“No, to the Right” – Online Language Corrections for Robotic Manipulation via Shared Autonomy

Yuchen Cui*  
yuchenc@cs.stanford.edu  
Stanford University

Siddharth Karamcheti*  
skaramcheti@cs.stanford.edu  
Stanford University

Raj Palleti  
Stanford University

Nidhya Shivakumar  
The Harker School

Percy Liang  
Stanford University

Dorsa Sadigh  
Stanford University

ABSTRACT

Modern systems for language-guided human-robot interaction require two key components for broad adoption: adaptivity and learning efficiency. Unfortunately, existing data-driven approaches for learning instruction-following agents cannot adapt, failing to incorporate additional natural language supervision, and even if they could, require hundreds of demonstrations to learn even simple policies. In this work, we address these problems by presenting a framework for incorporating and adapting to natural language corrections – “to the right”, or “no, towards the book” – as the robot executes. To focus on rich manipulation domains where the sample efficiency of existing methods is prohibitive, we work within a shared autonomy paradigm: instead of discrete turn-taking between a human and robot, our shared autonomy paradigm splits agency between the human and robot. In our approach, natural language is an input to a learned model that produces a meaningful, low-dimensional control space that the human can use to guide the robot. Each real-time correction refines the human’s control space, enabling the execution of precise, extended behaviors – with the added benefit of requiring only a handful of demonstrations to learn. We evaluate our approach via a user study, where users work with a Franka Emika Panda manipulator to complete complex manipulation tasks. Compared to existing learned baselines covering both open-loop instruction following and single-turn shared autonomy, we show that our corrections-aware approach obtains higher task completion rates, and is subjectively preferred by users because of its reliability, precision, and ease of use.1

1Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.


CCS CONCEPTS

• Computing methodologies → Cooperation and coordination; Natural language processing; Learning from demonstrations.

KEYWORDS

Online corrections, language & shared autonomy, robot learning

1 INTRODUCTION

Research in natural language for robotics has focused on dyadic, turn-based interactions between humans and robots, often in the instruction following regime [2, 3, 46, 49]. In this paradigm a human gives an instruction, and the robot executes autonomously – simultaneously resolving the human’s goal as well as planning a course of actions to execute in the environment, without any...
additional user input. While such systems have the potential to substantially impact household and assistive robotics, this explicit division of agency between humans and robots places a tremendous burden on learning algorithms; as a result existing systems either require enormous amounts of language-aligned demonstration data to learn general policies [10, 35, 44, 45], or make similarly restrictive assumptions about known environment dynamics, in addition to the ability to perform perfect object localization and affordance prediction to plug into task and motion planners [28, 38].

Coupled with the sample inefficiency of these approaches is their lack of adaptivity. Consider the robot in Figure 1, trying to execute the long-horizon task “Pick up the book and insert it into the bookshelf.” This is a long-horizon task with several critical states requiring precise manipulation – from grasping the book by its spine, to raising it above the table without hitting the side table, to lining it up precisely with the bookshelf for insertion (with less than a few millimeters of clearance on either side). In such circumstances, even the best of the prior approaches would fail to complete the task in a repeatable, generalizable fashion. Yet these “task failures” are almost never truly catastrophic; for a task like this, the robot would not crash into the table or cause irreversible damage to the environment. Instead, the difference between success and failure is innocent; a missed grasp (by a few centimeters), or failure to align the angle of the book with the opening to the bookshelf, as shown in Figure 1 – innocent errors that a human supervisor could easily rectify with a simple mechanism to refine a robot’s behavior.

One natural option for implementing such a mechanism is to augment the robot with a way to incorporate and adapt given streaming natural language corrections – from simple corrections like “left!” to “tilt down a little bit” (as in Figure 1), to more the more complex “no, towards the blue marker”. While recent work tries to get at the spirit of this idea by learning from dialogue [47, 48], post-hoc corrections [7, 8, 12, 43], or implicit feedback [23], none of these approaches work online as the robot executes in real-time.

