes $\mathbf{a}$, and corresponding primitive parameters $\mathbf{x}$. The dataset consists of two types of demonstrations: "successful" and "recovery" trials. In half of the trials, the demonstrator successfully inserts the object, while in the other half, failures occur, followed by recovery actions demonstrated by the operator. These recovery trials are crucial to making the learned policy more robust to errors.

In summary, we allow the demonstrator to select the discrete primitive and parameter to execute while the continuous input parameter comes from pose estimation. This is a novel way to collect demonstration data as it is often difficult and time-consuming to collect teleoperated demonstrations for high-precision tasks.

3.4 Pose Estimation

As shown in Figure 2, object pose is needed by the high-level policy and also serves as the input parameter to the low-level primitive skills, such as `grasp`. We integrate the pose estimation method CPPF++ [50, 51], which demonstrates strong generalization from simulation to real-world scenarios. Specifically, given a 2D RGB-D image $\mathcal{I}$, the model produces an accurate 6D pose $\xi \in \mathfrak{se}(3)$, where the first three components represent rotation and the last three correspond to translation. While the original CPPF++ is designed for category-level pose estimation, we adapt it for instance-level pose estimation by using distinct CAD models as different "categories". To enhance precision, inspired by the Iterative Closest Point (ICP) [52] algorithm, we propose a post-optimization procedure conducted in the tangent space of the Lie group.

Specifically, we compute the one-way Chamfer distance from the object-masked point cloud to the full CAD model, transformed by the current pose ξ:

$$L_{CD}(\xi) = \sum_i \min_j \|T_\xi(p_i) - p_j^*\|_2, \tag{3}$$

where p_j^* is the j-th point on the back-projected point cloud obtained from the predicted masks, p_i is the i-th point on the object mesh, and T_ξ is the rigid transformation that aligns the canonical object point with the scene, defined as $T_\xi(p_i) = R \cdot p_i + t$. We utilize the one-way Chamfer distance due to self-occlusion in the observation and optimize ξ iteratively using the Lietorch [53] library to obtain a refined pose. Figure 4 in the Appendix presents some qualitative pose detection results. The high accuracy of the pose estimator enables us to have a high success rate in terms of both the low-level MP policies and the high-level policy.

4 Experiments

4.1 Task Description and Setup

Recent advancements in robotic learning have also highlighted the need for benchmarks that effectively balance generalization and the complexity of manipulation tasks. The Functional Manipulation Benchmark (FMB) [54] aims to bridge this gap by defining functional manipulation as a series of relevant behaviors, such as grasping, repositioning, and physically interacting with objects. Building on FMB, we want to extend its emphasis on contact-rich dynamics and object diversity. For these reasons, we use the multi-peg assembly task introduced in FMB, along with two more challenging custom-designed multi-part assembly tasks, to showcase the generalizability of our approach.

We focus on three assembly tasks: FMB Multi-peg Assembly in [54], 5-part Beam Assembly, and 9-part Stool Assembly, as shown in Figure 1. FMB Assembly is tested on the real robot, whereas the other two assemblies are in simulation only. For FMB Assembly, 9 objects of different shapes must be inserted into the board. The robot must grasp the object that is randomly placed on the table, may need to reorient and regrasp it using the fixture depending on its initial pose, and then insert it into the board. More task and setup details can be found in the Appendix. We evaluate single-step and long-horizon categories and examine generalization to unseen objects and tasks.

Table 1: Success rate SR(%) of the multi-stage FMB Assembly of Hexagon, Beam Assembly, and Stool Assembly, comparing our method with baseline methods after training with 10 demonstrations, with the human oracle being the upper bound.

Methods	FMB (real)		Beam (sim)		Stool (sim)	
Metrics	SR (%) ↑	SPL ↑	SR (%) ↑	SPL ↑	SR (%) ↑	SPL ↑
E2E RL [47]	0	0.00	0	0.00	0	0.00
E2E IL [8]	0	0.00	0	0.00	0	0.00
MimicPlay [5]	20	0.16	15	0.12	10	0.08
Luo et al. [44]	25	0.19	15	0.14	20	0.18
ARCH (Ours)	55	0.51	55	0.51	45	0.44
Human Oracle [54]	65	0.55	60	0.62	50	0.49

4.2 Baselines

We evaluate two end-to-end (E2E) baselines: 1) **E2E IL**: Diffusion Policy (DP) [8], a goal-conditioned imitation learning framework that leverages a diffusion model for generating diverse action trajectories. Teleoperated demonstration data is collected in the real world. 2) **E2E RL**: PPO [47], a widely-used RL method that optimizes policy performance through proximal updates, balancing exploration and exploitation while ensuring stable training. Training is conducted in the IsaacLab simulation environment [46].

