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

Yuchen Cui*
yuchenc@cs.stanford.edu
Stanford University
Stanford, CA, USA

Siddharth Karamcheti*
skaramcheti@cs.stanford.edu
Stanford University
Stanford, CA, USA

Raj Palleti
Stanford University
Stanford, CA, USA

Nidhya Shivakumar
The Harker School
San Jose, CA, USA

Percy Liang
Stanford University
Stanford, CA, USA

Dorsa Sadigh
Stanford University
Stanford, CA, USA

ABSTRACT

Systems for language-guided human-robot interaction must satisfy two key desiderata for broad adoption: *adaptivity* and *learning efficiency*. Unfortunately, existing instruction-following agents cannot adapt, lacking the ability to incorporate online 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 Language-Informed Latent Actions with Corrections (LILAC), a framework for incorporating and adapting to natural language corrections – "to the right", or "no, towards the book" – *online, during execution*. We explore rich manipulation domains within a *shared autonomy* paradigm. Instead of discrete turn-taking between a human and robot, LILAC *splits agency* between the human and robot: 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 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]

CCS CONCEPTS

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

*Both authors contributed equally to this research.

[1]Project website with videos & study interface: https://sites.google.com/view/hri-lilac. Code for data collection, model definition, training, and evaluation: https://github.com/Stanford-ILIAD/lilac.

HRI '23, March 13–16, 2023, Stockholm, Sweden

ACM ISBN 978-1-4503-9964-7/23/03.
https://doi.org/10.1145/3568162.3578623

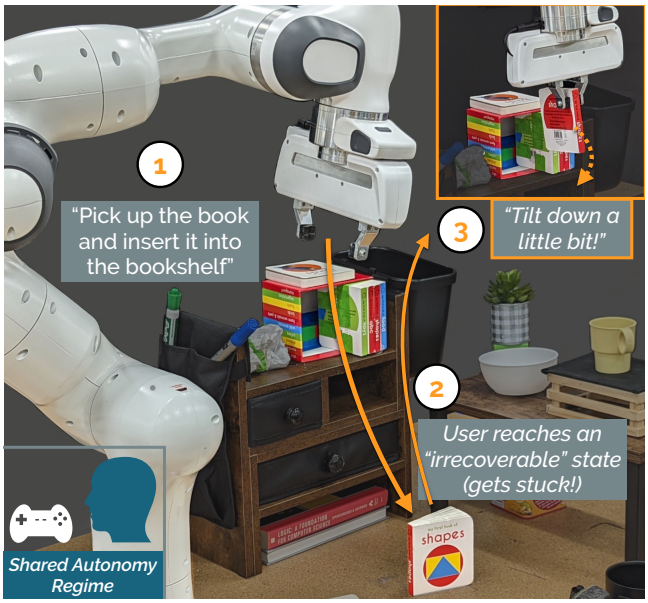

Figure 1: LILAC: Whereas prior work only allows for issuing a *single* language utterance for the entire task ("Pick up the book and insert it into the bookshelf" – solid line), our approach allows users to provide *language corrections* at any point during execution, allowing the robot to adapt online ("Tilt down a little bit!" – right window).

KEYWORDS

Online corrections, language & shared autonomy, robot learning

ACM Reference Format:
Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. 2023. *"No, to the Right"* – Online Language Corrections for Robotic Manipulation via Shared Autonomy. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction (HRI '23), March 13–16, 2023, Stockholm, Sweden.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3568162.3578623

