PRIME: Scaffolding Manipulation Tasks with Behavior Primitives for Data-Efficient Imitation Learning

Tian Gao[1*], Soroush Nasiriany[2], Huihan Liu[2], Quantao Yang[3*], Yuke Zhu[2]

***Abstract*—Imitation learning has shown great potential for enabling robots to acquire complex manipulation behaviors. However, these algorithms suffer from high sample complexity in long-horizon tasks, where compounding errors accumulate over the task horizons. We present PRIME (PRimitive-based IMitation with data Efficiency), a behavior primitive-based framework designed for improving the data efficiency of imitation learning. PRIME scaffolds robot tasks by decomposing task demonstrations into primitive sequences, followed by learning a high-level control policy to sequence primitives through imitation learning. Our experiments demonstrate that PRIME achieves a significant performance improvement in multi-stage manipulation tasks, with 10-34% higher success rates in simulation over state-of-the-art baselines and 20-48% on physical hardware.[1]**

***Index Terms*—Imitation Learning, Deep Learning in Grasping and Manipulation, Deep Learning Methods.**

I. Introduction

IMITATION learning (IL) has become a powerful paradigm for programming robots to perform manipulation tasks. Policies trained through imitation have exhibited diverse and complex behaviors, such as assembling parts [55], preparing coffee [57], making pizza [7], and folding cloth [52]. Deep IL methods aim at training policies that map sensory observations directly to low-level motor commands [4, 7, 27]. While conceptually simple, these methods usually require a large volume of human demonstrations, making them costly for tackling long-horizon tasks. Furthermore, the direct imitation of low-level motor actions leads to limited generalization abilities of the learned policy.

One solution to improve data efficiency and model generalization is incorporating temporal abstraction into policy learning [40]. Conventional methods afforded with temporal abstraction decouple learning new tasks into learning *what* subtasks to perform and *how* to achieve them. Among these methods, *skills* represent a popular form of temporal abstraction, offering a systematic approach to decomposing complex tasks for robots. Skills serve as fundamental building blocks, capturing necessary robot behaviors for specific tasks, such as grasping an object. The first step of incorporating skills as temporal abstraction is to obtain a repertoire of motor skills that capture *how* to perform behaviors, serving as reusable building blocks for various tasks. This reduces the problem of learning new tasks to learning *what* behaviors to perform rather than *how* to perform them, simplifying the learning process and enhancing generalization. The second step involves learning a policy for skill sequencing. To acquire skills, one popular approach is skill learning, which learns low-level skills that capture short-horizon sequences of robot actions by learning either continuous latent skill representations [1, 30, 37] or a discrete set of skills with continuous parameters [45, 58]. Prior work in skill learning extracts skills from a large amount of prior human data. While promising, a core limitation of these methods is the need for substantial human data to ensure the learned skills possess a high generalization capability.

Recent work has explored using robotic *behavior primitives* [5, 9, 21, 31] to decompose manipulation tasks, such as movement primitives [15, 32], motion planning [11, 20, 50], and grasping systems [2, 23]. A behavior primitive is a parameterized module designed to capture a certain movement pattern, usually with explicit semantic meaning (*e.g.*, grasping). The input parameters instantiate the behavior primitive into a specific movement, with the output being a sequence of motor actions to control the robot. These primitives enjoy the advantages of re-usability, modularity, and robustness toward variations. To utilize these primitives, recent work has

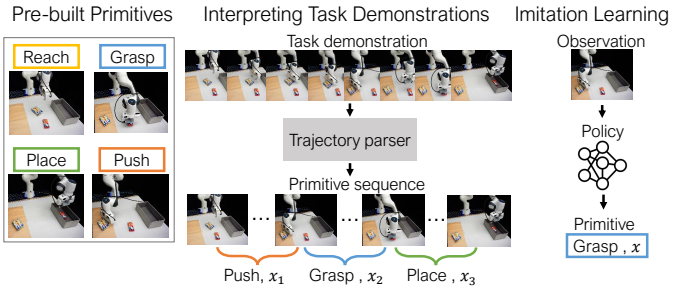

Fig. 1: **Overview of PRIME.** (Left) Our learning framework leverages a set of pre-built behavior primitives to scaffold manipulation tasks. (Middle) Given task demonstrations, we use a trajectory parser to parse each demonstration into a sequence of primitive types (such as "push", "grasp" and "place") and their corresponding parameters x_i. (Right) With these parsed sequences of primitives, we use imitation learning to acquire a policy capable of predicting primitive types (such as "grasp") and corresponding parameters x based on observations.

Manuscript received: March 1, 2024; Revised May 29, 2024; Accepted July 1, 2024.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments.

[1] Tian Gao is with the Department of Computer Science, Stanford University, tiangao@stanford.edu

[2]Soroush Nasiriany, Huihan Liu, and Yuke Zhu are with the Department of Computer Science, the University of Texas at Austin.

[3] Quantao Yang is with the Department of Computer Science, KTH Royal Institute of Technology.

* This work was done when Tian Gao and Quantao Yang were visiting researchers at UT Austin.

Digital Object Identifier (DOI): see top of this page.