We also evaluate two hierarchical baselines: 1) **MimicPlay** [5] learns high-level latent plans from human play data to guide low-level visuomotor control. Teleoperated demonstration data is collected in the real world. 2) **Luo et al.** [44] propose a Multi-Stage Cable Routing approach through hierarchical imitation learning. Teleoperated demonstration data for both low-level and high-level policies is collected in the real world. We adapt the above methods to our task.

4.3 Evaluation Metrics

A total of 20 trials are conducted for each task. For each trial, the selected object is initialized randomly in the grasp region. For the `GRASP` primitive to succeed, the object must be securely gripped by the gripper without falling after being lifted. For the `INSERT` primitive to succeed, the object must be fully inserted into the corresponding receptacle.

The main metric is the overall task success rate (%). We found that failure primarily stems from grasping and insertion. Hence, we also introduce two metrics: "% Grasped" and "% Inserted", to measure the success rates of these two primitives.

Success Rate (SR%): percentage of successful completion of the long-horizon tasks.

Success weighted by Path Length (SPL):

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \cdot \frac{l_i}{\max(p_i, l_i)}, \tag{4}$$

where N is the total number of trials, S_i is a binary indicator of success in trial i, l_i is the shortest path (number of primitives) to the goal, and p_i is the actual path length taken by the agent.

% Grasped: percentage of successful grasps as a single-stage task.

% Inserted: percentage of successful insertions as a single-stage task.

4.4 Experiment Analysis

Comparative Performance Analysis. First, we compare with end-to-end baselines and hierarchical approaches. For methods that utilize a low-level primitive library, we ensure they remain identical across all methods to ensure a fair comparison. As shown in Table 1, long-horizon tasks

Table 2: Success rate SR(%) of the multi-stage FMB Assembly task after training with 10 demonstrations and success rate of the single stages by object. Our system demonstrates the ability to generalize to unseen objects.

	Object	SR (%)	% Grasped	% Inserted
Seen	Hexagon	50	80	75
Unseen	Star	50	85	60
	SquareCircle	35	75	50
	3Prong	80	90	90
	Circle	75	95	80
	Oval	55	85	70
	Arch	40	65	65
	DoubleSquare	40	70	60
	Rectangle	65	90	75

often involve sparse rewards, which complicates RL training. Consequently, **E2E RL** [47] fails at such long-horizon assembly tasks. With diffusion models, **E2E IL** [8] falls short in handling high-precision tasks and requires a large number of expert demonstrations. The hierarchical baselines, **MimicPlay** [5] and **Luo et al.** [44], achieve better success rate than the E2E methods, but struggle with contact-rich steps such as insertion due to the small number of demonstrations available. Finally, **Human Oracle** is not a baseline but serves as the upper bound for high-level policy. The human oracle selects primitives with near-flawless accuracy and can self-correct mistakes. Failures with this method are due to pose estimation and grasp errors. **ARCH** demonstrates significantly higher success rates than baseline methods due to our hierarchical framework and the hybrid MP-RL design of the low-level library. The advantage of **ARCH** becomes more pronounced on the more challenging Beam Assembly and Stool Assembly tasks compared to baseline methods. By combining model-based and learning-based components, our approach achieves both high precision and flexibility. Failures with our approach are mainly due to inaccuracies in pose estimation and limitations in the RL policies. Since pose estimation and insertion policies are extensively studied areas, numerous established methods [55, 7] exist to enhance performance; however, exploring these improvements is beyond the scope of this work.

Generalization to Unseen Objects. Our system has been trained exclusively with the Hexagon object, both for the RL primitive and for the high-level policy. Table 2 shows that our system generalizes well to unseen objects without any fine-tuning. Objects with a narrow edge, such as the SquareCircle and DoubleSquare, present more challenge during the grasping stage. Additionally, certain object shapes, such as the Star, SquareCircle, and Arch, present more challenge to pose estimation.

