AdaDemo: Data-Efficient Demonstration Expansion for Generalist Robotic Agent

Tongzhou Mu[2,*], Yijie Guo[1], Jie Xu[1], Ankit Goyal[1], Hao Su[2], Dieter Fox[1,3], Animesh Garg[1,4]

Abstract— Encouraged by the remarkable achievements of language and vision foundation models, developing generalist robotic agents through imitation learning, using large demonstration datasets, has become a prominent area of interest in robot learning. The efficacy of imitation learning is heavily reliant on the quantity and quality of the demonstration datasets. In this study, we aim to scale up demonstrations in a data-efficient way to facilitate the learning of generalist robotic agents. We introduce AdaDemo (Adaptive Online Demonstration Expansion), a general framework designed to improve multi-task policy learning by actively and continually expanding the demonstration dataset. AdaDemo strategically collects new demonstrations to address the identified weakness in the existing policy, ensuring data efficiency is maximized. Through a comprehensive evaluation on a total of 22 tasks across two robotic manipulation benchmarks (RLBench and Adroit), we demonstrate AdaDemo's capability to progressively improve policy performance by guiding the generation of high-quality demonstration datasets in a data-efficient manner.

I. INTRODUCTION

The recent unprecedented success of language and vision foundation models [1], [2] have highlighted the importance of scaling up datasets as a key strategy for solving challenging tasks. This insight has similarly influenced the field of robotics, where the development of robotic foundation models through imitation learning, utilizing large datasets of demonstrations, has emerged as a prominent area of interest [3]–[6]. However, the success of these robotic foundation models is significantly dependent on the quantity, quality, and diversity of the demonstration data. Meanwhile, recent efforts of collecting large-scale robot demonstration datasets, such as RT-1 [4] and Open X-Embodiment [7], require extensive time and resources, involving years of data collection by numerous human teleoperators, which proves to be costly. Given these considerations, it is both crucial and timely to explore this question: how to *scaling up demonstrations in a data-efficient manner* for learning generalist robotic agents?

In this work, we delve into the direction of *actively expanding the demonstration dataset*. Imitation learning [8] is a widely-used method in training generalist robotic agents, which typically relies on datasets that are *pre-collected and static*. Despite various efforts to improve the demonstration datasets through data augmentation [9], [10] and dataset re-distribution methods [11], [12], such approaches remain fundamentally limited by the original dataset. Our study introduces a significant *paradigm shift*: rather than collecting the demonstration dataset once and using it forever, we
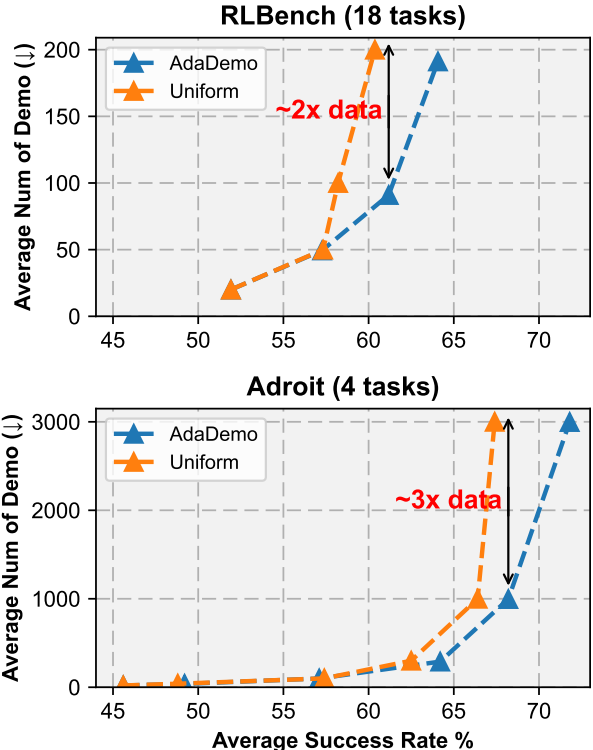
*Work is done at NVIDIA.
[1]NVIDIA, [2]UC San Diego, [3]University of Washington, [4]Georgia Tech.

RLBench (18 tasks)

Adroit (4 tasks)

Fig. 1: Comparison of data efficiency between **AdaDemo** (adaptively expanding the demo dataset) and **Uniform** (collecting more demonstrations uniformly). After achieving a mediocre success rate 57% on RLBench and 62% on Adroit, **Uniform** only gains slightly better success rate with a huge increase in demonstration numbers. While the baseline's performance plateaus, **AdaDemo** continues improving multi-task performance iteratively. Overall, it achieves better performance with only 1/2 the data on RLBench and 1/3 on Adroit. This data efficiency could translate into *substantial cost savings* in large-scale demonstration collection.

propose *actively and continuously expanding the dataset by collecting new demonstrations* to boost the performance of the learned agent. In the context of embodied AI, it is usually feasible to collect new demonstrations by interacting with the environments, especially in simulated environments, where the cost of interactions is relatively low.

We study the problem of data-efficient demonstration expansion in a multi-task visual policy learning setup, where the objective is to develop a single generalist policy capable of executing a variety of tasks. This setup aligns with the recent trend of creating versatile, generalist agents that can handle a broad spectrum of challenges [4]–[6]. A straightforward idea to expand the demonstration dataset is to uniformly collect more demonstrations across all tasks. However, our observations indicate that while this approach can improve policy performance in certain scenarios, it lacks data efficiency

and tends to reach a performance plateau easily (see Fig. 1). To improve the data efficiency in demonstration expansion, a crucial intuition is that the *new demonstrations should target scenarios where the current policy fails*. Stemming from this intuition, we propose a framework named **AdaDemo** (**Ada**ptive Online **Demo**nstration Expansion) that is grounded in three core principles: 1) prioritizing the collection of demonstrations for tasks with low policy performances; 2) within each task, focusing on acquiring demonstrations for the initial states where the policy underperforms; and 3) adapting sampling strategies in training to emphasize challenging tasks. By adopting this framework, we aim to tailor the demonstration expansion process to precisely address the weaknesses of the existing policy, thereby avoiding collecting unnecessary demonstrations.