[1]Additional materials are available at https://ut-austin-rpl.github.io/PRIME/

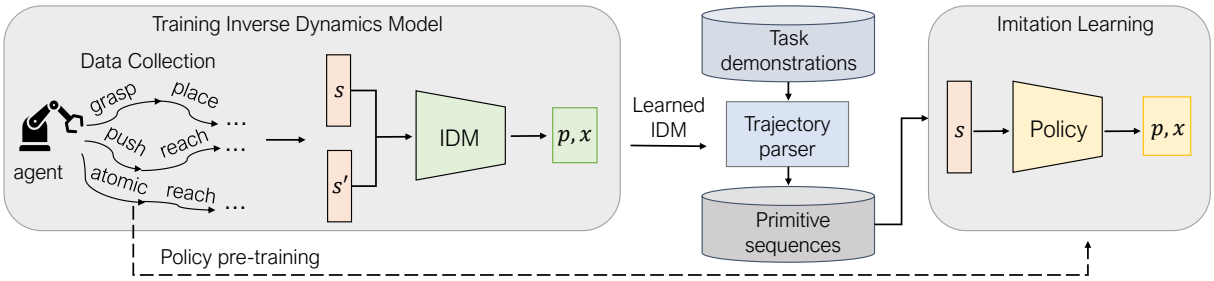

Fig. 2: **Method Overview.** We develop a self-supervised data collection procedure that randomly executes sequences of behavior primitives in the environment. With the generated dataset, we train an IDM that maps an initial state s and a final state s' from segments in task demonstrations to a primitive type p and corresponding parameters x. To derive the optimal primitive sequences, we build a trajectory parser capable of parsing task demonstrations into primitive sequences using the learned IDM. Finally, we train the policy using parsed primitive sequences.

proposed learning high-level policies that discover the optimal sequence of primitives using reinforcement learning (RL) [8, 9, 31]. However, RL requires expensive exploration even in the action space afforded by the primitives and is unsafe to train on real robots. Another notable line of work learns the policy with primitives from segmented demonstrations using imitation learning [13, 24, 47]. Segmented demonstrations require costly human efforts to manually segment demonstrations into primitive sequences.

In response to the above challenges, we introduce PRIME (PRimitive-based IMitation with data Efficiency), a data-efficient imitation learning framework based on behavior primitives (see Fig. 1). We provide a small set of task demonstrations with raw sensory observations and a discrete collection of behavior primitives. Our framework consists of a two-step learning process: first, parsing task demonstrations as primitive sequences via a trajectory parser without the need for any human annotations, and subsequently, training a policy through imitation learning to predict the sequence of primitives (such as "push" and "grasp") and their corresponding parameters given observations. By incorporating primitives, we break down long-horizon tasks into shorter sequences of primitives, significantly reducing the complexity and temporal horizon for imitation learning.

In this work, PRIME does not require access to segmentation labels, rendering the parsing of demonstrations a challenging task. To segment demonstrations into primitive sequences, it is essential to establish a mapping from raw observation-action sequences to primitives. To generate the data necessary for learning the mapping, we introduce a self-supervised data collection procedure [5, 39] that randomly samples sequences of primitives to execute within the environment, effectively reducing the need for human efforts in collecting prior data. Subsequently, we train an Inverse Dynamics Model (IDM) [3, 10, 34, 35, 56] on the collected data, which maps pairs of states to primitives. We use this IDM with a dynamic programming algorithm to identify the optimal primitive sequences derived from task demonstrations.

We evaluate our method's effectiveness in tabletop manipulation tasks, both in simulation and on real hardware. Our results highlight PRIME's substantial performance gains over state-of-the-art imitation learning baselines in a low-data regime. In simulations, success rates increase by 10.0% to 33.6%, and on real robots, by 20.0% to 48.3%. We further verify that our trajectory parser can effectively parse task demonstrations into primitive sequences that can be replayed to accomplish the task with success rates exceeding 90%. Moreover, our IDM generalizes to unseen environments, achieving performance levels comparable to those in training environments.

We highlight three contributions of this work: 1) We introduce PRIME, a data-efficient imitation learning framework that scaffolds robot tasks with behavior primitives; 2) We develop a trajectory parser that transforms task demonstrations into primitive sequences using dynamic programming without segmentation labels; 3) We validate the effectiveness of PRIME in simulation and on real hardware.

II. Related Work

A. Learning from Demonstration

Learning from Demonstration (LfD) has shown promise in robot manipulation tasks [6, 33, 36, 42]. LfD aims to enable an agent to observe and replicate expert behavior to effectively achieve a designated task. Within the domain of LfD, a diverse and extensive range of approaches has emerged, encompassing imitation learning [14, 25, 27], demonstration-guided RL [37, 38, 48], and offline RL [1, 18, 26, 53]. These approaches that rely on demonstration guidance often necessitate a substantial number of expert demonstrations, thereby limiting their data efficiency. To reduce the burdens of collecting expert demonstrations, a common line of work learns from task-agnostic play data [22, 29, 30], which is more cost-effective to acquire but still demands a certain level of human supervision. Instead of relying on additional human data, we propose utilizing pre-defined primitives and data acquired from random primitive rollouts.

B. Skill-based Imitation Learning

Skill-based imitation learning extracts low-level temporally-extended sensorimotor behaviors as skills from expert demonstrations [41, 54] or task-agnostic play data [22, 29, 30] and emulates the high-level behavior observed in the expert demonstrations to guide the execution of these low-level

skills. A common approach involves joint learning of low-level skills and a high-level policy, with skills acquired in an unsupervised manner [16, 17, 43, 44]. These skills can be either discrete [58] or continuous in a latent space [30, 37]. Unsupervised learning obviates the need for additional human annotation but often results in skills with limited reusability and low generalization capability. Alternatively, some research focuses on learning a high-level policy and low-level skills from structured demonstrations with additional segmentation labels using supervised learning, relying on either weak [47] or strong human supervision [13]. In our work, we use pre-built parameterized behavior primitives as low-level skills, which are highly robust, reusable, and generalizable. Furthermore, our method requires only raw sensory demonstrations without the need for additional human annotations.