Data Efficiency. As Table 1 shows, **ARCH** achieves high success rates with merely 10 demonstrations, whereas baseline methods would require significantly more data. This capability not only reduces the overall data collection burden but also enhances the practical applicability of our method in real-world scenarios, where data acquisition can be a limiting factor.

Robustness. Our high-level policy exhibits robustness to single-stage failures and human disturbances via retries. For example, we observed that when the grasp primitive fails, the policy automatically triggers another grasp attempt using the new pose estimate. This failure recovery capability contributes to the higher success rate of **ARCH**.

5 Conclusion

In this paper, we introduce a hierarchical hybrid learning system for long-horizon contact-rich robotic assembly. It includes a low-level parameterized skill library and an imitation-learned high-level policy for selecting and composing primitive skills. The hierarchical structure and the pre-trained primitive skills enable our system to be data efficient for long-horizon tasks, while satisfying the high-precision requirement of assembly.

6 Limitations

Cross-Task Generalization. Although the learned high-level policy transfers across novel objects within a given task, it remains task-specific: separate training is still required when moving from, for example, FMB Assembly to Stool Assembly. However, training a high-level policy for a new task requires only a handful of demonstrations and no teleoperation, keeping the adaptation effort minimal.

Object Identification Pipeline. All objects are manually labeled and indexed prior to execution. Replacing this offline step with a perception module—e.g., a vision-language model–based detector that auto-assigns semantic IDs—would streamline this process and improve overall efficiency.

Scalability and Generality of the Primitive Library. Our approach assumes that tasks can be decomposed into a finite set of parameterized primitives. This assumption may pose scalability challenges for tasks that require fundamentally new skills. However, in practice, we observe that a relatively small library of primitives—automatically discovered from demonstrations—generalizes effectively across a wide range of long-horizon tasks. Moreover, in industrial settings, tasks generally require only a small set of primitives. In addition, our framework is modular and can readily integrate newly introduced primitives when needed, making it naturally extensible to novel tasks.

References

[1] Y. Cohen, H. Naseraldin, A. Chaudhuri, and F. Pilati. Assembly systems in industry 4.0 era: a road map to understand assembly 4.0. *The International Journal of Advanced Manufacturing Technology*, 105:4037–4054, 2019.

[2] H. Christensen, N. Amato, H. Yanco, M. Mataric, H. Choset, A. Drobnis, K. Goldberg, J. Grizzle, G. Hager, J. Hollerbach, et al. A roadmap for us robotics–from internet to robotics 2020 edition. *Foundations and Trends® in Robotics*, 8(4):307–424, 2021.

[3] M. Ghallab, D. Nau, and P. Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[4] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana. Deep reinforcement learning for high precision assembly tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 819–825, 2017. doi:10.1109/IROS.2017.8202244.

[5] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[6] A. S. Morgan, B. Wen, J. Liang, A. Boularias, A. M. Dollar, and K. Bekris. Vision-driven compliant manipulation for reliable, high-precision assembly tasks. *arXiv preprint arXiv:2106.14070*, 2021.

[7] X. Zhang, M. Tomizuka, and H. Li. Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion. *ICRA*, 2024.

[8] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[9] Y. Zang, P. Wang, F. Zha, W. Guo, C. Zheng, and L. Sun. Peg-in-hole assembly skill imitation learning method based on promps under task geometric representation. *Frontiers in Neurorobotics*, 17:1320251, 2023.

[10] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada. Robotic imitation of human assembly skills using hybrid trajectory and force learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11278–11284. IEEE, 2021.

[11] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[12] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5548–5555. IEEE, 2020.

[13] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3080–3087. IEEE, 2019.

[14] J. Sun, L. Yu, P. Dong, B. Lu, and B. Zhou. Adversarial inverse reinforcement learning with self-attention dynamics model. *IEEE Robotics and Automation Letters*, 6(2):1880–1886, 2021.

