Bottom-Up Skill Discovery from Unsegmented Demonstrations for Long-Horizon Robot Manipulation

Yifeng Zhu[1], Peter Stone[1,2], Yuke Zhu[1]

***Abstract*—We tackle real-world long-horizon robot manipulation tasks through skill discovery. We present a bottom-up approach to learning a library of reusable skills from unsegmented demonstrations and use these skills to synthesize prolonged robot behaviors. Our method starts with constructing a hierarchical task structure from each demonstration through agglomerative clustering. From the task structures of multi-task demonstrations, we identify skills based on the recurring patterns and train goal-conditioned sensorimotor policies with hierarchical imitation learning. Finally, we train a meta controller to compose these skills to solve long-horizon manipulation tasks. The entire model can be trained on a small set of human demonstrations collected within 30 minutes without further annotations, making it amendable to real-world deployment. We systematically evaluated our method in simulation environments and on a real robot. Our method has shown superior performance over state-of-the-art imitation learning methods in multi-stage manipulation tasks. Furthermore, skills discovered from multi-task demonstrations boost the average task success by 8% compared to those discovered from individual tasks.[1]**

***Index Terms*—Deep Learning for Grasping and Manipulation, Imitation Learning, Sensorimotor Learning.**

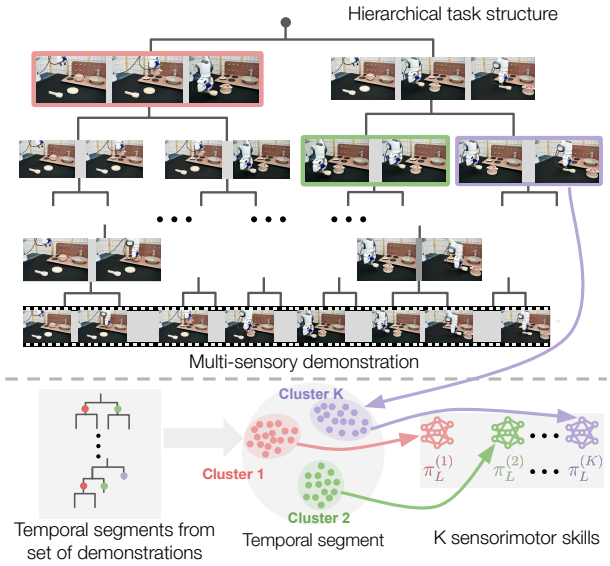


Fig. 1: Overview of BUDS. We construct hierarchical task structures of demonstration sequences in a bottom-up manner, from which we obtain temporal segments for discovering and learning sensorimotor skills.

I. Introduction

REAL-world manipulation tasks challenge autonomous robots to reason about long-term interactions with the physical environment through the lens of raw perception. To tackle this challenge, temporal abstraction [38, 47] offers a powerful framework to model the compositional structures of manipulation tasks. The key idea is to use (sensorimotor) skills as the basic building blocks for synthesizing temporally extended robot behaviors. Recently, skill-based manipulation algorithms, including task and motion planning [11, 18, 19] and hierarchical reinforcement learning [6, 47], have demonstrated promising results in some restrictive settings. Nonetheless, it remains challenging to devise an effective yet automated way of building a rich repertoire of skills without costly manual engineering, especially when incorporating real-world sensory data.

Several paradigms have been investigated for automating skill acquisition. Notable ones include the *options* framework [2, 10, 13, 23, 26, 50] and unsupervised skill discovery based on information-theoretic metrics [8, 15, 43]. While initial successes have been achieved for discovering skills from a robot's self exploration, primarily in simulation, these methods exhibit a prohibitively high sample complexity or require access to ground-truth physical states, hindering their applicability on real robots. An alternative strategy is to extract skills from human demonstrations, easing the exploration burdens. A direct approach is to manually annotate the demonstrations into coherent segments as skills [17], but manual annotations of temporal segments can be ambiguous and costly. Instead, our approach discovers skills from a small set of *unsegmented* human demonstrations with no temporal labels. It reduces the human efforts and improves the scalability. Prior work on learning from unsegmented demonstrations used Bayesian inference [23, 36], generative modeling [22, 42, 48], and dynamic programming [44] to temporally decompose the demonstrations into a sequence of shorter skill segments. However, these methods usually fall short of processing high-dimensional sensor data. Recent ad-

Manuscript received: September 9, 2021; Revised December 6, 2021; Accepted Jan 10, 2022.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work was partially supported by research grants from NSF, MLL Research Award, FLI, ARO, DARPA, Lockheed Martin, GM, and Bosch.

[1]Yifeng Zhu, Peter Stone, and Yuke Zhu are with the Department of Computer Science, the University of Texas at Austin. [2] Peter Stone is also with Sony AI.

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

[1]Project website with additional materials can be found at https://ut-austin-rpl.github.io/rpl-BUDS.

vances in hierarchical imitation learning [14, 31, 32], modeled with deep neural networks, have shown promise in tackling long-horizon manipulation tasks in a hierarchical manner. They have focused on solving individual tasks rather than discovering a library of reusable skills across tasks.

We develop a hierarchical approach to tackling real-world robot manipulation through skill discovery from unsegmented demonstrations. Our method BUDS (**B**uttom-**U**p **D**iscovery of sensorimotor **S**kills), presented in Figure 1, starts with an unsupervised clustering-based segmentation model that extracts a library of sensorimotor skills from human demonstrations collected through teleoperation. Each skill is modeled as a goal-conditioned sensorimotor policy that operates on raw images and robot proprioceptiion. BUDS further learns a high-level meta controller that selects a skill and predicts the subgoal for the skill to achieve at any given state. Both the skills and the meta-controllers are trained with imitation learning on human demonstrations. Together it presents a scalable framework for solving complex manipulation tasks from raw sensory inputs, amendable to real-robot deployment.