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

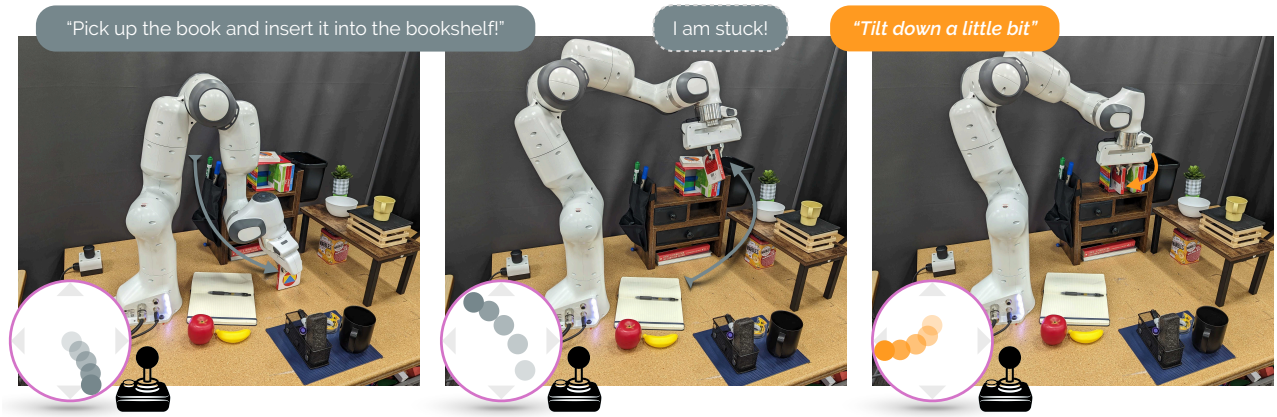

Figure 2: A user interacts with our system. [Left] The user utters "pick up the book and insert it into the bookshelf," inducing a low-dimensional controller (depicted with the joystick and shaded inputs). [Middle] This control space is state and language-conditioned: pressing down brings the end-effector close to the book, while holding up/left after grasping the book moves the end-effector towards the shelf. However, this *static* controller is not enough; the user gets stuck! [Right] Our approach allows users to provide real-time corrections ("tilt down a little bit") refining the control space so the user can complete the task.

gives an instruction, *then* 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. This explicit division of agency between humans and robots places a tremendous burden on learning; existing systems either require large amounts of language-aligned demonstration data to learn policies [10, 34, 44, 45], or make other 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 [27, 37].

Coupled with the sample inefficiency of these approaches is their *lack of adaptivity*. Consider the robot in Figure 1, trying to execute "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 existing approaches fail to complete the task repeatably. Yet these "task failures" are often predictable and recoverable. A user watching a robot diving towards a glass bowl knows that catastrophe is seconds away, and how to avert it; similarly, fine-grained errors such as a missed grasp, or a misaligned end-effector are similarly fixable – as long as the user is provided with the right mechanism to adapt the robot's behavior.

One way to enable such adaptation is through *natural language corrections* – from the simple "left!" or "tilt down a little bit" (as in Figure 1), to 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 [22], none of these approaches work in *real-time*. Instead, we argue that scalable systems for language-driven human-robot interaction must be able to handle online corrections in a manner that is both *adaptive* and *sample efficient*.

We introduce a novel approach – **LILAC: Language-Informed Latent Actions with Corrections** – that presents a generalizable framework for adapting to *online* natural language corrections built within a *shared autonomy* [1, 14, 20, 30] paradigm for human-robot collaboration. With LILAC, a user provides a stream of language utterances, starting with a high-level goal ("Pick up the book and insert it into the bookshelf"), with each utterance shaping the control the user is afforded over the robot. At any point during execution, a user can provide a new utterance – a correction like "tilt down a little bit!" – which updates that control space *online*, reflecting the user's intent in real time. Working in a shared autonomy setting like this gives us the *adaptivity* mentioned above, but also allows us to develop correction-aware systems with extreme *sample efficiency*. With LILAC, we can learn to 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 fully autonomous imitation or reinforcement learning approaches[10, 19, 33, 34]. 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.

We evaluate LILAC 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 [23], 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 [21, 32] 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/roll, 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) [23] – 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 we need *online language corrections* – the mechanism that allows the user to quickly diagnose the problem and refine the robot's behavior. With LILAC, the user provides the correction "tilt down a little bit," in the midst of execution and switch into a new 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 spanning methods for incorporating language corrections, learning language-conditioned policies, and incorporating other forms of corrective feedback.

Incorporating Language Corrections for Manipulation. Most relevant to our 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 data-efficient online corrections (similar to LILAC) using distributed correspondence 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 is limiting, especially in cases where tasks have irreversible or "hard-to-reset" components, and the trajectory optimizer requires non-trivial knowledge of the environment. Later work relaxes these prior knowledge assumptions, but 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 contrast to these approaches, LILAC is a shared autonomy approach that operates *online, in real-time*, without a need for massive amounts of data, 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, 27, 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 [19, 34, 38, 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 such as LILA [23] have turned to the shared autonomy regime, learning collaborative human-robot policies from orders of magnitude fewer demonstrations. LILA is the starting point of our proposed approach.

Incorporating Other Forms of Corrective Feedback. Other approaches tackle learning from other forms of corrective feedback, such as physical corrections [28, 31], targeted interventions wherein a human fully assumes control over a robot via remote teleoperation [17, 25, 35], 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 promising, 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 can be more context-dependent.

