LILA: Language-Informed Latent Actions

Siddharth Karamcheti*, **Megha Srivastava***, **Percy Liang, Dorsa Sadigh**
Department of Computer Science, Stanford University
{skaramcheti, megha, pliang, dorsa}@cs.stanford.edu

Abstract: We introduce Language-Informed Latent Actions (LILA), a framework for learning natural language interfaces in the context of human-robot collaboration. LILA falls under the shared autonomy paradigm: in addition to providing discrete language inputs, humans are given a low-dimensional controller – e.g., a 2 degree-of-freedom (DoF) joystick that can move left/right and up/down – for operating the robot. LILA learns to use language to *modulate* this controller, providing users with a language-informed control space: given an instruction like "place the cereal bowl on the tray," LILA may learn a 2-DoF space where one dimension controls the distance from the robot's end-effector to the bowl, and the other dimension controls the robot's end-effector pose relative to the grasp point on the bowl. We evaluate LILA with real-world user studies, where users can provide a language instruction while operating a 7-DoF Franka Emika Panda Arm to complete a series of complex manipulation tasks. We show that LILA models are not only more sample efficient and performant than imitation learning and end-effector control baselines, but that they are also qualitatively preferred by users.[1]

Keywords: Language for Shared Autonomy, Language & Robotics, Learned Latent Actions, Human-Robot Interaction

1 Introduction

Nearly a million American adults live with physical disabilities, requiring assistance for everyday tasks like taking a bite of food, or pouring a glass of milk [1] – assistance that robots could provide. Paradigms for efficient human-robot collaboration that strike a balance between robot autonomy and human control such as *shared autonomy* [2, 3, 4, 5] present a promising path towards building such assistive systems. Unlike full autonomy approaches that enforce a sharp separation between user intent and robot execution, falling prey to problems of sample efficiency and robustness, shared autonomy couples a human's input with automated robot assistance. Consider a kitchen or dining environment where a *high-dimensional* (high-DoF) robot such as a wheelchair-mounted manipulator aids a human who may be physically unable to perform tasks requiring fine-grained manipulation. While the human can manually teleoperate the arm by fully controlling individual "modes", or separate degrees-of-freedom of the robot's end-effector, past work has shown this to be unintuitive, slow, and frustrating [3, 6]. Shared autonomy approaches such as *learned latent actions* [5, 7, 8] however, build intuitive low-dimensional controllers for high-DoF robots via dimensionality reduction.

Specifically, learned latent actions models learn *state-conditioned* auto-encoders directly from datasets of (state, action) pairs; the encoder takes the current state and action, and compresses it to a latent action z with the same dimensionality as the human control interface (e.g., 2-DoF). This is fed to a decoder to try to reconstruct the original high-dimensional action. While these approaches are reliable and sample efficient, they are limited by their reliance on just the current state: given tasks like "grab the milk" and "shift the milk to the side" that overlap in state space, these controllers fail as the models lack sufficient information to disambiguate behavior.

To address this concern, we consider incorporating natural language within this framework to add an additional conditioning variable for structuring the control space. Prior work integrates language in

[1] Additional visualizations and supplemental experiments can be found at the following webpage: https://sites.google.com/view/lila-corl21. Code can be found here: https://github.com/siddk/lila.

5th Conference on Robot Learning (CoRL 2021), London, UK.

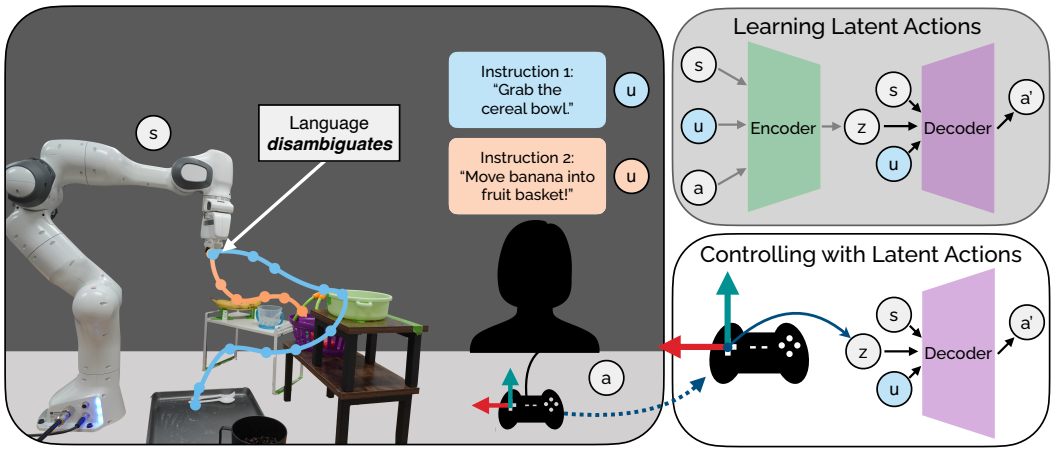

Figure 1: [**Left**] Our breakfast buffet environment with several diverse manipulation tasks. By providing both natural language input and low-dimensional joystick control [**Middle**], users *disambiguate* between different tasks while retaining the ability to maneuver through the environment. This is enabled by our [**Right**] language-informed latent actions (LILA) models that use auto-encoders to learn language & state-conditioned low-DoF latent spaces for meaningful control.

robotics settings for similar purposes within the full autonomy paradigm [9, 10, 11, 12, 13]. Unfortunately, these approaches suffer from poor sample efficiency, failure recovery, and generalization; many issues that shared autonomy methods including learned latent actions seek to address. By joining language and latent actions, a user can express an utterance $u =$ *"grab the cereal bowl"* and obtain a control space that is both state and language conditioned (Fig. 1 [Right]).

We introduce Language-Informed Latent Actions (LILA), a framework for incorporating language into learned latent actions. Key to LILA is the principle that language *modulates* a user's low-level controller based on their provided utterance; as intuition, given the utterance *"grab the cereal bowl"* as in Fig. 1, our assistive robot might learn a semantically meaningful, low-dimensional (2-DoF) control space where one dimension (one joystick axis) may control the distance from the robot's end-effector to the cereal bowl, whereas the other might control the angle of the end-effector relative to the bowl such that the robot's gripper can obtain a solid grasp of the object. Other utterances can modulate the controller in similar ways – "pour the milk into the cup" might result in a learned control space where one joystick dimension controls the jug's pouring angle, while the other may control its height. Language not only serves as a natural means for a human to communicate their intent to the robot, but also helps *disambiguate* across a wide variety of objectives as well, by inducing language and state-conditioned control spaces.