Four key properties of BUDS are crucial for its effectiveness: 1) it uses bottom-up agglomerative clustering to build hierarchical task structures of demonstrations. These hierarchical representations offer flexibility for the imitation learner to determine the proper granularity of the temporal segments. 2) BUDS segments the demonstrations based on multi-sensory cues, including multi-view images and proprioceptive features. It takes advantage of the statistical patterns across multiple sensor modalities to produce more coherent and compositional task structures than using a single modality. 3) BUDS extracts skills from human demonstrations on multiple tasks that achieve different manipulation goals, facilitating knowledge sharing across tasks and improve the reusability of the discovered skills. 4) we train our goal-conditioned skills with the recently developed Hierarchical Behavior Cloning algorithm [14, 31, 32, 49], producing perceptually grounded yet versatile skills for composition.

We systematically evaluate our method in simulation and on a real robot and perform ablation studies to validate our design choices. BUDS achieves an average 66% success rate on three challenging vision-based manipulation tasks, which the recent hierarchical imitation learning algorithms [14, 32] struggled to solve. It also outperformed the most competitive baselines over 20%. We further show that skills learned from multi-task demonstrations boost the success rate by 8% compared to those learned for each task separately. Moreover, we show that the skills can be reused for solving new task variants that require different subtask combinations. Finally, we deploy BUDS on a real robot for a complex kitchen task, achieving a 56% success rate on par with our simulation results. For all experiments, BUDS is trained on 50-120 demonstrations for each task collected within 30min.

We summarize the three key contributions of this work: 1) We present a bottom-up clustering algorithm to discover sensorimotor skills from unsegmented demonstrations; 2) We introduce a hierarchical policy learning method that composes the skills for long-horizon, vision-based robot manipulation tasks; 3) We show the practical advantages of BUDS both in simulation and on real hardware.

II. Related Work

Robot Skill Discovery. Skill discovery has been studied in a large body of existing works. A major line of works focuses on acquiring skills from self-exploration in environments. Many works fall into the options framework [47], discovering skills through hierarchical reinforcement learning [2, 10, 13, 23, 26, 50]. Other works use information-theoretic metrics to discover skills from unsupervised interaction [8, 15, 43]. These works typically require high sample complexity and operate on ground-truth physical states, hindering their applicability to real robot hardware. An alternative to self-exploration is to segment skills from human demonstrations, such as Bayesian inference [24, 35, 36] and trajectory reconstruction [42, 48]. These approaches produce temporal segmentation on low-dimensional physical states, difficult to scale to raw sensor data. Weakly supervised learning methods discover skill segments through temporal alignment on demonstrations [37, 44], but require manual human annotations of task sketch. Our work resonates with these works on skill discovery from human demonstrations; however, it directly operates on raw sensor data and requires no manual labeling on execution stages in demonstrations. Similar to prior works [5, 46], we take advantage of multi-sensory cues in demonstrations. An important difference is that our method produces closed-loop sensorimotor policies, while the others focus primarily on learning task structures.

Bottom-up Methods in Perception and Control. Bottom-up processing of sensory information traces back to Gibson's theory of direct perception [12], of which the basic idea is that the higher level of information is built up on the retrieval of direct sensory information. Bottom-up methods have been successfully employed in various perception tasks. These methods construct hierarchical representations by grouping more fine-grained visual elements, such as pixels/superpixels for image segmentation [9] and spatio-temporal volumes for activity understanding [27, 39, 40]. Recently, bottom-up deep visual recognition models [28, 33, 56] achieved competitive performances compared to the mainstream top-down methods. The bottom-up design principles have also been studied for robot control. A notable example is the subsumption architecture developed in behavior-based robotics, which decomposes a complex robot behavior into hierarchical layers of sub-behaviors [3, 4, 25, 34]. Our work leverages a similar bottom-up principle to discover hierarchical representations of human demonstrations. Furthermore, we demonstrate how imitation learners can exploit such hierarchies to scale to long-term manipulation behaviors.

Hierarchical Imitation in Robot Manipulation. We leverage hierarchical imitation learning [29] for learning policies of sensorimotor skills. Hierarchical imitation learning is a class of approaches which uses temporal abstractions to tackle longer-horizon tasks than vanilla imitation models. In particular, we

use hierarchical behavior cloning [14, 31, 32, 49], which recently shows great promises in robot manipulation. These methods learn a hierarchical policy where a high-level policy predicts subgoals and a low-level policy computes actions to achieve the subgoals, where the subgoals can be obtained through goal relabelling [1]. One-shot imitation learning is a meta-learning framework that aims to learn from a single demonstration of test tasks [7, 45, 52, 53]. Our work differs from prior work in that we extract a set of skills from multi-task demonstrations for task composition and directly handle raw sensory data.

III. Approach

We introduce BUDS, an approach for sensorimotor skill discovery and hierarchical imitation learning of closed-loop sensorimotor policies in robot manipulation. The core idea is to discover a set of reusable sensorimotor skills from multi-task, multi-sensory demonstrations, which can be composed to solve long-horizon tasks. An overview of BUDS is illustrated in Figure 1. In the following, we first formalize our problem, and then present the two key steps of our approach: 1) skill segmentation with hierarchical agglomerative clustering on unsegmented demonstrations, and 2) learning skills and meta-controllers with hierarchical behavioral cloning.