4 LILAC: FRAMING CORRECTIONS

LILAC builds off of LILA as introduced by Karamcheti et al. [23] by incorporating natural language corrections during the course of execution. The LILAC architecture is depicted in Figure 3; solid lines denote inference, while dashes denote training. LILAC incorporates natural language corrections in a data-driven way; this work focuses on directional corrections (e.g., "to the left") and referential corrections (e.g., "towards the blue marker") – all with a generalizable and scalable procedure that can be extended to other more complex types of corrections. This section outlines all the elements of our approach.

4.1 Problem Statement

Our setting is that of a sequential decision making problem defined by elements $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{U}, \mathcal{C}^*, \mathcal{Z})$ where $s \in \mathcal{S} \subseteq \mathbb{R}^n$ denotes the state of the robot and environment, $a \in \mathcal{A} \subseteq \mathbb{R}^k$ 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 $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a (stochastic) unobserved transition function. Furthermore, $u \in \mathcal{U}$ denotes a high-level natural language instruction provided by the user, $\mathbf{c} \in \mathcal{C}^*$ denotes the ordered

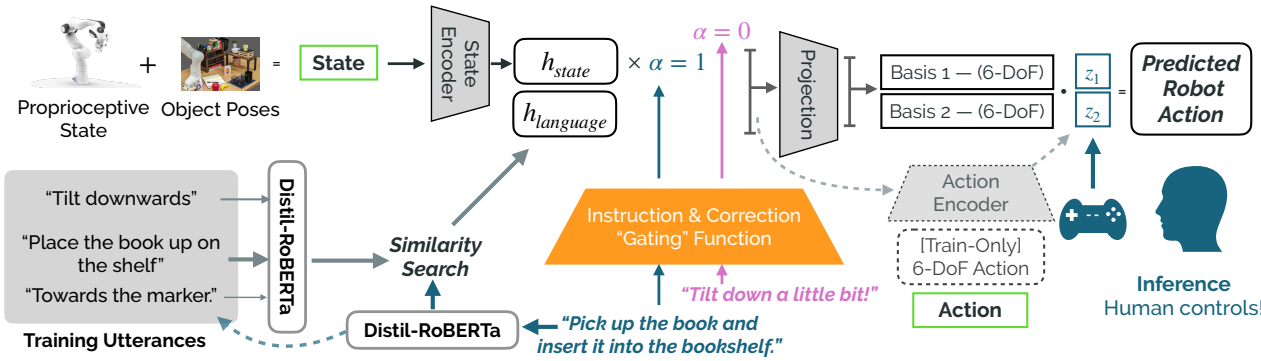

Figure 3: LILAC Overview – solid lines represent the inference pipeline, while dashed lines indicate training-only steps. Part of LILAC's ability to incorporate language corrections efficiently is the "gating" module (orange) which controls the amount of state-context for a given input – for example, grounding a correction such as "tilt down a little bit" requires no state context ($\alpha = 0$), whereas a high-level instruction such as "pick up the book and insert it into the bookshelf" does require context ($\alpha = 1$). We use GPT-3, a pretrained language model, to provide α (see §4.3 for discussion).

(possibly empty) stack of natural language corrections the user has provided, and $z \in \mathcal{Z} \subseteq \mathbb{R}^d$ where $d \ll k$ denotes a user-provided input via their low-dimensional control device (e.g., a 2-DoF joystick). Users can provide an arbitrary number of online corrections *throughout* the episode to adapt the robot's behavior.

The goal of LILAC is to learn a function $\mathcal{F}_\theta(s_t, z_t, u_t, \mathbf{c}_t) : \mathcal{S} \times \mathcal{Z} \times \mathcal{U} \times C^* \rightarrow \mathcal{A}$ 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 $\mathbf{c}_t$ to a high-dimensional robot action a_t that is to be executed in the environment. The corresponding low-DoF control manifold $\bigcup_{z_t \in \mathcal{Z}} \mathcal{F}_\theta(s_t, z_t, u_t, \mathbf{c}_t)$ provides an intuitive interface for the user to maneuver the robot towards satisfying the task in question. At each new 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 $\mathbf{c}_t$ 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 Modeling: Inference & Learning

