2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

SPRINT: Scalable Policy Pre-Training via Language Instruction Relabeling

Jesse Zhang[1], Karl Pertsch[2,3], Jiahui Zhang[1], Joseph J. Lim[4]

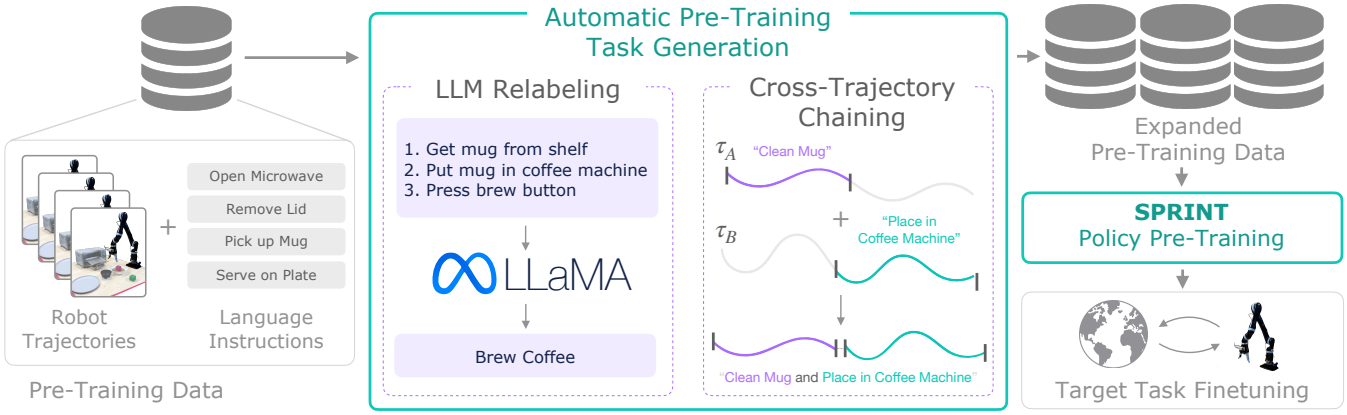

Fig. 1: SPRINT is a scalable approach for pre-training robot policies with a rich repertoire of skills while minimizing human annotation effort. Given a dataset of language-annotated trajectories for offline pre-training, SPRINT automatically expands the skill set via LLM-based **instruction relabeling** and **cross-trajectory skill chaining** to enable efficient finetuning on unseen target tasks.

***Abstract*— Pre-training robots with a rich set of skills can substantially accelerate the learning of downstream tasks. Prior works have defined pre-training tasks via natural language instructions, but doing so requires tedious human annotation of hundreds of thousands of instructions. Thus, we propose SPRINT, a scalable offline policy pre-training approach which substantially reduces the human effort needed for pre-training a diverse set of skills. Our method uses two core ideas to automatically expand a base set of pre-training tasks: instruction relabeling via large language models and cross-trajectory *skill chaining* with offline reinforcement learning. As a result, SPRINT pre-training equips robots with a richer repertoire of skills that can help an agent generalize to new tasks. Experiments in a household simulator and on a real robot kitchen manipulation task show that SPRINT leads to substantially faster learning of new long-horizon tasks than previous pre-training approaches. Website at https://clvrai.com/sprint.**

I. Introduction

When humans learn a new task, e.g., how to cook a new dish, we rely on a large repertoire of previously learned *skills*, like "*chopping vegetables*" or "*boiling pasta*", that make learning more efficient. Similarly, much work in robot learning aims to equip robots with a set of useful skills for improving learning efficiency [1]–[6]. A common approach to acquiring a rich skill set is to pre-train policies on a wide range of tasks. Recent works have employed *language instructions* as a way for humans to manually define such tasks for policy training, typically via hindsight annotation of large, pre-collected robot experience datasets [7]–[10]. While the resulting policies show impressive capabilities, generalization to new tasks requires a *large* set of pre-trained skills and thus many pre-training tasks. As a result, prior works resorted to annotating robot trajectory datasets with *hundreds of thousands* of human instruction labels [9], limiting their application outside industrial contexts. Can we instead devise a pre-training approach that similarly equips robots with a wide repertoire of skills but *minimizes* the need for human task annotations?

We introduce SPRINT (**S**calable **P**re-training via **R**elabeling Language **IN**s**T**ructions), a scalable pre-training approach that equips robots with a large set of skills while substantially reducing human labeling effort (see Figure 1). Given an initial set of language-labeled pre-training tasks, SPRINT uses extensive *automated* relabeling to greatly expand this task set without additional human effort. Given a dataset of robot trajectories with initial language instruction annotations, we leverage two core ideas to grow the number of tasks. First, we leverage the rich knowledge captured in large language models (LLMs) to iteratively combine consecutive language instructions into more complex tasks, e.g., "*place mug in coffee machine*" and "*press brew button*" into "*make coffee*". Second, we propose a language-conditioned offline reinforcement learning (RL) objective that "stitches" multiple trajectory segments from the data to form new tasks, a process we call "skill chaining" since it allows the policy to learn longer-horizon skills. Through the combination of both techniques, SPRINT creates a richer pre-training task set that can help the agent generalize to new tasks. We demonstrate that SPRINT-pre-trained robots can leverage their resulting larger skill repertoire to more efficiently learn new downstream tasks.

