Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

PATO: Policy Assisted TeleOperation for Scalable Robot Data Collection

Shivin Dass*† Karl Pertsch*† Hejia Zhang† Youngwoon Lee‡ Joseph J. Lim§ Stefanos Nikolaidis†
†USC ‡UC Berkeley §KAIST

***Abstract*—Large-scale data is an essential component of machine learning as demonstrated in recent advances in natural language processing and computer vision research. However, collecting large-scale robotic data is much more expensive and slower as each operator can control only a single robot at a time. To make this costly data collection process efficient and scalable, we propose Policy Assisted TeleOperation (PATO), a system which automates part of the demonstration collection process using a learned assistive policy. PATO autonomously executes repetitive behaviors in data collection and asks for human input only when it is uncertain about which subtask or behavior to execute. We conduct teleoperation user studies both with a real robot and a simulated robot fleet and demonstrate that our assisted teleoperation system reduces human operators' mental load while improving data collection efficiency. Further, it enables a single operator to control multiple robots in parallel, which is a first step towards scalable robotic data collection. For code and video results, see clvrai.com/pato.**

I. Introduction

Recently, many works have shown impressive robot learning results from diverse, human-collected demonstration datasets [34, 8, 32, 14, 7]. They underline the importance of scalable robot data collection. Yet, such human demonstration collection through "teleoperation" is tedious and costly: tasks need to be demonstrated repeatedly and each operator can control only a single robot at a time. Research in teleoperation has focused on exploring different interfaces, such as VR controllers [52] and smart phones [34], but does not address the aforementioned bottlenecks to scaling data collection. Thus, current teleoperation systems are badly equipped to deliver the scalability required by modern robot learning pipelines.

Our goal is to improve the scalability of robotic data collection by providing assistance to the human operator during teleoperation. We take inspiration from other fields of machine learning, such as semantic segmentation, where costly labeling processes have been substantially accelerated by providing human annotators with learned assistance systems, e.g., in the form of rough segmentation estimates, that drastically reduce the labeling burden [9, 1].

Similarly, we propose to train an assistive *policy* that can automate control of repeatedly demonstrated behaviors and ask for user teleoperation only when facing a novel situation or when unsure which behavior to execute. Thereby, we aim to reduce the mental load of the human operator and enable scalable teleoperation by allowing a single operator to perform data collection with multiple robots in parallel.

*Equal Contributions.

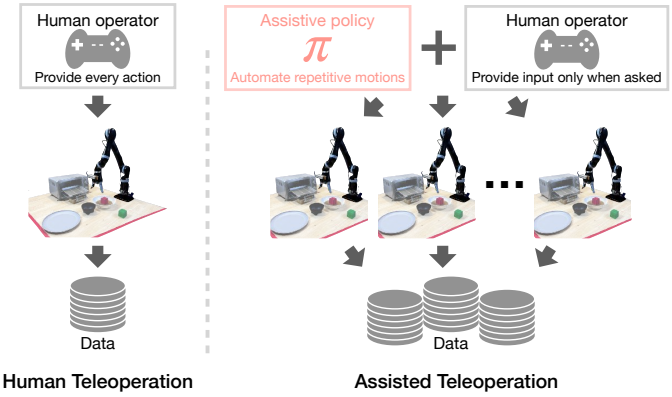


Fig. 1: **Policy Assisted TeleOperation** (PATO) enables large-scale data collection by minimizing human operator inputs and mental efforts with an assistive policy, which autonomously performs repetitive subtasks. This allows a human operator to simultaneously manage multiple robots.

In order to build an assistive system for robotic data collection, we need to solve two key challenges: (1) we need to learn assistive policies from diverse human-collected data, which is known to be challenging [37], and (2) we need to learn when to ask for operator input while keeping such interventions at a minimum. To address these challenges, we propose to (1) use a powerful hierarchical policy architecture that can effectively learn from diverse multi-modal human data, and (2) train a stochastic policy and use its output distribution to estimate its uncertainty about how to act in the current scene and which task to pursue. We then use this estimate to elicit operator input only if the assistive policy is uncertain about how to proceed.

The main contribution of this paper is a novel Policy Assisted TeleOperation (PATO) system, which enables scalable robotic data collection using a hierarchical assistive policy. We evaluate the effectiveness of our approach in a user study in which operators collect datasets of diverse kitchen-inspired manipulation tasks with a real robot. We find that our proposed *assisted* teleoperation approach reduces operators' mental load and improves their demonstration throughput. We further demonstrate that our approach allows a single operator to manage data collection with multiple robots simultaneously in a simulated manipulation environment – a first step towards more scalable robotic data collection.

II. Related Work

Robot Teleoperation. Demonstrations have played a key role in robot learning for many decades [44, 6, 4]; thus, many approaches have been explored for collecting such demonstrations. While initially kinesthetic teaching was common [2] in which a human operator directly moves the robot, more recently teleoperation has become the norm [52, 34, 18, 14, 30], since separating the human operator and the robot allows for more comfortable human control inputs and is crucial for training policies with image-based inputs. Research in teleoperation systems has focused on exploring different interfaces, like VR headsets [52, 14], joysticks [8] and smartphones [34]. Yet, none of these works explores active assistance of the human operator during teleoperation. Others have investigated controlling high-DoF manipulators via low-DoF interfaces through learned embedding spaces [31, 25] to allow people with disabilities to control robotic arms. In contrast, our approach trains assistive policies that automate part of the teleoperation process with the goal of enabling more scalable data collection.

Shared and Traded Autonomy. The idea of sharing efforts between humans and robots when solving tasks has a rich history in the human-robot interaction community [24, 47, 5, 43, 17, 40, 26, 3, 15, 16]. The literature differentiates between *shared* autonomy approaches (human and robot act concurrently, [12, 13, 49]) and *traded* autonomy approaches like ours (human and robot alternate control, [28, 42, 41]). Such approaches typically rely on a pre-defined set of goals and aim to infer the intent of the human operator to optimally assist them. Crucially, in the context of data collection, we cannot assume that all goals are known a priori, since a core goal of data collection is to collect previously unseen behaviors. Thus, instead of inferring the operator's intent over a fixed goal set, we leverage the model's estimate over its own uncertainty to determine when to assist and when to rely on operator input.

Interactive Human Robot Learning. In the field of robot learning, many approaches have explored leveraging human input in the learning loop and focused on different ways to decide when to leverage such input. Based on the DAgger algorithm [45], many works have investigated having the human themselves decide when to intervene [27], using ensemble-based support estimates [38], using discrepancies between model output and human inputs [51, 23], or risk estimates based on predicted future returns [22]. Yet, all these approaches focus on training a policy for a single task, not on collecting a diverse dataset. Thus, they are not designed to learn from multi-modal datasets or estimate uncertainty about the desired task. We show in our user study that these are crucial for enabling scalable robot data collection.