A. Problem Formulation

We formalize the problem of solving a robot manipulation task as a discrete-time Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \rho_0)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(\cdot|s,a)$ is the stochastic transition probability, $R(s,a,s')$ is the reward function, $\gamma \in [0,1)$ is the discount factor, and $\rho_0(\cdot)$ is the initial state distribution. Our goal is to learn a sensorimotor policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$. In our context, $\mathcal{S}$ is the space of the robot's sensor data including raw images and proprioceptions, $\mathcal{A}$ is the space of the robot's motor actions, and π is a closed-loop sensorimotor policy that we deploy on the robot to perform the task.

To tackle long-horizon tasks, we factorize the policy π with a two-level temporal hierarchy. The low level consists of a set of K skills, $\{\pi_L^{(1)}, \pi_L^{(2)}, \ldots, \pi_L^{(K)}\}$, each of which corresponds to a goal-conditioned policy $\pi_L^{(k)} : \mathcal{S} \times \Omega \to \mathcal{A}$, where $\omega \in \Omega$ is a vector that represents a goal, and k is the skill index. This is a standard formulation of hierarchical policy learning, under which prior work has explored different goal representations, with Ω being the original state space [41] or a learned latent space [50]. A latent goal for $\pi_L^{(k)}$ can be computed using an encoder E_k on the goal state, which is defined in the original state space. To harness these skills, we further design a high-level meta controller $\pi_H : \mathcal{S} \to \{1, 2, \ldots, K\} \times \Omega$. Intuitively, the meta controller outputs two pieces of information from a given state to invoke a low-level skill: a categorical distribution over the skill indices, of which we take the mode as the selected skill, and a vector that specifies the goal for this selected skill to reach. With this temporal abstraction, the policy π can be thus represented as

$$\pi(a_t|s_t) = \sum_{i=1}^{K} \mathbb{1}(i=k)\pi_L^{(i)}(a_t|s_t,\omega), \quad \text{where } (k,\omega) = \pi_H(s_t) \tag{1}$$

Prior works proposed different algorithms to train hierarchical policies with reinforcement learning [14] or imitation learning [31], but typically focusing on a single task. In contrast, we examine a multi-task learning formulation, where the skills are learned from multi-task human demonstrations. We assume the demonstrations come from M different tasks, each corresponding to a different MDP. We assume the MDPs of all M tasks share the same state space, action space, and transition probabilities, but differ in reward functions and initial state distributions. Our demonstrations are collected from human operators to complete instances of each task from different initial states. We denote $\mathcal{D}^{(m)} = \{\tau_i^{(m)}\}_{i=1}^{N_m}$ as the demonstration datasets for the m-th task, where $\tau_i^{(m)} = \{(s_t, a_t)\}_{t=0}^{T_i^{(m)}}$ is the i-th demonstration sequence of length $T_i^{(m)}$ and N_m is the total number of demonstrations for this task. Let $\mathcal{D} = \bigcup_{m=1}^{M} \mathcal{D}^{(m)}$ be the aggregated dataset of all tasks.

Our method learns the hierarchical policies from the multi-task demonstrations. We use hierarchical clustering (Sec. III-B) to identify the recurring temporal segments from the aggregated dataset, separating $\mathcal{D}$ into K partitions $\{\tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_K\}$. By learning on $\mathcal{D}$, we augment the training data for individual tasks and facilitate the learned low-level skills to be reusable across tasks. We use hierarchical behavior cloning (Sec. III-C) to learn the goal-conditioned skills $\pi_L^{(k)}$ on $\tilde{\mathcal{D}}_k$ for $k = 1, 2, \ldots, K$. Once obtaining skills, we learn to compose the skills with a task-specific meta-controller trained on the demonstration data of that task, for example, training π_H to solve the m-th task on $\mathcal{D}^{(m)}$.

B. Skill Segmentation with Hierarchical Clustering

We present how to split the aggregated dataset D into K partitions, which we use to train the K skills. Our objective is to cluster similar temporal segments from multi-task demonstrations into the same skill, easing the burden for the downstream imitation learning algorithm. To this end, we first learn per-state representations based on multi-sensory cues, which we use to form a hierarchical task structure of each demonstration sequence $\tau_i^{(m)}$ with bottom-up agglomerative clustering. We then identify the recurring temporal segments across the entire demo dataset via spectral clustering. The whole process is fully unsupervised without additional annotations beyond the demonstrations.

Learning Multi-Sensory State Representations. Our approach learns a latent representation per state in the demonstrations. It is inspired by research in event perception [54], which addresses the importance of correlation statistics presented in multiple sensory modalities for event segmentation. BUDS learns the representations from multi-modal sensory data to

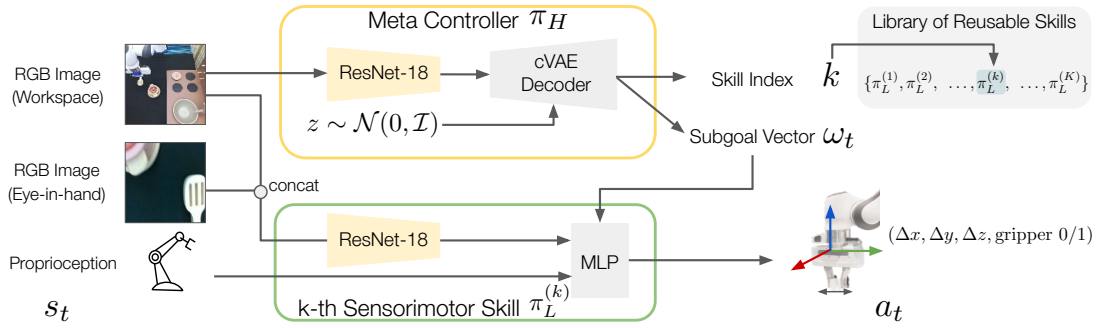


Fig. 2: Overview of the hierarchical policy. Given a workspace observation, the meta controller selects the skill index and generates the latent subgoal vector ω_t. Then the selected sensorimotor skill generates action a_t conditioned on observed images, proprioception, and ω_t.

capture their statistical patterns. Our method follows Lee et al. [30] which learns a joint latent feature of all modalities by fusing feature embeddings from individual modalities (multi-view images and proprioception) with the Product of Experts [16]. The feature is optimized over an adapted evidence lower bound loss [21] which reconstructs the current sensor inputs. The reconstruction is different from the one from Lee et al. which is optimized over reconstructing next states. This different design choice is motivated by the fact that the previous work focuses on policy learning with fused representation inputs which needs to encode future state information, while BUDS focuses on learning the statistical patterns of multi-sensory data at the current state. By learning the joint representation, it captures the congruence underlying multi-sensory observations while retaining the information necessary to decode s_t. We denote h_t as the latent vector computed from s_t.