[15] J. Sun, D.-A. Huang, B. Lu, Y.-H. Liu, B. Zhou, and A. Garg. Plate: Visually-grounded planning with transformers in procedural tasks. *IEEE Robotics and Automation Letters*, 7(2): 4924–4930, 2022.

[16] L. P. Kaelbling and T. Lozano-Perez. Hierarchical task and motion planning in the now. *ICRA*, 2011.

[17] L. P. Kaelbling and T. Lozano-Perez. Pre-image backchaining in belief space for mobile manipulation. *Robotics Research*, 2017.

[18] 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(1):265–293, 2021.

[19] J. Sun, H. Sun, T. Han, and B. Zhou. Neuro-symbolic program search for autonomous driving decision module design. In *Conference on Robot Learning*, pages 21–30. PMLR, 2021.

[20] J. Huang, S. Xie, J. Sun, Q. Ma, C. Liu, D. Lin, and B. Zhou. Learning a decision module by imitating driver's control behaviors. In *Conference on Robot Learning*, pages 1–10. PMLR, 2021.

[21] A. Adu-Bredu, N. Devraj, and O. C. Jenkins. Optimal constrained task planning as mixed integer programming, 2022. URL https://arxiv.org/abs/2211.09632.

[22] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. *CoRR*, abs/2108.04145, 2021. URL https://arxiv.org/abs/2108.04145.

[23] A. Curtis, G. Matheos, N. Gothoskar, V. Mansinghka, J. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Partially observable task and motion planning with uncertainty and risk awareness, 2024. URL https://arxiv.org/abs/2403.10454.

[24] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[25] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[26] G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *In Advances in Neural Information Processing Systems*, 2009.

[27] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 2012.

[28] C. Agiaa, T. Migimatsu, J. Wu, and J. Bohg. Taps: Task-agnostic policy sequencing. *arXiv preprint arXiv:2210.12250*, 2022.

[29] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim. Composing complex skills by learning transition policies. *In International Conference on Learning Representations*, 2019.

[30] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *CoRL*, 2021.

[31] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics*, 2018.

[32] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. *CoRL*, 2023.

[33] X. Mao, G. Giudici, C. Coppola, K. Althoefer, I. Farkhatdinov, Z. Li, and L. Jamone. Dexskills: Skill segmentation using haptic data for learning autonomous long-horizon robotic manipulation tasks. *arXiv preprint arXiv:2405.03476*, 2024.

[34] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *CoRL*, 2023.

[35] T. Gao, S. Nasiriany, H. Liu, Q. Yang, and Y. Zhu. Prime: Scaffolding manipulation tasks with behavior primitives for data-efficient imitation learning. *IEEE Robotics and Automation Letters*, 2024.

[36] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34: 21847–21859, 2021.

[37] S.-M. Yang, M. Magnusson, J. A. Stork, and T. Stoyanov. Learning extrinsic dexterity with parameterized manipulation primitives. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5404–5410. IEEE, 2024.

[38] J. Sun, R. Liu, and B. Zhou. Hiabp: Hierarchical initialized abp for unsupervised representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9747–9755, 2021.

[39] Z. Ye, J. Chen, J. Light, Y. Wang, J. Sun, M. Schwager, P. Torr, G. Li, Y. Chen, K. Yang, et al. Reasoning in reasoning: A hierarchical framework for better and faster neural theorem proving. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

[40] W. Zhang, S. Qiao, L. Luo, Y. Li, C. Zheng, Q. Xu, M. Li, Y. Gui, Y. He, J. Qiu, et al. Synapseroute: An auto-route switching framework on dual-state large language model. *arXiv preprint arXiv:2507.02822*, 2025.

[41] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. *CoRL*, 2023.

[42] X. Ma, S. Patidar, I. Haughton, and S. James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. *CVPR*, 2024.

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

[44] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine. Multi-stage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 2024.

[45] W. Masson and G. D. Konidaris. Reinforcement learning with parameterized actions. *CoRR*, abs/1509.01644, 2015. URL http://arxiv.org/abs/1509.01644.

[46] 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.

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

[48] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[49] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[50] Y. You, R. Shi, W. Wang, and C. Lu. Cppf: Towards robust category-level 9d pose estimation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6866–6875, 2022.