Assisted Robot Data Collection. Clever et al. [10] aims to assist in robot demonstration collection via a learned policy. They visualize the projected trajectory of the assistive policy to enable the human operator to intervene if necessary. However, they focus on collection of single-task, short-horizon demonstrations and require the operator to constantly monitor the robot to decide when to intervene. In contrast, our system can collect diverse, multi-task datasets and learn when to ask the user for input, enabling more scalable data collection, e.g., with multiple robots in parallel.

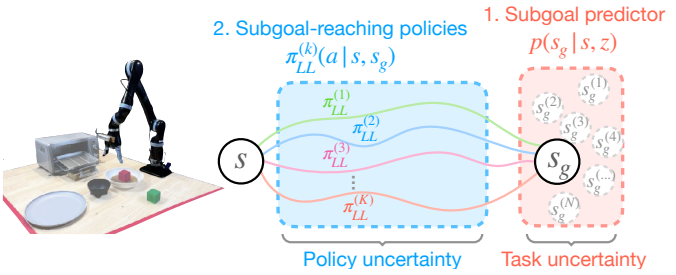


Fig. 2: PATO is hierarchical: a high-level subgoal predictor $p(s_g|s,z)$ and a low-level subgoal-reaching policy $\pi_{LL}(a|s,s_g)$. To decide when to follow the assistive policy, we measure uncertainty of both high-level (subgoal predictor) and low-level (subgoal-reaching policy) decisions. The task uncertainty is estimated using the subgoal predictor's variance, and the policy uncertainty is estimated as a disagreement among an ensemble of subgoal-reaching policies.

III. Approach

In this paper, we aim to improve the scalability of robotic data collection by learning an assistive policy to automate part of teleoperation when possible (in Section III-B) and ask for user inputs only when necessary (in Section III-C).

A. Problem Formulation

To enable scalable data collection of a dataset $\mathcal{D}$, we propose to automate control of repetitive behaviors using an *assistive policy*. The assistive policy controls the robot and minimize required human inputs, which allows the human operator to divert attention away from the robot over contiguous intervals, e.g., to attend to other robotic agents collecting data in parallel. The assistive policy can be defined as $\pi(a|s)$, which produces actions a, e.g., robot end-effector displacements, given states s, e.g., raw RGB images.

To train the assistive policy π, we assume access to a pre-collected dataset $\mathcal{D}_{\text{pre}}$ of diverse agent experience, e.g., from scripted policies, previously collected data on different tasks or human play [33]. Crucially, we explicitly require our approach to handle scenarios in which the newly collected dataset $\mathcal{D}$ contains behaviors that are *not* present in $\mathcal{D}_{\text{pre}}$. Thus, it is *not* possible to fully automate data collection given the pre-training dataset $\mathcal{D}_{\text{pre}}$. Instead, the system needs to request human input for unseen behaviors while providing assistance for known behaviors.

B. Learning Assistive Policies from Multi-Modal Data

Learning the assistive policy $\pi(a|s)$ from the diverse, multi-task data $\mathcal{D}_{\text{pre}}$ is challenging since the data is often highly multi-modal and long-horizon to imitate [37]. Thus, our assistive policy is built up on prior work in imitation of long-horizon, multi-task human data [35, 18, 33].

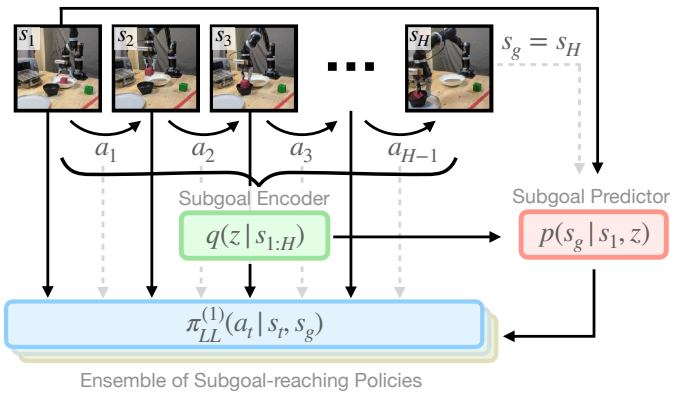


Fig. 3: Our hierarchical assistive policy is trained using a pre-collected dataset $\mathcal{D}_{\text{pre}}$. From a sampled trajectory $(s_1, a_1, \ldots, a_{\mathcal{H}-1}, s_{\mathcal{H}})$ of length $\mathcal{H}$, a subgoal predictor $p(s_g|s_1, z)$ is trained as a conditional VAE to cover a multi-modal subgoal distribution, where $s_g = s_{\mathcal{H}}$. Then, an ensemble of subgoal-reaching policies $\pi^{(k)}_{LL}(a_t|s_t, s_g)$ are trained to predict the ground truth actions. The gray dashed lines represent supervision for the prediction tasks of the subgoal predictor and subgoal-reaching policies.

Our hierarchical assistive policy consists of a (high-level) subgoal predictor $p(s_g|s, z)$ and a (low-level) subgoal-reaching policy $\pi_{\text{LL}}(a_t|s_t, s_g)$, as illustrated in Figure 2. Given a state s, the subgoal predictor first generates a subgoal s_g. Then, given the subgoal and the current state, the subgoal-reaching policy outputs an action for next $\mathcal{L}$ timesteps, leading the agent toward the subgoal.

To allow prediction of the *full distribution* of possible subgoals in multi-modal human demonstration data, we condition the subgoal predictor on a stochastic latent variable z [35]. For example, in a multi-modal dataset in which the robot moves vegetables into the oven in half of the trajectories and places them on a plate in the other half, z captures whether to predict a subgoal with vegetables in the oven or on the plate.

We train the subgoal predictor $p(s_g|s_t, z)$ as a conditional variational auto-encoder over subgoals [48]: given a randomly sampled starting state s_t from the pre-training dataset $\mathcal{D}_{\text{pre}}$ and a subgoal state $s_g = s_{t+\mathcal{H}}$ $\mathcal{H}$ steps later in the trajectory, we use a learned inference network $q(z|s_t, s_g)$ to encode s_t and s_g into a latent variable z. We then use the subgoal predictor $p(s_g|s_t, z)$ to decode back to the original subgoal state s_g. In addition to this subgoal reconstruction objective, we apply a latent regularization loss, which shapes the distribution of the latent variable z to a prior distribution $p(z)$.

The low-level subgoal-reaching policy $\pi_{\text{LL},\phi}(\mathbf{a}|s, s_g)$ is trained via the behavioral cloning objective [44]. Since the subgoal-reaching policy needs to predict a sequence of actions given a subgoal, we opt for a recurrent policy implemented using an LSTM [21], which autoregressively predicts the actions for the next $\mathcal{L}$ steps using s_t and s_g.