In summary, our contributions are threefold: (1) we propose SPRINT, a scalable pre-training approach for robot policies that minimizes human task annotation effort via LLM-based aggregation and cross-trajectory skill chaining, (2) we introduce ALFRED-RL, an RL benchmark for the

[1]University of Southern California, [2]UC Berkeley, [3]Stanford University,[4]Korea Advanced Institute of Science and Technology

popular ALFRED household task simulator [11], to test our pre-trained agents on a rich set of long-horizon, semantically meaningful tasks, (3) we demonstrate that policies pre-trained with SPRINT learn downstream tasks more efficiently than prior pre-training approaches, both on challenging ALFRED tasks and in a real robot kitchen manipulation setup.

II. Related Work

Language in RL. There is a large body of work at the intersection of natural language processing and behavior learning for robotics, and the field has been further accelerated by the recent successes in training large, general-purpose language models. Language has been used to structure agents' representations [12], [13], learn reward functions [14], guide task learning via recipes [15], [16] and perform long-horizon planning [17]–[20]. Another line of work has used language to define a wide range of tasks for pre-training policies, resulting in impressive generalization capabilities [8]–[10]. Yet, these works require collecting hundreds of thousands of costly human language instructions. Our approach SPRINT builds on this line of work but introduces two novel objectives for *automatic relabeling* of training task instructions, thereby substantially reducing the amount of human labeling required for successful pre-training. Prior works have also investigated automated language instruction generation [21]–[23], but they focus on online learning and make assumptions that are hard to scale, e.g., hand-defined grammars [21] or privileged state information [22], [23]. In contrast, we perform *offline* pre-training and use large language models for *scalable* task generation.

Pre-training Policies for RL. Developing policy pre-training approaches for faster downstream learning has been investigated for many years [24]–[26]. Recent advances in offline RL [27] enabled approaches that can pre-train agents offline and effectively finetune them on online tasks [28]–[31]. However, these approaches require target-task reward annotations on the pre-training data and the resulting policies are only pre-trained to solve the target task. Meta-RL approaches, on the other hand, pre-train on a range of tasks and thus allow fast adaptation to *unseen* downstream tasks [32]–[35], yet require the tedious manual definition of pre-training tasks by experts. To avoid manual task design, other works have explored unsupervised pre-training approaches based on behavior diversification [36]–[38], extraction of behavior priors from offline agent experience [5], [39], [40] or goal state reaching [41], [42]. Closest to ours, Chebotar, Hausman, Lu, *et al.* [42] proposes an objective that randomly selects states to chain together existing trajectories, while we propose a language skill chaining objective that allows SPRINT to execute new, composite language instructions. Such unsupervised pre-training approaches [42] learn skill repertoires without clear meaning, which, as we demonstrate in Section IV, lead to worse downstream task transfer.

Pre-trained Models for Data Augmentation. Obtaining robot (pre-)training data at scale is costly. Thus, recent works have explored using world knowledge captured in large pre-trained models for enriching robot learning datasets, e.g., by increasing the visual diversity of trajectories [43]–[45] or annotating unlabeled data [46]. Our approach similarly leverages pre-trained (language) models for automated data augmentation. By investigating an orthogonal augmentation direction, aggregation and chaining of natural language instructions, SPRINT is complementary to these methods.

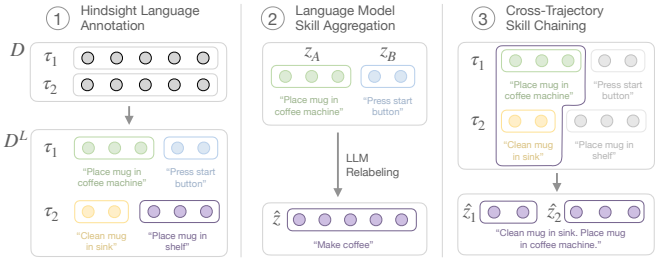

Fig. 2: SPRINT overview. We assume access to a dataset of agent experience with language instructions for the performed skills **(1)**. Collecting such instructions with human hindsight annotation is a flexible yet costly approach for defining pre-training tasks. Thus, SPRINT introduces two approaches for automatically growing the set of pre-training tasks without additional human effort: **(2)** by aggregating language instructions with an LLM and adding the relabeled trajectories back into the pre-training dataset (Section III-B), **(3)** by performing cross-trajectory chaining of skills to enable pre-training of skills that are unseen in the offline agent experience (Section III-C).

III. SPRINT: Scalable Policy Pre-Training with Language Instructions

In this paper, we propose SPRINT (**S**calable **P**re-training via **R**elabeling Language **IN**s**T**ructions), an approach for pre-training robot policies that equips them with a rich repertoire of skills to enable efficient finetuning on unseen tasks. Following prior work on agent pre-training, SPRINT assumes access to a large offline dataset $\mathcal{D}$ of agent experience [4], [5], [42], [47]–[49], collected, e.g., from prior RL runs or via teleoperation. We further assume that the data is annotated with an initial set of natural language task instructions, e.g., "*put a mug in the coffee machine*" or "*push the brew button*", that can be collected *in hindsight* via platforms like Amazon Mechanical Turk [8], [11]. Given a sequence τ of states and actions from the dataset $\mathcal{D}$, annotators can label sub-trajectories $\tau_1 = [s_0, a_0, s_1, \dots], \tau_2 = \dots$ with free-form language descriptions $z_1, z_2, \dots$ of the skills executed in the respective sub-trajectories (see Figure 2, left), resulting in a *language-annotated* dataset $\mathcal{D}^L$.