C. Learning with Behavior Primitives

One line of research focuses on policy learning with primitives, which involves augmenting the motor action space through the integration of parameterized primitives [8, 9, 12, 19, 31, 49]. Dalal et al. [9] propose to manually specify a comprehensive library of robot action primitives. These primitives are carefully parameterized with arguments that are subsequently fine-tuned and learned by an RL policy. Similarly, Nasiriany et al. [31] augments standard RL algorithms by incorporating a pre-defined library of behavior primitives. Chitnis et al. [8] decomposes the learning process into learning a state-independent task schema. The discrete-continuous augmented action space imposes a significant exploration burden in RL. Recently, Chen et al. [5] proposed an imitation learning framework that integrates primitives for solving stowing tasks, which involves a complex graph construction for Graph Neural Networks to predict forward dynamics. Another tangential work by Shi et al. [46] decomposes demonstrations into sequences of waypoints, which are interpolated through linear motion. The interpolated linear motion between waypoints can be regarded as a type of primitive. In contrast, our primitive-based framework can be viewed as a more versatile form of waypoint extraction, capable of encompassing a broader spectrum of skills.

III. Method

We introduce PRIME, our primitive-based imitation learning framework, which decomposes complex, long-horizon tasks into concise, simple sequences of primitives. We begin by formulating the problem and providing an overview of our framework. We then describe two components of our framework: the trajectory parser and the policy.

A. Problem Formulation

We formulate a robot manipulation task as a Parameterized Action Markov Decision Process (PAMDP) [28], defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, p_0, \mathcal{R}, \gamma)$ representing the continuous state space $\mathcal{S}$, the discrete-continuous parameterized action space $\mathcal{A}$, the transition probability $\mathcal{P}$, the initial state distribution p_0, the reward function $\mathcal{R}$, and the discount factor γ. In our setting, the motor action space is afforded by the primitives into discrete-continuous primitive action space, $a = (p, x),\ a \in \mathcal{A},\ p \in \mathcal{L}, x \in \mathcal{X}_p$, where $\mathcal{L}$ is a discrete set of primitive types and $\mathcal{X}_p$ is the parameter space of primitive p. We aim to learn a policy, $\pi(a|s) = \pi(p, x|s),\ s \in \mathcal{S}$, to maximize the expected sum of discounted rewards. We assume access to a small set of task demonstrations for imitation learning.

In our framework, we first parse task demonstrations into concise primitive sequences to reduce the complexity and temporal horizon of imitation learning. The challenge in parsing is to map unsegmented demonstrations into sequences of parameterized primitive actions. To establish this mapping, we build an IDM capable of identifying segments of task demonstrations into primitives. Utilizing the learned IDM, we develop a trajectory parser that uses dynamic programming to determine the optimal primitive sequences derived from task demonstrations. Subsequently, we train a policy from parsed primitive sequences to compose primitives via imitation learning. By leveraging primitives, the policy only needs to focus on primitive selection and their parameters rather than low-level motor actions. See Fig. 2 for an overview of our framework.

B. Trajectory Parser

We develop a trajectory parser to parse task demonstrations into primitive sequences. This parser comprises an IDM and a dynamic programming algorithm. The IDM learns the probability of mapping from segments of task demonstrations to primitives. The dynamic programming algorithm determines the optimal primitive sequence by maximizing the product of probabilities in the parsed primitive sequences.

1) Inverse Dynamics Model: To parse a demonstration with primitives, it is important to identify behaviors shown in the demonstration that can be reproduced using a primitive. Toward this objective, we seek the initial and final states of behaviors and develop an IDM to infer primitives that can transition from a specified initial state to a targeted final state. We construct the IDM, $\text{IDM}(p, x|s, s')$, which predicts a primitive type p and its parameters x based on a pair of initial and final states (s, s'). The predicted primitive type falls within a categorical distribution that encompasses the types of primitives contained in the pre-built primitive set, in addition to an "other" category. Given that not all segments of a demonstration can be reproduced by a primitive within our pre-built primitive set, we introduce a new category named "other". This category is designated for classifying pairs of initial and final states that do not correspond to any of the predefined types of primitives. The predicted parameters belong to a continuous distribution, representing the parameter space associated with the primitive.

To collect the training dataset for the IDM, we introduce a self-supervised data collection procedure by randomly

executing primitives and atomic motor actions within the environment. Specifically, our random policy either uniformly samples a primitive type p from a pre-built set or selects a random atomic motor action, each with a 50% probability. If a primitive is sampled, its parameters x are randomly chosen and the primitive is executed. Otherwise, the sampled motor action is executed directly. This process yields a set of trajectories comprising the rollouts of these primitives. These primitive rollouts are subsequently utilized to train the IDM in a supervised manner. To gather data for the "other" category, we randomly sample a selection of state pairs from the generated trajectories. Specifically, we uniformly sample K state pairs in each episode and label them as "other" category, enabling us to balance the dataset by selecting an appropriate K (see Alg. 1). To further address data imbalance during training, we reweight the data for each primitive type p in the IDM training dataset by a factor of $1/(\text{number of rollouts for primitive type } p)$.

The collected dataset is an offline dataset and differs from online RL samples. The data collection policy we used is not task-specific, generating a domain-specific dataset. Consequently, all tasks within the same domain share a single IDM, requiring only one dataset collection procedure.

To enhance training data quality, we filter out unsuccessful rollouts using the success criteria for each primitive. For example, a successful grasping primitive is defined by the gripper securely holding an object. Without this filtering, many rollouts would include ineffective actions, degrading the accuracy and quality of the learned IDM and demonstration segmentation.