We summarize the components of our training model in Figure 3. Our final training objective is:

$$\max_{\theta,\phi,\mu} \mathbb{E}_{\substack{(s,\mathbf{a},s_g)\sim\mathcal{D}_{\text{pre}} \\ z\sim q(\cdot|s,s_g)}} \underbrace{p_\theta(s_g|s,z)}_{\text{subgoal reconstruction}} + \underbrace{\pi_{\text{LL},\phi}(\mathbf{a}|s,s_g)}_{\text{behavioral cloning}} - \underbrace{\beta D_{\text{KL}}\big(q_\mu(z|s,s_g), p(z)\big)}_{\text{latent regularization}}. \quad (1)$$

Here, $\mathbf{a}$ represents the sequence of actions from current state until s_g. We use θ, ϕ, μ to denote the parameters of the subgoal predictor, goal reaching policy, and inference network, respectively. β is a regularization weighting factor, D_{KL} denotes the Kullback-Leibler divergence, and we use a unit Gaussian prior $p(z)$ over the latent variable. More implementation details can be found in Appendix A.

To execute our assistive policy $\pi(a|s)$, we first sample a latent variable z from the unit Gaussian prior, and then generate a subgoal by passing z and s through the subgoal predictor $p(s_g|s, z)$. With the sampled subgoal s_g, we use the goal-reaching policy to predict an executable action $\pi_{\text{LL}}(a|s, s_g)$ for $\mathcal{L}$ timesteps. Every $\mathcal{L}$ actions, we sample a new subgoal.

C. Deciding When to Request User Input

A core requirement of our approach is that it can actively ask for operator input while minimizing the number of such requests. This is crucial since it allows the operator to divert their attention to other tasks, e.g., controlling other robots, while the assistive policy is controlling the robot. Thus, a key question is: when should the assistive policy ask for human inputs?

Intuitively, the assistive policy should request help when it is *uncertain* about what action to take next. This can occur in two scenarios: (1) the policy faces a situation that is not present in the training data, so it does not know which action to take, or (2) the policy faces a seen situation, but the training trajectories contain multiple possible continuations and the policy is not sure which one to pick. The latter scenario commonly occurs during the collection of diverse datasets since trajectories for different tasks often intersect. For example, in a kitchen environment, multiple tasks could start by tossing vegetables in a pan, but then diverge into putting the pan on the stove or in the oven. During teleoperation, human inputs are required for such situations to decide which task to continue with. This is in contrast to single-task demonstration collection, e.g., during DAgger training, where such uncertainty over tasks usually does not occur.

Specifically, we let the assistive policy to ask for user input when it has a high uncertainty due to (1) an out-of-distribution state or (2) a multi-modal action distribution. Our hierarchical model allows us to separately estimate both classes of uncertainty.

First, to estimate whether a given state is unseen, we follow prior work on out-of-distribution detection [29, 38] and train an ensemble of K low-level subgoal-reaching policies $\pi^{(1)}_{\text{LL}}(a^{(1)}|s), \ldots, \pi^{(K)}_{\text{LL}}(a^{(K)}|s)$, all on the same data $\mathcal{D}_{\text{pre}}$ but with different initializations and batch ordering. Then, the disagreement $D(a^{(1)}, \ldots, a^{(K)})$ between the actions predicted

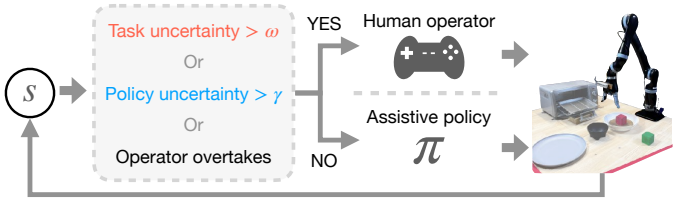


Fig. 4: Our approach asks for human inputs when the assistive policy is uncertain about which subtask or action to take. If both the task uncertainty and policy uncertainty are lower than their thresholds, our assistive policy can reliably perform a subtask, reducing the workload of the human operator.

by this ensemble of policies, i.e., the mean of the variance of actions in each dimension, can be used to approximate how out-of-distribution a state is, which we call **policy uncertainty**. The states with high policy uncertainty (i.e. high disagreement [29]) can be considered as unseen states. When the assistive policy encounters unseen states, it requests a user for deciding following actions.

Even if we determine that a state is seen in the training data, we further identify whether there are multiple possible task options in the current state before proceeding to follow the assistive policy. To estimate the **task uncertainty**, the assistive policy's uncertainty about the task, we propose to compute the inter-subgoal variance $Var(s_g^{(1)}, \ldots, s_g^{(N)})$, where the subgoals are sampled from the subgoal predictor $p_\theta(s_g^{(i)}|s, z^{(i)})$. If there is only one task to perform from a given state, the sampled subgoals will be similar to each other and thus, the variance of the sampled subgoals will be nearly 0. On the other hand, in cases with multiple possible task continuations, this variance will be high since the distribution of possible subgoals will widen. If the task uncertainty is high, the policy should also stop and ask a user to choose which task to perform next.

In summary, we leverage both policy and task uncertainty estimates to decide on whether the assistive policy should continue controlling the robot or whether it should stop and ask for human input. We found a simple thresholding scheme sufficient, with threshold parameters γ, ω for the policy uncertainty (i.e. ensemble disagreement) and task uncertainty (i.e. subgoal variance), respectively. Future work can investigate more advanced mixing schemes, e.g., with auto-tuned thresholds [22] or hysteresis between switching from robot to human and back [23]. We also include a human override H_t that allows the human to actively take control at any point during the teleoperation, e.g., in order to demonstrate a new task from a seen state, in which case the assistive policy would not ask for human input automatically. By combining the policy uncertainty, task uncertainty, and human override, we decide to *continue* executing the assistive policy if:

$$Assist = \neg \underbrace{(D(a^{(1)}, \ldots, a^{(K)}) > \gamma)}_{\text{unseen state}} \wedge \neg \underbrace{(Var(s_g^{(1)}, \ldots, s_g^{(N)}) > \omega)}_{\text{uncertain task}} \wedge \neg \underbrace{H_t}_{\text{human override}} . \quad (2)$$

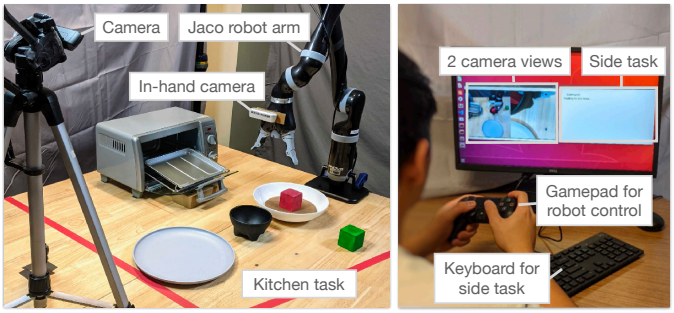


Fig. 5: User study setup. **(left)** A Kinova Jaco arm, front-view and in-hand cameras, and objects for kitchen-inspired tasks are placed on the workspace. **(right)** A human operator can watch a monitor, which shows either the camera inputs or a side task. The operator uses a gamepad to control the robot, and uses a keyboard to solve the side task.