A core part of our method is its ability to handle diverse, realistic language. To this end, we collect a small, crowdsourced dataset of natural language descriptions to describe each of our training demonstrations; we use this real, natural language as the only input while training our models. To allow for out-of-the-box generalization to novel user utterances such as those that describe similar behaviors but with different words or phrases, we tap into the power of pretrained models [14, 15]. We perform a *comprehensive user study* across 10 users who use natural language and our learned LILA controllers to complete a variety of diverse manipulation tasks in a simplified assistive "breakfast buffet" setting. Our results show that LILA models are not only more reliable, performant, and sample efficient than fully autonomous imitation learning and fully human-driven end-effector control baselines, but are qualitatively preferred by users as well.

2 Related Work

We build LILA within a shared autonomy framework [2, 16], applied to assistive teleoperation [17, 18]. We additionally build off of work at the intersection of language and robotics [19, 20, 21].

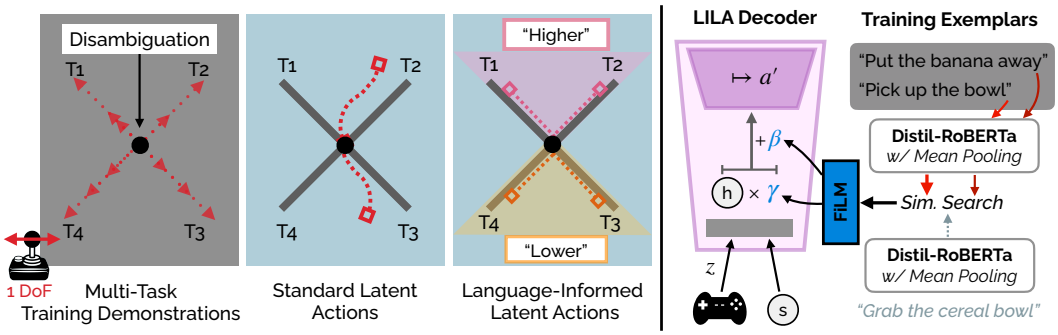

Figure 2: [**Left**] Stylized example: navigating toward four points on a cross with a 1-DoF latent action, where disambiguation is required. Standard latent action models fail, while LILA accurately reaches the corners with the help of language. [**Right**] LILA decoder architecture. We embed an utterance using a pretrained language model, then identify the closest exemplar in the training set via similarity search. We feed the embedding for this exemplar through a feature-wise linear modulation, or FiLM [46], layer that fuses language and state representations within the decoder.

Shared Autonomy & Assistive Teleoperation. Shared autonomy casts robot control as a collaborative process between humans and robots [2, 4, 16, 22]. While other work focuses on "blending" human inputs with possibly task-agnostic policies within the same action space [23, 24], in this work, we focus on assistive teleoperation, where a user is provided a low-dimensional controller (e.g., a joystick, sip-and-puff device) to directly control a high-dimensional robot manipulator. Using these controllers for end-effector control – e.g., via operational space control [25] – is incredibly difficult, requiring frequent mode-switching to control specific robot DoFs [3, 26]. Instead, we adopt *learned latent actions* [5, 7, 8, 27, 28] a framework that uses conditional auto-encoders [29] to learn task-specific latent "action" spaces from demonstrations. These latent spaces match the dimensionality of the low-DoF interface and provide semantically meaningful control. However, existing methods fail to differentiate between tasks with overlapping states, hindering the ability to perform diverse behaviors in a workspace (e.g., manipulating a jug of milk in different ways – pouring, placing in the fridge, etc.). In this work, we use language for *disambiguation*; users naturally speak their intent, conditioning latent action models to produce intuitive control spaces that align with user objectives.

Language-Informed Robotics. A variety of methods have sought to combine language and robotics, spanning approaches that map language to planning primitives [9, 30, 31, 32, 33], perform imitation learning from demonstrations and instructions [11, 34, 35, 36], and pair language instructions with reward functions for reinforcement learning [12, 37, 38]. Other approaches use language in more nuanced ways, such as learning language-conditional reward functions directly [39, 40, 41], or within adaptive frameworks, where language is used to correct or define new behavior [42, 43]. This list is not exhaustive; we present further discussion – including approaches that combine language with other modalities – in the Appendix. However, all these approaches fall within full autonomy: after providing an instruction, human users cede control over to the robot policy, which then takes the actions necessary to perform a task.

While robots trained with these approaches can perform diverse tasks and generalize to new instructions, it is not without cost. Paramount is sample efficiency; imitation learning approaches often require hundreds to thousands of demonstrations for learning to navigate [35, 44], and reinforcement learning approaches can require millions of episodes of experience to learn robust policies [12, 45]. Whereas coarse behaviors are easy to learn, learning to recover from slight deviations from the training data, or to perform precise motions in a sequence, is incredibly difficult. By casting our approach, LILA, within a shared autonomy framework instead, we intelligently offload these parts that are harder for robots – but easier and intuitive for humans – onto the user.

3 Formalizing Language for Assistive Teleoperation

Formalism. We formulate a user's objective, or task, (on a *per-user basis*) as a fully-observable, language-augmented, Markov Decision Process (MDP) $\mathcal{M}$ defined by the tuple $(\mathcal{S}, \mathcal{U}, \mathcal{A}, T, R, \gamma)$

Task Name	Success	Example User Study Input	Mapped Training Data
`Pick Banana`	100%	*yellow in purple*	→ `pick up the yellow banana and place it into the purple basket`
`Pick Fruit Basket`	100%	*bring basket to center of pan*	→ `place the basket onto the tray`
`Pick Cereal`	100%	*go to the left side of the cream bowl, go down, grab the cereal bowl, and place it on the try*	→ `grab the cereal bowl and put it on the tray`
`Pour Bowl`	67%	*pick up the cup of marbles and pour them into the cereal bowl*	→ `pick and pour the cup of white balls into the bowl of cereal`
`Pour Cup`	100%	*pick up the clear cup with marbles in it and pour it in the black mug with the coffee beans in it*	→ `pick up the cup and pour the contents in the mug`

Table 1: Example utterances provided by study participants paired with the retrieved exemplar per §4.1. Success rate refers to the percentage of the time a user study utterance (over all utterances in the study) was grounded to the correct task via our retrieval method. As each participant only attempted 2 tasks, success rate can fluctuate significantly, as is the case with the **`Pour Bowl`** task.

similar to prior work in language-conditional robotics [10, 34, 42]. Let $u \in \mathcal{U}$ denote a user's language utterance provided at the start of each episode, where $\mathcal{U}$ is the full set of language utterances a user could provide to a robot. Let $s \in \mathcal{S} \subseteq \mathbb{R}^n$ be the robot's state, and $a \in \mathcal{A} \subset \mathbb{R}^m$ be the robot's action: taking action a in state s results in a next state s' according to the transition function $T(s, a)$. Given the language utterance u, the user implicitly defines a reward function $R(s, u, a) \in \mathbb{R}$; the human and robot collaboratively maximize this reward subject to discount factor $\gamma \in [0, 1]$.