Instead, we argue that scalable systems for language-driven human-robot interaction must be able to handle streaming corrections in a manner that is both adaptive and sample efficient.

To address both of these concerns, we introduce a novel approach – LILAC: Language-Infused Latent Actions with Corrections – that presents a generalizable framework for adapting to online natural language corrections built within a shared autonomy [1, 14, 21, 31] paradigm for human-robot collaboration. Working in a shared autonomy setting allows us to develop correction-aware systems with extreme sample efficiency, learning to collaboratively perform complex manipulation tasks like those in Figure 1 from 10-20 demonstrations instead of the thousands to tens of thousands of demonstrations required by imitation or reinforcement learning [10, 20, 34, 35]. These gains are rooted in the idea of splitting agency between the human and robot; during execution, both parties influence the ultimate actions of the robot, sharing the burden of reasoning over actions. Specifically, we build atop the assistive teleoperation with learned latent actions literature [22, 24, 25, 33], where we use demonstrations to learn task-aware, low-dimensional control spaces that a human can use to control the robot. By factoring the difficulty of manipulation between the human and the robot, we can play to the strengths of both: the robot learns the coarse, temporally-extended features of a task, while offloading the short, fine-grained portions to the human.

We evaluate the efficacy of our correction-aware LILAC method via a within-subjects user study (n = 12), where users complete a complex set of manipulation tasks on a Franka Emika Panda arm using LILAC and two baselines – the state-of-the-art language-informed latent actions (LILA) model [24], as well as a fully autonomous language-conditioned imitation learning approach. We find that LILAC obtains higher task success rates than either baseline because of its ability to adapt given online language instructions, and that users qualitatively find LILAC to be more reliable, precise, and easy to use.
2 MOTIVATING EXAMPLE

The learned latent actions paradigm \[22, 25, 33\] was initially conceived of in the context of assistive teleoperation; given users with limited mobility, finding an intuitive manner of controlling a 7-DoF+ assistive robotic arm with low-dimensional controllers (e.g., a 2-DoF joystick attached to a wheelchair) is extremely difficult. Naive approaches for mapping the high-dimensional robotic control problem to a low-DoF interface – for example, by controlling the \((x/y, z/\text{roll}, \text{pitch/yaw})\) axes of the end-effector independently with the joystick – lead to high amounts of user discomfort, with frequent mode-switching, imprecise controls, and high cognitive load for users \[1, 16\]. Learned latent actions – and specifically, the latest work on Language-Informed Latent Actions (LILA) \[24\] – offer a compromise: use a small number of task-specific demonstrations to learn a nonlinear mapping from joystick axes to end-effector control axes, such that each axis of the joystick represents semantically meaningful movement through task space.

As a concrete example, consider Figure 2 for the task of “pick up the book and insert it into the bookshelf.” LILA learns a single, static mapping to use for the entirety of the episode, and hits a key failure mode; due to compounding errors as the user navigated the book from the table up towards the shelf, the end-effector is misaligned with the shelf, making a clean insertion impossible! This is where the need for adaptivity is demonstrated, and where LILAC shines: by providing a language correction “tilt down a little bit,” – in the midst of robot execution – LILAC refines the user’s control space. Pressing left on the joystick now provides explicit, precise control over the robot’s orientation, allowing the user to even-out the end-effector and complete the task.

3 RELATED WORK

LILAC builds off of a rich body of work in language-informed robot learning, spanning methods for incorporating language corrections for robotic manipulation, learning language-conditioned policies, and incorporating other forms of corrective feedback.