In our experiments, we compute the values of γ and ω by plotting the policy uncertainty and task uncertainty over about 5 trajectories in the pre-training dataset, similar to the plots in Figure 8. We then choose the thresholds that correctly differentiate between the high and low uncertainty regions with the highest margin.

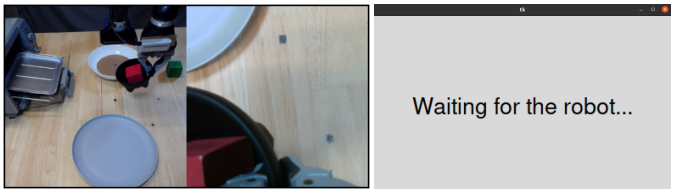
IV. Experiments

This paper proposes PATO, an assistive policy for teleoperation that reduces user's mental load during data collection as well as enable one user to handle multiple robots simultaneously. In this section, we aim to answer the following questions: (1) Does PATO reduce the mental load of human operators? (2) Does it allow the operators to divert their attention to other tasks during teleoperation? (3) Can PATO scale robotic data collection by allowing a single operator to teleoperate multiple robotic systems in parallel? To answer these questions, we perform two user studies in which (1) users teleoperate a real robot arm to perform diverse manipulation tasks (Section IV-A), and (2) users teleoperate *multiple* simulated robotic arms simultaneously (Section IV-B). For implementation details of our policy architecture and used training hyperparameters, see Appendix, Section A.

A. Reducing Mental Load during Data Collection

To evaluate the effectiveness of PATO, we conduct a user study ($N = 16$) in which users teleoperate a Kinova Jaco 2 robot arm to collect diverse robot manipulation demonstrations for kitchen-inspired long-horizon tasks, e.g., "place ingredients in bowl and place bowl in oven".

Real-World Kitchen Environment. As illustrated in Figure 5, we evaluate our algorithm and the baselines in a real-world kitchen environment with a Kinova Jaco 2 robot arm. The observations are $224 \times 224 \times 3$ cropped RGB images recorded from a Logitech Pro Webcam C920 for front-view and an Intel RealSense D435 for wrist-view (see Figure 7). In addition to images, the neural net policies have access to



(a) Side task is hidden when participant operates robot

(b) Robot view is hidden when participant performs the side task

Fig. 6: Participant's screen. (a) When a user teleoperates the robot, the side task is not visible. (b) Similarly, while performing the side task, the robot observation screen is hided. Users can switch between robot teleoperation and side task by pressing a button on a controller.

robot end-effector position, velocity, and gripper state. We use a Dual Shock 4 gamepad to control the robot's end-effector position. The action space for all policies is the delta in end-effector position and the gripper opening / closing commands. The actions are communicated at a rate of 10Hz and translated into desired joint poses using the inverse kinematics module of PyBullet [11].

The environment contains 8 long-horizon kitchen tasks composed by combining sub-tasks: {cook meat, serve meat}x{cook vegetables, serve vegetables}x{2 starting locations}. Each sub-task itself may involve the chaining of several skills. For example the sub-task *cook meat* requires sequential execution of the skills *pick meat from table* and *put meat in oven*.

Baseline and Prior Approach. To train our method, PATO, we collect a pre-training dataset of 120 demonstrations. Crucially, during the user study the operators need to collect *unseen* long-horizon tasks. We compare PATO to (1) human teleoperation with no assistance, the current standard approach to collecting robot demonstration data and (2) ThriftyDAgger [22], the closest prior work to ours for interactive human-robot data collection. ThriftyDAgger is designed to minimize human inputs during *single-task* demonstration collection by requesting human input only in *critical* states where a learned value function estimates low probability for reaching the goal. We initially implemented ThriftyDAgger with a flat policy as in the original work, but found it could not learn to model the multi-modal trajectories in our training dataset, leading to poor performance. Thus, we compare to an improved version of ThriftyDAgger that uses the same hierarchical policy as ours, which we found better suited to learn from the multi-modal data. We train all policies directly from raw RGB inputs without requiring additional state estimation.

User Study Design. We recruited 16 ($M = 13$, $F = 3$) participants from the graduate student population of our university. Participants were compensated with a 25 USD gift card. The study protocol was approved by the Institutional Review Board (IRB) at our university.

All participants teleoperate the robot's end-effector via joystick and buttons on a standard gamepad controller. We give each participant 5 minutes of unassisted training time before starting the experiment and an additional training task for each of the assisted methods (PATO and ThriftyDAgger) so that participants can familiarize themselves with the behavior of the policy. During the experiment, the participants are required to perform 3 randomly sampled long horizon tasks from a list of 8 possible tasks using each method. We counterbalance the order of the methods to guard against any sequencing effects.

The users are also asked to solve simple side tasks during teleoperation to measure their ability to divert attention and conduct other tasks. Specifically, they are shown a string of randomly selected characters, one of which is different from the others (e.g., 000100), and asked to specify the index of the odd character (e.g., 4 in the example above). As soon as the participant provides an answer, we present them with the next string. We leave it up to the participant how to allocate time between the robot manipulation task and the side task.

To ensure that the users can only attend to teleoperation *or* the side task at a time, they perform teleoperation purely via a live-stream video of the robot setup, without being able to see the physical robot (see Figure 5). Crucially, they can only see *either* the video stream *or* the side task and can switch between the two views with a button press, as shown in Figure 6.

To measure the user's mental workload during teleoperation, after each teleoperation session, we administer the NASA TLX survey [19, 20], which measures the user's perceived workload. We aggregate the responses to obtain a single score for each user. Participants also answer a survey adopted from previous work [40], which assesses their perception of the robot's intelligence, their satisfaction and trust in the system (Table I).

Comparison of Assisted Methods with Unassisted Baseline. We first compare the standard unassisted teleoperation baseline to the two methods with assistance in Table II. The

TABLE I: Post-execution survey (Likert scales with 7-option response format)

Trust:
Q1. I trusted the robot to do the right thing at the right time.
Robot intelligence ($\alpha = 0.95$)**:**
Q2. The robot was intelligent.
Q3. The robot perceived accurately what my goals are.
Q4. The robot and I worked towards mutually agreed upon goals.
Q5. The robot's actions were reasonable.
Q6. The robot did the right thing at the right time.
Human satisfaction ($\alpha = 0.91$)**:**
Q7. I was satisfied with the robot and my performance.
Q8. The robot and I collaborated well together.
Q9. The robot was responsive to me.