Problem Statement. This MDP forms the basis of a shared autonomy task wherein a human is equipped with a low-dimensional control interface for the robot. Let $z \in \mathcal{Z} \subset \mathbb{R}^d$ where $d \ll m$ be the human's control input to the robot, such as the $d = 1$ DoF controller in Fig. 2. Previous work on learning latent actions for assistive teleoperation [5, 28] learn a decoder $\text{Dec}(s, z) : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$ that maps user low-dimensional inputs $z \in \mathcal{Z}$ and current state $s \in \mathcal{S}$ to a high-dimensional action $a \in \mathcal{A}$. However, in situations where state-conditioning is not enough to disambiguate a users' intent, too low of a control input dimension d may lead to failure. Recalling the milk jug example, we have multiple different behaviors we could execute if the end-effector were next to the milk. For example, one might want to pick up the jug, shift it to the side, pour it, etc; conditioning only on the state with a 2-DoF action space is not enough to recover all possible behaviors.

Instead, we aim to learn $\text{Dec}(s, u, z) : \mathcal{S} \times \mathcal{U} \times \mathcal{Z} \to \mathcal{A}$ that takes the user's control input *and* utterance u, and predicts the high-DoF action that matches the user's objective. The utterance u acts as additional conditioning information, producing control spaces that depend on both language and state; this circumvents the disambiguation problem above.

4 Language-Informed Latent Actions (LILA)

We are given a dataset of demonstrations, where each demonstration contains an utterance u and a trajectory $\tau = \{s_0, a_0, s_1, a_1 \dots s_T\}$. We split each demonstration into triples of (u, s_i, a_i) and use these to learn a conditional auto-encoder, consisting of a language-conditional encoder $\text{Enc}: \mathcal{S} \times \mathcal{U} \times \mathcal{A} \to \mathcal{Z}$ that maps to a latent z and a decoder $\text{Dec}: \mathcal{S} \times \mathcal{U} \times \mathcal{Z} \to \mathcal{A}$ that attempts to reconstruct the original action a. We minimize the mean-squared error between the predicted and the original action:

$$L_{\text{Enc,Dec}} = \frac{1}{N} \sum_{i=1}^{N} (\text{Dec}(s_i, u_i, \text{Enc}(s_i, u_i, a_i)) - a_i)^2 \tag{1}$$

We next discuss how we integrate language into the architecture of the encoder and decoder.

4.1 Integrating Language within the Latent Actions Architecture

We implement the encoder and decoder as multi-layer feed-forward networks, with the ReLU activation as in prior work [7, 28]. We focus on the decoder here, but the encoder is symmetric. For the decoder, we first concatenate the robot state s and latent action z, then feed the corresponding vector through multiple ReLU layers (usually 2-3), upsampling to produce the high-dimensional robot action. We next discuss how to incorporate language within this simple scaffold.

Pretrained language models such as BERT, T5, and GPT-3 [47, 48, 49] have revolutionized NLP, providing powerful language representations. Inspired by their success when applied to robotics and reinforcement learning tasks [38, 50, 51], we use a distilled RoBERTa-Base model [14], from `Sentence-Transformers` [15] to encode utterances. This model is fine-tuned on a corpus of paraphrases, allowing it to pick up on sentence-level semantics. We generate utterance embeddings by performing mean-pooling over token embeddings for an utterance, as in prior work [38, 50]. We incorporate these embeddings using feature-wise linear modulation (FiLM) layers [46] that fuse language information with other features h by mapping language embeddings to parameters (γ, β) of an affine transformation: $h' = \gamma * h + \beta$ (Fig. 2). Notably, this h is the representation received after feeding the state and latent action (s, z) through the *first* layer of the decoder as described above. Once the language transforms $h \rightarrow h'$, we feed h' to the subsequent layers of the decoder.[2]

Nearest-Neighbor Retrieval at Inference. A major concern for work in language-conditioned robotics is generalizing to novel language inputs. While it may be unreasonable to expect generalization to completely new tasks, for user-facing systems with a clear set of behaviors seen at train time as in our work, there is an expectation that any language-informed system is capable of handling moderate variations of utterances from the training set. To do this, adding linguistically diverse data has been the gold standard [44, 51]; however, a new class of approaches have emerged that sidestep additional data requirements by tapping into the potential of pretrained language models [43, 50]. These approaches frame language interpretation at inference, when interfacing with real users, as a *retrieval* problem: each new user utterance u' is embedded(with the same pretrained model as above, then used to query a nearest neighbors store containing all training exemplars; once the nearest neighbor u_i has been identified, it replaces u' as an input to LILA.

The key benefit of such an approach is the minimal mismatch between train and test language inputs: all "test inputs" are drawn from the training set. This does mean, however, that user utterances that describe new tasks, or are otherwise unachievable also get mapped to language seen at training. While this limits the ability to perform novel tasks, it again highlights the benefits of the shared autonomy paradigm – doubly so, considering the cost of a mistake in an assistive domain like the one we consider in this work: if, while providing control inputs to the robot, a user feels the robot is not acting in alignment with the user's desired objective, they can always stop execution.

5 User Study

We evaluate LILA with a real-world user study on a 7-DoF Franka Emika Panda Arm, on a series of 5 complex manipulation tasks. Each user is provided a 2-DoF joystick for control. We compare against a non-learning, end-effector (EE) control baseline where users "mode switch," controlling the velocity of 2 axes of the end-effector pose at a given time – [(X, Y), (Z, Roll), (Pitch, Yaw)]. Language utterances u are typed into a text console for simplicity; future work could extend this work by using off-the-shelf speech recognition systems. We also compare against a fully autonomous imitation learning (IL) strategy where users solely provide language inputs, and the robot attempts to perform a task without additional input. Ostensibly missing is a *no-language* variant of the latent actions model, in keeping with prior work; however, upon evaluating this model, we found it to be unintuitive and unable to make progress or solve any task, so we omit it from our user study. However, further experiments and analysis can be found in the Appendix.[3]

Environment. Fig. 1 shows our "breakfast buffet" setting, a scaled down version of an assistive feeding domain. We define 5 tasks: 1) **`Place Banana`**: placing the banana in the purple fruit basket,

[2]Implementation can be found in the open-source code repository: `https://github.com/siddk/lila`.