Incorporating Language Corrections for Manipulation. Most relevant to our proposed approach are recent methods for incorporating various types of natural language corrections in the context of robotic manipulation. These methods can be stratified based on the assumptions they make, and when during execution a user provides a correction. For example, Broad et al. \[5\] enables online corrections (similar to LILAC) using distributed correspondance graphs to ground language, via use of a semantic parser that maps language to a predefined space of correction groundings; these groundings are brittle and hand-designed, additionally requiring access to a motion planner (and fully known environment dynamics) to identify a fulfilling set of actions for the robot to execute. In contrast, later work in incorporating corrections removes the need for brittle, hand-designed correction primitives, instead using post-hoc corrections provided at the end of task execution to define composable cost functions that are fed to a trajectory optimizer \[43\]; the post-hoc nature of these corrections are limiting, especially in cases where tasks have irreversible or “hard-to-reset” components, and the trajectory optimizer still requires non-trivial prior knowledge of the environment. Later work relaxes the prior knowledge assumption, but still can only incorporate correction information post-hoc, directly “modifying” trajectory waypoints following a language correction in both 2D \[7\] and 3D \[8\] environments using massive datasets of paired corrections and demonstrations, in addition to data-hungry Transformer \[50\] sequence models. In contrast to these approaches, LILAC is a shared autonomy approach that operates in real-time, without a need for massive amounts of data, or prior environment dynamics or full state knowledge.

Learning Language-Conditioned Policies. More general than incorporating corrections is a tremendous body of work on learning language-conditioned policies in both the full and shared autonomy regimes. Early work in this space used semantic parsers to map natural language instructions to predefined motion planning primitives, given modest sized datasets \[2, 3, 28, 46\]. While these approaches were able to accomplish a limited range of tasks with high reliability, reliance on predefined primitives and motion planners made it hard to scale these approaches to more complex manipulation domains, where environment knowledge is hard to come by, and hand-defined primitives are brittle and limiting. As a result, more recent work in this space learn language-conditioned policies directly via imitation learning from large datasets of paired (language, demonstration) pairs \[20, 35, 39, 44, 45\]. While expressive, these approaches are still tremendously data hungry, requiring hundreds or thousands of examples to learn even the simplest tasks. To address the sample efficiency problem, other work has turned to the shared autonomy regime, learning collaborative human-robot policies from orders of magnitude fewer demonstrations \[18, 24\].

Incorporating Other Forms of Corrective Feedback. Finally, a broad set of approaches tackle learning from other forms of corrective feedback, such as physical corrections \[29, 32\], targeted interventions wherein a human fully assumes control over a robot via remote teleoperation \[17, 26, 36\], critiques \[9, 13\], as well as trajectory preferences \[4, 11\]. More recently, Schmittle et al. \[42\] have proposed a meta-algorithm for online learning from multiple types of corrective feedback (excluding language). While this work is highly impactful, we focus our approach on language corrections, a natural communication modality for human users. Learning to incorporate language corrections also allows for transfer across tasks as correction language such as “to the left” are general and often independent of the current state of the robot, whereas other correction modalities are more context-dependent.

4 LILAC: FRAMING CORRECTIONS

LILAC builds off of LILA as introduced by Karamcheti et al. \[24\] by incorporating natural language corrections during the course of execution. The LILAC architecture, with the highlighted gating mechanism for sample-efficient correction learning is depicted in Figure 3; solid lines denote inference, while dashes denote training. LILAC incorporates natural language corrections in a data-driven way. While corrections like “tilt,” “move left,” are minimally necessary for providing expressive, reliable control, there is no bound on the corrections a LILAC model could handle; this work focuses on both these directional corrections (e.g., “to the left”) and referential corrections (e.g., “towards the blue marker”) – all with a scalable procedure. We dive into the process by which we learn to refine control spaces from corrections in the following section.
### 4.1 Problem Statement

Our setting is that of a sequential decision-making problem defined by elements $(S, A, T, U, C, Z)$ where $s \in S \subseteq \mathbb{R}^n$ denotes the state of the robot and environment, $a \in A \subseteq \mathbb{R}^k$ denotes a robot’s $k$-dimensional action (in our case, a 6-DoF delta in end-effector pose – Cartesian coordinates for position and Euler angles for orientation), and $T : S \times A \rightarrow S$ is a (stochastic) unobserved transition function. Furthermore, $u \in U$ denotes a high-level natural language instruction provided by the user, $c \in C$ denotes a natural language correction, and $z \in Z \subseteq \mathbb{R}^d$ where $d \ll k$ denotes a user-provided input via their low-dimensional control device (e.g., a 2-DoF joystick). We assume that a high-level instruction $u$ is provided once, at the beginning of each episode, defining the user’s goal for a given interaction. However, users can provide an arbitrary number of corrections $c$ throughout the episode in real-time to adapt the robot’s behavior. We model corrections as a last-in, first-out stack $[c]$: users can “push” new corrections onto the stack simply by speaking (e.g., “no, tilt down a little bit!”), and can “pop” corrections off of the stack by pressing a button, signaling that their latest correction has been properly addressed.