[51] Y. You, W. He, J. Liu, H. Xiong, W. Wang, and C. Lu. Cppf++: Uncertainty-aware sim2real object pose estimation by vote aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[52] K. S. Arun, T. K. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

[53] Z. Teed and J. Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10338–10347, 2021.

[54] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. Fmb: A functional manipulation benchmark for generalizable robotic learning. *arXiv preprint arXiv:2401.08553*, 2024.

[55] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.

[56] R. Bohlin and L. Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 521–528 vol.1, 2000. doi:10.1109/ROBOT.2000.844107.

A Appendix

A.1 Task Descriptions

In addition to FMB Assembly, we introduce two more challenging long-horizon assembly tasks, both of which involve multi-stage dependencies. These dependencies make the tasks more difficult—for instance, the feet can only be inserted after the legs are correctly assembled. If the legs are not properly inserted, subsequent steps will fail.

FMB Assembly: This task involves assembling 9 distinct objects with varying shapes. For each object, the robot must first grasp it, then perform a sequence of reorientation actions using an environment fixture, followed by an insertion into the board. The time horizon for each object ranges from 20 to 40 seconds.

5-part Beam Assembly: The base part is fixed to the table. The task involves inserting two legs into the base, followed by inserting two feet onto the legs. The time horizon for each object ranges from 20 to 40 seconds.

9-part Stool Assembly: As shown in Figure 3, the base is also fixed to the table. This task involves inserting four legs into the base, and then inserting four feet onto the legs. The increased number of objects and stages makes this task significantly more challenging and long-horizon. The time horizon for each object ranges from 20 to 40 seconds.

For visualizations of the initial and final states of two new task, Beam Assembly and Stool Assembly, please refer to Figure 3.

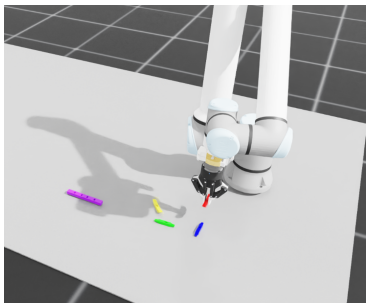
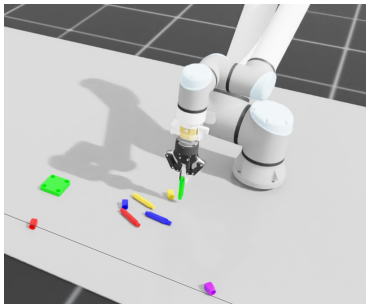
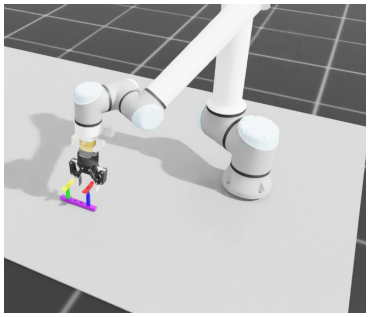
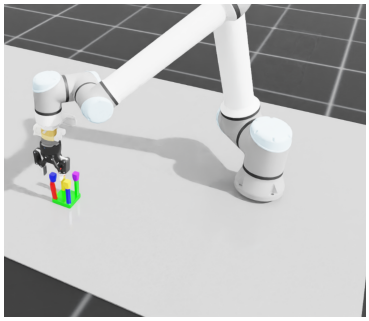
Task Name	5-part Beam Assembly	9-part Stool Assembly
Initial State		
Final State		

Figure 3: Simulation setup for Beam Assembly and Stool Assembly in IsaacLab.

A.2 Experiment Details

Real Workcell Setup. The real workcell setup includes a UR10e robotic arm and a third-person view Zivid camera (Figure 1). The simulation platform for RL training is built on the IsaacLab en-

gine [46], while motion-level manipulation plans are computed using lazyPRM [56]. Model training is performed on an Ubuntu machine equipped with an Intel i7 CPU and a GeForce RTX 4090 GPU. The Beam Assembly and the Stool Assembly are evaluated in simulation.

Hyperparameters and Computation. For pose estimation, we use 512×512 RGB-D image as input. For high-level policy π_θ, We adopt the Diffusion Transformer model [48] for predicting action feasibility score. Our RL primitive is trained using 1,000 parallel environments in IsaacLab. The maximum task horizon is 25,000 action steps, equivalent to 200 seconds of robot execution at a 125Hz control frequency.