[3]Experiments with the no-language baseline and extra analysis showing the necessity of extra conditioning information can be found here: `https://sites.google.com/view/lila-corl21/home/no-lang-baseline`

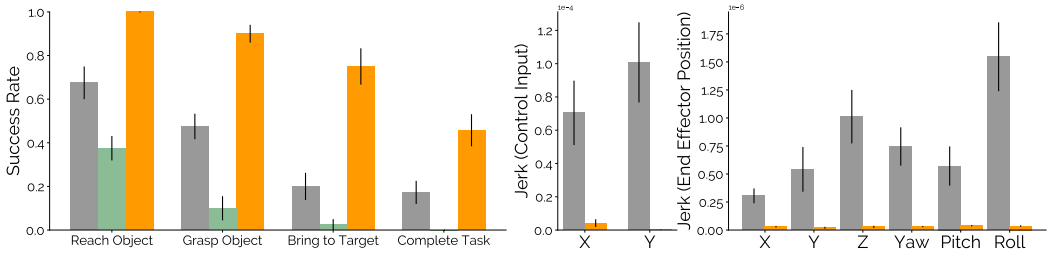

Figure 3: Quantitative Results. We average success rate [**Left**] across all sub-tasks for each control method, and find that LILA is significantly ($p < 0.05$) more performant. However, the steep drop in performance when completing the full task shows the difficulty of fine-grained control. We also calculate jerk as an indicator of controller smoothness, for both user control inputs [**Middle**] and end-effector position [**Right**]. Averaged across tasks and users, we find LILA leads to significantly smoother control for users than end-effector control.

2) **`Place Basket`**: grasping the purple basket by the handles and dropping it on the tray, 3) **`Place Bowl`**: grasping the green cereal bowl by its edge and moving it to the tray, 4) **`Pour Bowl`**: pouring the blue cup of marbles (a proxy for milk) into the cereal bowl positioned on the tray, and 5) **`Pour Cup`**: pouring the blue cup of marbles into the yellow coffee cup. Fig. 1 shows idealized example trajectories for the **`Place Bowl`** (blue) and **`Place Banana`** (orange) tasks.

These tasks vary in difficulty, requiring precise grasping and dexterous manipulation. We evaluate partial success based on how many of the following 4 subtasks users are able to complete: 1) **`Reaching`**: touching the desired object, 2) **`Grasping`**: executing a successful grasp, 3) **`Bring to Target`**: successfully transporting the manipulated object, and 4) **`Task Completion`**.

Demonstration Collection. Both LILA and IL models require learning from (language, demonstration) pairs for all 5 tasks. We collect demonstrations kinesthetically as in prior work [7, 28], recording joint states at a fixed frequency. We initially collected 15 demonstrations per task for each method. However, on testing the IL model, we found it incapable of performing even rudimentary reaching behaviors. To give IL the best chance, we collected twice the number of demonstrations (30 per task; 150 total), requiring an extra 2 hours of labor.

Crowdsourcing Language Annotation. To build a *natural* language interface for human-robot collaboration, we collect language annotations for each task by crowdsourcing utterances. Our goal was to capture the diverse ways users may refer to the objects and actions our tasks entail without any additional information, simulating a real user interacting with our environment for the first time. We recruited 30 workers on Amazon Mechanical Turk, showing only a video of a recorded demonstration, and asked them to provide *"a short instruction that you would want to provide the robot to complete this task independently in the box below."*. However, this procedure resulted in some annotations containing "spam", or extremely out-of-domain text. To address this without introducing our own bias on what constitutes "spam", we filtered the data to identify workers who consistently provided "noisy" annotations, measured by the cosine distance between the sentence embedding (using any pretrained embeddings) of an annotator's provided text and the *average* sentence embedding aggregated over all other annotators for a given video. We used annotations from the 15 least "noisy" annotators under this metric as our ground-truth utterances. Further details, as well as example "spam" annotations that were filtered out, are in the Appendix. Table 1 provides examples of crowdworker utterances from our final dataset (rightmost column).

Participants & Procedure. We conducted our study with a participant pool of 10 university students (5 female/5 male, age range 23.2 ± 1.87). Four subjects had prior experience teleoperation a robot arm.[4] We conduct a within-subjects study, where each participant completed 2 tasks, chosen randomly, with each of the three methods. Users were given 2 trials to complete each tasks, and an

[4]Due to the COVID-19 pandemic and university restrictions, only those with pre-authorized access could participate. See the COVID-19 considerations document in the Appendix for more details.

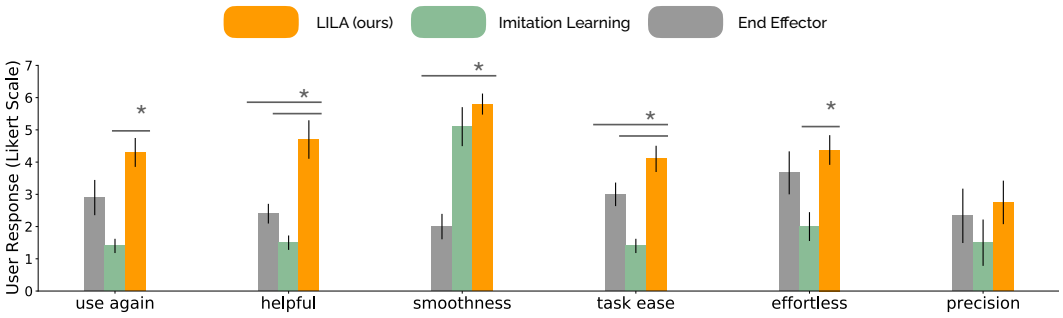

Figure 4: Qualitative Results. Using a 7-point Likert scale, we ask users to evaluate each of the 3 control methods for different properties. With high significance ($p < 0.05$), we find that LILA outperforms both imitation learning and end-effector control baselines on several metrics, including degree of helpfulness provided and ease in completing tasks.

allotted 3 minutes per control strategy to practice. Users were also given a sheet describing controller inputs and details for each control method, which we include in the Appendix. For imitation learning and LILA controllers, which require language inputs, participants provided a natural language utterance which which a proctor entered into the model – participants were allowed to verify the proctor entered their input accurately. This user-provided language utterance is used as the query in the nearest-neighbor retrieval described in §4.1; the retrieval set consists of all training utterances collected via the crowdsourcing procedure above. In addition to tracking quantitative success rates (normalized, based on progress relative to each of the 4 defined subtasks), time taken per task, and controller logs, we ask users to fill out a qualitative survey evaluating each method at the end of each study. We present both quantitative and qualitative results below.

Quantitative Results. Fig. 3 summarizes our objective results. We evaluate both full- and partial-task success rates for each task across all control methods, in addition to computing smoothness metrics directly on the logged user inputs and robot actions. Smoothness is a measure for intuitiveness when measured on user 2-DoF joystick inputs, ease of use when measured on the robot's end-effector pose, and implicit safety: a trajectory with high discontinuity in acceleration can lead to rapid, unpredictable changes in the environment. Smoothness is negatively correlated with *jerk*, the time-derivative of acceleration. We compute jerk by taking the second-order derivative of velocity, and report average jerk across fixed windows.