Given the current state s and language u and $\mathbf{c}$, $\mathcal{F}_\theta$ maps low-dimensional user control inputs z to the high-dimensional robot actions a. Crisply, we define $\hat{a} = \mathcal{F}_\theta(s, z, u, \mathbf{c})$ as:

$$
\begin{aligned}
h_{\text{state}} \in \mathbb{R}^m &= \texttt{EncodeState}_\theta(s) \\
h_{\text{language}} \in \mathbb{R}^m &= \texttt{EncodeLanguage}_\theta(u, \mathbf{c}) \\
\alpha \in [0, 1] &= \texttt{GPTGating}(u, \mathbf{c}) \\
h_{\text{gated}} \in \mathbb{R}^m &= \alpha \cdot h_{\text{state}} + (1 - \alpha) \cdot \text{bias}_\theta \\
h_{\text{fused}} \in \mathbb{R}^m &= \texttt{FiLM}_\theta(h_{\text{gated}}, h_{\text{language}}) \\
B_{\text{bases}} \in \mathbb{R}^{k \times d} &= \text{Gram-Schmidt}(\texttt{Projection}_\theta(h_{\text{fused}})) \\
\hat{a} \in \mathbb{R}^k &= B_{\text{bases}} \cdot z
\end{aligned}
$$

where m is a hyperparameter denoting the hidden dimensionality of the model (we set $m = 128$ for this work).

We first learn a state encoder $\texttt{EncodeState}_\theta$. The state space we use in this work consists of the robot's proprioceptive state (joint angles & end-effector pose) concatenated with a vector of (x, y, z) positions for each object in the scene; $\texttt{EncodeState}_\theta$ is a two-layer MLP that takes this input and outputs h_{state}.

To encode language (u and $\mathbf{c}$), we adopt a last-in-first-out strategy for selectively encoding utterances, only encoding on the most recent utterance – u at the beginning of an interaction, then the most recent correction c'. We embed this utterance with a frozen variant of the Distil-RoBERTa language model [41] as released by the Sentence-BERT project [40]. This process is backed by an "unnatural language processing" nearest neighbors index [36] where inference-time utterances are mapped onto the closest existing training exemplars, which are then retrieved and fed to the rest of the model. This process, similar to that used in LILA [23] prevents LILAC from degenerating in the presence of slight variations of language, which could lead to practical user safety issues. $\texttt{EncodeLanguage}_\theta$ is another two-layer MLP that takes in the retrieved embedding and outputs h_{language}.

Next, we consider how to fuse the state and language embeddings. A key component of LILAC and its ability to remain data efficient is the GPT-3 [6] "gating" module (shown in orange in Figure 3), and denoted by the scalar value $\alpha \in [0, 1]$ in the equations above. The gating value α reflects a simple insight – certain utterances require different amounts of object/state dependence. For example, an utterance such as "go left" does not require reasoning over any objects in the environment ($\alpha = 0$). A detailed discussion on gating and how we operationalize GPT-3 can be found in §4.3. We use this gating value α to module the amount of state information in h_{gated} by taking the convex combination of α with h_{state} and a vector bias_θ, where $\alpha = 0$ obviates x_{state}.

We incorporate language by using FiLM [39], mapping h_{language} to affine transformation parameters $\gamma \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^m$ via separate learned two-layer MLPs. We then apply the resulting transformation elementwise to h_{gated}, producing $h_{\text{fused}} = \gamma \cdot h_{\text{gated}} + \beta$.

Finally, we predict basis vectors $B_{\text{bases}} \in \mathbb{R}^{k \times d}$ (recall that k is the dimensionality of the high-DoF robot action space, while d is the dimensionality of the user's low-DoF control interface). B_{bases} uniquely defines the user's control manifold for the given timestep; we learn a two-layer MLP $\texttt{Projection}_\theta$ that takes h_{fused} and outputs a matrix with the required dimensions. To serve as an appropriately conditioned control manifold, we run modified Gram-Schmidt to orthonormalize the bases. The final high-DoF robot action $\hat{a} \in \mathbb{R}^k$ is then the matrix-vector product between B_{bases} and the user's input $z \in \mathbb{R}^d$.

Learning from Language & Demonstrations. To learn $\mathcal{F}_\theta$, we assume a dataset of (u = language, τ = trajectory) pairs, where each trajectory is comprised of a sequence of (s, a) pairs; $\tau = \{(s_1, a_1) \dots (s_T, a_T)\}$, and a desired latent action dimensionality d (e.g., $d = 2$ in this work for the number of axes on a joystick). The action space we use are deltas in end-effector space (Cartesian position, Euler angle orientation). Note that these inputs don't fully line up with the signature of $\mathcal{F}_\theta$ – notably, *we do not* have access to "ground-truth" latent actions z for each given robot action a. To address this, we adopt the insight used in prior work on learned latent actions [23, 24, 32]: frame the training process as learning a state-and-language conditional autoencoder, using *compression* as a way to induce meaningful latent action control manifolds.