The effectiveness of AdaDemo was evaluated through a series of multi-task visual policy learning experiments on two robotic manipulation benchmarks: RLBench [13] and Adroit [14], including a total of 22 tasks. These experiments demonstrate AdaDemo's capability to progressively improve policy performance by guiding the generation of high-quality demonstration datasets in a *data-efficient* manner, as depicted in Fig. 1. Notably, AdaDemo exhibits significant data efficiency, *particularly at points where performance plateaus*. Compared to the baseline, it achieves better performance with only 1/2 the data on RLBench and 1/3 on Adroit. This efficiency **could translate into substantial cost savings**, especially in large-scale demonstration collection.

To summarize, our contributions go as follows:

1) We propose **AdaDemo**, a **data-efficient demonstration expansion framework**, designed to improve multi-task visual policy learning by adaptively expanding the demonstration dataset and employing an appropriate sampling strategy for training.

2) Through extensive **experiments on two robotic manipulation benchmarks**, including a total of **22 tasks**, we demonstrate the effectiveness of AdaDemo in expanding demonstration datasets in a data-efficient way.

II. RELATED WORK

Data Augmentation in Policy Learning Data augmentation is a widely utilized technique in machine learning that derives additional data samples by applying varied transformations to the existing data. In the context of policy learning, specialized techniques have been proposed: prior works [15]–[17] discuss enhancing the robustness of visual policies by applying image-based data augmentation techniques to observations. On the other hand, some recent studies [18], [19] propose to add additional trajectories by adapting existing demonstration trajectories to novel situations. While data augmentation can produce novel data instances, these instances are intrinsically limited in informational value as they are derived from the original dataset. AdaDemo distinguishes itself from these approaches by directly collecting new demonstrations to address the weakness of the learned policy.

Data Collection for Foundation Models Foundation models, such as large language models [1], vision language models

[20], generalist agents [21], and robotic agents [4], [5], usually rely on datasets that are pre-collected and static. This means the data collection phase is conducted once, with the dataset then used forever. Although straightforward to implement, this one-off approach to data collection can restrict the models' potential for learning and adaptation. Recently, there has been a shift towards training foundation models using dynamically expanding datasets. For instance, SAM [2] employs a bootstrapping approach, leveraging pre-trained segmentation models to help annotate a more extensive dataset, which is then used to train stronger models. Similarly, in the robotics field, RoboCat [3] utilizes a pre-trained policy to collect additional demonstrations for new tasks or robots, enlarging the dataset for subsequent agent training. Compared to these methods, AdaDemo prioritizes data efficiency by selectively targeting data collection efforts towards scenarios where the current policy underperforms, rather than indiscriminately adding data.

Online Policy Learning In contexts where online interaction is feasible, a straightforward strategy for policy learning is using reinforcement learning (RL) [22]. However, due to the sample efficiency issue and the unstable training dynamics, RL is rarely used in training large-scale policies. Also, online imitation learning [23]–[27] presents a viable approach for leveraging online interactions in policy learning. Many of these methods are based on DAgger [23], which requires an expert to provide action supervision for all encountered states during online interactions. The requirement for an expert capable of providing immediate supervision across *all possible states* can be impractical, making such methods less feasible for a wide range of applications. In contrast, AdaDemo simplifies this requirement by only requiring a demonstration collector for the *initial states* of tasks. This more manageable assumption positions AdaDemo as a less demanding alternative compared to methods like DAgger.

III. PROBLEM SETUP

In this paper, we study a multi-task setting where a single visual policy (generalist) is trained to solve multiple tasks, each having variants in terms of different goals and initial states. This setup mirrors practical scenarios, such as a robotic arm performing diverse tasks like stacking plates and opening drawers, depending on the given instructions and observed objects. We assume that tasks come with a success metric, allowing us to verify the task completion and compute the success rate of the policy.

To train the multi-task visual policy, we employ behavior cloning [8], a widely adopted algorithm in learning generalist robotic agents [3], [4]. Behavior cloning serves as a general foundation that can accommodate various modern visual policy learning architectures [28], [29]. Following prior work, we assume the demonstration dataset in behavior cloning has the format: $\mathcal{D}_0 := \{\mathcal{D}_0^1, ..., \mathcal{D}_0^M\}$, where M is the number of tasks, the superscript denotes the index of the task, and the subscript 0 denotes this is an initial pre-collected demonstration dataset. The demonstration dataset of each task m contains a set of demonstration trajectories τ:

$\mathcal{D}_0^m := \{\tau_0^m, \tau_1^m, ...\}$. For each demonstration τ, it contains a goal description g and a sequence of transition: $\tau := \langle g, \{(o_0, a_0), (o_1, a_1), ...\}\rangle$, where g is a goal description (e.g., language or low-dim vectors) and o_i is a visual observation, , which serves as the input of the visual policy. a_i is an action, which provides the supervision for output of visual policy.

An important aspect of our problem setup is the **flexibility in collecting demonstration**, which can be sourced through varied methods such as task and motion planning, state-based reinforcement learning, or teleoperation from human demonstrators. We assume access to a demonstration collector to *generate demonstrations for any state from the initial state distribution of the tasks*. Note that the demonstration collector **only needs to handle the initial states** of tasks, thus the assumption is less demanding than methods like DAgger [23], which require an oracle providing action supervision across **all possible states**. Such a demonstration collector is also feasible in real world via human teleoperation [30] or real-world motion planning [31], [32].

Generating demonstrations for all possible states is significantly more challenging than focusing on initial states, because most human-designed tasks are typically solvable from the initial states. For instance, in a "Stacking Cup" task, it is feasible to generate demonstrations where cups are initially standing on a table. However, if a cup is knocked down or falls from the table during the task execution, creating a demonstration for that state becomes difficult or impossible. Thus, our assumption is more realistic, only requiring a demonstration collector capable of handling initial task states.

IV. ADAPTIVE ONLINE DEMO EXPANSION

A. Overview

We aim to *actively* and *continuously* expand the demonstration dataset for training generalist robotic agents, particularly, for multi-task visual policy learning. Contrary to the conventional use of pre-collected and static datasets in imitation learning [8], **AdaDemo** (**Ada**ptive Online **Demo**nstration Expansion), differentiates itself by adopting an online and adaptive method. It operates over multiple iterative rounds to progressively improve the demonstration dataset, and consequently, the performance of the policy. Before delving into the technical details, we first highlight the key properties of AdaDemo below:

1) Online Demonstration Expansion: AdaDemo advocates the concept of online expansion of the demonstration dataset to overcome the limitations associated with pre-collected and static datasets in imitation learning. By incorporating new demonstrations, it becomes possible to address the weakness of the learned policy by directly providing additional supervision. This strategy is also supported by recent successes in foundation models [2], [3], suggesting that expanding the dataset is a potent method for enhancing model performance.