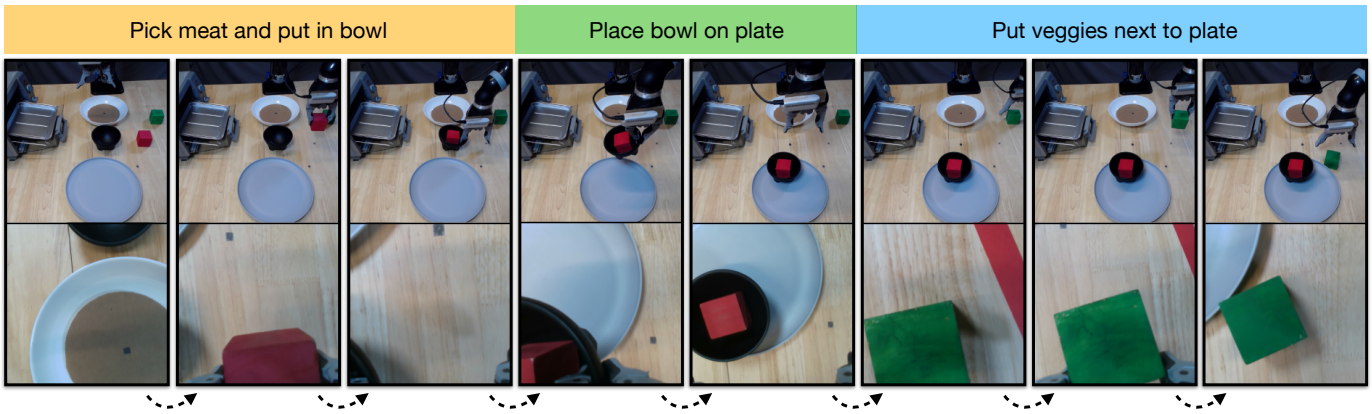


Fig. 7: Visualization of the front view (top) and wrist view (bottom) of a (serve meat, serve vegetables) task trajectory in the real-world kitchen environment.

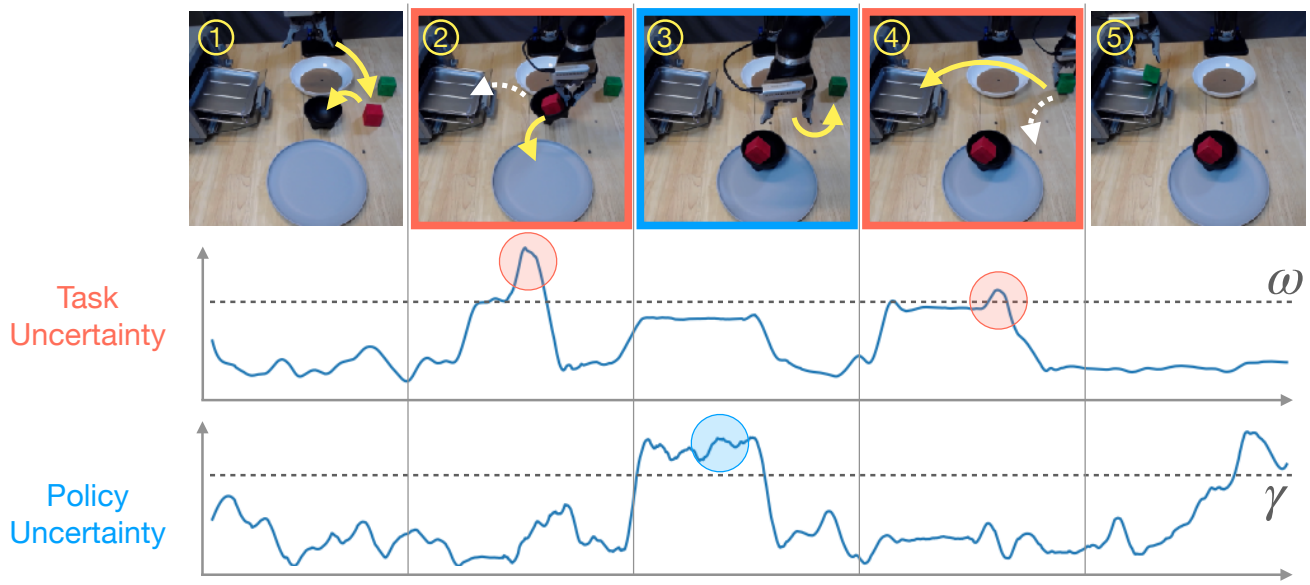


Fig. 8: Visualization of PATO on a task from the real-robot user study: *place red block in bowl; place bowl on plate; put green block in oven*. PATO autonomously executes familiar behaviors, but asks for user input in frames (2) and (4) to determine where to place bowl and green block (white vs. yellow arrow). In these cases, the task uncertainty surpasses the threshold ω since the subgoal predictor produces subgoals for both possible targets. Further, PATO asks for user input in frame (3) since the required transition between placing the bowl and picking up the green block was not part of its training data. Thus, the policy uncertainty estimate surpasses its threshold γ.

unassisted teleoperation baseline requires user's full attention by definition and thus, they are nearly unable to solve any side task during teleoperation. In contrast, the assisted approaches allow the users to solve additional side tasks without sacrificing teleoperation speed by diverting their attention to the side task during phases of data collection in which the policy acts autonomously.

Comparison of PATO with ThriftyDAgger. We then compare PATO with the prior assisted teleoperation method, ThriftyDAgger. First, we statistically evaluate the user responses to the surveys to elicit the participants' trust, satisfaction and perception of the robot's intelligence, as well as their mental workload when teleoperating with the two approaches.

During the study, participants agreed more strongly that they trusted the robot to perform the correct action at the correct time for PATO (Wilcoxon signed-rank test, $p = 0.001$)[1]. Further, they found the robot to be significantly more intelligent with the proposed method (repeated-measures ANOVA, $F(1, 15) = 5.14, p = 0.039$, Cronbach's $\alpha = 0.95$) and were significantly more satisfied with their collaboration with the robot ($F(1, 15) = 5.05, p = 0.040$, $\alpha = 0.91$). Finally, during the NASA TLX survey, participants showed a

[1]Following recommended practices for Likert scales in HRI [46], we use a non-parametric Wilcoxon signed-rank test for individual Likert items, while we assume equidistant scores and use parametric ANOVA tests in multi-item scales.

lower mental workload using PATO compared to the baseline ($F(1, 15) = 5.52, p = 0.033$).