The goal of LILAC is to learn a function $F_\theta(s_t, z_t, u^*, [c]) : S \times U \times [C] \times Z \rightarrow \mathcal{A}$ that maps the current robot and environment state $s_t$, low-dimensional control input $z_t$, initial high-level utterance $u^*$ provided by the user (held constant throughout the given episode), and (possibly empty) stack of language corrections $[c]$, to a high-dimensional robot action $a_t$ that is to be executed in the environment. The corresponding low-DoF control manifold $\cup_{z_t} F_\theta(s_t, z_t, u^*, [c])$ provides an intuitive interface for the user to maneuver the robot towards satisfying the task in question. At each subsequent timestep $t + 1$, a user can either provide a new language correction $c'$ which is “pushed” onto the stack, press a button to “pop” their latest correction off of the stack $[c]$, signalling that their correction has been addressed, or provide a control input $z_t$ that is mapped to the corresponding robot action $a_t$.

### 4.2 Learning

We learn $F_\theta$ by treating the mapping of low-dimensional user inputs $z$ to high-dimensional robot actions $a$ as learning a state and language conditional autoencoder as in prior learned work [24, 31], with added structural elements. In this formulation, the encoder takes in (language, state, action) triples $\{s_t, a_t, u_t\}$ where $u_t$ is a high-level natural language instruction provided by the user (held constant throughout the episode), and $s_t$ and $a_t$ are the state and action components, respectively. The encoder then learns to map these triples to a representation in the latent space $h_t = F_\theta(s_t, a_t, u_t)$, $z_t = g_t(s_t, a_t, u_t)$, where $g_t$ provides an appropriate assignment to the corresponding robot action $a_t$.

Inference

We use GPT-3, a strong, pretrained language model, in lieu of a heuristic to provide $\alpha$ (see §4.4 for discussion).
We use a pretrained Distil-Roberta model from Sentence-BERT [41] to encode language utterances, in tandem with an "unnatural-language processing" nearest neighbors index [37]; inference-time utterances are projected onto existing training exemplars, preventing the LILAC model from generalizing poorly as language embeddings drift, which could lead to practical issues of user safety.

4.3 Gating Instructions vs. Corrections

Key to scaling LILAC is the insight that various forms of correction language are generalizable across states – in other words, different language utterances require different amounts of object/environment state-dependence. Formally defining the "state-dependence" of a language utterance is hard; one heuristic might be to categorize different utterances based on the number of referents present; an utterance like "grab the thing on the side table and place it on the table" as in Figure 3 has 3 referents, indicating a large degree of state dependence; the robot must ground the utterance in the objects of the environment to resolve the correct behavior. However, an utterance like "no, to the left!" has no explicit referents; one can resolve the utterance by relying solely on the user’s static reference frame and induced deltas in end-effector space.

To operationalize this idea with LILAC, further contributing to the sample efficiency of our approach, we use a gating function (orange, in Figure 3) that given language, predicts a discrete value $\alpha \in \{0, 1\}$. A value of 0 signifies a state-independent utterance – for example, the correction "tilt down a little bit." Appropriately, in our architecture, this zeroes out any state-dependent information (see the $\alpha$ term in Figure 3), and predicts an action solely based on the provided language. In this work, we construct a prompt harness with GPT-3 [6] to output $\alpha$ – a discussion on this process can be found in the following section.