2) Iterative Improvement Process: AdaDemo employs an iterative approach, progressively incorporating new demonstrations into the dataset with each round. This iterative process offers several key advantages over the bulk accumulation
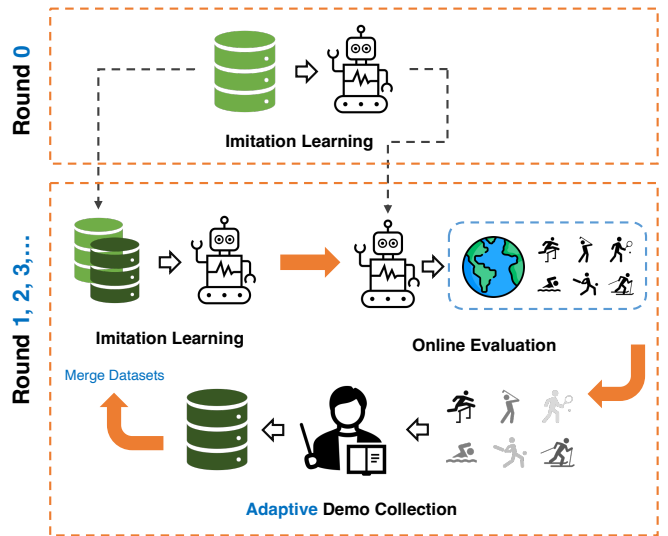

Fig. 2: AdaDemo iteratively expands the demonstration dataset through online evaluation of the trained policy, adaptively collecting additional demonstrations to target cases where the multi-task policy most needs improvement.

of demonstrations: **1) Data efficiency:** With the policy re-evaluated at each round, it is possible to specifically collect new demonstrations that address the current policy's weakness, thus eliminating the collection of unnecessary demonstrations; **2) Adaptability to Dynamic Budgets:** Resource budgets for demonstration collection may vary over time. This iterative approach accommodates such fluctuations, allowing for the initial use of available resources and the integration of new demonstrations as more resources become available; **3) Alignment with Current Research:** The iterative expansion of data is a strategy that has been utilized in modern foundation models [2], underscoring its effectiveness and relevance to current practices in the field.

3) Adaptive Demonstration Expansion: Instead of indiscriminately adding data, AdaDemo strategically focuses on demonstrations that are likely to yield the most significant improvements in policy performance, thus ensuring enhanced data efficiency. The adaptiveness is mainly grounded in **three core principles**: 1) Emphasizing collecting more demonstrations for initial states where the current policy fails (Sec. IV-B); 2) Prioritizing the collection of demonstrations for tasks where the current policy gets low success rates (Sec. IV-C); 3) to efficiently utilize the additional data, AdaDemo adapts the sampling strategy in training to emphasize tasks considered more challenging, ensuring that the model is continually exposed to and learns from the most demanding scenarios (Sec. IV-D).

In summary, AdaDemo actively expands the demonstration dataset for training generalist robotic agents. It emphasizes an online, iterative, and adaptive strategy to expand the dataset, ensuring that each newly incorporated demonstration is strategically selected to boost the policy's capability across a diverse range of tasks. See Fig. 2 for a visual illustration of our framework. The full framework is summarized in Algorithm 1. In the following subsections, we delve into the

details of the three core principles mentioned in Sec. IV-A.3.

B. Demo Collection on Failed Initial States

One core principle of AdaDemo is to focus on collecting demonstrations for tasks specifically from those *initial states* where the current policy falters. To systematically identify these failed initial states, we evaluate the multi-task policy on each task, pinpointing specific initial states where the current policy cannot successfully complete the task. For the sake of brevity, we treat the goal description as a part of the initial state of a task. Focusing on these failed initial states ensures that the newly collected demonstrations provide direct guidance for solving scenarios where the policy previously failed, thereby enhancing the data efficiency without wasting resources on collecting demonstrations for initial states the policy already navigates successfully.

In practice, we evaluate the policy on the default initial state distribution for each task. Upon encountering an initial state that leads to an unsuccessful episode, we engage a demonstration collector to generate an expert demonstration trajectory from that specific initial state.

Our framework is compatible with a variety of tools for expert demonstration collection, including task and motion planning systems, state-based reinforcement learning, model-predictive control, and even human teleoperation. The key requirement for these tools is their capability to complete the task from a given initial state. In cases where a demonstration collector fails to complete a task from a particular initial state, we can either opt for retrying the collection process a few more times, or choose to skip collecting a demonstration for that specific initial state. This flexible approach accommodates the reality of imperfect demonstration collectors, ensuring our framework's broad applicability.

C. Demo Collection on Unsolved Tasks

With the above strategy for collecting new demonstrations within a task, a critical consideration emerges: how should we allocate our demonstration collection budget across various tasks? Specifically, we need to decide the number of demonstrations to collect for each task.

Our guiding principle is straightforward: prioritize the collection of demonstrations for tasks that remain unsolved, i.e., the tasks where the current policy gets low success rates. These tasks are crucial in improving the overall performance across all tasks, because they offer large room for improvement. For each task, we keep evaluating the policy until it reaches a predetermined number of successful episodes. During this process, demonstrations are collected from the initial states where the policy fails, as described in Sec. IV-B.

Consequently, tasks with lower success rates naturally lead to more failures during the evaluation, thereby getting more new demonstrations. This approach creates a direct correlation between the task's difficulty (gauged by its success rate) and the number of demonstrations collected for this task. It ensures a concentrated effort on those tasks where the current policy underperforms, leading to a more efficient and focused data-collection process.