Approach Overview. SPRINT equips policies with a diverse repertoire of skills via language-instruction-conditioned offline RL: given a natural language task description z, the policy $\pi(a|s, z)$ is rewarded for successfully executing the instruction (Section III-A). Intuitively, the richer the set of task instructions during pre-training, the more skills the policy will learn and the more downstream tasks it can finetune on efficiently. Thus, SPRINT introduces two

approaches for increasing the scale and diversity of the pre-training task instructions without requiring additional costly human inputs. Firstly, SPRINT leverages pre-trained language models to aggregate consecutive instructions into new tasks (Figure 2, middle, Section III-B). Secondly, SPRINT introduces an objective for cross-trajectory skill-chaining via offline RL that generates novel instruction chains *across different trajectories* (Figure 2, right, Section III-C). SPRINT pre-trains policies on the combined set of tasks and thereby equips them with a richer skill repertoire. In our experiments (Section IV) we demonstrate that this leads to more effective learning of new tasks.

A. Instruction-Conditioned Offline RL

To pre-train our policy π with the natural language instruction dataset $\mathcal{D}^L$, we take inspiration from goal-conditioned RL [42], [50], [51]: instead of rewarding the policy for reaching goal states, we condition our policy $\pi(a|s,z)$ on *language instructions* z from $\mathcal{D}^L$ and provide a scalable sparse reward $R(s,a,z)$ to the agent for reaching the end-state s_T of the sub-trajectory. Formally, we define the reward as:

$$R(s,a,z) = \begin{cases} 1, & \text{for } s = s_T \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

We train our policy $\pi(a|s,z)$ to maximize this reward with offline RL [27] using an instruction-conditioned critic $Q(s,a,z)$. Specifically, we use Implicit Q-Learning [31] as it is performant and easy to tune.

B. Language-Model-Based Instruction Aggregation

LLM Prompt Example

Summarize the following steps.

1: Pick up the tomato slice.
2: Heat it up in the microwave.
Summary: Microwave a tomato slice.

1: [SKILL 1]
2: [SKILL 2]
...
Summary:

Fig. 3: A shortened example of the LLM prompt.

Large language models (LLMs), trained on massive corpora of internet text data, have been shown to be effective at performing a variety of tasks – from question answering to program synthesis – when prompted with relevant text [52]–[58]. Here we use LLMs to *aggregate*, i.e., paraphrase, the existing language instructions in $\mathcal{D}^L$ (see Figure 2, middle). Given a trajectory that contains multiple sub-trajectories, we can aggregate adjacent sub-trajectories into a longer trajectory and relabel its natural language annotation with a summary of the individual instructions generated by the LLM, thereby generating a new *higher-level* pre-training task that encompasses instructions from multiple sub-trajectories.[1] We use a simple summarization prompt to instruct the language model (see Figure 3). Specifically, we aggregate with LLAMA-13B [59], an open-source 13 billion parameter LLM which is able to retain important information from individual instructions in the overall summary. Like in Section III-A, the reward for this new aggregated sub-trajectory is 1 at the last transition and 0 otherwise. For example, we prompt the LLM to summarize the two skills (z_1 : "*Put a mug in the coffee machine*," z_2 : "*Push the brew button*"), resulting in a new annotation $\hat{z}_{1:2}$ describing both skills (e.g., "*Make coffee*"). We then add the new trajectory back to our dataset $\mathcal{D}^L$. Using this technique, we generate new language annotations for all combinations of consecutive sub-trajectories in our dataset. In practice, this increases the number of task instructions by 2.5x in ALFRED and 2x in our robot manipulation dataset (see Section IV).

C. Cross-Trajectory Chaining

In addition to generating new pre-training tasks composed of behaviors within the *same* trajectory (Section III-B), we also want to be able to generate pre-training tasks containing behaviors across *different* trajectories. For example, if trajectory (A) shows cleaning the mug in the sink while trajectory (B) starts with placing the mug in the coffee machine, the agent should be able to learn to clean the mug in the sink and then place it in the coffee machine (see Figure 2, right), thus learning long-horizon behaviors that are unseen in the training data. Agents trained with standard offline RL can implicitly combine tasks described from multiple trajectories into longer-horizon behaviors via value propagation, i.e., perform "stitching" [27]. In our case of *instruction-conditioned* offline RL, values do not naturally propagate from trajectory (B) back to trajectory (A) due to the different language instruction conditionings for the critic $Q(s,a,z_A)$ and $Q(s,a,z_B)$. However, we can actively add "chaining examples" [42], which encourage learning longer-horizon behaviors, to our training dataset by first *combining language instructions* and then appropriately *relabeling rewards*. To build such chaining examples, we first sample two sub-trajectories τ_{z_A} and τ_{z_B} from *different* trajectories (see Figure 2, right). Next, we create an aggregate instruction $\hat{z}$ which indicates that the agent first finishes (A) and then finishes (B), e.g., "*clean the coffee mug (A) and place it in the coffee machine (B).*"[2]

Unlike in Section III-B, we cannot simply concatenate the two trajectories together and relabel the reward of the last transition to 1. Since we sampled the two sub-trajectories at random, the last state of the first, s_{T_A}, does not directly transition into the first state of the second. To solve this issue, we relabel both τ_{z_A} and τ_{z_B} with the aggregate instruction $\hat{z}$ and treat them as *separate* trajectories with appropriately labeled rewards. For transitions in τ_{z_B}, we simply relabel the last transition with a reward of 1 to be

[1]Other relabeling operations, such as splitting an instruction into lower-level instructions, can also be performed by the LLM. However, such operations require grounding the LLM in the agent's observations to determine sub-trajectory split points. We leave investigating this to future work.