A key factor in these strong subjective differences between the two approaches is their ability to elicit user feedback at appropriate times: when the robot is at a decision point between two possible task continuations (see Figure 8, frames (2) and (4)). ThriftyDAgger's risk-based objective is not sensitive to such decision points and thus, it rarely asks for user feedback. It instead executes one of the possible subtasks at random. In our study, we found that this leads to erroneous skill executions in 48 % of the cases. Such errors require tedious correction by the user, deteriorating their trust in the system and their teleoperation efficiency. In contrast, as illustrated in Figure 8, PATO leverages its estimate of task uncertainty to correctly elicit user feedback in 82 % of cases, leading to higher perceived levels of trust and reduced mental load. As a result, we also find that PATO allows for more efficient data collection (see Table II) since fewer corrections are required.

B. Scaling Data Collection to Multiple Robots

In the previous section, we showed that PATO allows users to divert their attention to other tasks during data collection. An important application of this is multi-robot teleoperation, in which a single operator performs data collection with multiple robots in parallel and periodically attends to different robots. We test this in an environment that requires simultaneous teleoperation of *multiple* simulated robots.

Simulated IKEA Furniture Assembly Environment. We evaluate the scalability of data collection with assisted teleoperation on the simulated IKEA Furniture Assembly environment of Lee et al. [30], which builds on the MuJoCo physics simulator [50]. We use the ground truth pose and orientation measurements of the end-effector and the blocks provided by the simulator as observations for all policies. They control the robot via end-effector pose control, like in our real-world setup. We use multiple instances of this environment to simulate data collection on multiple robots, as illustrated in Figure 9. We use a Dual Shock 4 gamepad to control each robot's end-effector position, gripper open / close state and to allow switching between controlling the different robots with a button press.

To pre-train our assistive policy, we collect 60 demonstrations for a block stacking task in which the middle block needs to be stacked on either the left block or the right block. During teleoperation time we only accept stacking on the left block as a successful trajectory. Thus, a useful assistive policy needs to elicit user input on which block to stack the middle block on, left or right, and the user needs to point the policy to the left.

TABLE II: Average number of completed side tasks and teleoperation time per demonstration during the real-robot teleoperation user study.

Approach	Avg. # completed side tasks	Avg. teleop time (sec)
Unassisted	0.25 ($\pm$0.66)	109.5 ($\pm$31.4)
ThriftyDAgger++[2]	**13.06** ($\pm$**9.63**)	105.9 ($\pm$29.5)
PATO (ours)	**15.88** ($\pm$**7.11**)	**85.0** ($\pm$**18.2**)

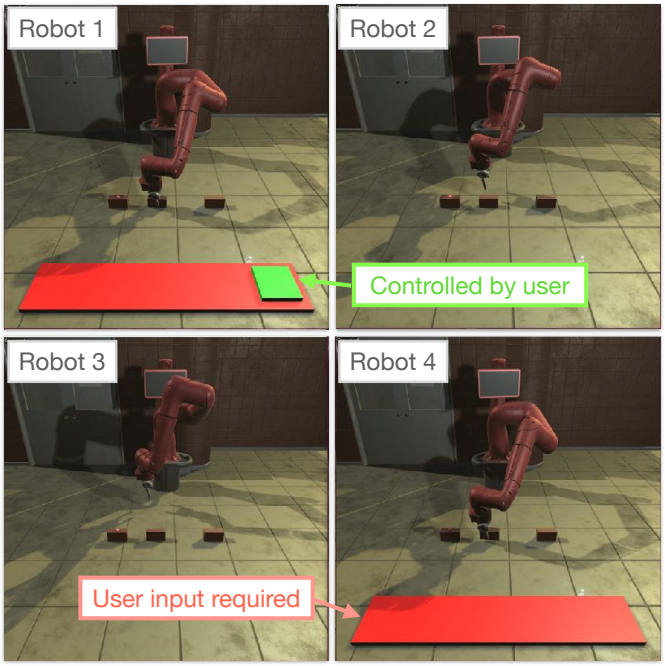


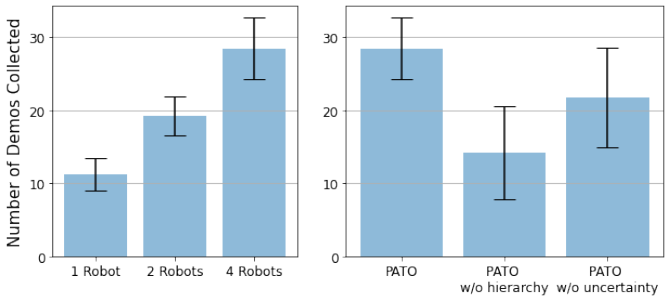
Fig. 9: We use up to four simulated robots to collect demonstrations. The assistive policy asks for human input using the red indicator. The small green indicator represents which environment is being controlled by a user.

In this way we can measure both, whether the system asks for user input at the correct times and whether it allows users to give meaningful input to achieve their desired behavior, even when interacting with multiple robots in parallel.

User Study Design. We give each participant ($N = 10$) 1 minute of training time with each of the unassisted and assisted teleoperation approaches in a single-robot version of the environment to allow them to familiarize themselves with the task and the teleoperation system. During the experiment, the participants are asked to collect as much data as possible while controlling multiple robots simultaneously (1, 2 or 4). Each participant completes data collection with all three robot fleet sizes. They are given 4 minutes for each data collection session. For 2 or more robots, the participants can switch between which robot they want to control via a button press and the policies can request user feedback via a visual indicator, as illustrated in Figure 9.

Multi-Robot Data Collection Results. We compare the number of collected demonstrations with increasing numbers of simultaneously teleoperated robots in Figure 10a. Our approach enables strong scaling of data collection throughput. As expected, the scaling is not linearly proportional, i.e., four robots do not lead to four times more demonstrations collected. This is because simultaneous teleoperation of a larger fleet requires more context switches between the robots, reducing the effective teleoperation time. Yet, PATO enables effective parallel teleoperation, leading to higher demonstration

[2]ThirftyDAgger [22] using our improved, hierarchical policy architecture.



(a) Scaling experiment (b) Ablation study

Fig. 10: Average number of demonstrations collected in 4 minutes using multiple robots in simulation. **(a)** With PATO, users can manage multiple robots simultaneously and collect more demonstrations with four robots. **(b)** The ablated systems with four robots show inferior demonstration collection throughput.

TABLE III: Average success rate of behavior cloning policy trained on data from single-robot vs. multi-robot teleoperation.

No. of robots	Time Taken	# Traj. Collected	Success Rate (%)
1-robot	4 mins	11	34.6 ($\pm$8.3)
1-robot	~10 mins	28	**61.0** ($\pm$**9.3**)
4-robot	**4 mins**	28	**56.8** ($\pm$**13.0**)

throughput with larger robot fleets. In contrast, with standard unassisted teleoperation, a single operator can only control a single robot. Thus, the demonstration throughput would not increase even if a larger fleet of robots was available.