Algorithm 1 Adpative Online Demo Expansion

Require: Initial demonstration dataset for M tasks $\mathcal{D}_0 := \{\mathcal{D}_0^0, \mathcal{D}_0^1, ..., \mathcal{D}_0^M\}$, where demonstration dataset for task k $\mathcal{D}_0^k := \{\tau_0^k, \tau_1^k, ...\}$; Demonstration collector G;
1: Train a multi-task policy π on $\mathcal{D}_0$ by imitation learning
2: **for** each round i **do**
3:     Determine a target number of success episodes E_i for this round according to the budget
4:     **for** each task T^k **do**
5:         $\mathcal{D}_i^k \leftarrow \{\}$
6:         Counter $c \leftarrow 0$
7:         **repeat**
8:             Random sample an initial state s in task T^k
9:             Evaluate π on s
10:           **if** π failed on s **then**
11:              Collect a demo trajectory on s: $\tau \leftarrow G(s)$
12:              $\mathcal{D}_i^k \leftarrow \mathcal{D}_i^k \cup \{\tau\}$
13:           **else**
14:              $c \leftarrow c + 1$
15:         **until** $c = E_i$
16:     $\mathcal{D}_i \leftarrow \{\mathcal{D}_i^0, \mathcal{D}_i^1, ..., \mathcal{D}_i^M\}$
17:     Merge the datasets from all rounds $\mathcal{D} \leftarrow \bigcup_{j=0}^{i} \mathcal{D}_i$
18:     Re-train π on $\mathcal{D}$ with the selected sampling strategy

---

To provide a quantitative analysis, the formula for estimating the number of demonstrations N_{demo}^k required for each task k is given by $N_{demo}^k = \frac{E}{SR^k} - E$, where SR^k denotes the success rate of the current policy on task k, and E is a hyperparameter specifying the target number of success episodes, which could be adjusted based on the available resource budget. In practice, we also impose a cap on the demonstrations to be collected per task. The rationale behind this is to prevent the potential endless evaluation process to reach a target number of successful episodes for some tasks with extremely low success rates.

D. Sampling Strategy in the Collected Dataset

Beyond increasing the number of demonstrations, it is also crucial to optimize the use of the expanded dataset through a suitable sampling strategy during training. A simplistic strategy might involve uniformly sampling across the entire dataset. However, this method unintentionally biases towards tasks with longer trajectories, which does not necessarily correlate with their actual importance to the overall policy performance. To address this bias, we need to adopt alternative sampling strategies that more accurately prioritize demonstrations which are crucial for enhancing policy performance. One possible approach could be to ensure uniform sampling with respect to tasks, granting each task an equal chance of being selected. This method offers a more balanced distribution compared to uniform sampling with respect to the entire dataset, but falls short in highlighting the significance of the more difficult tasks, which are vital for the overall policy performance.

Therefore, we adopt a strategy that draws samples uniformly with respect to the demonstration trajectories, i.e., the sampling probability of each task's data is proportional to the number of demonstration trajectories in the dataset. This method aligns with our focused approach of collecting more demonstrations for unsolved tasks. By sampling in this

manner, we naturally place more emphasis on these unsolved tasks within the learning process. This not only ensures that harder tasks receive more attention, but also maximizes the utility of the additional demonstrations we have collected. In practice, we also set a minimum sampling weight for each task to avoid some tasks with a very small number of demonstrations being undersampled during training.

This sampling strategy plays a crucial role in our framework, guaranteeing that the expanded demonstration dataset is fully utilized, thereby directly supporting our objective of achieving maximal data efficiency.

V. EXPERIMENTS

A. Experimental Setup

The goal of our experimental evaluation is to study *whether AdaDemo can effectively expand the demonstration dataset in a data-efficient manner, thereby enhancing the learned multi-task visual policy.* Conceptually, AdaDemo is expected to be compatible with a variety of core components within the learning system, including the demonstration collector, network architecture, and type of controller. To confirm AdaDemo's wide-ranging applicability, our experimental designs incorporate variations across these dimensions:

- **2 Demonstration collectors**: Task and Motion Planning (TAMP), and state-based reinforcement learning (RL)
- **2 Controllers**: Keyframe-based end-effector control, and joint position control
- **2 Network architectures**: Vision Transformer [29], and Convolutional Neural Network [33].
- **A total of 22 tasks across 2 benchmarks**: RLBench [13] (table-top robotic manipulation, 18 tasks) and Adroit [14] (dexterous manipulation, 4 tasks). Fig. 4 shows sample tasks from each benchmark.

We summarize the key details of our setups as follows.

1) Environments: RLBench:

a) Task Description: We follow the standard setting used in [28], [29]. A 7-DoF Franka Panda robot equipped with a parallel gripper is directed to solve a total of 18 tasks, including pick-and-place, tool use, drawer opening, and high-accuracy peg insertions. Each task includes several variations specified by the associated language description. The visual observations are from four RGB-D cameras positioned at the robot's front side, left shoulder, right shoulder, and wrist, each providing a resolution of 128 × 128.

b) Demonstration Collector: The demonstrations are collected by a task and motion planning (TAMP) system from RLBench [13]. Notably, this TAMP system can only solve the tasks from the default task initial states. It lacks the capacity to tackle the tasks from certain intermediate states.

c) Visual Policy and Controller: In our RLBench experiments, we employ RVT [29] as the backbone visual policy. RVT features a multi-view transformer architecture and performs keyframe-based control, predicting actions at low frequency that specify the next keyframe's end-effector pose. A motion planner then directs the end effector to this predicted pose. The original RVT (as well as its predecessor [28], [34])

has a legacy issue leading to that makes the training dataset unnecessarily large. By refining the data processing scripts, we have mitigated this, resulting in a more storage-efficient RVT with a minor performance trade-off. Importantly, since the same modifications are consistently applied across all our RLBench experiments, we ensure a fair comparison between AdaDemo and baselines. Additionally, to accommodate the increasing demonstration data, we also raise the number of epochs to ensure that the model adequately fits the data.

2) Environments: Adroit:

a) Task Description: Adroit tasks are introduced in [14], where a 24-DoF ADROIT hand is tasked to master four dexterous manipulation skills: object relocation, in-hand manipulation, tool use, and opening doors. The visual observations are acquired from a pre-defined camera with a resolution of 256 × 256.

b) Demonstration Collector: The demonstrations are collected by a well-trained state-based RL agent. It is a common practice to employ state-based RL agents for demonstration collection, followed by rendering to images [35]–[37]. This approach is favored because generally it is easier to train a state-based agent compared to a visual agent.

c) Visual Policy and Controller: In our Adroit experiments, we utilize LfS [33] as the backbone visual policy. This approach employs a CNN enhanced with random shifting augmentation to generate actions at a high frequency, controlling the absolute joint positions of the dexterous hand. For the sake of simplicity, the observation space is limited to a single image frame rather than multiple frame stacking. Originally, the four tasks in Adroit have different action dimensions since certain tasks constrain the free movement of the robot hand. To learn a single multi-task policy, we set a unified action space as 6D movement of the root link and the 24-DoF joint control signal. Contrary to [33], which evaluates the policy at every epoch and selects the top three performances, we evaluate and report solely on the final policy's performance to avoid cherry-picking. It is crucial to note that these adjustments make our results not directly comparable to [33]. However, by employing the same visual policy backbone across all experiments, we ensure fair comparisons between our framework and the baselines.