[2]Note that we could generate $\hat{z}$ using the same LLM summarization as in Section III-B. Yet we found the resulting summaries to often be confusing since randomly paired instructions *from different trajectories* can rarely be summarized meaningfully. We got the best empirical results by simply concatenating the sampled instructions with the word "and". Note that we perform chaining on both the original trajectories and those generated by LLM aggregation in Section III-B.

consistent with the 0-1 rewards in Sections III-A and III-B. Meanwhile, we would like to relabel the reward of the last, terminal transition in τ_{z_A} so that the learned Q-value for this transition, $Q(s_{T_A}, a_{T_A}, \hat{z})$, will also be consistent with the prior labeling schemes. What reward should we use here?

Recall that Q-functions trained for sparse reward (Eq. 1) intuitively represent a value proportional to the probability of reaching goal state s_{T_z} at time T [42], [60]:

$$Q^{\pi}(s_t, a_t, z) = \mathbb{E}[\sum_{t'=t} \gamma^{t'-t} R(s_{t'}, a_{t'}, z)] \\ = \mathbb{E}\left[\gamma^{T-t} \mathbb{1}\left[s_T = s_{T_z}\right]\right] \propto P^{\pi}(s_T = s_{T_z} | s_t, a_t). \tag{2}$$

where $\gamma \in (0, 1)$ denotes the discount factor. Following this intuition, the Q-value learned for the last transition of (A) should be proportional to the probability of finishing *the remainder* of the combined task $\hat{z}$, i.e., proportional to the likelihood of finishing (B) from s_{T_A} when taking action a_{T_A}. Following Eq. 2, $Q(s_{T_A}, a_{T_A}, z_B)$ *is* this probability. Intuitively, if there are transitions in the dataset which indicate that finishing (B) from s_{T_A} by taking action a_{T_A} is possible, then this Q-value should be non-zero and the agent will learn to chain (A) and (B) together through their aggregate instruction $\hat{z}$. Our reward labels for the two trajectories with aggregate instruction $\hat{z}$ are therefore:

$$R(s, a, \hat{z}) = \begin{cases} 1, & \text{for } s = s_{T_B} \\ Q(s, a, z_B), & \text{for } s = s_{T_A} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Since Q changes during training, we compute the rewards in Eq. 3 in each batch while training. Full SPRINT pseudocode is listed in Alg. 1.

Algorithm 1 SPRINT Algorithm

Require: Dataset $\mathcal{D}^L$ w/ language instruction labels, LLM
AGGREGATESKILLS($\mathcal{D}^L$, LLM)
while not converged **do**
  $\tau_z \leftarrow \mathcal{D}^L$: Sample an annotated skill (sub-)trajectory
  Train offline RL on τ_z
  $\tau_{\text{agg}_1}, \tau_{\text{agg}_2} \leftarrow$ CHAINSKILLS($\mathcal{D}^L$, LLM)
  Train offline RL on $\tau_{\text{agg}_1}, \tau_{\text{agg}_2}$
procedure AGGREGATESKILLS($\mathcal{D}^L$, LLM) ▷ Sec. III-B
  for composite trajectory $\tau_{\hat{z}}$ in $\mathcal{D}^L$ **do**
    for all adjacent sub-trajectories $\left[\tau_{z_i} ... \tau_{z_j}\right]$ **do**
      Assign name from LLM: LLM$(z_i ... z_j) = \hat{z}_{i:j}$
      $\tau_{\hat{z}_{i:j}} \leftarrow$ Concat $\left[\tau_{z_i}, ..., \tau_{z_j}\right]$ and relabel with $\hat{z}_{i:j}$ and reward from Eq. 1.
      $\mathcal{D}^L = \mathcal{D}^L \cup \{\tau_{\hat{z}_{i:j}}\}$
procedure CHAINSKILLS($\mathcal{D}^L$, LLM) ▷ Sec. III-C
  Sample random $\tau_{z_1}, \tau_{z_2} \sim \mathcal{D}^L$
  Assign new name : $\hat{z} =$ "$\{z_1\}$ and $\{z_2\}$"
  $\tau_{\text{agg}_1} \leftarrow$ Relabel τ_{z_1} w/ $\hat{z}$ and rew from Eq. 3
  $\tau_{\text{agg}_2} \leftarrow$ Relabel τ_{z_2} w/ $\hat{z}$ and rew from Eq. 3
  return $\tau_{\text{agg}_1}, \tau_{\text{agg}_2}$

IV. EXPERIMENTS

In our experiments, we investigate how well an agent pre-trained with SPRINT performs on challenging unseen tasks. Thus, we answer the following questions: (1) Does SPRINT enable more efficient finetuning on unseen target tasks than previous pre-training approaches? (2) Can SPRINT agents execute unseen language instructions zero-shot? (3) Does augmentation via *language* relabeling lead to more generalizable policies than through goal image relabeling?

A. Experimental Setup

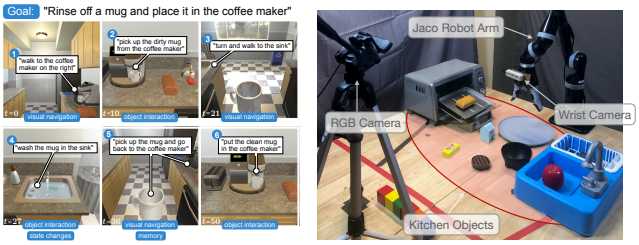

Fig. 4: **Left**: ALFRED provides a rich set of long-horizon, meaningful tasks and a dataset of 6.6k language-annotated demos. We introduce the ALFRED-RL Benchmark which tests finetuning of RL agents on unseen tasks and scenes. **Right**: Our Jaco robot arm with RGB image-based control.