Specifically, we implement this by adding a layer to compress high-DoF robot actions down to a d-dimensional latent:

$$z_{\text{compressed}} \in \mathbb{R}^d = \texttt{Compress}_\theta(a)$$

$$a_{\text{reconstruct}} \in \mathbb{R}^k = B_{\text{bases}} \cdot z_{\text{compressed}}$$

where B_{bases} is computed from state and language as above.

Given this reconstruction objective, we can write a compact loss function for training: $\mathcal{L}(\theta) = ||a - a_{\text{reconstruct}}||_2^2$ – in other words, minimize the mean squared error between the action high-DoF robot action a and $a_{\text{reconstruct}}$. $\texttt{Compress}_\theta$ is implemented as a two-layer MLP, and is discarded after training.

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 1 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,[2] 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 α term in Figure 3), and predicts an action solely based on the provided language. Critically, the fused representation of language and state provided to the rest of the network defining in $\mathcal{F}_\theta$ is modulated by α – more detail in §4.4.

Using GPT-3 to Identify Corrections. In this work, we construct a prompt harness with GPT-3 [6] to output α. We do this because characterizing the state-dependence of an utterance is difficult; 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 to be grounded in the environment state*. Many other phenomena make it hard to define heuristics for computing α – 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]. To define our prompt, we allocated a 10 minute budget to iteratively engineer the input/output examples and task description, using a held-out set of 5 language utterances to provide signal.[3]

4.4 Reproducibility

To facilitate reproducibility and future work, we 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. All MLPs detailed in §4.2 use $m = 128$ and the `GELU` activation [15]. For stability, we add a single Batch Normalization layer [18] before feeding the concatenated state representation to the state encoder. As implemented, LILAC is extremely lightweight at only 188K parameters.

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 [26] 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.

As mentioned in §4.2, we assume a dataset of utterances and corresponding trajectories. These utterances consist of both the high-level task utterances (e.g., "pick up the book and insert it into the bookshelf") and correction utterances (e.g., "tilt down a little bit"). We run the GPT-3 alpha labeling procedure as a preprocessing step, marking each example with the degree of context-dependence required, then train $\mathcal{F}_\theta$ jointly, on all of our data.

5 USER STUDY PRELIMINARIES

To evaluate LILAC with respect to prior methods for language-informed policy learning, we conduct a *within-subjects* user study with $n = 12$ participants, with each participant evaluating LILAC

[2] A 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 "gating" approach, we only need one!

[3] Prompt: https://github.com/Stanford-ILIAD/lilac/tree/main/scripts/alphas.py.

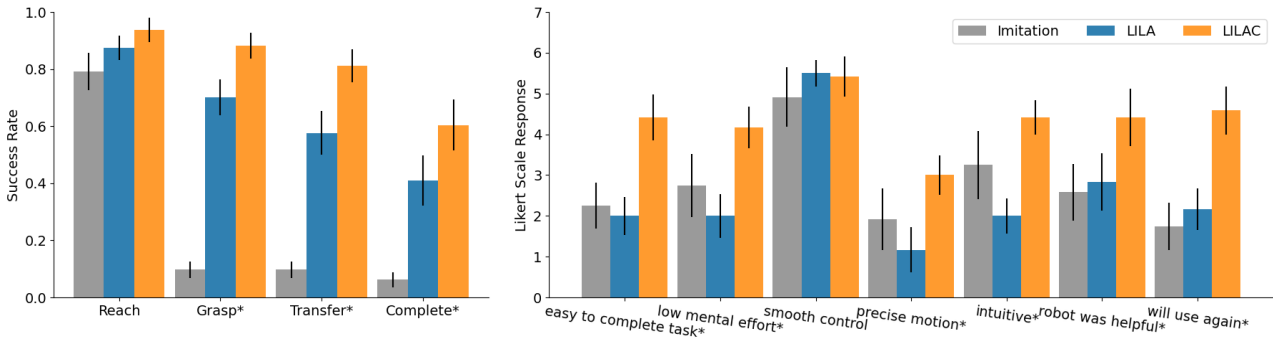

Figure 4: Results from our user study ($n = 12$) across three conditions: 1) Language-Conditioned Imitation Learning, 2) Language-Informed Latent Actions (LILA) – an instantiation of language-informed shared autonomy *without* online corrections, and 3) LILAC – our approach where users can provide online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

against language-conditioned approaches for full and shared autonomy – namely, a language-informed imitation learning baseline ("Imitation") trained on the same demonstrations as LILAC, as well as a non-corrective shared autonomy baseline ("LILA"), also trained on the same demonstrations. The following sections detail the environment, tasks, data collection process, as well as user study procedure. Finally, we list our independent variables, dependent measures and concrete hypotheses.