B. Experimental Results

This subsection aims to substantiate two key points: 1) AdaDemo can *progressively improve* the performance of the multi-task visual policy across multiple rounds; 2) Compared to the uniform approach of data collection across all tasks and initial states, AdaDemo can expand the demonstration datasets in a more *data-efficient* manner.

We implemented AdaDemo on the RLBench and Adroit benchmarks, with results presented in Fig. 1, Table I and II. In these results, "Uniform" denotes the baseline method for expanding demonstrations, where demonstrations are evenly distributed across all tasks and initial states. Despite appearing simplistic, this uniform collection strategy is actually widely used in multi-task policy learning [28], [29], [34]. For each round, we trained visual policies of the same architectures
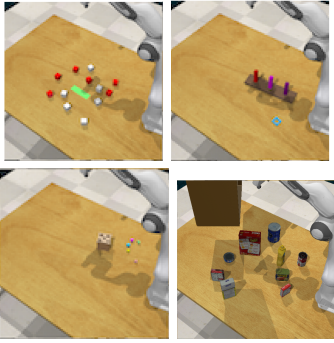
(a) RLBench (showing 4 of 18 tasks)
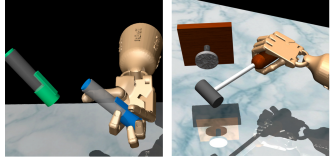
(b) Adroit (showing 2 of 4 tasks)

Fig. 4: **Tasks:** We consider challenging and diverse robotic manipulation tasks spanning two benchmarks: RLBench (table-top robot arm manipulation) and Adroit (dexterous manipulation).

	Initial	Round 1		Round 2		Round 3	
		● AdaDemo	● Uniform	● AdaDemo	● Uniform	● AdaDemo	● Uniform
Put in Drawer	85.3±8.3	80.0±4.0	80.0±17.4	72.0±12.0	70.7±31.1	82.7±12.2	**94.7**±2.3
Drag Stick	78.7±15.1	78.7±16.2	82.7±23.1	**97.3**±2.3	65.3±22.7	92.0±4.0	58.7±40.3
Turn Tap	92.0±4.0	90.7±2.3	96.0±4.0	92.0±10.6	93.3±2.3	**96.0**±4.0	93.3±2.3
Slide Block	**56.0**±8.0	33.3±8.3	54.7±22.0	16.0±4.0	29.3±18.0	17.3±6.1	13.3±8.3
Open Drawer	78.7±8.3	85.3±8.3	76.0±10.6	**85.3**±6.1	81.3±6.1	80.0±4.0	74.7±4.6
Put in Cupboard	34.7±8.3	61.3±4.6	58.7±8.3	64.0±6.9	65.3±2.3	**77.3**±12.9	60.0±13.9
Sort Shape	29.3±2.3	28.0±4.0	36.0±8.0	40.0±4.0	40.0±4.0	**41.3**±10.1	37.3±8.3
Put in Safe	76.0±4.0	78.7±8.3	90.7±2.3	86.7±6.1	**93.3**±2.3	93.3±2.3	93.3±4.6
Push Buttons	74.7±4.6	86.7±2.3	88.0±0.0	92.0±4.0	96.0±4.0	96.0±4.0	**98.7**±2.3
Close Jar	29.3±6.1	45.3±2.3	38.7±12.2	**58.7**±4.6	48.0±10.6	48.0±6.9	48.0±13.9
Stack Blocks	13.3±8.3	29.3±10.1	17.3±12.9	37.3±9.2	25.3±2.3	**46.7**±2.3	37.3±8.3
Place Cups	1.3±2.3	0.0±0.0	0.0±0.0	2.7±2.3	**8.0**±0.0	1.3±2.3	4.0±4.0
Place Wine	92.0±4.0	84.0±8.0	**94.7**±2.3	78.7±4.6	85.3±2.3	90.7±6.1	92.0±4.0
Screw Bulb	38.7±4.6	50.7±8.3	49.3±4.6	57.3±2.3	58.7±18.0	**78.7**±4.6	62.7±8.3
Sweep to Dustpan	45.3±11.5	52.0±4.0	48.0±6.9	48.0±12.0	46.7±11.5	45.3±2.3	**57.3**±4.6
Insert Peg	9.3±4.6	13.3±6.1	4.0±4.0	**21.3**±4.6	16.0±8.0	12.0±0.0	16.0±4.0
Meat off Grill	96.0±4.0	97.3±2.3	96.0±0.0	97.3±2.3	96.0±0.0	94.7±4.6	**100.0**±0.0
Stack Cups	4.0±4.0	37.3±8.3	20.0±21.2	54.7±6.1	29.3±6.1	**60.0**±6.9	45.3±20.5
Average SR	51.9	57.3	57.3	61.2	58.2	**64.1**	60.4
Average # Demo	20	49.4	50	91.0	100	191.2	200

TABLE I: **RLBench Results:** "SR" stands for success rate, and "Average # Demo" indicates the average number of demonstrations allocated per task. "Uniform" refers to the baseline where demonstrations are uniformly collected across all tasks and initial states. SRs are averaged over 3 random seeds. All agents are evaluated for 100 episodes on each task.