We evaluate our approach on two image-based environments (see Figure 4): ALFRED-RL, a simulated RL benchmark we introduce, and a real robot kitchen.

ALFRED-RL. Our goal is to compare different pre-training approaches on a diverse set of semantically meaningful, long-horizon tasks. Yet, existing multi-task RL environments typically evaluate only on short-horizon or semantically meaningless tasks [7], [61]. Thus, we introduce a new RL benchmark based on the ALFRED household task simulator [11]. While ALFRED abstracts away low-level agent control into discrete actions like "pick up" or "turn left," its 100+ rich indoor scenes with many interactable objects allow to evaluate an agent's capabilities for solving long-horizon household tasks from a rich task distribution. The original benchmark focuses on imitation learning, but we extend it to support training RL agents through a gym interface with egocentric RGB observations and an action space consisting of 12 discrete action choices and 82 interactable object types [62]. We create three evaluation task sets that test progressively more challenging axes of generalization: ***EVAL***$_{\textit{INSTRUCT}}$ uses unseen human-generated instructions on familiar scenes, ***EVAL***$_{\textit{LENGTH}}$ uses tasks that are longer than any observed in pre-training, testing "stitching" capabilities, and ***EVAL***$_{\textit{SCENE}}$ uses tasks in unseen floorplans.

Real-World Robot Kitchen Manipulation. To evaluate pre-training approaches on end-to-end *low-level* robot control, we design a set of stylized kitchen manipulation tasks with a Kinova Jaco 2 robot arm. The policy's inputs are RGB images from a wrist-mounted and a third-person camera and it produces continuous end-effector (3-dim) displacement actions and a discrete gripper open/stay/close action at a control frequency of 10Hz. We collect a dataset of 329 long-horizon trajectories via human teleoperation with the setup from Dass, Yapeter, Zhang, *et al.* [63], each consisting of

multiple language-annotated sub-trajectories like *"pick up the apple fruit,"*, *"place the black bowl in the dish rack,"* etc. For evaluation, we construct three long-horizon tasks, sequencing 2 to 8 "primitive skills" like the ones mentioned above, in environment configurations that are unseen in the pre-training data. We collect 25 demonstrations for each of the three tasks to evaluate offline fine-tuning performance of different pre-trained policies.

Comparisons. We compare SPRINT against common policy pre-training approaches, behavioral cloning and offline goal-conditioned RL: **Language-conditioned BC (L-BC)** [8], [64]: Behavior cloning (BC) conditioned on the individual language instructions; **Episodic Transformers (ET)** [62]: BC conditioned on sequences of language instructions – ET is the best-performing end-to-end learned policy on the ALFRED leaderboard that *does not* use privileged domain knowledge like hand-engineered policies or voxel maps; **Actionable Models (AM)** [42]: Goal-conditioned offline RL with randomly sampled goal observations from the same training data as SPRINT. We also evaluate **SayCan** [18]: Top-down LLM planning over pre-trained, language-conditioned policies.

All methods use the same architectures, hyperparameters, and training data $\mathcal{D}^L$ where possible. In ALFRED-RL, all methods use the same language token conditioned transformer policy architecture proposed by Pashevich, Schmid, and Sun [62] specifically for ALFRED; we use a transformer critic model with a separate output head for each critic, following Snell, Kostrikov, Su, *et al.* [65]. On the real robot, all methods use an RNN architecture with "action chunking" [66] proposed by Dass, Pertsch, Zhang, *et al.* [67]. Results are means and standard deviations over 3 seeds.

B. SPRINT Solves Long-Horizon Tasks Zero-Shot

We first test the effectiveness of SPRINT's pre-training by analyzing zero-shot performance across 100 unseen tasks in the $EVAL_{INSTRUCT}$ evaluation set. We report results in Figure 5 (left). Our approach, SPRINT, achieves **2-8x** higher zero-shot task performance than prior pre-training approaches AM and L-BC. Even though ET also trains to condition on long-horizon instruction sequences like SPRINT, ours still outperforms it overall by 2x. To better understand the differences between the methods, we report the breakdown of returns by length of the evaluation task in Figure 5 (middle). We find that all methods except AM achieve similar performance on length 1 tasks. However, on long-horizon tasks, SPRINT achieves much higher returns than all baselines since it can leverage the LLM to automatically generate longer-horizon pre-training tasks. In contrast, L-BC trains only on the human-provided, shorter-horizon annotations and thus cannot zero-shot perform longer tasks. Meanwhile SayCan, with the same LLM as used for SPRINT, commonly generates incorrect plans that lead to incorrect behaviors. This problem is exacerbated on longer tasks; the chance of planning errors increases with task length. In contrast, SPRINT's pre-training enables more robust long-horizon task execution. Similar to our approach, AM trains to reach long-horizon goals during pre-training but the results in Figure 5 (left) show that its pre-training with goal-state conditioning is *less* effective than our language-conditioned pre-training. These results also hold for the $EVAL_{LENGTH}$ task set, which tests generalization to task horizons beyond the ones seen during training. On these most challenging tasks, SPRINT outperforms the best baseline by 2.5x.

C. SPRINT Finetunes Effectively in Unseen Environments

ALFRED-RL. We test SPRINT's finetuning performance to unseen tasks on the most challenging $EVAL_{SCENE}$ task set in unseen household floor plans with 50k environment interactions. This corresponds to a realistic scenario in which an agent is placed in a new household environment and needs to leverage skills learned during pre-training to solve new tasks with minimal environment interaction. To implement finetuning for SPRINT and AM, we condition the policy on a language instruction or goal image from the target task respectively and then run IQL with online data collection. For L-BC and ET, we first pre-train a language-conditioned critic with IQL on the pre-training dataset and then finetune both the policy and critic with online IQL. Sparse, per-subtask completion reward is given to agents during fine-tuning.