4.4 Using GPT-3 to Identify Corrections

Characterizing the state-dependence of an utterance is hard; while the aforementioned reference counting heuristic may work in some cases, utterances such as "no, the blue!" have implicit referents (in this case, perhaps a marker, or cup) that need state. Many other phenomena make it hard to define heuristics for computing $\alpha$ – anaphora, null referents ("move the robot left"), etc. Instead of crafting grammars or heuristics, we opt to tap into the power of large language models with in-context learning abilities, that learn to extrapolate given a prompt and small set of examples. We specifically build off of GPT-3 text-davinci-002 (175B parameters) [6].

We defined a prompt for GPT-3 by choosing a series of examples from our collected dataset of language and demonstrations, holding out 5 language utterances as a validation set. We allocated a 5-minute budget for defining a prompt that maximized performance on the unseen validation utterances.3

4.5 Reproducibility: Architecture & Training

We provide full details on model architecture and training for reproducibility in this section. We additionally release an open-source codebase (https://github.com/Stanford-ILIAD/lilac) with the complete pipeline spanning data collection, model definition, training, and real-robot deployment.

Model Architecture. LILAC consists of the two subproblems identified in §4.2: 1) learning the set of $d$ basis vectors given language and state, and 2) finding the latent action weights $z$ on each of these bases. For generating basis vectors, we first encode state by concatenating the proprioceptive state of the robot (both joint angles and end-effector pose) with a fixed cardinality set of $n$ object coordinates $(x, y, z)$ (absolutely ordered). This state is then fed to a Batch Normalization layer [19] for training stability, then projected to a state embedding via a 2-layer MLP. Language is similarly encoded; after performing the nearest-neighbors retrieval to identify the closest training exemplar, the respective Distil-Roberta embedding [41] is mapped via a FiLM layer [40] into affine transformation parameters $\gamma$ and $\beta$ via separate 2-layer MLPs. Based on the computed $\alpha$ under GPT-3 for the given utterance, we then take the convex combination of the state and learned "no-context" embedding (using $\alpha$ to set the weighting), resulting in a gated embedding $h_{\text{gated}}$: We then applying the FiLM parameters to obtain a fused embedding $h_{\text{fused}} = \gamma \cdot h_{\text{gated}} + \beta$ before feeding the fused embedding to a separate 2-layer MLP that predicts the $d \times k$ basis matrix $B$ (where $k$ is the dimensionality of our action space). We orthonormalize these basis vectors using a Modified Gram-Schmidt process.

To produce the latent action weights $z$, we learn a 2-layer MLP that takes in the concatenated $h_{\text{fused}}$ and high-DoF action $a$. The "predicted action" $\hat{a}$ is the linear combination of the bases $B$ and weights $z$: $\hat{a} = \sum_{i=1}^{k} z_i B_i$. We optimize reconstruction loss (mean-squared error) between the original robot action $a$ and $\hat{a}$ to train the entire network end-to-end. All MLPs use the same hidden dimension ($h = 128$) and the GELU activation [15].

Training Details. Training LILAC is efficient, and can be run on consumer laptop CPUs, eschewing the need for expensive GPUs. We train for 50 epochs, with a batch size of 512, using the AdamW optimizer [27] with default learning rate of 0.001 and weight decay of 0.01. We do not use any other form of regularization (e.g., Dropout). We select models based on validation loss with respect to a small number ($n = 5$) of held out (language, trajectory) pairs.

5 USER STUDY PRELIMINARIES

To evaluate the efficacy of LILAC with respective to prior methods for language-informed policy learning, we conduct a within-subjects user study with $n = 12$ participants, with each participant evaluating LILAC against language-conditioned approaches for full and shared autonomy – namely, a language-informed imitation learning baseline trained with the same number of demonstrations as LILAC, as well as LILA, also trained using the same number of demonstrations. The following sections detail the environment, set of defined tasks, data collection process, as well as participant demographics and user study procedure. Finally, we list our independent variables, dependent measures and concrete hypotheses before moving on to the results in the following section.