Environment & Tasks. We consider a multi-task "desk" environment (Figure 5) with the following tasks listed by complexity:

(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).

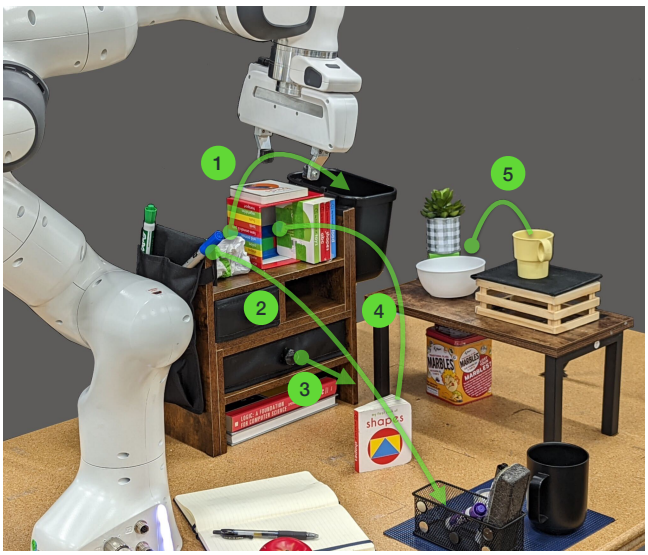

Figure 5: Setup of our tabletop manipulation environment with sketches of our high-level tasks (further details in §5).

(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 *subtasks* 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 [34, 38]). 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 [29] 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

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.

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 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 (the 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.

Hypotheses. In this study, we vary the control strategy (Imitation, LILA and LILAC) and use the objective measures of subtask and full-goal success rates to assess efficacy. We additionally track qualitative aspects such as "ease of use," "smooth control," and "likelihood of using this control strategy again" (the full set of qualitative measures 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 – LILAC 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 Imitation (language-conditioned behavioral cloning) baselines are trained on the *exact same data* as LILAC following nearly identical processes.[4] LILA is trained following the exact same architecture as LILAC from Figure 3, *without* the GPT-3 gating component. The imitation learning model is implemented as a *history-aware* policy that is able to attend to prior states (in this work, we truncate history at 1 second). We use the same state and language encoder as LILAC, but use a 2-layer Transformer [50] to encode the full history sequence. We use again use a FiLM layer to fuse language and state embeddings, then use a final 2-layer MLP to directly predict the action to execute in the environment (for open-loop control).

6 USER STUDY RESULTS

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 survey results, we find that LILAC is significantly ($p < 0.05$) preferred on 6 out of the 7 metrics, including "ease of use," "intuitiveness," and "willingness to use again," amongst others. The subjective results support **H2**; LILAC is favored for its *adaptability*, allowing users to execute targeted, precise motions.

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`

[4] Implementations: https://github.com/Stanford-ILIAD/lilac/tree/main/models.

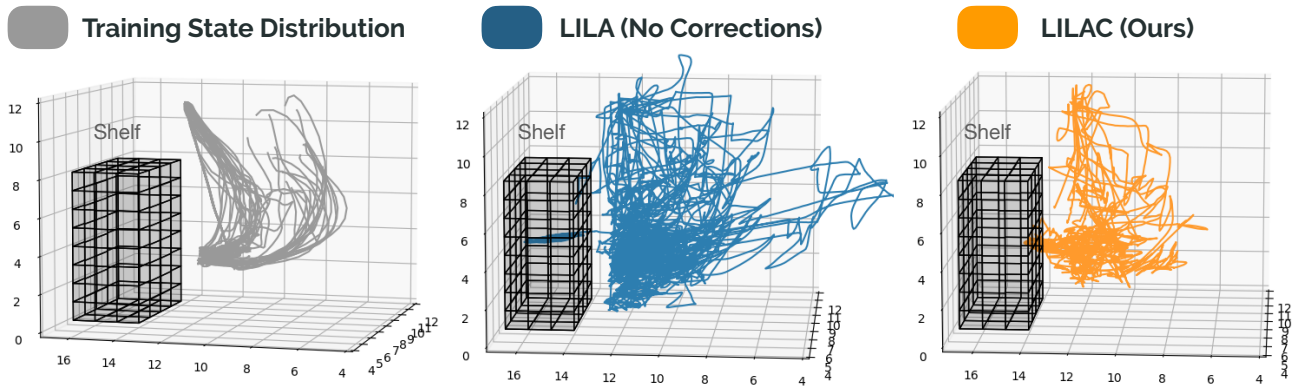

Figure 7: Observed trajectories for LILA and LILAC on the `open-drawer` task (with train trajectories shown on the left). While LILA deviates from the observed state distribution, states traversed with LILAC are close to those seen at training.