Our results show that LILA significantly ($p < 0.05$) outperforms both methods across all sub-tasks, and is also smoother to use both in control input space (2-DoF input) and end-effector space (6-DoF). However, the relative drop in performance of LILA for the final sub-task "Complete Task" shows the room for improvement in fine-grained control, such as pouring motions. Additionally of note is the poor performance of imitation learning. To explore this fully, we perform an ablation, and show that sample inefficiency is a likely cause – especially since we are in a low-data regime. These results can be found in the Appendix.[5]

Qualitative Results. Fig. 4 summarizes our subjective results. We administered a 7-point Likert scale survey after users finished performing tasks with each method; this survey included questions around the perceived helpfulness of the model in completing the tasks (*helpful*) and whether the participant would use the control method again (*use again*). The results show that LILA outperforms both imitation learning and end-effector control across most qualitative metrics, with significant results ($p < 0.05$) marked with an $*$. We additionally visualize samples of the observed end-effector trajectories by individual users collected during our study for 3 of our 5 tasks in Fig. 5. Across all tasks, LILA results in smoother end-effector trajectories than end-effector control, while imitation learning comes close to the target object but is unable to complete the entire trajectory for the task.

[5] Additional experiment videos, and a results of the imitation learning ablation can be found here: https://sites.google.com/view/lila-corl21/home/il-ablation

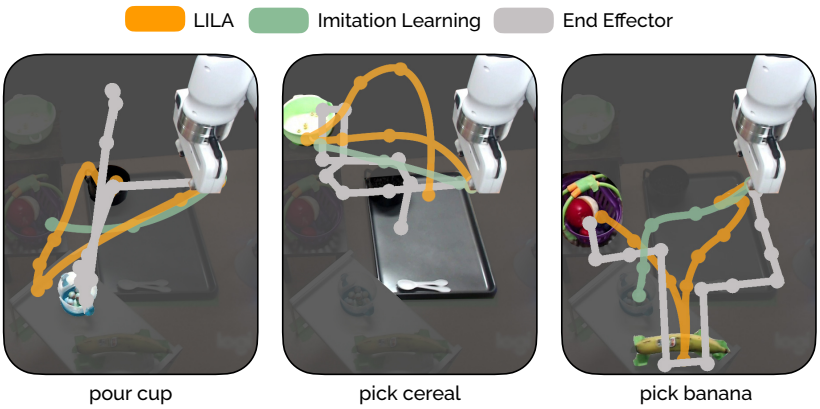

Figure 5: Trajectories from one user comparing three different control methods 3 out of our 5 tasks – **`Pour Cup`**, **`Pick Cereal`**, and **`Pick Banana`**. LILA provides smooth actions that immediately approach the target object for a task, while end-effector control shows more rigid motions that result in users diverging from their intended paths. While imitation learning also enables smooth motions, it often fails shortly after reaching objects, hence the shortened trajectories.

6 Discussion

Summary. We present Language-Informed Latent Actions (LILA) a framework that marks the first step in combining the expressiveness and naturalness of language for *specifying* and *executing* on a human user's objective within the context of assistive teleoperation. Our user study results show that when compared to fully autonomous, imitation learning approaches, LILA is more sample-efficient and performant, training on half the number of task demonstrations, but obtaining significantly higher success rates. Compared with no-learning end-effector control methods, we again show LILA's effectiveness at obtaining high success rates, but also demonstrate its ability to produce intuitive low-dimensional control spaces from language input. Qualitatively, we find that users prefer LILA to alternative methods across the board, opening the door for additional work on language & latent actions.

Limitations and Future Work. Currently, LILA uses language as a mechanism for *task disambiguation* – in the current results, there is no mechanism for generalizing to completely unseen tasks or language specifications. We believe that the ability to disambiguate with language, and the integration of language within the latent actions framework is a strong research contribution, and hope that future work looks to dynamic states – perhaps by leveraging visual latent actions [28] – and to adapting to new utterances and tasks dynamically [43, 52]. Furthermore, while users found LILA intuitive and natural, they found themselves wanting to further modulate the robot's behavior with language instructions *during the course of execution*. Many users, upon seeing the robot make slight deviations from a desired path would instinctively provide spoken corrections – *"a little to the right"*, *"no, grasp it by the handle!"* – indicating a desire for *multi-resolution* language control.

Shared Autonomy and LILA. LILA fits within the shared autonomy paradigm, where the role of language has been underexplored. With LILA and shared autonomy approaches in general, humans retain *agency* – they are responsible for robot motion, and if the robot moves in a way that is not safe, or does not align with their objectives, they stop providing control and possibly reset, give a new instruction, or drop into a more complex control mode. Behavior is *interpretable* – the latent actions model, *critically informed by language*, produces intuitive control spaces that humans can quickly grasp. Finally, language is *natural* – users specify their objectives as they would if speaking to another person, and the robot uses that language to shape their control space. These properties – preserving agency, maintaining interpretability, and leveraging the expressive and natural features of language for specifying objectives – are critical for widespread human-robot collaboration, and we hope this work presents a concrete step towards achieving that goal.

Acknowledgments

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. This project was also supported by NSF Awards 2006388 and 2132847. Siddharth Karamcheti is grateful to be supported by the Open Philanthropy Project AI Fellowship. Megha Srivastava is supported by the NSF Graduate Research Fellowship Program under Grant No. DGE-1656518.

We would additionally like to thank the participants of our user study. We further extend a special thank you to Madeline Liao and Raj Palleti for helping with the visual intuition for our imitation learning analysis. Finally, we are grateful to our anonymous reviewers, Suneel Belkhale, Ajay Mandelkar, Kaylee Burns, and Ranjay Krishna for their feedback on earlier versions of this work.

References

[1] D. M. Taylor. *Americans With Disabilities: 2014*. US Census Bureau, 2018.

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

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

[4] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming. *International Journal of Robotics Research (IJRR)*, 37:717–742, 2018.

[5] H. J. Jeon, D. P. Losey, and D. Sadigh. Shared autonomy with learned latent actions. In *Robotics: Science and Systems (RSS)*, 2020.

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

[7] D. P. Losey, K. Srinivasan, A. Mandlekar, A. Garg, and D. Sadigh. Controlling assistive robots with learned latent actions. In *International Conference on Robotics and Automation (ICRA)*, pages 378–384, 2020.

[8] D. P. Losey, H. J. Jeon, M. Li, K. P. Srinivasan, A. Mandlekar, A. Garg, J. Bohg, and D. Sadigh. Learning latent actions to control assistive robots. *Autonomous Robots (AURO)*, pages 1–33, 2021.

[9] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2011.

[10] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. S. Wong, and S. Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. In *Robotics: Science and Systems (RSS)*, 2017.