Gathering successful primitive rollouts through uniform parameter sampling is inefficient. To reduce the sampling burden and speed up data collection, we use a prior distribution of Gaussian mixtures centered on task objects. This directs the agent to interact more effectively with objects, improving rollout success rates. The pseudo-code for the data generation process is summarized in Alg. 1.

Algorithm 1: Self-Supervised Data Collection Procedure

Notations
  $\mathcal{D}$: training dataset of IDM
  $\mathcal{C}_p$: number of episodes during data generation
  $\mathcal{M}$: horizon of episodes
  $\mathcal{K}$: number of negative samples for each episode

for $e \leftarrow 1, 2, \cdots \mathcal{C}_p$ **do**
  for $i \leftarrow 1, 2, \cdots \mathcal{M}$ **do**
    Sample a primitive and its parameters (p^i, x_p^i) or an atomic motor action a_{t_i} and set $p^i =$ atomic
    Execute it and get $\tau^i = (s_{t_i}, a_{t_i}, s_{t_i+1}, ..., s_{t_{i+1}})$
    if $p^i \neq$ atomic **and** `is_success`(τ^i, p^i) **then**
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_{t_i}, s_{t_{i+1}}, p^i, x_p^i)\}$
    end if
  end for
  Sequence $\{\tau^i\}_{i=1}^{M}$ into an episode trajectory:
    Get $\tau = (s_0 = s_{t_0}, a_0, ..., s_{t_1}, a_{t_1}, ..., s_{t_2}, ..., s_{t_M})$
  for $k \leftarrow 1, 2, \cdots K$ **do**
    Sample a segment $\tau' = (s_j, a_j, ..., s_l)$ from τ
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_j, s_l, \texttt{other}, \texttt{none})\}$
  end for
end for

2) Dynamic Programming Algorithm: The learned IDM predicts a hybrid discrete-continuous distribution when provided with a pair of input initial and final states. In this context, the probability $\text{IDM}(p, x|s, s')$ represents the likelihood of interpreting a segment between states s and s' as belonging to the primitive type p and its parameter x. Once this mapping is established, the remaining task is to identify an optimal sequence of primitives that maximizes the probability of being parsed from the task demonstrations. To optimize the likelihood of primitive sequences interpreted from given task demonstrations, we leverage dynamic programming to find the optimal segmentation with a maximal product of probabilities in the parsed primitive sequences.

Specifically, considering a task demonstration denoted as $\tau = (s_0, a_0, ..., s_\mathcal{T})$ where a_t represents low-level motor actions. We define the objective function in our dynamic programming process as $f(i)$, which represents the probability of decomposing $(s_0, a_0, ..., s_i)$ into an optimal sequence of primitives $(s_0, (p_0, x_0), s_{j_1}, (p_1, x_1), ..., s_i)$. We iteratively update the objective function

$$f(i) = \max_{p,x,t<i} f(t) \cdot (\alpha \cdot \text{IDM}(p, x|s_t, s_i)) \tag{1}$$

by maximizing the product of probabilities of primitive sequences. We multiply a factor α to $\text{IDM}(p, x|s_t, s_i)$, where α is a small constant ($\alpha = 0.0001$ in our implementation) if p is in "other" category; otherwise, $\alpha = 1$. We use the factor α to penalize mappings to the category "other". Upon completing the dynamic programming, we can extract the optimal primitive sequences denoted as $(s_0, p_0, x_0, s_{t_1}, p_1, x_1, \ldots, s_{t_\mathcal{M}} = s_\mathcal{T})$ from the final value $f(\mathcal{T})$, where $\mathcal{M}$ represents the length of the segmented sequence.

C. Policy Learning

In this section, we describe the process of learning the policy from parsed primitive sequences. We train the policy $\pi(p, x|s)$ with behavioral cloning using the segmented primitive sequences.

Given that the segmented primitive sequences are notably shorter than the task demonstrations, leading to significantly smaller segmented data than the size of the demonstrations, we introduce a stepwise augmentation technique to enrich the segmented data and increase the scope of supervision for imitation learning. We assume that beginning from any point within the demonstration, the decomposition will remain consistent in subsequent segments. In other words, we presume that start and end points of each segment in parsed sequences are the same across all "suffixes" of demonstrations, where a suffix is defined as a portion of the demonstration beginning from any intermediate state and continuing to the final state.

With this assumption, for each segmented primitive sequences $\tau' = (s_0, (p_0, x_0), s_{t_1}, (p_1, x_1), ..., s_{\mathcal{T}})$ parsed from a task demonstration $\tau = (s_0, a_0, ..., s_{\mathcal{T}})$, we can get an augmented tuple $(s_l, p'_d, x'_d, s_{t_{d+1}})$ at each timestep l, where $s_{t_d} < l < s_{t_{d+1}}$, $(p'_d, x'_d) = \text{argmax}\, \text{IDM}(.|s_l, s_{t_{d+1}})$. We incorporate these augmented tuples $\{(s_l, p'_d, x'_d)\}$ into the training dataset of policy.

To leverage additional prior knowledge, we pretrain the policy using the training dataset of IDM and fine-tune the policy using parsed primitive sequences.