Discovering Temporal Segments. BUDS uses the per-state representations to effectively group states in temporal proximity to build a hierarchical representation of a demonstration sequence. The strength of a hierarchical representation, as opposed to flat segmentation, is the flexibility to decide the segmentation granularity for imitation learning. Here we use hierarchical aggomerative clustering, where in each step we combine two adjacent temporal segments into one based on similarity until all segments are combined into the entire demonstration sequence. This process produces a tree of segments. To reduce the tree depth, we start with the bottom-level elements that contain a temporal segments of a demonstration of W steps ($W = 10$ in our case). The clustering process selects two adjacent segments that are most similar to each other among all pairs of adjacent segments, and merges them into one longer segment. The similarity between two segments is computed according to the ℓ_2 distance between their *segment features*, defined to be the average of latent vectors $\{h_t\}$ of all states in each segment. The process is repeated until we have only one segment left for each $\tau_i^{(m)}$. We discover a collection of intermediate segments, which we term as *temporal segments*, from the formed hierarchies. This concept is inspired by the concept of Mid-level Action Elements [27] in the action recognition literature. The way we determine the temporal segments is to breadth-first search from the root node of the hierarchy. During the search, we stop on one branch if the length of the intermediate segment is not longer than a given threshold of minimum length. And the whole breadth-first search is stopped when we have the number of segments at the lowest levels on every branch are more than a given threshold, and each segment at the lowest levels on every branch in $\tau_i^{(m)}$ is a temporal segment.

Partitioning Skill Datasets. After we have a set of temporal segments for every $\tau_i^{(m)}$, we aggregate them from all demonstrations into one set, and apply another clustering process to group them into K partitions $\{\tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_K\}$, and we use each partition $\tilde{\mathcal{D}}_k$ to train the skill $\pi_L^{(k)}$. By training the skills on datasets from multiple tasks, it improves the reusability of the skills. We use spectral clustering [51] with RBF kernel on the features of temporal segments. The feature of a segment is computed as the concatenation of representations of the first, middle (or several frames in the middle), and last states of the segment. The number of keyframes chosen in the middle of a segment can vary based on the average length of demonstrations. The spectral clustering step results in K datasets of temporal segments for skill learning. In practice, we set the maximum number of clusters in spectral clustering and merge any classified skill into an adjacent skill if its average length is below a threshold. The number of remaining classes is denoted as K, which is the final number of skills we partition demonstrations into.

C. Policy Learning with Hierarchical Behavioral Cloning

We use the obtained K datasets of temporal segments from the segmentation step to train our hierarchical policies using a hierarchical behavioral cloning algorithm, including two parts: 1) skill learning with goal-conditioned imitation; 2) skill composition with a meta controller. Figure 2 visualizes the model structure of the hierarchical policy.

Skill Learning with Goal-Conditioned Imitation. We train each skill $\pi_L^{(k)}$ on the corresponding dataset $\tilde{\mathcal{D}}_k$ $\forall k = 1, \ldots, K$. Every skill $\pi_L^{(k)}(a_t|s_t, \omega_t)$ takes a sensor observation $s_t \in \mathcal{S}$ and a subgoal vector ω_t as input, and produces a robot's motor action $a_t \in \mathcal{A}$. By conditioning $\pi_L^{(k)}$ on a subgoal vector, we enable the meta controller to invoke the skills and specify the subgoals that these skills should achieve. Instead of defining the subgoals in the original sensor space, which is typically high-dimensional, we instead learn a latent space of subgoals Ω, where a subgoal state s_g is mapped to a low-dimensional feature vector $\omega_t \in \Omega$. For each state s_t, we define

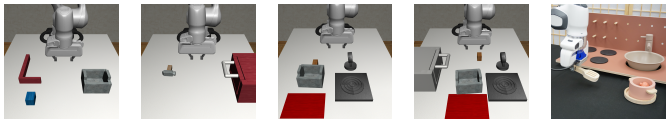
its subgoal as the future state either H steps ahead of s_t in the demonstration or the last state of a skill segment if it reaches the end of the segment from s_t within H steps. The reason we define a subgoal as a look-ahead state a constant number of steps in the future, as opposed to the final goal state of the task, is to exploit the temporal abstraction and reduce the computational burden of individual skills — skills only need to reach short-horizon subgoals, without the need for reasoning about long-term goals. Concretely, we train such a goal-conditioned skill on skill segments in $\tilde{\mathcal{D}}^{(k)}$. For each $\pi_L^{(k)}$, we sample $(s_t, a_t, s_{g_t}) \sim \tilde{\mathcal{D}}^{(k)}$ where $g_t = \min(t + H, T)$ (T is the last timestep of end of the segment), and we generate a latent subgoal vector ω_t for the subgoal of s_t, s_{g_t}, using a subgoal encoder $\omega_t = E_k(s_{g_t})$, where $E_k(\cdot)$ is a ResNet-18 backboned network jointly trained with the policy $\pi_L^{(k)}$.