[11] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Computer Vision and Pattern Recognition (CVPR)*, pages 6077–6086, 2018.

[12] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. Czarnecki, M. Jaderberg, D. Teplyashin, M. Wainwright, C. Apps, D. Hassabis, and P. Blunsom. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

[13] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor. Language-conditioned imitation learning for robot manipulation tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[15] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[16] D. Gopinath, S. Jain, and B. D. Argall. Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters (RA-L)*, 2:247–254, 2016.

[17] M. T. Ciocarlie and P. K. Allen. Hand posture subspaces for dexterous robotic grasping. *International Journal of Robotics Research (IJRR)*, 28:851–867, 2009.

[18] S. Jain and B. Argall. Probabilistic human intent recognition for shared autonomy in assistive robotics. *ACM Transactions on Human-Robot Interaction (THRI)*, 9:1–23, 2019.

[19] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[20] C. Matuszek. Grounded language learning: Where robotics and NLP meet. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[21] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):25–55, 2020.

[22] B. A. Newman, R. M. Aronson, S. Srinivasa, K. Kitani, and H. Admoni. Harmonic: A multimodal dataset of assistive human-robot collaboration. *arXiv preprint arXiv:1807.11154*, 2018.

[23] S. Reddy, A. D. Dragan, and S. Levine. Shared autonomy via deep reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2018.

[24] C. B. Schaff and M. R. Walter. Residual policy learning for shared autonomy. In *Robotics: Science and Systems (RSS)*, 2020.

[25] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3:43–53, 1987.

[26] W. J. Wilson, C. W. Hulls, and G. S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics (T-RO)*, 12:684–696, 1996.

[27] M. Li, D. P. Losey, J. Bohg, and D. Sadigh. Learning user-preferred mappings for intuitive robot control. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[28] S. Karamcheti, A. Zhai, D. P. Losey, and D. Sadigh. Learning visually guided latent actions for assistive teleoperation. In *Learning for Dynamics & Control Conference (L4DC)*, 2021.

[29] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[30] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *Human-Robot Interaction*, pages 259–266, 2010.

[31] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *International Symposium on Experimental Robotics (ISER)*, 2012.

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

[33] S. Karamcheti, E. C. Williams, D. Arumugam, M. Rhee, N. Gopalan, L. L. S. Wong, and S. Tellex. A tale of two draggns: A hybrid approach for interpreting action-oriented and goal-oriented instructions. In *First Workshop on Language Grounding for Robotics @ ACL*, 2017.

[34] V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Robotics: Science and Systems (RSS)*, 2018.

[35] X. E. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[37] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[38] F. Hill, S. Mokra, N. Wong, and T. Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

[39] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang. Grounding English commands to reward functions. In *Robotics: Science and Systems (RSS)*, 2015.

[40] J. Fu, A. Korattikara, S. Levine, and S. Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations (ICLR)*, 2019.

[41] D. Bahdanau, F. Hill, J. Leike, E. Hughes, S. A. Hosseini, P. Kohli, and E. Grefenstette. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations (ICLR)*, 2019.

[42] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. DeNero, P. Abbeel, and S. Levine. Guiding policies with language via meta-learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[43] S. Karamcheti, D. Sadigh, and P. Liang. Learning adaptive language interfaces through decomposition. In *EMNLP Workshop for Interactive and Executable Semantic Parsing (IntExSemPar)*, 2020.

[44] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[45] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[46] E. Perez, F. Strub, H. D. Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[47] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Association for Computational Linguistics (ACL)*, pages 4171–4186, 2019.

[48] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[49] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[50] A. Marzoev, S. Madden, M. Kaashoek, M. J. Cafarella, and J. Andreas. Unnatural language processing: Bridging the gap between synthetic and natural language data. *arXiv preprint arXiv:2004.13645*, 2020.

[51] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[52] S. I. Wang, S. Ginn, P. Liang, and C. D. Manning. Naturalizing a programming language via interactive learning. In *Association for Computational Linguistics (ACL)*, 2017.

[53] F. Duvallet, T. Kollar, and A. Stentz. Imitation learning for natural language direction following through unknown environments. In *International Conference on Robotics and Automation (ICRA)*, pages 1047–1053, 2013.

[54] S. Ross, G. Gordon, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*, 2011.

[55] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[56] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014.

[57] T. Kollar, J. Krishnamurthy, and G. P. Strimel. Toward interactive grounded language acqusition. In *Robotics: Science and Systems (RSS)*, 2013.

[58] C. Kennington, S. Kousidis, and D. Schlangen. Interpreting situated dialogue utterances: an update model that uses speech, gaze, and gesture information. In *SIGDIAL Conference*, 2013.

[59] D. Whitney, M. Eldon, J. G. Oberlin, and S. Tellex. Interpreting multimodal referring expressions in real time. In *International Conference on Robotics and Automation (ICRA)*, pages 3331–3338, 2016.

[60] N. Reimers and I. Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[61] M. Laskey, J. N. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning (CORL)*, 2017.

We provide further details, experiments, and descriptions of the attached media, to reinforce the results and conclusions from the main body of our paper. For a more fluid viewing experience please look through our project website, where videos (and corresponding descriptions) are side-by-side: https://sites.google.com/view/lila-corl21. The top-level page contains videos from our actual user study, showing the various models in practice, while the sub-pages contain additional experiments exploring the poor performance of the imitation learning baseline in our work, as well as justifications for the omission of the no-language latent actions baseline.

We additionally provide a full version of our code repository at the following url: https://github.com/siddk/lila, with a detailed README spanning the entire LILA pipeline from demonstration recording to training models, to deploying them on a robot.

A COVID-19 Impact Statement

Due to the ongoing COVID-19 Pandemic, the ability to run larger-scale user studies, and even have reliable lab access for preparing experiments, was limited. We are in a University setting, and as such, were susceptible to University restrictions.

All members of the authorship team had to go through a COVID-19 safety training, frequent testing, as well as an official approval process to be granted permission to work in the robot lab. For User Study participants, we were mostly limited to those with pre-existing access to the Engineering Building (spanning both robotics and non-robotics students), as well as a limited number of other University students granted approval to access other nearby buildings. This limited our ability to launch a larger scale user study, with more than 10 participants from a more diverse population.

That being said, we strongly believe that our existing participant statistics – 10 students (5 female/5 male, age range 23.2 ± 1.87 – reflect a broader user pool. Coupled with the statistical significance of the results we have already collected, we feel that the User Study results remain compelling, and our conclusions hold. That being said, we would like to run additional studies once COVID-19 restrictions relax in our area.

B Related Work Discussion

The main body of our paper contains a cursory discussion of different approaches for language-informed robotics, spanning a multitude of full-autonomy solutions. While this discussion helps provide contrast for our *shared-autonomy & language* based approach, LILA, it does not do the existing work in the field justice, nor does it discuss the data, implementation, and feature-engineering considerations that are coupled with these different approaches.