D. Implementation Details

We use the primitives from Nasiriany et al. [31] to implement our library of primitives with minor modifications. These task-independent, hard-coded APIs can be directly adapted to new situations within the same domain, as these APIs only require robot proprioceptive information as input. We implement the following four primitives:

- **Reaching:** The robot moves its end-effector to a target location (x, y, z) and yaw angle ψ in a collision-free path.
- **Grasping:** Same behavior and parameters as the reaching primitive, followed by the robot closing its gripper.
- **Placing:** Same behavior and parameters as the reaching primitive, followed by the robot opening its gripper.
- **Pushing:** The robot reaches a starting location (x, y, z) at a yaw angle ψ in a collision-free path and then moves its end-effector by a displacement $(\delta_x, \delta_y, \delta_z)$.

To reduce the complexity of the mapping, we factorize the IDM into a primitive IDM, i.e. $\text{IDM}_{\text{prim}}(p|s, s')$, capable of identifying primitive types and a parameter IDM, i.e. $\text{IDM}_{\text{param}}(x|s, s', p)$, capable of predicting parameters of the primitive, where

$$\text{IDM}(p, x|s, s') = \text{IDM}_{\text{prim}}(p|s, s') \cdot \text{IDM}_{\text{param}}(x|s, s', p). \quad (2)$$

Our primitive IDM and parameter IDM are two separate networks. Both networks take a pair of observations, s and s', and encode these observations with a pair of ResNet-18 encoders. A 2-layer MLP follows the image encoder in both networks. The primitive IDM outputs a Softmax distribution over primitives, while the parameter IDM outputs a Gaussian mixture model over parameter values. Our model architecture is similar to the default architecture in the RoboMimic framework [27], as the paper reported that ResNet encoder and GMM policy uniformly perform better than other design choices. Similarly to the IDM, we factorize the policy $\pi(p, x|s)$ into a primitive policy, i.e. $\pi_{\text{prim}}(p|s)$, and a parameter policy, i.e. $\pi_{\text{param}}(x|s, p)$. During deployment, the primitive policy first predicts the next primitive type p given the current state s. Following this, the parameter policy predicts the corresponding parameters x given inputs s and p. We train these two policy networks separately with behavioral cloning. These networks take a single observation as input (the current observation s) and have architectures and output spaces similar to their counterparts in the IDM. We use RoboMimic [27] to implement and train these networks.

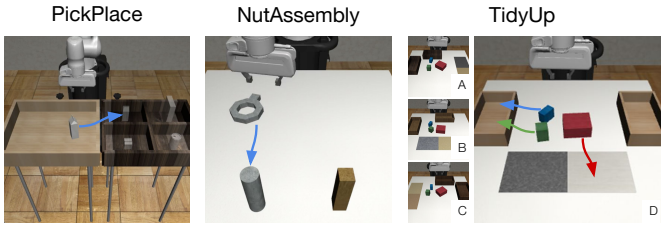

Fig. 3: **Simulated Tasks**. We perform evaluations on three tasks from the RoboSuite simulator [59]. The first two, `PickPlace` and `NutAssembly`, are from the RoboSuite benchmark, with `NutAssembly` featuring less initial randomization than the original task. We introduce a third task, `TidyUp`, to study long-horizon tasks and test the inverse dynamics model's generalization to unseen environments. We create four environment variants in this domain, denoted as (A, B, C, D). `TidyUp` task is designed in environment (D), and we collect human demonstrations for `TidyUp` in the same environment (D). To gauge the inverse dynamics model's generalization capability, we train two IDMs: IDM-D, based solely on data from environment (D), and IDM-ABC, trained on data from environments (A, B, C). While IDM-D is our default model for experiments, we use IDM-ABC to evaluate generalization in unseen environments.

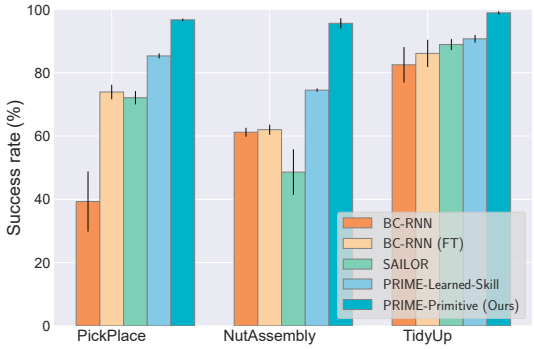

Fig. 4: **Quantitative evaluation in three simulated tasks.** Our method significantly outperforms state-of-the-art imitation learning approaches, with success rates surpassing 95% in all three tasks.

IV. Experiments

Our experiments are designed to answer the following questions: 1) How does PRIME perform for imitation learning in low-data regimes? 2) Which design choices are critical to PRIME? 3) How effective is the trajectory parser in PRIME? 4) Is PRIME feasible for practical deployment to real-world robot tasks? 5) Are learned skills compatible with PRIME?

A. Experimental Setup

1) Manipulation Tasks: We validate our approach and examine the above questions in simulated and real-world tasks. We perform evaluations in three simulated tasks from the RoboSuite simulator [59] (see Fig. 3) and two real-world tasks (see Fig. 6):

- **PickPlace**. The robot picks up a milk carton from the left and places it in the corresponding bin.
- **NutAssembly**. The robot picks up the nut and inserts it over the peg. The high precision of nut insertion is challenging when learning under a low-data regime.
- **TidyUp**. A new domain to study long-horizon tasks and generalization of IDM in unseen environments. The setup