We report finetuning results in Figure 5 (right). SPRINT quickly achieves higher downstream task return than the best prior work. Specifically, L-BC converges to lower peak performance than SPRINT and ET performs poorly, perhaps because transferring from instruction sequences to high-level task descriptions is challenging. Meanwhile, AM performs similarly to L-BC, possibly because unseen goal states are more difficult to learn from. In contrast, SPRINT's pre-training with language conditioning allows for effective transfer even to unseen environments since the semantics of the tasks transfer well: the language description "*place cup in coffee machine*" transfers to many environments while the goal image for the same task might look very different. Thus, pre-training with language instructions can enable better transfer for learning tasks in new environments than pre-training to reach goal states. SayCan performs poorly due to both planning and execution errors as it does not fine-tune. We also attempted to first fine-tune SayCan's primitive policies before running SayCan, but its performance did not change as fine-tuning its policies on high-level task instructions did not improve primitive instruction execution.

Real Robot. We also measure finetuning performance on an unseen environment on our real robot setup. We evaluate on three tasks consisting of 2, 4, and 8 subgoals, respectively:

1) *Bake bread in the oven*: The robot must (1) pick up the bread, (2) place it in the oven.
2) *Serve heated milk in the bowl*: The robot must (1) pick up the milk, (2) place it in the black bowl, (3) pick up the bowl with milk, (4) place the bowl in the oven.
3) *Serve milk in the bowl and butter and baked bread in the plate*: (1) pick up milk, (2) put it in the black bowl, (3) pick up butter, (4) put it in the plate, (5) pick up the bread, (6) bake it in the oven, (7) pick up the bread from the oven, (8) place the bread in the plate.

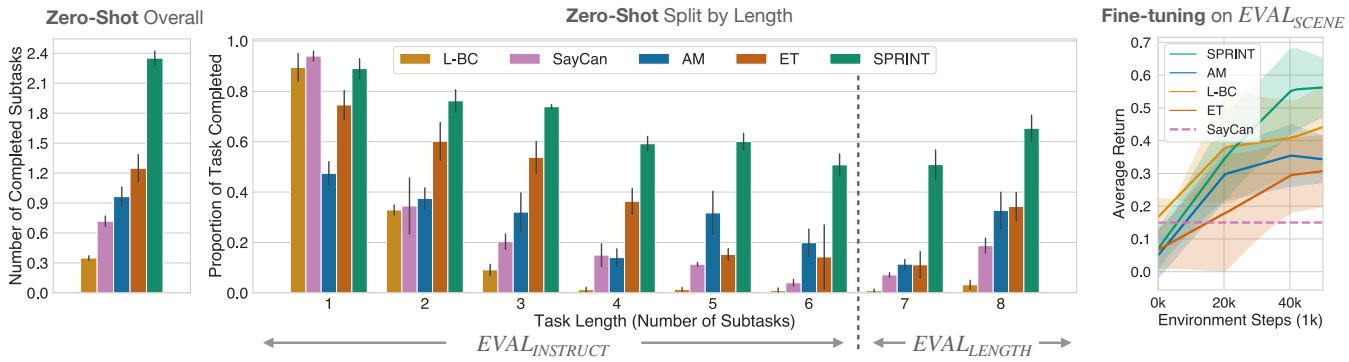

Fig. 5: ALFRED-RL evaluation results. **Left**: Zero shot performance on $\text{EVAL}_{\text{INSTRUCT}}$ and $\text{EVAL}_{\text{LENGTH}}$. SPRINT is able to complete substantially more subtasks than prior approaches. **Middle**: Breakdown of performance by task length. SPRINT performs well on challenging, long tasks. **Right**: Finetuning performance in unseen floor plans of $\text{EVAL}_{\text{SCENE}}$. SPRINT learns in new floorplans more effectively by reaching higher performance.

TABLE I: Success rates and number of subgoals completed after fine-tuning on the tabletop arrangement displayed on the left with unseen object combinations over 5 trials.

	Length 2		Length 4		Length 8	
Method	Success	# Tasks	Success	# Tasks	Success	# Tasks
SPRINT	**100%**	**2.0**	**60%**	**3.4**	**40%**	**6.2**
L-BC Comp.	**100%**	**2.0**	40%	2.8	20%	5.2
L-BC	**100%**	**2.0**	40%	0.4	0%	2.0
No pre-train	0%	1.0	0%	0.0	0%	0.0

We collect 25 demonstrations per task for offline fine-tuning. We compare SPRINT against **L-BC**, a version of L-BC trained on full sequences of concatenated language instructions (**L-BC Composite**), and a method that is trained only on the downstream task demonstrations (**No pre-train**).

Results in Table I demonstrate that *No Pre-train* performs poorly, indicating that pre-training is necessary. SPRINT achieves the best success rates and completes the most subgoals on all tasks. Compared to L-BC Composite, SPRINT achieves higher returns and success rates on challenging, longer tasks. See Figure 6 for an example evaluation.