	Initial	Round 1		Round 2		Round 3		Round 4		Round 5	
		● AdaDemo	● Uniform	● AdaDemo	● Uniform	● AdaDemo	● Uniform	● AdaDemo	● Uniform	● AdaDemo	● Uniform
Relocate	0.6±0.4	1.9±0.7	0.9±0.7	6.0±1.7	6.0±3.1	18.6±4.0	12.3±0.9	28.8±1.3	25.2±5.5	**32.2**±6.2	23.6±9.6
Door	50.0±3.1	55.2±8.5	55.7±6.6	70.1±3.3	70.5±3.2	73.7±3.5	76.1±1.2	77.2±3.0	79.7±3.8	**83.0**±3.9	74.4±4.1
Pen	35.8±1.4	42.7±1.9	39.8±0.7	53.5±2.4	55.5±5.5	66.6±6.4	64.3±4.9	68.9±1.3	63.0±3.2	**74.8**±1.5	74.6±3.2
Hammer	95.7±4.9	97.2±1.6	**98.9**±1.7	98.8±1.0	97.7±1.8	97.6±1.7	97.5±2.4	97.7±3.5	97.2±2.6	97.3±4.4	96.9±4.7
Average SR	45.6	49.2	48.8	57.1	57.4	64.2	62.5	68.2	66.4	**71.8**	67.4
Average # Demo	20	37.8	40	99.0	100	290.0	300	997.0	1000	2994.3	3000

TABLE II: **Adroit Results:** For definitions of specific terms, please refer to Table I.

but utilized datasets collected separately by AdaDemo and the baseline, with the baseline dataset being comparable or slightly larger in size. The same demonstration collector was employed across both methods to ensure a fair comparison.

Fig. 1 shows a clear trend: the average performance of the multi-task visual policy can be progressively improved by expanding the demonstration datasets. Additionally, AdaDemo outperforms the baseline in two respects:

- **Data Efficiency:** AdaDemo demonstrates remarkable data efficiency, particularly at junctures where performance tends to plateau. It surpasses the baseline by requiring only 1/2 data for RLBench and 1/3 for Adroit. This efficiency could translate into significant cost reductions, especially in scenarios of large-scale data collection.
- **Performance Upper Bound:** While the baseline method quickly hits performance plateaus, AdaDemo consistently improves the multi-task performance, broadening the performance gap between the two approaches with each round. This trend indicates a greater potential in performance upper bound as the demonstration dataset is scaling up.

Tables I and II offer more detailed insights into performance differences. The RLBench results indicate that AdaDemo consistently outperforms the baseline in terms of average performance across tasks, although it does not lead in every single task. We hypothesize that the performance drop in certain RLBench tasks might be related to the issue of conflicting gradients in multi-task learning, as studied in [38]–[40]. AdaDemo's strategy of accumulating more demonstrations for harder tasks means that when gradients from these tasks do not align well with those from other tasks, following their gradient direction might adversely affect the performance on other tasks. Intuitively, focusing more on some tasks naturally detracts attention from others, and hence might hurt performance on other tasks. This effect appears more pronounced in the RLBench experiments, where the gradients may be noisier due to a relatively small average number of demonstrations per task (~ 200), particularly impacting the easier tasks. Conversely, in the Adroit benchmark, which comprises a larger average number of demonstrations per task (~ 3000) and only four tasks, the issue of gradient conflict is potentially less severe. Here, AdaDemo outperforms the baseline across all tasks, with the Hammer task being an exception (given its relative ease, all learned policies perform nearly optimally).

We noted that the contribution to the overall performance improvement is concentrated in a few tasks, such as "Relocate"

	● Uniform	A	A+B	● AdaDemo (A+B+C)
Put in Drawer	94.7±2.3	80.0±10.6	**97.3**±4.6	82.7±12.2
Drag Stick	58.7±40.3	84.0±20.8	88.0±4.0	**92.0**±4.0
Turn Tap	93.3±2.3	88.0±6.9	94.7±2.3	**96.0**±4.0
Slide Block	13.3±8.3	**21.3**±26.6	17.3±4.6	17.3±6.1
Open Drawer	74.7±4.6	73.3±11.5	**81.3**±4.6	80.0±4.0
Put in Cupboard	60.0±13.9	64.0±10.6	70.7±4.6	**77.3**±12.9
Sort Shape	37.3±8.3	**49.3**±2.3	38.7±8.3	41.3±10.1
Put in Safe	**93.3**±4.6	90.7±6.1	92.0±8.0	**93.3**±2.3
Push Buttons	**98.7**±2.3	94.7±2.3	97.3±2.3	96.0±4.0
Close Jar	48.0±13.9	45.3±9.2	**58.7**±8.3	48.0±6.9
Stack Blocks	37.3±8.3	42.7±4.6	**48.0**±12.0	46.7±2.3
Place Cups	**4.0**±4.0	2.7±2.3	0.0±0.0	1.3±2.3
Place Wine	**92.0**±4.0	89.3±9.2	86.7±8.3	90.7±6.1
Screw Bulb	62.7±8.3	64.0±10.6	65.3±8.3	**78.7**±4.6
Sweep to Dustpan	**57.3**±4.6	42.7±12.9	44.0±0.0	45.3±2.3
Insert Peg	16.0±4.0	**21.3**±2.3	22.7±4.6	12.0±0.0
Meat off Grill	**100.0**±0.0	98.7±2.3	96.0±0.0	94.7±4.6
Stack Cups	45.3±20.5	45.3±18.9	48.0±6.9	**60.0**±6.9
Average SR	60.4	61.0	63.7	**64.1**
Average # Demo	200	191.2	191.2	191.2

TABLE III: Ablation Study: A, B, C denote the 3 core principles in AdaDemo, see Sec. V-C for a detailed explanation.

and "Door" in Adroit, as well as "Put in Cupboard" and "Stack Cups" in RLBench. These tasks, which are generally hard, significantly benefit from our adaptive demonstration expansion strategy. By allocating more demonstrations to these tasks, rather than to easier ones, AdaDemo enhances both *data efficiency* and *overall performance*. However, it is important to note that the most difficult tasks (e.g., "Place Cups" and "Insert Peg") do not exhibit notable improvement with our adaptive demonstration expansion strategy. This is mainly due to these tasks being somewhat beyond the model's capabilities under the current setup. In such cases, exploring improvements to the model architecture or control method could be a worthwhile direction.

C. Ablation Study

We conduct an ablation study to examine the importance of the three guiding principles (Sections IV-B, IV-C, and IV-D) underlying AdaDemo. These principles, denoted as A, B, and C, are delineated as follows:

- A: Collecting demonstrations on unsolved tasks.
- B: Collecting demonstrations on failed initial states.
- C: Sampling data uniformly with respect to the demonstration trajectories.