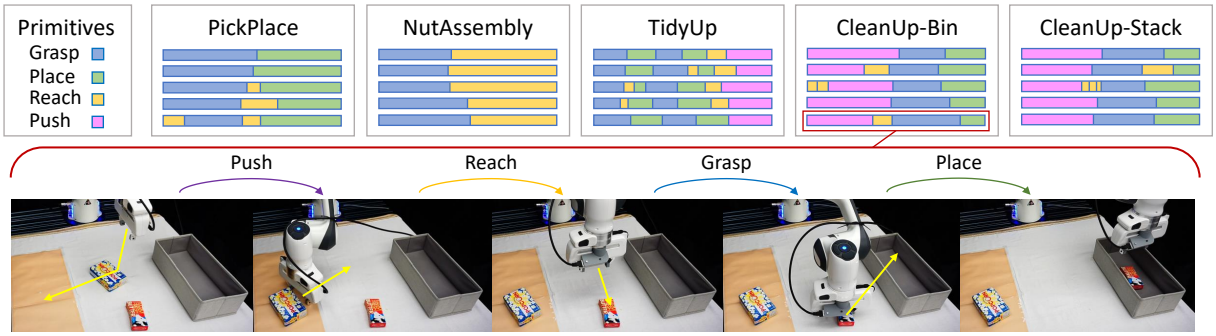

Fig. 5: **Visualization of output primitive sequences from trajectory parser.** For each task, we select five human demonstrations and visualize the segmented primitive sequences as interpreted by the trajectory parser.

includes a large box, 2 smaller boxes, 2 bins, and a table mat. Across four environment variants, objects differ in placement and texture. The robot's task is to move the large box onto the table mat and place the smaller boxes into the bins. See Fig. 3 for the detailed design.

- **CleanUp**. This domain replicates the real-world experimental setup introduced by Nasiriany et al. [31], involving two tasks: `CleanUp-Bin` and `CleanUp-Stack`. The tasks require the robot to push a popcorn box to a serving area and move butter to a target location. See Fig. 6 for the tasks.

We employ a Franka Emika Panda robot with operational space control for all tasks. In simulation and the real world, we opt for 5-DoF control at 20 Hz: 3 for end-effector position $(\delta_x, \delta_y, \delta_z)$, 1 for yaw orientation, and 1 for the gripper. The agent observes wrist-camera and third-person images and receives proprioceptive data. Vision-based policies are trained for all tasks, with only **30 human demonstrations** per task. For the training data of IDM, we collect 1M transitions for `PickPlace`, 3M transitions for `NutAssembly`, and 5M transitions for `TidyUp`. Notably, all tasks are completed based on the **same set of primitives**.

2) Evaluation Protocol: In the simulation, we run 50 trials for each checkpoint, averaging success rates from the top 5 checkpoints per seed. The policy is executed for 20 trials per checkpoint on the real robot. We choose the best checkpoint from each seed. We compute the mean and standard deviation of success rates over the three seeds.

3) Baselines: We contrast our method with three imitation learning baselines: BC-RNN [27], its fine-tuning variant BC-RNN (FT), and SAILOR [30]. BC-RNN uses LSTM as the backbone network architecture and learns through behavioral cloning. BC-RNN (FT) pre-trains on prior data and fine-tunes with target task demonstrations. SAILOR pre-trains a skill encoder to extract skill latents from prior data and learns the policy using task demonstrations and pre-trained skill representations through imitation learning. The prior data for pre-training BC-RNN (FT) and SAILOR consists of all low-level transitions in the training dataset of IDM in PRIME. PRIME-Learned-Skill runs our framework PRIME with pretrained primitives which learn from human motions by first manually segmenting the demonstrations into sequences of primitives, then training each primitive using the data from its corresponding segments. The set of primitive types remains the same as the hard-coded primitives: {reach, grasp, place, push}. To make a fair comparison, we use the default network architecture in RoboMimic [27] to implement BC-RNN, using ResNet-18 as the image encoder and a GMM policy, similar to our method. SAILOR utilizes a VAE architecture for skill encoding and decoding and also uses ResNet-18 and a GMM.

TABLE I: Success rates in simulation tasks for ablation studies.

Task	Ours	No Pretraining	Greedy Algo
`PickPlace`	$\mathbf{0.967 \pm 0.004}$	0.463 ± 0.004	0.881 ± 0.064
`NutAssembly`	$\mathbf{0.956 \pm 0.016}$	0.705 ± 0.005	0.554 ± 0.080
`TidyUp`	$\mathbf{0.989 \pm 0.005}$	0.944 ± 0.003	0.859 ± 0.004

TABLE II: Effectiveness of trajectory parser.

Task	Success Rate	Primitive Seq Len / Demo Len
`PickPlace`	0.967 ± 0.027	3.5 / 314
`NutAssembly`	0.956 ± 0.031	2.1 / 232
`TidyUp`	0.911 ± 0.016	6.6 / 403

B. Experimental Results

1) Quantitative Results: Fig. 4 demonstrates our method's substantial superiority over all baselines, achieving success rates exceeding 95% across all tasks with remarkable robustness. This showcases the effectiveness of our approach in achieving data-efficient imitation learning through the decomposition of raw sensory demonstrations into concise primitive sequences. Furthermore, our policy often attempts the same primitive type and slightly adjusts primitive parameters after a failure. The results indicate that our primitive policy is more capable of learning this recovery attempt following a failure than baseline policies which require predicting a sequence of actions for recovery. Failures in our method are typically caused by prediction errors in the parameter policy.

TABLE III: Generalization capability of IDM.