Environment & Tasks. We consider a tabletop multi-task "work desk" environment as depicted in Figure 5, consisting of the following 5 high-level tasks, listed in order of complexity:

3A valid question is why not treat all utterances as requiring uniform, or the same amount of state-dependence; the answer is rooted in the small data regime we operate in. We'd need to collect several instances of the same correction "to the left" in different states to generalize, whereas with LILAC's "gating" approach, we only need one!

Figure 4: Results from our within-subjects user study results \((n = 12)\) across three conditions: 1) Language-Conditioned Imitation Learning – demonstrating the status quo in language-informed human-robot interaction, 2) Language-Informed Latent Actions (LILA) – an instantiation of language-informed shared autonomy without online corrections, and 3) LILAC – our shared autonomy approach where users can provide online corrections at any point during robot execution.

(1) **clean-trash**: throw away a piece of crumpled paper (deformable) into the black trash bin.

(2) **transfer-pen**: transfer the blue marker (upper left of Figure 5) from the shelf into the metal tin holder (lower left).

(3) **open-drawer**: Open the bottom drawer on the shelf by grasping the small knob, and sliding out horizontally (requires fine-grained end-effector orientation control).

(4) **insert-book**: Pick up the book on the table by its spine, and insert it into the bookshelf (has only a few millimeters of clearance on either side).

(5) **water-plant**: Water the succulent (white bowl on the upper right of Figure 5) using the water in the yellow cup (rather than actual water, we use marbles for easy cleanup).

Each of the 5 tasks we define are difficult from a manipulation perspective, especially in the small data regime we operate in. For fine-grained comparison between the three approaches under study, we define a set of sub-tasks to use to measure partial task success (turning a sparse full-task success rate into a denser measure of progress): a) reaching the desired object to manipulate (e.g., the book in the **insert-book** task), b) successfully grasping the manipulable object, c) transferring the desired object to the target location (e.g., moving the water cup above the plant for the **water-plant** task), and finally, d) completing the full task.

**Demonstration & Correction Data Collection.** For each task, we collected dense, human-guided kinesthetic demonstrations – 50 full-task demonstrations total (orders of magnitude fewer demonstrations than what is typically required for fully autonomous instruction following approaches [35, 39]). The robot’s proprioceptive state is encoded as the concatenation of its joint states (7-DoF; in radians), as well as the end-effector poses computed via forward kinematics (expressed as 3-DoF Cartesian position, and 3-DoF Euler angle orientation), and compute the high-level robot actions as the deltas in end-effector space between consecutive time steps in our demonstrations (resulting in 6-DoF high-level actions). We record our demonstrations and run our controllers at 10 Hz; we use Polymetis [30] as the basis for our robot control platform.

For LILAC, we additionally collect a small set of correction demonstrations (collected under 2 hours of interaction with the robot) with associated correction language utterances, spanning two loose categories: 1) directional corrections such as “tilt down,” “to the right,” “rotate counterclockwise”, and 2) contextual referential corrections such as “towards the blue marker” or “no, move down towards the knob on the drawer.” We collect these demonstrations by replaying the full-task demonstrations, and sampling random intermediate states during playback to initiate corrections. The authors of this work served as the expert demonstrators for both full-task and correction demonstrations.

**Participants & Procedures.** All user studies were conducted subject to a university-approved IRB protocol, with participants recruited from a pool of 12 university students (8 male/4 female, age...
imitation learning baselines are trained on the exact same data as the LILA and language-conditioned Baseline Implementations.


Hypotheses. In this study, we vary the control strategy (imitation model, LILA and LILAC) and use the objective measures of subtask and full-goal success rate to assess the various strategies. We additionally track qualitative aspects such as “ease of use,” “smooth control,” and “likelihood of using this control strategy again” (full set can be found in Figure 4) by surveying our users via a 7-point Likert scale. We test the following two hypotheses regarding LILAC’s performance relative to the baseline strategies:

H1 – LILA allows users to obtain higher subtask and full-goal success rates when completing complex manipulation tasks relative to specifying tasks for a language-conditioned imitation learning agent, or LILA trained on the same amount of data.