Figure 10a clearly shows that with the same amount of human time and effort, we can collect a larger number of demonstrations with multiple robots via our approach. We additionally verify the quality of demonstrations collected by our parallel data collection pipeline. To this end, we train a behavioral cloning (BC) policy on the data from the multi-robot teleoperation experiment, and compare its success rates to a BC policy trained on single-robot teleoperation data. In Table III, we first compare the performance of BC on data collected in 4 minutes by unassisted single-robot teleoperation and assisted multi-robot teleoperation using PATO. The larger demonstration throughput with four robots results in a policy with higher performance using the same amount of human teleoperator effort. Additionally, we compare the BC performance on the same number of trajectories collected using single- and multi-robot teleoperation and demonstrate that the quality of data collected using PATO does not deteriorate as we increase the number of robots while only taking approximately a third of the teleoperation time. This exhibits the potential of assisted teleoperation systems to drastically increase the effectiveness of a team of operators simply by intelligently automating redundant teleoperation sequences.

Ablation Study. Our approach, PATO, has two key components: the hierarchical policy (Section III-B) and the uncertainty-based requesting of user input (Section III-C). To evaluate the importance of each component, we perform ablation studies in the simulated IKEA Furniture Assembly Environment with the 4-robot teleoperation setup described above. We compare our full method against two ablated methods: (1) **PATO w/o hierarchy**, which trains an ensemble of *flat* stochastic policies $\pi(a|s)$, and (2) **PATO w/o uncertainty**, which removes the uncertainty-based requesting of user input. The flat policy ablation, "PATO w/o hierarchy", uses the ensemble disagreement and action distribution variance (instead of subgoal distribution variance in PATO) to determine policy and task uncertainty, respectively.

We report the number of collected demonstrations per method within 4 minutes averaged across $N = 10$ users in Figure 10b. Both ablated methods show much smaller throughputs than PATO. Specifically, we find that the flat policy in "PATO w/o hierarchy" is neither able to accurately model the multi-modal training data nor to accurately estimate the task uncertainty. As a result, the assistive policy does not ask for user input in critical decision states and exhibits frequent control failures, leading to reduced teleoperation throughput. Similarly, the ablation without uncertainty estimation, "PATO w/o uncertainty", does not elicit user input in critical states and thus, requires the user to correct it whenever it tries to complete an incorrect task. As a result, the demonstrations collected with the ablated methods are less optimal, requiring an average 398 and 375 steps until task completion for "PATO w/o hierarchy" and "PATO w/o uncertainty", respectively, compared to 322 steps for our method.

V. Conclusion

A large-scale robot demonstration dataset is key to enabling the next breakthrough in robot learning. As a step towards large-scale robot data, we propose an efficient and scalable system for robotic data collection, which automates part of human teleoperation using a learned assistive policy and actively asks for human input in critical states. This allows a human operator to handle multiple robots simultaneously and significantly improve data collection throughput. The user study supports that our assisted teleoperation system requires infrequent inputs from users and users feel less mental burden when collecting robot demonstrations. We further show significantly improved data collection throughput of our system in the multi-robot control experiments, which leads to higher downstream policy performance for the same amount of human operator effort.

For simplicity, we assume access to pre-collected data $\mathcal{D}_{\text{pre}}$ to train the assistive policy. However, our assistive policy can, in theory, be learned from scratch and continuously improved as more data is collected. In this way, operators can also teach the assistive policy new skills over time and tailor its capabilities to their needs. We leave an investigation of such continually evolving assistance systems as an exciting direction for future work. Moreover, deploying our multi-robot data collection in the real world is a plausible avenue for future work.

ACKNOWLEDGMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant (No.2019-0-00075, Artificial Intelligence Graduate School Program, KAIST) and National Research Foundation of Korea (NRF) grant (NRF-2021H1D3A2A03103683), funded by the Korea government (MSIT).

REFERENCES

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, pages 859–868, 2018.

[2] Heni Ben Amor, Erik Berger, David Vogt, and Bernhard Jung. Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction. In *Annual conference on artificial intelligence*, 2009.

[3] Brenna D Argall. Autonomy in rehabilitation robotics: An intersection. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:441, 2018.

[4] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483, 2009.

[5] Pierre Berthet-Rayne, Maura Power, Hawkeye King, and Guang-Zhong Yang. Hubot: A three state human-robot collaborative framework for bimanual surgical tasks based on learned models. In *ICRA*, pages 715–722. IEEE, 2016.

[6] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

[8] Serkan Cabi, Sergio Gomez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *RSS*, 2019.

[9] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.

[10] Henry M Clever, Ankur Handa, Hammad Mazhar, Kevin Parker, Omer Shapira, Qian Wan, Yashraj Narang, Iretiayo Akinola, Maya Cakmak, and Dieter Fox. Assistive tele-op: Leveraging transformers to collect robotic task demonstrations. *arXiv preprint arXiv:2112.05129*, 2021.

[11] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[12] Jacob W Crandall and Michael A Goodrich. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 2, pages 1290–1295. IEEE, 2002.

[13] Anca D Dragan and Siddhartha S Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7):790–805, 2013.

[14] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *RSS*, 2022.

[15] Matthew Fontaine and Stefanos Nikolaidis. A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy. *arXiv preprint arXiv:2012.04283*, 2020.

[16] Matthew C Fontaine and Stefanos Nikolaidis. Evaluating human–robot interaction algorithms in shared autonomy via quality diversity scenario generation. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(3):1–30, 2022.

[17] Yang Gao and Steve Chien. Review on space robotics: Toward top-level science through space exploration. *Science Robotics*, 2(7):eaan5074, 2017.

[18] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *CoRL*, 2019.

[19] Sandra G Hart. Nasa task load index (tlx). 1986.

[20] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *CoRL*, 2021.

[23] Ryan Hoque, Ashwin Balakrishna, Carl Putterman, Michael Luo, Daniel S Brown, Daniel Seita, Brijen Thananjeyan, Ellen Novoseller, and Ken Goldberg. Lazydagger: Reducing context switching in interactive imitation learning. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2021.

[24] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. In *RSS*, 2015.

[25] Hong Jun Jeon, Dylan P Losey, and Dorsa Sadigh. Shared autonomy with learned latent actions. *RSS*, 2020.

[26] Mishel Johns, Brian Mok, David Sirkin, Nikhil Gowda, Catherine Smith, Walter Talamonti, and Wendy Ju. Exploring shared control in automated driving. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 91–98. IEEE, 2016.

[27] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *ICRA*, 2019.

[28] David Kortenkamp, R Peter Bonasso, Dan Ryan, and Debbie Schreckenghost. Traded control with autonomous robots as mixed initiative interaction. In *AAAI Symposium on Mixed Initiative Interaction*, volume 97, pages 89–94, 1997.

[29] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

[30] Youngwoon Lee, Edward S Hu, Zhengyu Yang, Alex Yin, and Joseph J Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. *ICRA*, 2021. URL https://clvrai.com/furniture.

[31] Dylan P Losey, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, and Dorsa Sadigh. Controlling assistive robots with learned latent actions. In *ICRA*, 2020.