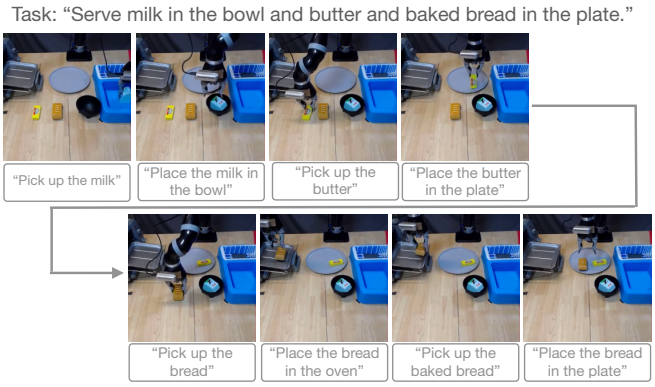

Fig. 6: Successful rollout of a SPRINT agent offline finetuned for the task above with object combinations not in the pre-training data. SPRINT solves all 8 tasks in sequence.

D. Ablation Studies

We verify the effectiveness of the components of our approach, with the following ablations: **SPRINT w/o chain** removes cross-trajectory chaining (Section III-C), instead trains only on within-trajectory human-provided and LLM-aggregated tasks; **SPRINT Naïve Chain** replaces Q-value reward labels when chaining with 0's to test naïve offline RL "stitching" with language instruction-conditioned agents. **SPRINT w/o LLM-agg** additionally removes LLM aggregation (Section III-B) and chaining, thus training only on the human-provided task annotations. We report zero-shot ALFRED evaluation results in Table II: each component of our approach improves zero-shot evaluation performance. There is a large performance loss when removing LLM aggregation, underlining the importance of leveraging LLMs for automatically generating long-horizon training tasks. We also see that naïve chaining is worse than not chaining.

TABLE II: Ablations. SPRINT achieves the highest return.

Ablation	$\text{EVAL}_{\text{INSTRUCT}}$	$\text{EVAL}_{\text{LENGTH}}$
SPRINT (ours)	**1.94 ± 0.04**	**4.40 ± 0.39**
SPRINT w/o Chain	1.75 ± 0.11	3.98 ± 0.29
SPRINT Naïve Chain	0.50 ± 0.04	0.26 ± 0.05
SPRINT w/o LLM-agg	0.37 ± 0.01	0.15 ± 0.10

V. Discussion and Acknowledgements

We presented SPRINT, an approach for scalable agent pre-training that automatically generates training tasks for offline RL via LLM relabeling and cross-trajectory skill chaining. SPRINT pre-training leads to higher zero-shot and finetuning performance on diverse household tasks in the ALFRED simulator and on real-robot kitchen manipulation tasks.

This work was supported by a USC Viterbi Fellowship, Institute of Information & Communications Technology Planning & Evaluation (IITP) grants (No.2019-0-00075, Artificial Intelligence Graduate School Program, KAIST; No.2022-0-00077, AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data, No.2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation), a National Research Foundation of Korea (NRF) grant (NRF-2021H1D3A2A03103683) funded by the Korean government (MSIT) and Samsung Electronics Co., Ltd (IO220816-02015-01).

REFERENCES

[1] 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, pp. 181–211, 1999.

[2] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*. Springer Tokyo, 2006.

[3] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *ICLR*, 2018.

[4] C. Lynch, M. Khansari, T. Xiao, *et al.*, "Learning latent plans from play," in *CoRL*, 2020.

[5] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," in *CoRL*, 2020.

[6] T. Haarnoja, B. Moran, G. Lever, *et al.*, *Learning agile soccer skills for a bipedal robot with deep reinforcement learning*, 2023. arXiv: 2304.13653.

[7] 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 (RA-L)*, vol. 7, no. 3, pp. 7327–7334, 2022.

[8] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," *Robotics: Science and Systems*, 2021. [Online]. Available: https://arxiv.org/abs/2005.07648.

[9] C. Lynch, A. Wahid, J. Tompson, *et al.*, "Interactive language: Talking to robots in real time," *arXiv preprint arXiv:2210.06407*, 2022.

[10] A. Brohan, N. Brown, J. Carbajal, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[11] M. Shridhar, J. Thomason, D. Gordon, *et al.*, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [Online]. Available: https://arxiv.org/abs/1912.01734.

[12] J. Andreas, D. Klein, and S. Levine, "Learning with latent language," in *North American Chapter of the Association for Computational Linguistics*, 2017.

[13] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," in *CoRL*, 2022.

[14] L. Fan, G. Wang, Y. Jiang, *et al.*, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," *arXiv preprint arXiv: Arxiv-2206.08853*, 2022.

[15] S. R. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay, "Reinforcement learning for mapping instructions to actions," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009, pp. 82–90.

[16] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *International Conference on Machine Learning*, 2017.

[17] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *arXiv preprint arXiv:2201.07207*, 2022.

[18] M. Ahn, A. Brohan, N. Brown, *et al.*, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[19] W. Huang, F. Xia, T. Xiao, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *arXiv preprint arXiv:2207.05608*, 2022.

[20] I. Singh, V. Blukis, A. Mousavian, *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *ICRA*, 2023.

[21] C. Colas, T. Karch, N. Lair, *et al.*, "Language as a cognitive tool to imagine goals in curiosity driven exploration," *Advances in Neural Information Processing Systems*, 2020.

[22] G. Cideron, M. Seurin, F. Strub, and O. Pietquin, "Higher: Improving instruction following with hindsight generation for experience replay," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2020, pp. 225–232.

[23] S. Li, X. Puig, C. Paxton, *et al.*, "Pre-trained language models for interactive decision-making," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=FWMQYjFso-a.

[24] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 2, 2002, pp. 1398–1403.

[25] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 2397–2403.