Skill Composition with A Meta Controller. Now we that have a set of skills, we need a meta controller to decide which skill to use at s_t and specify the desired subgoal for it to reach. We train a task-specific meta controller π_H for each task. Given the current state, π_H outputs an index $k \in \{1, \ldots, K\}$ to select the k-th skill, along with a subgoal vector ω_t on which the selected skill is conditioned. As human demonstrations are diverse and suboptimal by nature, the same state could lead to various subgoals in demonstration sequences (e.g., grasp different points of an object, push an object at different contact points). Thus, the meta controller needs to learn distributions of subgoals from demonstration data of a task $\mathcal{D}^{(m)}$, and we choose conditional Variational Autoencoder (cVAE) [21]. To obtain the training data of skill indices and subgoal vectors, we sample $(s_t, s_{g_t}) \sim \mathcal{D}^{(m)}$, and from the clustering step we have the correspondence between a skill index k and state s_t while from the skill learning step we can generate a per-state subgoal vector $\omega_t = E_k(s_{g_t})$. The meta controller π_H is trained to generate a skill index k and a subgoal vector ω_t conditioned on state s_t. During evaluation, the controller generates the skill index and the subgoal vector conditioned on the current state s_t and a latent vector z from the prior distribution $\mathcal{N}(0, \mathcal{I})$. The controller for evaluation is typically chosen to operate at a lower frequency than skills so that it can avoid switching among skills too frequently.

Training meta controllers follows the same cVAE training convention in prior works [31, 32] which minimizes an ELBO loss on demonstration data. To obtain the training supervision for skill indices and subgoal vectors, we augment the demonstrations with results from the clustering and skill learning steps: 1) The training labels of skill indices come from the cluster assigments; 2) The latent subgoal vectors are computed on the demonstration states, and the encoders for computing the vectors were jointly trained with skill policies.

IV. Experiments

We design our experiments to examine three questions: 1) How does BUDS perform in long-horizon vision-based manipulation tasks compared to baseline methods? 2) Do learning from multi-task demonstrations and using multimodal features improve the quality and reusability of skills? and 3) Does BUDS work with real-world sensor data and physical hardware?



(a) (b) (c) (d) (e)

Fig. 3: Visual illustrations of the four simulation tasks and one real robot task used in our experiments. (a) `Tool-Use`; (b) `Hammer-Place`; (c) `Kitchen`; (d) `Multitask-Kitchen`; (e) `Real-Kitchen`.

A. Experimental Setup

We perform baseline comparisons and model analysis in simulation environments developed with the `robosuite` framework [57], and present quantitative results on real hardware. Figure 3 illustrates all the tasks. The first three single-task environments, `Tool-Use`, `Hammer-Place`, and `Kitchen`, are designed primarily for baseline comparisons and ablation studies. The multi-task domain `Multitask-Kitchen` is designed for investigating the quality and reusability of skills discovered from multi-task demonstrations. The `Real-Kitchen` task is for real-world validation and deployment. We provide detailed task descriptions below. For all the experiments, we use a 7-DoF Franka Emika Panda arm with a position-based Operational Space Controller [20] and a binary command for controlling parallel-jaw gripper. The meta controller runs at 4Hz and the skills run at 20Hz. We release our datasets, simulation environments, and model implementations on our project website for reproducing purpose.

Single tasks. The three single tasks require prolonged interactions with the environment and entail a broad range of prehensile and nonprehensile behaviors, such as tool use and manipulating articulated objects. The goal of `Tool-Use` is to put the cube into the metal pot. In order to fetch the cube which is initially beyond the robot's reach, the robot needs to first grasp an L-shape tool and pull the cube with the tool. After fetching the cube, it needs to put the tool aside, pick up the cube, and place it into the pot. The goal of `Hammer-Place` is to put the hammer in the drawer and close the drawer, where the hammer is small and hard to grasp. To achieve this goal, the robot needs to open the drawer, place the hammer into the drawer, and close the drawer. The goal of `Kitchen` is to cook and serve a simple dish in the serving region and turn off the stove. This task is the most complex among the three, requiring a sequence of subtasks, including turning on the stove, placing the pot on the stove, putting the ingredient into the pot, putting the pot on the table, pushing it to the serving region (red region on the table), and turning off the stove at the end.

`Multitask-Kitchen`. The multi-task domain includes three tasks (referred to as `Task-1`, `Task-2`, and `Task-3`) that are distinct from each other by their end goals. The goal state of `Task-1` entails the drawer closed, the stove on, the cube in the pot, and the pot placed on the stove. The goal of `Task-2` entails the drawer closed, the stove on, the cube in the pot, and the pot in the serving region. The goal of `Task-3` entails the drawer closed, the stove off, the cube in the pot, and the pot in the serving region. To study the reusability of our skills, we design three variants for each task based on

their initial configurations, which we refer to as `Variant-1`, `Variant-2`, `Variant-3`. We describe all task variants in details at our project website. Different initial configurations require solving different combinations of subtasks. Therefore, we examine whether skills learned in a subset of task variants can be reused in new variants.

`Real-Kitchen`. The task requires versatile behaviors including grasping, pushing, and tool use. The robot needs to remove the lid of the pot, place the pot to the plate, pick up the tool, use the tool to push the pot along with the plate to the side of the table, and put down the tool in the end. We capture RGB images from the workspace camera (Kinect Azure) and the eye-in-hand camera (Intel Realsense D435i).

Data Collection. We collect human demonstrations through teleoperation with a 3Dconnexion SpaceMouse. We collect 100 demonstrations for each of the three single tasks (less than 30 minutes each task), 120 demonstrations for each task (40 for each of the three task variants) in `Multitask-Kitchen`, and 50 demonstrations for `Real-Kitchen`. Each demonstration consists of a sequence of sensory observations (images from the workspace and the eye-in-hand camera, proprioception) and actions (the end-effector displacement and the gripper open/close).