[32] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, and Sergey Levine. Aw-opt: Learning robotic skills with imitation and reinforcement at scale. In *CoRL*, 2021.

[33] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *CoRL*, 2020.

[34] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, 2018.

[35] Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *ICRA*, 2020.

[36] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Gti: Learning to generalize across long-horizon tasks from human demonstrations. In *RSS*, 2020.

[37] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *CoRL*, 2021.

[38] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *IROS*, 2019.

[39] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *CoRL*, 2022.

[40] Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 294–302. IEEE, 2017.

[41] Yoojin Oh, Tim Schäfer, Benedikt Rüther, Marc Toussaint, and Jim Mainprice. A system for traded control teleoperation of manipulation tasks using intent prediction from hand gestures. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 503–508. IEEE, 2021.

[42] Calder Phillips-Grafflin, Halit Bener Suay, Jim Mainprice, Nicholas Alunni, Daniel Lofaro, Dmitry Berenson, Sonia Chernova, Robert W Lindeman, and Paul Oh. From autonomy to cooperative traded control of humanoid manipulation tasks with unreliable communication: Applications to the valve-turning task of the darpa robotics challenge and lessons learned. *Journal of Intelligent & Robotic Systems*, 82:341–361, 2016.

[43] Andreas Pichler, Sharath Chandra Akkaladevi, Markus Ikeda, Michael Hofmann, Matthias Plasch, Christian Wögerer, and Gerald Fritz. Towards shared autonomy for robotic tasks in manufacturing. *Procedia Manufacturing*, 11:72–82, 2017.

[44] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, pages 305–313, 1989.

[45] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, pages 627–635, 2011.

[46] Mariah L Schrum, Michael Johnson, Muyleng Ghuy, and Matthew C Gombolay. Four years in review: Statistical practices of likert scales in human-robot interaction studies. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 43–52, 2020.

[47] Mario Selvaggio, Marco Cognetti, Stefanos Nikolaidis, Serena Ivaldi, and Bruno Siciliano. Autonomy in physical human-robot interaction: A brief survey. *IEEE Robotics and Automation Letters*, 2021.

[48] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[49] Justin G Storms and Dawn M Tilbury. Blending of human and obstacle avoidance control for a high speed mobile robot. In *2014 American Control Conference*, pages 3488–3493. IEEE, 2014.

[50] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[51] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving.

AAAI, 2017.

[52] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018.

APPENDIX

A. Implementation Details

Here we provide a detailed overview of the model architecture and used hyperparameters (for a detailed listing of all hyperparameters, see Table IV).

1) Sub-goal Predictor: The conditional-VAE for the sub-goal predictor consists of an encoder and a decoder, both implemented with 5-layer MLPs with 128 hidden units in each layer, batch norm and LeakyReLU activations. The input to the encoder is the state and goal $\mathcal{H}$ steps in the future, where $\mathcal{H}$ is the goal horizon. For image observations in the real-world kitchen environment we first run the inputs through a pre-trained visual encoder, R3M [39], to obtain a 2048-dimensional vector representation. The output of the encoder is the mean and log standard deviation of a 128-dimensional multivariate Gaussian latent space. The decoder uses the current state and a sample from the encoder's output distribution to generate a goal prediction. During inference, we generate goals with the decoder by sampling the latent vector from a standard Gaussian prior. $\mathcal{H}$ is set to 35 for the real-world kitchen environment and 7 for the IKEA Furniture Assembly environment.

To model the task uncertainty when deciding whether to request human input, we sample $N = 1024$ goals and measure the variance between the end-effector positions of the robot. In case of high task uncertainty, the goals predicted by the sub-goal predictor diverge, leading to high variance. When the variance is higher than a threshold τ_{task}, we query the human to disambiguate which task to perform next.

2) Low-level Sub-goal Reaching Policy: The inputs to the sub-goal reaching policy is the current state s_t and the goal state $s_{t+\mathcal{H}}$ where $\mathcal{H}$ is the goal horizon used in the sub-goal predictor. The policy is implemented as a LSTM [21] with 256-dimensional hidden units, which autoregressively predicts the actions for the next $\mathcal{L}$ steps using s_t and $s_{t+\mathcal{H}}$. Notice that in prior works [36] skill horizon $\mathcal{L}$ and goal horizon $\mathcal{H}$ are set to the same value. However, in our experiments we noticed that setting a higher goal horizon helps model the task uncertainty better but a higher skill horizon would reduce the decision frequency of the policy. Hence, in this work, we decouple the two and show that the goal horizon $\mathcal{H}$ can be much greater than the skill horizon $\mathcal{L}$ without affecting the policy performance. The state and goal are processed by input MLPs separately and concatenated before being processed by the LSTM. The hidden state of the LSTM is processed by an output MLP to generate actions.

To model policy uncertainty, we use an ensemble of K=5 sub-goal reaching policies. Similar to Menda et al. [38], we measure the variance between the actions predicted by each of the policies. When the state input to the policies is seen in the training data, we expect the action predictions of the ensemble policies to agree, leading to a low variance. When the state input is outside the distribution of seen training data, the policy predictions diverge, leading to a high variance. When variance is higher than a threshold τ_{policy}, the policy queries human help.

TABLE IV: PATO hyperparameters. Parameters that differ between environments are marked in red.

Hyperparameters	Kitchen	IKEA Assembly
Sub-goal Predictor		
train iterations	200	500
batch size	16	16
learning rate	0.001	0.001
optimizer	Adam(0.9, 0.999)	Adam(0.9, 0.999)
encoder-mlp (width x depth)	128x5	128x5
decoder-mlp (width x depth)	128x5	128x5
latent dimension	128	128
normalization	batch	batch
activation	LeakyReLU(0.2)	LeakyReLU(0.2)
goal horizon ($\mathcal{H}$)	35	7
loss	ELBO	ELBO
Low-level Sub-goal Reaching Policy		
train iterations	300	4000
batch size	16	16
learning rate	0.001	0.001
optimizer	Adam(0.9, 0.999)	Adam(0.9, 0.999)
input-mlp (width x depth)	256x3	256x3
output-mlp (width x depth)	256x3	256x3
lstm hidden dimension	256	256
normalization	batch	batch
activation	LeakyReLU(0.2)	LeakyReLU(0.2)
no. of ensemble policies	5	5
skill horizon ($\mathcal{L}$)	15	7
loss - delta EEF positions	MSE	MSE
loss - gripper	CE Loss	CE Loss

3) Q-Function (ThriftyDAgger): The ThriftyDAgger baseline [22] uses a risk measure derived from a trained Q-function for choosing when to request human input. The Q-function is modeled using a 3-layer MLP with 128 hidden units and LeakyReLU(0.2) activation. More details on training the Q-funtion and using it for risk calculation can be found in [22].