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 allows the users to stay closer to the training state distribution compared to LILA, further explaining LILAC's strong performance.

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.

Limitations. Future work should address 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 limiting for interpreting phenomena such as anaphora or implicit coreference – corrections such as "no, the other way," or "undo that" cannot be correctly interpreted using the current instantiation of the framework. 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 making the underlying high-level control spaces more natural and intuitive – *naturalizing* the control interface – 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. Some scenarios where this may occur would be in tasks requiring modulating 3+ degrees-of-freedom simultaneously, or where sequences of low-level corrections only allow users to make frustratingly slow progress at the task at hand (e.g., for long-horizon tasks like making a cup of tea). 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.

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*. We identified the ability to handle *online natural language corrections* as a way to enrich existing systems with such adaptivity, presenting LILAC – Language-Informed Latent Actions with Corrections – as a potential answer. LILAC is 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. Our user study comparing LILAC with language-conditioned imitation learning and language-informed shared autonomy shows the importance of being able to adapt to online corrections, as LILAC is both subjectively preferred by users and objectively performant than both 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.

ACKNOWLEDGMENTS

Toyota Research Institute ("TRI") provided funds to support this work. This project was additionally supported by the Office of Naval Research, as well as by NSF Awards 2006388 and 2132847. Siddharth Karamcheti is grateful to be supported by the Open Philanthropy Project AI Fellowship. Finally, we would like to thank our anonymous reviewers.

REFERENCES

[1] Brenna D Argall. 2018. Autonomy in rehabilitation robotics: an intersection. *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), 441–463.

[2] Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)* 1 (2013), 49–62.

[3] Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Lawson L. S. Wong, and Stefanie Tellex. 2017. Accurately and Efficiently Interpreting Human-Robot Instructions of Varying Granularities. In *Robotics: Science and Systems (RSS)*.

[4] Erdem Biyik and Dorsa Sadigh. 2018. Batch Active Preference-Based Learning of Reward Functions. In *Conference on Robot Learning (CoRL)*.

[5] Alexander Broad, Jacob Arkin, Nathan D. Ratliff, Thomas M. Howard, and Brenna Argall. 2017. Real-time natural language corrections for assistive robotic manipulators. *International Journal of Robotics Research (IJRR)* 36 (2017), 684–698.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165* (2020).

[7] Arthur Fender C. Bucker, Luis F. C. Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, and Rogerio Bonatti. 2022. Reshaping Robot Trajectories Using Natural Language Commands: A Study of Multi-Modal Data Alignment Using Transformers. In *International Conference on Intelligent Robots and Systems (IROS)*. 978–984.

[8] Arthur Fender C. Bucker, Luis F. C. Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, Sai Vemprala, and Rogerio Bonatti. 2022. LaTTe: Language Trajectory TransformEr. *arXiv preprint arXiv:2208.02918* (2022).

[9] S. Chernova and Andrea Lockerd Thomaz. 2014. Robot Learning from Human Teachers. In *Robot Learning from Human Teachers*.

[10] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *International Conference on Learning Representations (ICLR)*.

[11] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[12] John D. Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine. 2019. Guiding Policies with Language via Meta-Learning. In *International Conference on Learning Representations (ICLR)*.

[13] Yuchen Cui and Scott Niekum. 2018. Active Reward Learning from Critiques. In *International Conference on Robotics and Automation (ICRA)*. 6907–6914.

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

[15] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415* (2016).

[16] Laura V Herlant, Rachel M Holladay, and Siddhartha S Srinivasa. 2016. Assistive teleoperation of robot arms via automatic time-optimal mode switching. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. 35–42.

[17] Ryan Hoque, Ashwin Balakrishna, Ellen R. Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. 2021. ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning. In *Conference on Robot Learning (CoRL)*.

[18] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*. 448–456.

[19] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. 2021. BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning. In *Conference on Robot Learning (CoRL)*.

[20] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S Srinivasa, and J Andrew Bagnell. 2018. Shared autonomy via hindsight optimization for teleoperation and teaming. *International Journal of Robotics Research (IJRR)* 37 (2018), 717–742.

[21] Hong Jun Jeon, Dylan P. Losey, and Dorsa Sadigh. 2020. Shared Autonomy with Learned Latent Actions. In *Robotics: Science and Systems (RSS)*.