First, we discuss work that leverages structured logical forms as an intermediate representation for mapping language. The benefits of these forms are that they induce *logical forms* – functional, often programmatic – representations of meaning, that can then be executed on a robot, either by directly formulating a plan (requiring a model of the environment), or learning a lightweight policy from data that can fulfill these logical forms. A perceived benefit of these types of approaches is their sample efficiency – by leveraging a highly structured logical form and possibly hand-engineered features, one can learn the language to logical form mapping with 10s of examples. An example of this work is MacGlashan et al. [39] that learns reward functions given hand-engineered linguistic features; however, these reward functions are fed to a planner (requiring full knowledge of world dynamics – a huge assumption) to generate robot behavior. Follow-up work by Arumugam et al. [10] relax the hand-engineered language feature assumption by using more recent neural approaches, but still hinge on using planners. While these approaches are sample-efficient, many of the assumptions around planning and full dynamics are strong, and limit the potential of scaling this work (and for manipulation, are not necessarily straightforward!).

Other work that relaxes the planning/dynamics assumption is Duvallet et al. [53]; this work uses hand-engineered features on top of a popular logical form – Spatial Description Clauses (SDCs) – to learn policies for robot navigation. While seemingly as sample efficient as the prior approaches without the downsides, there is still a high cost for *policy learning*. Though the approach only required 10s of examples to learn to map language to the appropriate SDC, learning an effective policy (note this is just discrete node navigation – not continuous manipulation) required running DAgger [54] for 25 iterations on top of their existing data, collecting an order of magnitude more

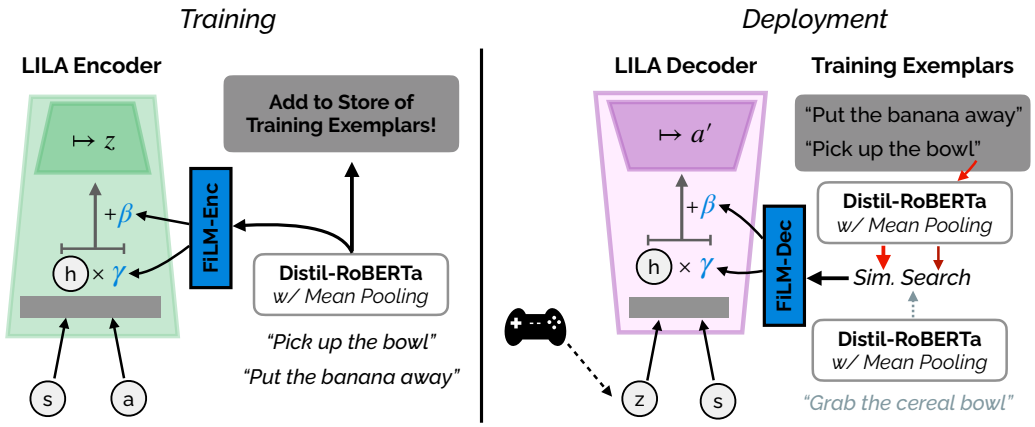

Figure 6: An overview of our language-informed latent actions (LILA) models with the FiLM modules highlighted. For additional clarity, we provide partial views of both the Encoder (during training) and Decoder (during deployment).

demonstration data (or demonstration edits/corrections as in DAgger) than originally given – 100s - 1000s of demonstrations.

On the other end of the spectrum are more recent approaches in end-to-end robot learning for more complicated tasks spanning navigation [44, 51] and manipulation [55]. This work is done in simulation, where one has the ability to do virtually infinite policy rollouts given a language instruction paired with a reward function, to learn robust policies via reinforcement learning. Other work in this paradigm that uses imitation learning, but with real-world deployments include works for quadcopter flight [34] and some limited manipulation [13]. These works still require 1000s of demonstrations, but use smart tricks for data augmentation and synthetic generation to learn robust policies.

LILA hopes to fill a void between these two classes of approaches; retaining the sample-efficiency of the earlier approaches, without the need for hand-engineered features, strong assumptions about known dynamics, or limited generalization potential. The experiments in the main body show that LILA is *extremely* sample efficient; however, the scope of this work is mostly using language as a means for *disambiguation*. It is our ardent hope that future work in language & latent actions (and shared autonomy more generally) turns to more dynamic settings, richer language, and hard forms of generalization; this work is just the first step.

Finally, we want to help fill out the story of language-informed robotics with work that does not necessarily involve *execution* (learning a policy or control space for robots to follow language instructions), but that can help language-based methods generalize better, from less data, or that leverage other modalities to help with specification. For example, Matuszek et al. [56] learn to map *unscripted* interactions consisting of language and gestures to object localizations from relatively few interactions; such methods are crucial for scaling up LILA to multiple objects, referring expressions, and compositional language instructions.

Other work looks at other modalities like speech and gesture to learn logical forms that allow for efficient generalization [57]. Other work combines several modalities on top of language like gaze, gesture, and intonation to further help lift representations of human intent from language [58, 59]. All this work, though not directly related to LILA in that they do not help learn meaningful control spaces, do present possible avenues through which we might scale this approach to new contexts, language instructions, and behaviors.

C Model Architectures & Training

Following from the main body of the paper, we provide additional details on the model architectures, and additional data processing/augmentation we use in this work. We start with a more thorough de-

scription of the feature-wise linear modulation (FiLM [46]) mechanism we use to integrate language into the latent actions pipeline.

FiLM Architecture for LILA. Prior work in latent actions [7, 28] implement the latent action models as simple feed-forward multi-layer perceptrons (MLPs) with `tanh` activations as the non-linearity between layers. The Encoder and Decoder are symmetrical, with the Encoder encoding a combination of (s, a) pairs down to the latent space z (via an intermediate layer or two), and the decoder decoding from (s, z) back up to the 7-DoF action a. The intermediate layers in both the Encoder and Decoder have the same dimensionality of 30 – we refer the reader to consult the attached code for more detail.

With LILA, the two differences are that 1) we use the `GELU` activation instead of the `tanh` due to its better stability, and 2) we incorporate language into this existing pipeline. Recall that we use a version of the pretrained Distil-RoBERTA language model [15, 60] to generate embeddings of each user utterance. These embeddings have dimensionality of 768, which far exceeds the dimensionality of the 7-DoF (s, a) of the typical pipeline. Initial experiments attempting to naively concatenate the language embedding with the 7-DoF states and actions (or states and z values, in the case of the decoder) were not fruitful, as we were unable to learn (loss failed to decrease when training).