[26] T. Hester, M. Vecerik, O. Pietquin, *et al.*, "Deep q-learning from demonstrations," in *Association for the Advancement of Artificial Intelligence*, 2018.

[27] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[28] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, *Advantage-weighted regression: Simple and scalable off-policy reinforcement learning*, 2019. DOI: 10.48550/ARXIV.1910.00177. [Online]. Available: https://arxiv.org/abs/1910.00177.

[29] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine, "Cog: Connecting new skills to past experience with offline reinforcement learning," *Conference on Robot Learning*, Nov. 2020. arXiv: 2010.14500 [cs.LG].

[30] A. Nair, M. Dalal, A. Gupta, and S. Levine, "Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[31] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=68n2s9ZJWF8.

[32] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL ^{2}: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[33] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *ICML*, 2017.

[34] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *ICML*, 2019.

[35] T. Nam, S.-H. Sun, K. Pertsch, S. J. Hwang, and J. J. Lim, "Skill-based meta-reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2022.

[36] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel, "Variational option discovery algorithms," *arXiv*, 2018.

[37] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *ICLR*, 2019.

[38] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," *arXiv*, vol. abs/1907.01657, 2019.

[39] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "Opal: Offline primitive discovery for accelerating offline reinforcement learning," *arXiv preprint arXiv:2010.13611*, 2020.

[40] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," *ICLR*, 2021.

[41] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak, *Discovering and achieving goals via world models*, 2021.

[42] Y. Chebotar, K. Hausman, Y. Lu, *et al.*, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 1518–1528. [Online]. Available: https://proceedings.mlr.press/v139/chebotar21a.html.

[43] T. Yu, T. Xiao, A. Stone, *et al.*, "Scaling robot learning with semantically imagined experience," in *arXiv preprint arXiv:2302.11550*, 2023.

[44] Z. Chen, S. Kiami, A. Gupta, and V. Kumar, "Genaug: Retargeting behaviors to unseen situations via generative augmentation," in *RSS*, 2023.

[45] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar, *Cacti: A framework for scalable multi-task multi-scene visual imitation learning*, 2023. arXiv: 2212.05711 [cs.RO].

[46] T. Xiao, H. Chan, P. Sermanet, *et al.*, "Robotic skill acquistion via instruction augmentation with vision-language models," in *Proceedings of Robotics: Science and Systems*, 2023.

[47] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," *CoRL*, 2019.

[48] F. Ebert, Y. Yang, K. Schmeckpeper, *et al.*, "Bridge data: Boosting generalization of robotic skills with cross-domain datasets," in *RSS*, 2022.

[49] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Demonstration-guided reinforcement learning with learned skills," in *CoRL*, 2021.

[50] L. P. Kaelbling, "Learning to achieve goals," in *IN PROC. OF IJCAI-93*, Morgan Kaufmann, 1993, pp. 1094–1098.

[51] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 1312–1320. [Online]. Available: https://proceedings.mlr.press/v37/schaul15.html.

[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805.

[53] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. DOI: 10.48550/ARXIV.2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165.

[54] B. Wang and A. Komatsuzaki, *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*, https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[55] J. W. Rae, S. Borgeaud, T. Cai, *et al.*, *Scaling language models: Methods, analysis & insights from training gopher*, 2021. DOI: 10.48550/ARXIV.2112.11446. [Online]. Available: https://arxiv.org/abs/2112.11446.

[56] J. Hoffmann, S. Borgeaud, A. Mensch, *et al.*, *Training compute-optimal large language models*, 2022. DOI: 10.48550/ARXIV.2203.15556. [Online]. Available: https://arxiv.org/abs/2203.15556.

[57] S. Zhang, S. Roller, N. Goyal, *et al.*, *Opt: Open pre-trained transformer language models*, 2022. DOI: 10.48550/ARXIV.2205.01068. [Online]. Available: https://arxiv.org/abs/2205.01068.

[58] A. Chowdhery, S. Narang, J. Devlin, *et al.*, *Palm: Scaling language modeling with pathways*, 2022. DOI: 10.48550/ARXIV.2204.02311. [Online]. Available: https://arxiv.org/abs/2204.02311.

[59] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[60] B. Eysenbach, T. Zhang, R. Salakhutdinov, and S. Levine, *Contrastive learning as goal-conditioned reinforcement learning*, 2022. DOI: 10.48550/ARXIV.2206.07568. [Online]. Available: https://arxiv.org/abs/2206.07568.

[61] T. Yu, D. Quillen, Z. He, *et al.*, *Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning*, 2019. arXiv: 1910.10897 [cs.LG].

[62] A. Pashevich, C. Schmid, and C. Sun, "Episodic Transformer for Vision-and-Language Navigation," in *ICCV*, 2021.

[63] S. Dass, J. Yapeter, J. Zhang, *et al.*, *Clvr jaco play dataset*, version 1.0.0, 2023. [Online]. Available: https://github.com/clvrai/clvr_jaco_play_dataset.

[64] E. Jang, A. Irpan, M. Khansari, *et al.*, "BC-z: Zero-shot task generalization with robotic imitation learning," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=8kbp23tSGYv.

[65] C. Snell, I. Kostrikov, Y. Su, M. Yang, and S. Levine, *Offline rl for natural language generation with implicit language q learning*, 2023. arXiv: 2206.11871 [cs.CL].

[66] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[67] S. Dass, K. Pertsch, H. Zhang, Y. Lee, J. J. Lim, and S. Nikolaidis, *Pato: Policy assisted teleoperation for scalable robot data collection*, 2023. arXiv: 2212.04708 [cs.RO].