To evaluate the influence of each principle, we start with the variant that uses none of these principles, and add one principle at one time, to create four variants. This ablation study allows us to dissect the specific role and importance of each component towards improving a multi-task visual policy, utilizing a comparable amount of data. The ablated variants are:

- **None of ABC:** This variant is the same as the "Uniform" baseline mentioned above, where demonstrations are uniformly collected across all tasks and initial states, without the targeted strategies of AdaDemo.

- **A**: Here, we collect demonstrations across tasks with the same number as in our full framework, but unlike our proposed approach, these demonstrations are sourced from all initial states, not just the failed ones.
- **A + B:** This variant combines the collection of more demonstrations on unsolved tasks (A) with the focus on failed initial states (B). However, unlike our full framework, it employs uniform sampling with respect to tasks, not demonstration trajectories.
- **A + B + C**: Our complete framework incorporates all three principles, providing a reference to compare against the ablated variants.

All the variants are applied on RLBench, and the results are shown in Table III. Analyzing the experimental results, we observe that each principle contributes to the overall performance gain, albeit to varying degrees. Notably, principle B, which emphasizes collecting demonstrations on failed initial states, appears to have a more substantial impact than principles A (focusing on unsolved tasks) and C (uniform sampling w.r.t. demonstration trajectories). This finding highlights the critical role of targeted demonstration collection in areas where the policy currently fails, underscoring the importance of addressing specific weaknesses in the policy for effective learning.

VI. LIMITATIONS

We would like to discuss a few limitations of our work. Firstly, AdaDemo requires policy evaluation to determine the scenarios for collecting new demonstrations, and this cost is not precisely accounted for in comparison to baselines. Nevertheless, it is important to note that the primary cost arises from the demonstration collection itself, which potentially involves human effort. Policy evaluation, by contrast, is a more autonomous procedure, and its costs are likely to be much lower than those incurred by demonstration collection. Secondly, AdaDemo assumes the availability of a success metric for tasks, a condition readily met in simulations but potentially hard in the real world. However, obtaining success signals in the real world is still feasible, and it has been achieved by methods like visual detection/tracking [41] or employing large vision-language models [42].

VII. DISCUSSIONS AND CONCLUSIONS

This study introduces AdaDemo, a novel framework designed to actively and continuously expand the demonstration dataset for training generalist robotic agents. AdaDemo can be categorized as a form of active learning tailored for robotics. Active learning methods have seen broad application in robotics, spanning active sensing [43], [44], active coverage [45], and active SLAM [46]. Our research aligns with the spirit of active learning, focusing on efficiently collecting additional demonstrations by a demonstration collector that only handles initial task states. A standout feature of AdaDemo is its exceptional data efficiency, especially evident at points where the performance of the policy reaches a plateau. This efficiency holds the potential for considerable cost reductions, especially in scenarios of large-scale demonstration collection.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[3] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, *et al.*, "Robocat: A self-improving foundation agent for robotic manipulation," *arXiv preprint arXiv:2306.11706*, 2023.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[6] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[7] O. X.-E. Collaboration, "Open X-Embodiment: Robotic learning datasets and RT-X models." https://arxiv.org/abs/2310.08864, 2023.

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

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[10] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4918–4927, 2019.

[11] E. Bronstein, S. Srinivasan, S. Paul, A. Sinha, M. O'Kelly, P. Nikdel, and S. Whiteson, "Embedding synthetic off-policy experience for autonomous driving via zero-shot curricula," in *Conference on Robot Learning*, pp. 188–198, PMLR, 2023.

[12] M. Jiang, E. Grefenstette, and T. Rocktäschel, "Prioritized level replay," in *International Conference on Machine Learning*, PMLR, 2021.

[13] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[14] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[15] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems*, vol. 33, pp. 19884–19895, 2020.

[16] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv preprint arXiv:2004.13649*, 2020.

[17] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering visual continuous control: Improved data-augmented reinforcement learning," *arXiv preprint arXiv:2107.09645*, 2021.

[18] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," *arXiv preprint arXiv:2310.17596*, 2023.

[19] S. Pitis, E. Creager, A. Mandlekar, and A. Garg, "Mocoda: Model-based counterfactual data augmentation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18143–18156, 2022.

[20] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[21] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[23] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[24] X. Yan, B. Boots, and C.-A. Cheng, "Explaining fast improvement in online imitation learning," in *Uncertainty in Artificial Intelligence*, pp. 1874–1884, PMLR, 2021.

[25] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*, PMLR, 2022.

[26] F. Liu, Z. Ling, T. Mu, and H. Su, "State alignment-based imitation learning," *arXiv preprint arXiv:1911.10947*, 2019.

[27] T. Mu, Z. Li, S. W. Strzelecki, X. Yuan, Y. Yao, L. Liang, and H. Su, "When should we prefer state-to-visual dagger over visual reinforcement learning?," *arXiv preprint*, 2024.

[28] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, pp. 785–799, PMLR, 2023.

[29] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, "Rvt: Robotic view transformer for 3d object manipulation," *arXiv preprint arXiv:2306.14896*, 2023.

[30] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *arXiv*, 2024.

[31] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox, "Object rearrangement using learned implicit collision functions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6010–6017, IEEE, 2021.

[32] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *Conference on Robot Learning*, pp. 967–977, PMLR, 2023.

[33] N. Hansen, Z. Yuan, Y. Ze, T. Mu, A. Rajeswaran, H. Su, H. Xu, and X. Wang, "On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline," *arXiv:2212.05749*, 2022.

[34] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1612–1619, 2022.

[35] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," *arXiv preprint arXiv:2107.14483*, 2021.

[36] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, *et al.*, "Maniskill2: A unified benchmark for generalizable manipulation skills," *arXiv preprint arXiv:2302.04659*, 2023.

[37] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang, "Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning," *arXiv preprint arXiv:2304.00464*, 2023.

[38] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[39] G. Shi, Q. Li, W. Zhang, J. Chen, and X.-M. Wu, "Recon: Reducing conflicting gradients from the root for multi-task learning," *arXiv preprint arXiv:2302.11289*, 2023.

[40] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18878–18890, 2021.

[41] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*, pp. 651–673, PMLR, 2018.

[42] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-language models as success detectors," *arXiv preprint arXiv:2303.07280*, 2023.

[43] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 26, p. eaav1488, 2019.

[44] N. Sornkarn and T. Nanayakkara, "Can a soft robotic probe use stiffness control like a human finger to improve efficacy of haptic perception?," *IEEE transactions on haptics*, vol. 10, no. 2, pp. 183–195, 2016.