B. Quantitative Results

For all simulation experiments, we evaluate BUDS and baseline methods in each task for 100 trials with random initial configurations (repeated with 5 random seeds). We use the success rate over trials as the evaluation metric, and a trial is considered successful if the task goal is reached within the maximum number of steps.

Single Task Experiments. Here we compare BUDS with imitation learning baselines on the single-task environments. To examine the efficacy of hierarchical modeling for long-horizon tasks, we first compare with a Behavior Cloning (BC) baseline [55] which trains a flat policy on the demonstrations. To examine our bottom-up clustering-based segmentation method, we compare with a second baseline that uses a classical Change Point Detection (CP) algorithm [35] to temporally segment the demonstrations while keeping the rest of the model design identical to ours.

Table I reports the quantitative results. BUDS outperformed both baselines for all three tasks, by over 20% on average. The comparison between BC and BUDS shows that while BC is able to solve short-horizon task reasonably well, it suffers a significant performance drop in longer tasks, such as `Kitchen`. In contrast, BUDS breaks down a long-horizon task with skill abstraction, leading to a consistent high performance across tasks of varying lengths. The comparison between CP and BUDS suggests that the quality of skill segmentation plays an integral role in final performance. Qualitatively, we found that the CP baseline failed to produce coherent segmentation results across different demonstrations, hindering the efficacy of policy learning.

We observe two major failure modes in BUDS: 1) Incorrect selection of skills due to out-of-distribution states, 2) Manipulation failures due to imprecise grasps. We quantify the failure modes in the `Kitchen` task with 5 repeated runs. Failures due to the first mode take up $12.3\% \pm 2.9\%$ of the evaluation trials, and failures due to the second one take up $9.0\% \pm 3.6\%$. Both failure types pertain to the fundamental limitations of imitation learning on small offline datasets. We believe the model performance could be improved with large-scale training and online robot experiences. We leave it for future work.

TABLE I: Success rate (%) in single task environments.

Environments	BC [55]	CP [35]	BUDS(Ours)
`Tool-Use`	54.0 ± 6.3	36.8 ± 5.1	**58.6** ± 3.1
`Hammer-Place`	47.8 ± 3.7	60.4 ± 4.5	**68.6** ± 5.7
`Kitchen`	24.4 ± 5.3	23.4 ± 3.4	**72.0** ± 4.0

Comparisons to Hierarchical Imitation Learning Algorithms. BUDS shares the same principle with recent works on hierarchical imitation learning, including IRIS [31], GTI [32], and RPL [14]. One notable distinction is that the prior works consider a single low-level skill, rather than a library of skills.

To compare BUDS with GTI, we evaluate our method with varying numbers of skills through a parameter sweep on the number of clusters K in the spectral clustering step. In the special case when BUDS has only a single skill ($K = 1$), our method is equivalent to a variant of GTI without the image reconstruction term. Table II reports the results in the `Kitchen` task. We observe that the number of skills has a salient impact on model performance. Intuitively, when K is too small, each skill will have difficulty dealing with diverse subgoals and various visual observations; and when K is too large, each skill has fewer data points to train on, as the dataset is fragmented into smaller partitions. The peak performance is observed with $K = 6$ skills, which is the value we used for the main experiments.

The GTI variant with a single skill ($K = 1$) fails to achieve non-zero task success. We also implemented the original GTI with the image reconstruction term, but observed no significant change in performance. After analyzing the qualitative behaviors of the GTI policy, we find that it works fine if the initial state is close to the task goal, but it cannot handle initial states that are further away. For quantitative evidence, we conduct an additional evaluation with the `Tool-Use` task, where we reset the robot to the state when it has already fetched the cube and placed the tool down. To complete the task, the robot only needs to pick up the cube and place it in the pot. In this shorter subtask, the GTI variant and BUDS achieved 63.0% and 60.3% success rates respectively. In comparison, they achieved 0.0% and 58.6% (Table I) success rates when starting from the original initial states. These results imply that GTI does not generalize well to longer tasks studied in this work.

Comparisons to IRIS and RPL require additional efforts as they were designed for low-dimensional states. We adapt RPL to handle image inputs by extracting visual features with a ResNet-18 module in the policy network, but the adapted model achieves no task success. After a closer examination, RPL fails to generalize to various object placements. Further-

TABLE II: Results in `Kitchen` with varying numbers of skills

	$K = 1$ (GTI [32])	$K = 3$	$K = 6$	$K = 9$	$K = 11$
`Kitchen`	0.0 ± 0.0	24.2 ± 3.6	72.0 ± 4.0	60.6 ± 6.53	44.6 ± 3.38

TABLE III: Success rate (%) in `Multitask-Kitchen`.

	Train (Multi)	**Train (Single)**	**Test**
`Task-1`	70.2 ± 2.2	52.6 ± 5.6	59.0 ± 6.4
`Task-2`	59.8 ± 6.4	60.8 ± 1.9	55.3 ± 3.3
`Task-3`	75.0 ± 2.0	67.6 ± 1.8	28.4 ± 1.5

more, it also uses a single low-level policy similar to GTI, which we have shown lower performance with the GTI-variant. On the other hand, IRIS is more difficult to adapt as its high-level policy predicts subgoals in the original state space, in our case, the raw sensory space. We expect it to suffer the similar issues as the other two methods.