H2 – LILAC is qualitatively preferred by users over both baseline strategies in terms of overall usability measured by their subjective responses to the survey questions.

Baseline Implementations. Both the LILA and language-conditioned imitation learning baselines are trained on the exact same data as the language feedback from Method C (LILAC) was helpful."

"Control method A (Imitation) wasn’t accurate enough; if it was perfect, it would be my favorite though. Control method B (LILA) helped me get the general motions, but was not precise. Control method C (LILAC) allowed for precise corrections that let me complete the task."

We also find that LILAC is subjectively preferred by users, as evidenced by Figure 4 (Right). Looking at the results from each survey question (framed on a 7-point Likert scale), we find that LILAC is significantly (p < 0.05) preferred on 6 out of the 7 metrics, including "ease of use," "low mental effort," "intuitiveness," and "willingness to use again," amongst others. These subjective results overwhelmingly support H2: we find that LILAC is favored due to its adaptability, allowing users to execute targeted periods of precise motions. We also collected open responses from users during the survey – a representative subset can be found below:

Figure 6: Qualitative trajectories across the different control strategies for the open-drawer and water-plant tasks. The fully autonomous imitation learning approach fails to make it beyond the first stage of the task, while LILA is able to reach the drawer as well as the cup but fails to precisely aim and grasp the object. LILAC gets stuck at the same place, but is able to recover as the user issues low-level corrections to precisely maneuver the end-effector and fully complete the tasks.

range 20-30 with mean 24.8). Of the 12 total participants, only 4 users had prior experience teleoperating a robot. All our studies used a Franka Emika Panda robot (as depicted in Figure 5), a 7-DoF fixed-arm manipulator with a parallel-jaw gripper. In all settings, the maximum joint velocity norm of the Panda was bounded to 1 rad/sec, with conservative torque limits of 40.0 Nm.

We conducted a within subjects user study where each participant used all three candidate methods (denoted as Imitation, LILA, and LILAC (Ours)) in Figure 4 to complete 3 of the 5 high-level tasks. We shuffled the order of candidate methods between users to ensure a fair comparison. Upon starting, each user read a detailed written description of each control method (explicit text can be found on the project website), viewed a video depicting the high-level task to perform, and were allowed a single "practice" session to with the control method in question. Each user performed two trials for a given task; we recorded partial success rates and asked the user to fill out a qualitative survey before switching to the next strategy.

We report the success rate for each subtask averaged across users in Figure 4 (Left). Objectively, we find that LILAC achieves the highest success rate across all subtasks, and is significantly (p < 0.05) more performant than the imitation learning and LILA baselines for the latter three subtasks – grasping, transfer, and full task completion – results that fully support H1.

We also find that LILAC is subjectively preferred by users, as evidenced by Figure 4 (Right). Looking at the results from each survey question (framed on a 7-point Likert scale), we find that LILAC is significantly (p < 0.05) preferred on 6 out of the 7 metrics, including “ease of use,” “low mental effort,” “intuitiveness,” and “willingness to use again,” amongst others. These subjective results overwhelmingly support H2: we find that LILAC is favored due to its adaptability, allowing users to execute targeted periods of precise motions. We also collected open responses from users during the survey – a representative subset can be found below:

"It was hard to get precise control with Method B (LILA), so for that, the language feedback from Method C (LILAC) was helpful."

"Control method A (Imitation) wasn’t accurate enough; if it was perfect, it would be my favorite though. Control method B (LILA) helped me get the general motions, but was not precise. Control method C (LILAC) allowed for precise corrections that let me complete the task."

While the results of the user study are compelling, we find it important to be transparent about the shortcomings of the current approach, addressing possible avenues for future work.