For our DiT-based high-level policy, the number of reverse diffusion timesteps is a critical parameter that significantly impacts both the sampling time and the quality of the generated samples. We choose 128 diffusion steps, which provide the best balance for most tasks. Additional hyperparameters for the DiT network are listed in Table 3.

A.3 Pose Estimation Results

Accurate pose estimation is essential for enabling high-precision assembly. This is particularly true for the `GRASP` primitive in our low-level primitives library, where precise object localization directly affects execution success. Figure 4 presents some qualitative pose detection results. The high accuracy of the pose estimator enables us to have a high success rate in terms of both the low-level MP policies and the high-level policy.

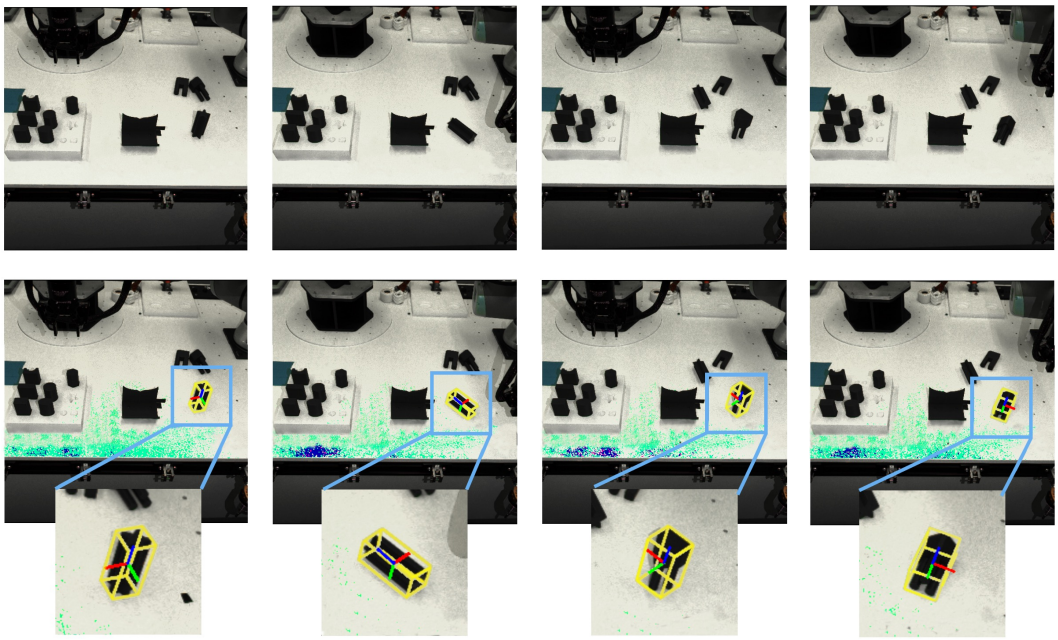


Figure 4: **Pose Estimation Examples**. Our method gives accurate pose estimation of differently shaped parts. The top row shows the input images, and the bottom row shows the pose estimation. Red, green, blue colors indicate the xyz axes in the canonical space, respectively.

A.4 Baseline Description

To evaluate the effectiveness of our proposed approach, we compare it with several adapted hierarchical baselines. However, due to differences in their design, it is challenging to directly apply their methods. In this section, we provide further details about these baselines, including their structure and the adaptations made to align them with our long-horizon assembly tasks.

Table 3: Hyperparameters for high-level policy

Hyper-parameter	Value
Hidden Dimension	128
Number of Blocks	4
Number of Heads	4
MLP Ratio	4
Dropout Prob	0.1

- **MimicPlay** [5]: Since we do not have access to human play data, we retain the hierarchical architecture of MimicPlay but train it using our own expert demonstrations. We modify the high-level policy to predict motion trajectories, and adapt the low-level policy to produce executable actions accordingly. This baseline does not incorporate low-level action primitives.
- **Luo et al.** [44]: We adopt the hierarchical structure from their framework. However, for this baseline, we construct the entire low-level primitive library using imitation learning rather than motion planning or reinforcement learning, which leads to primitives that are less robust and adaptable in diverse scenarios.