Learning from Multi-task Demonstrations. We investigate if BUDS is effective in learning from multi-task demonstrations. BUDS discovers $K = 8$ skills in the `Multitask-Kitchen` domain, and we examine the skills from two aspects: 1) *quality*: are skills learned from multi-task demonstrations better than those from individual tasks? 2) *reusability*: can these skills be composed to solve new task variants that require different subtask combinations?

We evaluate three settings: 1) **Train (Multi)**: the skills are discovered and trained on the multi-task demonstrations, and the meta controller is trained for each task respectively; 2) **Train (Single)**: the skills are discovered from demonstrations of each individual task; so is the meta-controller; and 3) **Test**: the skills are trained on demonstrations of `Variant-1` and `Variant-2` and the meta-controller is trained on `Variant-3`. Table III presents the evaluation results. The comparisons between **Train (Multi)** and **Train (Single)** indicate that skills learned across multi-task demonstrations improve the average task performance by 8% compared to those learned on demonstrations of individual tasks. We hypothesize that the performance gain roots from our method's ability to augment the training data of each skill with recurring patterns from other tasks' demonstrations. The results on **Test** show that we can effectively reuse the skills to solve the new task variants that require different combinations of the skills by solely training a new meta controller to invoke the pre-defined skills. We also observe the low performance of `Variant-3` on **Test**, because it has more subtasks than its training counterparts, and the execution failure of each skill compounds, leading to the low success rate. We provide additional visualizations of our skills and policy rollouts on our project website.

TABLE IV: Ablation study on demonstration state representations.

	BUDS	**BUDS-Image**	**BUDS-WS-Image**	**BUDS-Proprio**
`Kitchen`	72.0 ± 4.0	41.4 ± 2.2	7.4 ± 2.4	36.8 ± 5.8

Ablation Study on Demonstration State Representations. The quality of segmentation heavily relies on the choice of features we use to represent the demonstration data. A critical design of BUDS is to use multimodal representations learned from multi-view images and proprioceptive data for each state. This ablation study analyzes its impact. We compare BUDS with three ablative models, which learns the state representations from both workspace and eye-in-hand cameras (BUDS-Image), from only the workspace camera (BUDS-WS-Image), and from the proprioceptive data (BUDS-Proprio). The rest of these ablative models remains identical to BUDS. Table IV reports the comparisons. The use of multimodal representations in BUDS substantially outperforms the ablative models in task success rate. We note that the ablative model BUDS-WS-Image results in significantly lower performance. The workspace images without the aid of eye-in-hand images do not capture all task-relevant information of objects due to occlusion, leading to the poor segmentation results. This study shows that the use of multi-sensory observation leads to a more coherent task structure and better skill learning for solving challenging manipulation tasks.

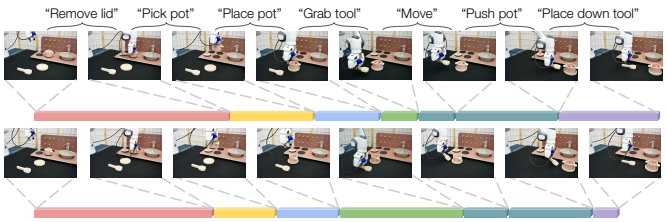


Fig. 4: Visualization of temporal segments in two demonstrations of `Real-Kitchen` with different lengths. Each cuboid corresponds to a temporal segment, and each color represents one skill that the temporal segment is clustered into. Text annotations are our interpretations of the segmented skills. We show that the temporal segments, though discovered without supervision, nicely capture semantically meaningful subtasks. They are consistent across demonstrations despite the differences in motions, e.g., the states of the tool during pushing are very different from each other.

Real Robot Experiments. We perform evaluations in the `Real-Kitchen` task to validate the practicality of BUDS for solving real-world manipulation tasks. Quantitatively, we evaluate 50 trials on varying initial configurations, achieving a 56% success rate. The performance is at the same level as our simulation evaluations, showing that BUDS generalizes well to real-world data and physical hardware. We also evaluate the most competitive baseline **CP** model on the real robot, which only achieved a 18% success rate. A consistent failure mode of this baseline is that the robot failed to place the pot correctly on the plate. We also qualitatively visualize the temporal segments of two demonstration sequences collected for the `Real-Kitchen` task in Figure 4. While our clustering-based segmentation algorithm is fully unsupervised, our quantitative inspection identifies consistent segments that can be interpreted with semantic meanings.

V. Conclusion

We presented BUDS, a hierarchical approach to tackling vision-based manipulation by discovering sensorimotor skills from unsegmented demonstrations. BUDS identifies recurring patterns from multi-task human demonstrations based on multi-sensory cues. Then it trains the skills on the recurring temporal segments with imitation learning and design a meta controller to compose these skills for tasks. The results show the effectiveness of BUDS in simulation and on real hardware. We also examine the impacts of different model designs through ablation studies.

While BUDS achieved superior performances over baselines in our evaluation tasks, it suffers from the common limitations of learning from offline demonstration datasets. One future direction is to improve its performance with the robot's online experiences. Another limitation of our current approach is the need of task-specific meta controllers to compose the skills for individual tasks. For future work, we would like to develop planning methods that integrate these acquired skills with a high-level task planner, such that they can compose the skills to solve novel manipulation tasks without training a new meta controller.