IDM	Ours	BC-RNN	BC-RNN (FT)
`IDM-D`	$\mathbf{0.989 \pm 0.005}$	0.825 ± 0.056	0.861 ± 0.043
`IDM-ABC`	$\mathbf{0.975 \pm 0.030}$	0.825 ± 0.056	0.860 ± 0.042

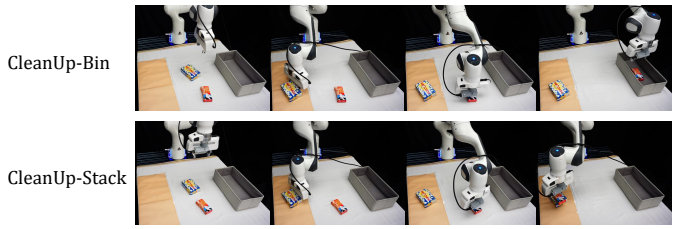

Fig. 6: **Real-world Tasks.** In `CleanUp-Bin`, the robot must push the popcorn box to the serving area and place the butter inside the bin. In `CleanUp-Stack`, the robot must push the popcorn box to the serving area and stack the butter on top of the popcorn.

PRIME-Learned-Skill outperforms all other baselines, confirming that learned skills are compatible with our PRIME framework. However, the fact that PRIME-Learned-Skill performs worse than PRIME with hard-coded primitives suggests that primitives pretrained from a small set of annotated human demonstrations are suboptimal, leading to a suboptimal low-level controller.

2) Ablation Studies: We perform ablation studies in all three simulation tasks to study the importance of design choices in our framework. We compare our methods with ablations: 1) without policy pretraining (**No Pretraining**), 2) utilizing a greedy algorithm to interpret demonstrations into primitive sequences instead of dynamic programming (**Greedy Algorithm**), where the greedy algorithm selects next primitive by examining all states in the demonstration that come after the current state and choosing the one with the highest probability. As shown in Table I, omitting policy pretraining or substituting dynamic programming with the greedy algorithm leads to decreased performance, highlighting their essential roles.

3) Model Analysis: **Effectiveness of the Trajectory Parser.** To evaluate our trajectory parser's performance, we execute the parsed primitive sequences in the simulated task environment. As shown in Table II, our trajectory parser consistently achieves over 90% success rates. Moreover, the ratio of average primitive sequence length (primitive seq len) to average demonstration length (demo len) illustrates that the trajectory parser is capable of reducing the task horizon from hundreds of steps to just a few steps. In Fig. 5, we present a visualization of the output primitive sequences generated by the trajectory parser.

Generalization Capability of the IDM. We assess the generalization capability of IDM in `TidyUp` task. As depicted in the caption of Fig. 3, we train an IDM-ABC on data collected from the environment (A, B, C) and an IDM-D on data collected merely from the environment (D). As shown in Table III, the policy performance achieved using IDM-ABC is comparable to that achieved using IDM-D, highlighting the IDM's generalization capability in an unseen environment.

C. Real-World Evaluation

We evaluate the performance of PRIME against an imitation learning baseline (BC-RNN) on two real-world `CleanUp` task variants: `CleanUp-Bin` and `CleanUp-Stack`. To ensure safe data collection, we perform self-supervised data collection in simulation to train an IDM and apply it to real-world demonstrations. The real-world observations include camera images, object poses, and robot proprioceptive states. Our state-based IDM uses object poses and proprioceptive states, transferring directly to real-world demonstrations. A single IDM segments demonstrations for two tasks within the same domain. Additionally, we develop a visual-based policy to predict motor actions from camera images and robot states. For object poses on the real robot, we use a pose estimator [51].

TABLE IV: Results on the real robot.

Task	Ours	BC-RNN
`CleanUp-Bin`	$\mathbf{0.900 \pm 0.041}$	0.417 ± 0.246
`CleanUp-Stack`	$\mathbf{0.683 \pm 0.062}$	0.483 ± 0.131

The results in Table IV show our method's notable advantage over BC-RNN, primarily due to its effective task scaffolding. Similar to the results in the simulation, our method also exhibits robust recovery behavior on real robots. Our method has a success rate below 70% in `CleanUp-Stack`, mainly because stacking demands greater precision and sim2real training for IDM leads to errors.

V. Conclusion

We present PRIME, a data-efficient imitation learning approach that decomposes task demonstrations into sequences of primitives and leverages imitation learning to acquire the high-level control policy for sequencing parameterized primitives. While we have already evaluated PRIME with pretrained primitives from annotated demonstrations, a promising direction for future research is to learn a scalable library of low-level skills and compose these diverse skills. These skills can include hard-coded primitives, learned primitives, and skills pretrained from large datasets. This approach holds the potential to facilitate curriculum learning, enabling the progressive acquisition of increasingly complex tasks. A limitation of this study is the use of sim2real experiments for IDM training, which may not be fully applicable to challenging real-world tasks. Extending IDM training to real-world settings is left for future research. Another limitation is that all tasks in this work can be fully decomposed into primitives from the primitive library. Extending our work to include tasks that are not fully decomposable would enhance the generalizability of our framework.

Acknowledgment

The authors would like to thank Yifeng Zhu, Jake Grigsby, Mingyo Seo, Rutav Shah, and Zhenyu Jiang for their valuable feedback. Tian Gao's visit to UT Austin was supported by IIIS, Tsinghua University. Quantao Yang's visit to UT Austin was supported by his advisor, Todor Stoyanov, and the Wallenberg AI, Autonomous Systems, and Software Program (WASP).

This work has been partially supported by the National Science Foundation (EFRI-2318065, FRR-2145283), the Office of Naval Research (N00014-22-1-2204), UT Good Systems, and the Machine Learning Laboratory.

REFERENCES

[1] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "Opal: Offline primitive discovery for accelerating offline reinforcement learning," in *ICLR*, 2021.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on robotics*, vol. 30, no. 2, pp. 289–309, 2013.