**Limitations.** While LILAC proves effective relative to fully autonomous and correction-free shared autonomy baselines, much more work can be done addressing different types of language corrections that are more context sensitive; as a concrete example, due to the way we encode incoming corrections (as described in §4), we interpret each utterance on its own, independent of what was said previously. This is hugely limiting for interpreting phenomena such as anaphora or implicit coreference – corrections such as “no, the other way,” or “undo that” are currently beyond LILAC’s abilities. Furthermore, we find that while corrections offer additional flexibility over the base shared autonomy control space, they can be easily overused – we noticed that certain participants in our study quickly departed from the control space induced by the high-level instruction, instead opting to complete the bulk of the task in correction mode, effectively turning LILAC into a glorified end-effector control, with users moving one axis at a time. Work on naturalizing the underlying high-level control spaces will be crucial for scaling LILAC to more complex, temporally extended tasks where low-level corrections alone may not afford users enough expressivity to solve tasks (e.g., if tasks requiring modulating 3+ degrees-of-freedom simultaneously), or would result in extremely slow, frustrating task completion. Finally, we find that corrections such as “rotate” or “tilt” can be ambiguously interpreted, with some users intending for the correction to be interpreted subject to their reference frame rather than the robot’s reference frame, or vice-versa.

**7 DISCUSSION**

While the results of the user study are compelling, we find it important to be transparent about the shortcomings of the current approach, addressing possible avenues for future work.

**Visualizations.** To further understand the value of LILAC and incorporating online corrections, we visualize example trajectories for each of the three control strategies for two high-level tasks in Figure 6. On the left are trajectories for the simple open-drawer task: we see that the fully autonomous imitation learning model fails to reach the drawer entirely, whereas LILA and LILAC are able to successfully reach the drawer, but get stuck trying to precisely aim and grasp the small knob. While LILA cannot recover, LILAC is receptive to the user’s correction, producing a refined control space allowing for the user to complete the grasp and finish out the task. We see a similar story with the more difficult water-plant task: imitation learning fails catastrophically by knocking over the cup, causing irreversible damage to the environment, while LILA reaches the cup, but does not afford the user enough precision to make a successful grasp. With LILAC, two tightly sequenced corrections allow the user precise, targeted control, first in acquiring the cup, then in aligning the end-effector orientation to complete the pouring motion successfully. These trajectory visualizations offer insight into where and when corrections are most useful – specifically showing the need for adaptivity in critical states.

Figure 7 additionally plots the 3D end-effector trajectories (position; orientation is omitted for clarity) across all users for the open-drawer task for both LILA and LILAC, in addition to the trajectories represented in the training data. We find that states LILAC is receptive to the user’s correction, producing a refined control space allowing for the user to complete the grasp and finish out the task. We see a similar story with the more difficult water-plant task: imitation learning fails catastrophically by knocking over the cup, causing irreversible damage to the environment, while LILA reaches the cup, but does not afford the user enough precision to make a successful grasp. With LILAC, two tightly sequenced corrections allow the user precise, targeted control, first in acquiring the cup, then in aligning the end-effector orientation to complete the pouring motion successfully. These trajectory visualizations offer insight into where and when corrections are most useful – specifically showing the need for adaptivity in critical states.

**Conclusion.** Throughout this work, we have argued that scalable systems for language-driven human-robot interaction must be able to exhibit both adaptivity and sample efficiency to enable widespread adoption. We identified the ability to handle online natural language correction as a way to enrich existing systems with such adaptivity, presenting LILAC – Language-Informed Latent Actions with Corrections – as a potential answer. LILAC is a concrete approach built within the shared autonomy paradigm whereby natural language utterances are mapped to meaningful, low-dimensional control spaces that humans can use to guide the robot, with each correction provided by the user working to refine the underlying control space, allowing for precise, targeted control of the robot in critical states. Our user study comparing LILAC with fully autonomous imitation learning and language-informed shared autonomy baselines shows the importance of being able to adapt to online corrections, as LILAC is both qualitatively preferred by users, and is able to obtain significantly higher task success rates than baselines. LILAC marks a strong step forward in adaptive language-driven approaches for shared autonomy, and we hope that its core tenets of reliability, precision, and ease of use are carried forward throughout future work.