ACKNOWLEDGEMENT

This work has taken place in the Robot Perception and Learning Group (RPL) and Learning Agents Research Group (LARG) at UT Austin. RPL research has been partially supported by NSF CNS-1955523, the MLL Research Award from the Machine Learning Laboratory at UT-Austin, and the Amazon Research Awards. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARO (W911NF-19-2-0333), DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

REFERENCES

[1] M. Andrychowicz *et al.*, "Hindsight experience replay," in *NIPS*, 2017.
[2] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *ICLR*, 2019.
[3] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.
[4] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, no. 1-3, pp. 139–159, 1991.
[5] V. Chu *et al.*, "Real-time multisensory affordance-based control for adaptive object manipulation," in *ICRA*, IEEE, 2019.
[6] T. G. Dietterich, "The MAXQ method for hierarchical reinforcement learning.," in *ICML*, vol. 98, 1998.
[7] Y. Duan *et al.*, "One-shot imitation learning," *arXiv:1703.07326*, 2017.
[8] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," *arXiv:1802.06070*, 2018.
[9] A. Farag, L. Lu, H. R. Roth, J. Liu, E. Turkbey, and R. M. Summers, "A bottom-up approach for pancreas segmentation using cascaded superpixels and (deep) image patch labeling," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 386–399, 2016.
[10] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg, "Multi-level discovery of deep options," *arXiv:1703.08294*, 2017.
[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] J. J. Gibson and L. Carmichael, *The senses considered as perceptual systems*, 1. Houghton Mifflin Boston, 1966, vol. 2.
[13] K. Gregor, D. J. Rezende, and D. Wierstra, "Variational intrinsic control," *arXiv:1611.07507*, 2016.
[14] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *CoRL*, 2020, pp. 1025–1037.
[15] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *ICLR*, 2018.
[16] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.
[17] G. E. Hovland, P. Sikka, and B. J. McCarragher, "Skill acquisition from human demonstration using a hidden markov model," in *ICRA*, vol. 3, 1996.
[18] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *ICRA*, IEEE, 2011.
[19] ——, "Integrated task and motion planning in belief space," *IJRR*, vol. 32, no. 9-10, pp. 1194–1227, 2013.
[20] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.
[21] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013.
[22] T. Kipf *et al.*, "Compositional imitation learning: Explaining and executing one task at a time," *arXiv:1812.01483*, 2018.
[23] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *NIPS*, vol. 22, pp. 1015–1023, 2009.
[24] G. D. Konidaris, S. Kuindersma, A. G. Barto, and R. A. Grupen, "Constructing skill trees for reinforcement learning agents from demonstration trajectories.," in *NIPS*, 2010.
[25] S. Krishnan *et al.*, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *IJRR*, 2017.
[26] V. C. Kumar, S. Ha, and C. K. Liu, "Expanding motor skills using relay networks," in *CoRL*, 2018.
[27] T. Lan, Y. Zhu, A. R. Zamir, and S. Savarese, "Action recognition by hierarchical mid-level action elements," in *ICCV*, 2015, pp. 4552–4560.
[28] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *ECCV*, 2018.
[29] H. Le, N. Jiang, A. Agarwal, M. Dudıék, Y. Yue, and H. Daumé, "Hierarchical imitation and reinforcement learning," in *ICML*, 2018, pp. 2917–2926.
[30] M. A. Lee *et al.*, "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *IEEE Transactions on Robotics*, 2020.
[31] A. Mandlekar *et al.*, "Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data," in *ICRA*, IEEE, 2020.
[32] A. Mandlekar *et al.*, "Learning to generalize across long-horizon tasks from human demonstrations," in *RSS*, 2020.
[33] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.
[34] M. N. Nicolescu and M. J. Matarić, "A hierarchical architecture for behavior-based robots," in *AAMAS*, 2002.
[35] S. Niekum, S. Osentoski, C. G. Atkeson, and A. G. Barto, "Online bayesian changepoint detection for articulated motion models," in *ICRA*, IEEE, 2015.
[36] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *IROS*, IEEE, 2012.
[37] S. Pirk, K. Hausman, A. Toshev, and M. Khansari, "Modeling long-horizon tasks as sequential interaction landscapes," *arXiv:2006.04843*, 2020.
[38] D. Precup, *Temporal abstraction in reinforcement learning*, 2000.
[39] M. S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen, "Temporally-weighted hierarchical clustering for unsupervised action segmentation," *arXiv:2103.11264*, 2021.
[40] S. Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient parameter-free clustering using first neighbor relations," in *CVPR*, 2019.
[41] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *ICML*, 2015.
[42] T. Shankar and A. Gupta, "Learning robot skills with temporal variational inference," in *ICML*, 2020, pp. 8624–8633.
[43] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," in *ICLR*, 2019.
[44] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner, "Taco: Learning task decomposition via temporal alignment for control," in *ICML*, 2018, pp. 4654–4663.
[45] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "Avid: Learning multi-stage tasks via pixel-level translation of human videos," *arXiv:1912.04443*, 2019.
[46] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Learning manipulation graphs from demonstrations using multimodal sensory signals," in *ICRA*, IEEE, 2018.
[47] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.
[48] D. Tanneberg, K. Ploeger, E. Rueckert, and J. Peters, "Skid raw: Skill discovery from raw trajectories," *RA-L*, vol. 6, no. 3, pp. 4696–4703, 2021.
[49] A. Tung *et al.*, "Learning multi-arm manipulation through collaborative teleoperation," in *ICRA*, 2021.
[50] A. S. Vezhnevets *et al.*, "Feudal networks for hierarchical reinforcement learning," in *ICML*, 2017.
[51] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
[52] D. Xu *et al.*, "Neural task programming: Learning to generalize across hierarchical tasks," in *ICRA*, IEEE, 2018.
[53] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," *arXiv:1810.11043*, 2018.
[54] J. M. Zacks and B. Tversky, "Event structure in perception and conception," *Psychological bulletin*, vol. 127, no. 1, p. 3, 2001.
[55] T. Zhang *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.
[56] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *CVPR*, 2019, pp. 850–859.
[57] Y. Zhu, J. Wong, A. Mandlekar, and R. Martıén-Martıén, "Robosuite: A modular simulation framework and benchmark for robot learning," *arXiv:2009.12293*, 2020.