[3] D. Brandfonbrener, O. Nachum, and J. Bruna, "Inverse dynamics pretraining learns good representations for multitask imitation," *arXiv preprint arXiv:2305.16985*, 2023.

[4] A. Brohan *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *Robotics: Science and Systems (RSS)*, 2022.

[5] H. Chen *et al.*, "Predicting object interactions with behavior primitives: An application in stowing tasks," in *Conference on Robot Learning*, PMLR, 2023, pp. 358–373.

[6] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '07, Honolulu, Hawaii: Association for Computing Machinery, 2007.

[7] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Robotics: Science and Systems (RSS)*, 2023.

[8] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta, "Efficient bimanual manipulation using learned task schemas," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 1149–1155.

[9] M. Dalal, D. Pathak, and R. R. Salakhutdinov, "Accelerating robotic reinforcement learning via parameterized action primitives," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 847–21 859, 2021.

[10] Y. Du *et al.*, "Learning universal policies via text-guided video generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[11] C. R. Garrett *et al.*, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[12] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv preprint arXiv:1511.04143*, 2015.

[13] D.-A. Huang *et al.*, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8565–8574.

[14] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[15] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, 2013.

[16] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[17] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg, "Ddco: Discovery of deep continuous options for robot learning from demonstrations," in *Conference on robot learning*, PMLR, 2017, pp. 418–437.

[18] A. Kumar, A. Singh, F. Ebert, Y. Yang, C. Finn, and S. Levine, "Pre-training for robots: Offline rl enables learning new tasks from a handful of trials," *arXiv preprint arXiv:2210.05178*, 2022.

[19] Y. Lee, J. Yang, and J. J. Lim, "Learning to coordinate manipulation skills via skill behavior diversification," in *International conference on learning representations*, 2019.

[20] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 3684–3691.

[21] J. Luo *et al.*, "Multi-stage cable routing through hierarchical imitation learning," *arXiv preprint arXiv:2307.08927*, 2023.

[22] C. Lynch *et al.*, "Learning latent plans from play," in *Conference on robot learning*, PMLR, 2020, pp. 1113–1132.

[23] J. Mahler *et al.*, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *RSS*, 2017.

[24] P. Mahmoudieh, T. Darrell, and D. Pathak, "Weakly-supervised trajectory segmentation for learning reusable skills," *ICLR 2020 Workshop on Bridging AI and Cognitive Science*, 2020.

[25] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020.

[26] A. Mandlekar *et al.*, "Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 4414–4420.

[27] A. Mandlekar *et al.*, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[28] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[29] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.

[30] S. Nasiriany, T. Gao, A. Mandlekar, and Y. Zhu, "Learning and retrieval from prior data for skill-based imitation learning," *arXiv preprint arXiv:2210.11435*, 2022.

[31] 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)*, IEEE, 2022, pp. 7477–7484.

[32] G. Neumann, C. Daniel, A. Paraschos, A. Kupcsik, and J. Peters, "Learning modular policies for robotics," *Frontiers in computational neuroscience*, vol. 8, p. 62, 2014.

[33] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

[34] K. Paster, S. A. McIlraith, and J. Ba, "Planning from pixels using inverse dynamics models," *arXiv preprint arXiv:2012.02419*, 2020.

[35] B. S. Pavse, F. Torabi, J. Hanna, G. Warnell, and P. Stone, "Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6262–6269, 2020.

[36] C. V. Perico, J. de Schutter, and E. Aertbeliën, "Learning robust manipulation tasks involving contact using trajectory parameterized probabilistic principal component analysis," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8336–8343.

[37] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on robot learning*, PMLR, 2021, pp. 188–204.

[38] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," *arXiv preprint arXiv:2107.10253*, 2021.

[39] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 3406–3413.

[40] D. Precup, *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[41] A. Rajeswaran *et al.*, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[42] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[43] T. Shankar and A. Gupta, "Learning robot skills with temporal variational inference," in *International Conference on Machine Learning*, PMLR, 2020, pp. 8624–8633.

[44] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta, "Discovering motor programs by recomposing demonstrations," in *International Conference on Learning Representations*, 2019.

[45] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta, "Discovering motor programs by recomposing demonstrations," in *International Conference on Learning Representations*, 2020.

[46] L. X. Shi, A. Sharma, T. Z. Zhao, and C. Finn, "Waypoint-based imitation learning for robotic manipulation," *arXiv preprint arXiv:2307.14326*, 2023.

[47] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner, "Taco: Learning task decomposition via temporal alignment for control," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4654–4663.

[48] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," *arXiv preprint arXiv:2011.10024*, 2020.

[49] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid, "Learning to combine primitive skills: A step towards versatile robotic manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 4637–4643.

[50] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning.," in *IJCAI*, 2015, pp. 1930–1936.

[51] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *CoRL*, 2018.

[52] C. Wang *et al.*, "Mimicplay: Long-horizon imitation learning by watching human play," *arXiv preprint arXiv:2302.12422*, 2023.

[53] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, and S. Levine, "How to leverage unlabeled data in offline reinforcement learning," in *International Conference on Machine Learning*, 2022.

[54] T. Zhang *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 5628–5635.

[55] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robotics: Science and Systems (RSS)*, 2023.

[56] Q. Zheng, M. Henaff, B. Amos, and A. Grover, "Semi-supervised offline reinforcement learning with action-free trajectories," in *International conference on machine learning*, PMLR, 2023, pp. 42 339–42 362.

[57] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *arXiv preprint arXiv:2210.11339*, 2022.

[58] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4126–4133, 2022.

[59] Y. Zhu *et al.*, "Robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.