[45] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2012.

[46] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

APPENDIX

VIII. IMPLEMENTATION DETAILS

A. RVT on RLBench

In our RLBench experiments, we employed RVT [29] as the foundational visual policy. While adhering to most aspects of the original RVT implementation, we introduced some modifications to address a legacy issue that resulted in unnecessarily large training datasets. Optimizing storage was crucial, especially considering that our framework expands the demonstration dataset. We detail our three key modifications below:

1) In the original RVT implementation (and its predecessor [28], [34]), each keyframe was redundantly added to the dataset k times, with k being the keyframe's index. This redundancy, acknowledged as a leftover artifact by the authors of [28], was eliminated in our approach. We revised the process to include each keyframe only once in the dataset, effectively reducing the size of the resulting dataset.

2) The removal of duplicate keyframes initially led to a performance drop in RVT. We found a common error pattern of the robot getting stuck at certain keyframes. We hypothesized that this problem was due to the training dataset's structure of $(s, a), ...$, where s represents the current state, and a is the end-effector pose for **the next keyframe**. If the current frame is too close to the next keyframe, the policy tends to yield minimal end-effector movement, causing the robot to stay still. To address this issue, we developed a new approach for assigning target end-effector poses. For frames between keyframes k and $k + 1$, the target pose is determined based on proximity: if closer to $k + 1$, the target is set from keyframe $k + 2$, otherwise from $k + 1$. By our empirical evaluation, this adjustment significantly mitigated the "stuck" behavior and enhanced overall performance.

3) Additionally, to further diminish the occurrence of the robot getting stuck, we modified the timestep encoding in the dataset. The original RVT augmented each frame with a timestep dimension, linearly decaying from 1 to -1 based on a predefined maximum episode length. We randomized timestep encoding during both training and evaluation, adding noises to help the agent break out of potential loops.

Please note that the effectiveness of the aforementioned modifications has only been validated empirically. However, we do not claim these modifications as the contributions of our work.

As a result of these three modifications, our adapted version of RVT is more storage-efficient, with only a minor impact on performance. Crucially, the consistent application of this modified algorithm across all RLBench experiments ensures a fair comparison between our framework and the baselines.

When collecting demonstrations from the failed initial states of each task, we set an upper limit of 100, 200, and 200 new demonstrations for each round, respectively. The rationale behind this is to prevent the potential indefinite duration required to attain a targeted number of successful episodes for some tasks with extremely low success rates.

B. LfS on Adroit

In our experiments on Adroit tasks, we followed [33], employing a simple convolutional neural network with random shifting augmentation as our visual policy backbone. While [33] limited their experiments to only two Adroit tasks (pen and relocate), we expanded our scope to include all four Adroit tasks: pen, relocate, hammer, and door. While retaining most of the implementation aspects from [33], we made a few adjustments, detailed as follows:

- In the original Adroit setup, each task had varying action dimensions, primarily because certain tasks restricted the Adroit hand's free movement in 3D space. To facilitate multi-task learning, we unified the action space to include 6D movement of the root link and a 24-DoF joint control signal. Note that we did not modify the tasks themselves, and the extra action dimensions for certain tasks will be ignored in the evaluation.
- For simplicity, our observation space utilizes only a single frame of images rather than stacking multiple frames.
- The evaluation approach in [33] involves averaging results from the top-3 performing checkpoints out of 50. To eliminate potential bias from such selective reporting, we assess and report performance based solely on the final checkpoints.

It is important to note that due to these modifications, our results are not directly comparable with those in [33]. Nonetheless, by using the same visual policy backbone for both our framework and the baselines, we still maintain fairness in the comparisons presented in our paper.

When collecting demonstrations from the failed initial states of each task, we set an upper limit of 40, 120, 400, 1400, and 6000 new demonstrations for each round, respectively. This cap is determined as twice the average number of demonstrations targeted for collection in each corresponding round. The rationale behind this is to prevent the potential indefinite duration required to attain a targeted number of successful episodes for some tasks with extremely low success rates.

IX. NUM OF TRAJECTORIES FOR EACH TASK

Our framework iteratively expands the demonstration dataset, and these demonstrations are not uniformly distributed across each task. In this section, we present the number of trajectories for each task in the demonstration dataset, as accumulated in each round of our framework. The numbers are shown in Table IV and V.

	Initial	Round 1		Round 2		Round 3	
		• AdaDemo	• Uniform	• AdaDemo	• Uniform	• AdaDemo	• Uniform
Put in Drawer	20	21	50	31	100	106	200
Drag Stick	20	21	50	31	100	49	200
Turn Tap	20	20	50	21	100	41	200
Slide Block	20	54	50	144	100	344	200
Open Drawer	20	21	50	24	100	39	200
Put in Cupboard	20	73	50	98	100	177	200
Sort Shape	20	36	50	93	100	246	200
Put in Safe	20	22	50	29	100	55	200
Push Buttons	20	26	50	29	100	40	200
Close Jar	20	38	50	98	100	231	200
Stack Blocks	20	86	50	186	100	386	200
Place Cups	20	120	50	220	100	420	200
Place Wine	20	22	50	23	100	69	200
Screw Bulb	20	43	50	77	100	190	200
Sweep to Dustpan	20	26	50	76	100	245	200
Insert Peg	20	120	50	220	100	420	200
Meat off Grill	20	21	50	22	100	24	200
Stack Cups	20	120	50	220	100	359	200
Average # Demo	20	49.4	50	91.0	100	191.2	200

TABLE IV: RLBench: Number of trajectories for each task in demonstration datasets.

	Initial	Round 1		Round 2		Round 3		Round 4		Round 5	
		• AdaDemo	• Uniform	• AdaDemo	• Uniform	• AdaDemo	• Uniform	• AdaDemo	• Uniform	• AdaDemo	• Uniform
Relocate	20	60	40	180	100	580	300	1980	1000	7980	3000
Door	20	35	40	79	100	206	300	922	1000	1903	3000
Pen	20	36	40	116	100	353	300	1060	1000	2068	3000
Hammer	20	20	40	21	100	21	300	26	1000	26	3000
Average # Demo	20	37.8	40	99.0	100	290.0	300	997.0	1000	2994.3	3000

TABLE V: Adroit: Number of trajectories for each task in demonstration datasets.