[22] Siddharth Karamcheti, Dorsa Sadigh, and Percy Liang. 2020. Learning Adaptive Language Interfaces through Decomposition. In *EMNLP Workshop for Interactive and Executable Semantic Parsing (IntEx-SemPar)*.

[23] Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. 2021. LILA: Language-Informed Latent Actions. In *Conference on Robot Learning (CoRL)*.

[24] Siddharth Karamcheti, A. Zhai, Dylan P. Losey, and Dorsa Sadigh. 2021. Learning Visually Guided Latent Actions for Assistive Teleoperation. In *Learning for Dynamics & Control Conference (L4DC)*.

[25] Michael Kelly, Chelsea Sidrane, K. Driggs-Campbell, and Mykel J. Kochenderfer. 2019. HG-DAgger: Interactive Imitation Learning with Human Experts. In *International Conference on Robotics and Automation (ICRA)*. 8077–8083.

[26] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[27] T. Kollar, J. Krishnamurthy, and Grant P. Strimel. 2013. Toward Interactive Grounded Language Acqusition. In *Robotics: Science and Systems (RSS)*.

[28] Mengxi Li, Alper Canberk, Dylan P. Losey, and Dorsa Sadigh. 2021. Learning Human Objectives from Sequences of Physical Corrections. In *International Conference on Robotics and Automation (ICRA)*. 2877–2883.

[29] Yixin Lin, Austin S. Wang, Giovanni Sutanto, Akshara Rai, and Franziska Meier. 2021. Polymetis. https://facebookresearch.github.io/fairo/polymetis/.

[30] Dylan P. Losey, Hong Jun Jeon, Mengxi Li, Krishna Parasuram Srinivasan, Ajay Mandlekar, Animesh Garg, Jeannette Bohg, and Dorsa Sadigh. 2021. Learning latent actions to control assistive robots. *Autonomous Robots (AURO)* (2021), 1–33.

[31] Dylan P Losey, Craig G McDonald, Edoardo Battaglia, and Marcia K O'Malley. 2018. A review of intent detection, arbitration, and communication aspects of shared control for physical human-robot interaction. *Applied Mechanics Reviews* 70 (2018).

[32] Dylan P. Losey, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, and Dorsa Sadigh. 2020. Controlling Assistive Robots with Learned Latent Actions. In *International Conference on Robotics and Automation (ICRA)*. 378–384.

[33] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A Survey of Reinforcement Learning Informed by Natural Language. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[34] Corey Lynch and Pierre Sermanet. 2020. Grounding Language in Play. *arXiv preprint arXiv:2005.07648* (2020).

[35] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. 2020. Human-in-the-Loop Imitation Learning using Remote Teleoperation. *arXiv preprint arXiv:2012.06733* (2020).

[36] Alana Marzoev, S. Madden, M. Kaashoek, Michael J. Cafarella, and Jacob Andreas. 2020. Unnatural Language Processing: Bridging the Gap Between Synthetic and Natural Language Data. *arXiv preprint arXiv:2004.13645* (2020).

[37] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*. 1671–1678.

[38] Oier Mees, Lukás Hermann, and Wolfram Burgard. 2022. What Matters in Language Conditioned Robotic Imitation Learning Over Unstructured Data. *IEEE Robotics and Automation Letters (RA-L)* 7 (2022), 11205–11212.

[39] Ethan Perez, Florian Strub, Harm D. Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. FiLM: Visual Reasoning with a General Conditioning Layer. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

[40] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[41] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[42] Matthew Schmittle, Sanjiban Choudhury, and Siddhartha S Srinivasa. 2020. Learning Online from Corrective Feedback: A Meta-Algorithm for Robotics. In *Conference on Robot Learning (CoRL)*.

[43] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. 2022. Correcting Robot Plans with Natural Language Feedback. In *Robotics: Science and Systems (RSS)*.

[44] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. 2021. CLIPort: What and Where Pathways for Robotic Manipulation. In *Conference on Robot Learning (CoRL)*.

[45] Simon Stepputtis, J. Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and H. B. Amor. 2020. Language-Conditioned Imitation Learning for Robot Manipulation Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[46] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

[47] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2019. Vision-and-Dialog Navigation. In *Conference on Robot Learning (CoRL)*.

[48] Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin W. Hart, Peter Stone, and Raymond J. Mooney. 2019. Improving Grounded Natural Language Understanding through Human-Robot Dialog. In *International Conference on Robotics and Automation (ICRA)*.

[49] Jesse Thomason, Shiqi Zhang, Raymond J. Mooney, and Peter Stone. 2015. Learning to Interpret Natural Language Commands through Human-Robot Dialog. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv preprint arXiv:1706.03762* (2017).