Instead, we turned to FiLM. The core principle with FiLM is that it's a fusion mechanism that does not increase the intrinsic parameter count of the core neural network (e.g., the original Latent Action MLP described above). FiLM has found great success in many multi-modal tasks for this reason, integrating with pre-existing pipelines in image classification to enable visual-question answering [46], as well as integrating into existing pipelines for instruction following in reinforcement learning [45, 41]. FiLM works as follows: given an intermediate representation from the Encoder or Decoder h with dimensionality d (say after the first layer of the corresponding MLPs), and a language embedding e, FiLM works in the following fashion:

$$\text{FiLM-Gen}_\theta(e) = \gamma_e, \beta_e$$
$$h' = \gamma_e \odot h + \beta_e$$

where h' is the representation fed to later layers in the Encoder/Decoder MLP, and $\odot$ denotes the Hadamard product (component-wise multiplication). Simply put, Film-Gen$_\theta$ is a module that learns to *shape* the representations learned by the core Latent Actions MLP, injecting language information through this affine transformation defined by γ_e, β_e. We implement Film-Gen$_\theta$ as a separate two-layer MLP that also uses the `GELU` activation. Fig. 6 breaks this down visually, showing how the FiLM modules are added for both the Encoder and Decoder.

Imitation Learning Architecture. For Imitation Learning, we do not have this Encoder-Decoder structure, with two separate FiLM modules. Instead, Imitation Learning is implemented as a single MLP (of same parameter count/number of layers as the Encoder + Decoder in LILA) that conditions on the state s; we add a single FiLM module after the first-layer of this MLP.

Data Augmentation. There are two key aspects to our data augmentation procedure: 1) enforcing latent action consistency, and 2) adding robustness to noise.

A key desire in latent action models is consistency in nearby states – executing the same latent action z in nearby states should be roughly similar. More formally, $d_T(\mathcal{T}(s_1, \phi(z, l, s_1)), d_T(\mathcal{T}(s_2, \phi(z, l, s_2)))) < \epsilon$ for $\|s_1 - s_2\| < \delta$, for some $\epsilon, \delta > 0$, where d_T is some distance metric (e.g., Euclidean distance between states). We enforce this with a sliding window approach; given a sequence of states within a fixed window size, we train the decoder to predict the same actions given the (z, l, s) triples for all states within the window.

The second, and most important augmentation we do is adding robustness to noise. Though we collect a dataset of (s, l, a) triples, we use a unique property of our control space to obtain better robustness: because our states are the actual joint states (let's call this q), and actions a are just joint velocities ($\dot{q}$), we can "re-compute" actions between two sequential states (s_1, s_2) by just taking the finite difference $a' = s_2 - s_1$. This allows us to do the following: add noise to each initial state s_i subject to $\epsilon = \mathcal{N}(0, \sigma)$ (we use $\sigma = 0.01$), then compute the "corresponding action" by taking $a' = s_{i+1} - (s_i + \epsilon)$. This can be viewed as a simulated version of the DART paradigm [61] for noise-robust imitation learning.

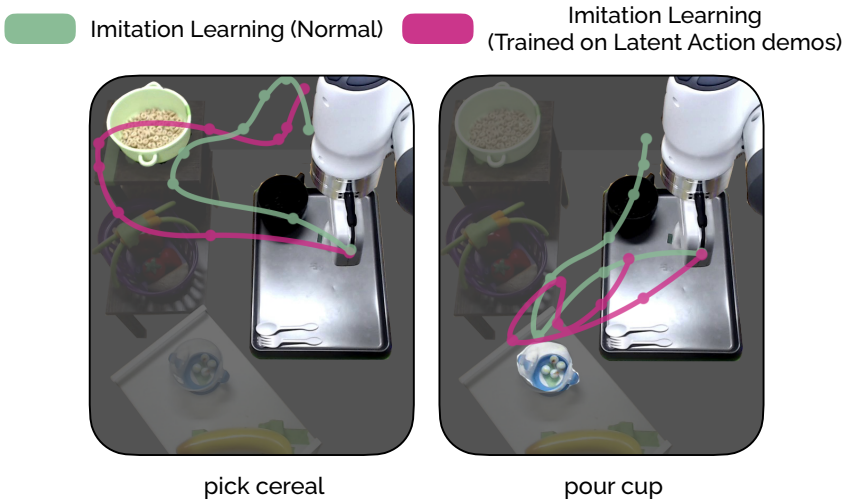

Figure 7: Visualized Trajectories for an Imitation Learning model trained on the "pure" data (that is reported in the paper), and an Imitation Learning model trained on the "LILA-style" demonstrations with the "sweeping" motions. On the left are trajectories for the instruction *"grab the cereal bowl"* and on the right, *"pour the blue cup into the black coffee mug"* – these two examples are from our training language set. We observe that Imitation Learning trained on the "LILA-style" demonstrations performs slightly worse by not following the complete ideal task trajectory, although neither approach is able to successfully complete the task.

We train LILA models with both the above augmentations. While the former augmentation mode is not directly applicable to Imitation Learning approaches, the second noise augmentation mode is – indeed, we find we have to triple the amount of such augmentations, in order to get even slightly meaningful behavior from Imitation Learning models.

D Demonstration Collection

Both Latent Action models and Imitation Learning models require a dataset of language paired with corresponding demonstrations to perform learning. As mentioned in the main body of the paper, we collect these demonstrations *kinesthetically*, manually moving the robot arm to complete specific tasks, recording the joint states and actions along the way. Critically, for imitation learning, these demonstrations consist solely of what we call the "forward", or "pure" demonstration of a task: starting at the home position, perform each of the individual subtasks smoothly and continuously, until the task has been satisfied. For a task like "put the banana in the fruit basket" this corresponds to 1) smoothly reaching for the banana and grasping it, 2) lifting the banana up and moving over to the basket, and 3) inserting the banana into the basket and releasing the gripper.

However, when collecting demonstrations for latent action models, we find that we can learn *better, more reliable* models by changing up the demonstration process a little, incorporating discrete segments of the demonstration where we *back off and then repeat* a motion. Concretely, for a task like "put the banana in the fruit basket" this corresponds to 1) smoothly reaching for the banana, pulling back to the home position, *and then* reaching for the banana again and grasping it, 2) lifting the banana up, lowering it, then taking it to the basket, and 3) finally dropping the banana in the basket. These "sweeping" back-and-forth motions intuitively help the latent actions model induce control spaces that give users *reversible* control – the ability to move the robot back, rather than just press forward; this is critical to usability and recoverability.

On the Fairness of Comparing Imitation Learning and LILA. Because LILA and Imitation Learning are trained with different demonstrations, there is a plausible fear that our comparison between LILA and Imitation Learning is unfair – specifically, because the latent actions demonstration incorporates extra motion, LILA technically may be seeing more (s, a) pairs per demonstration compared to Imitation Learning, and can therefore learn better.

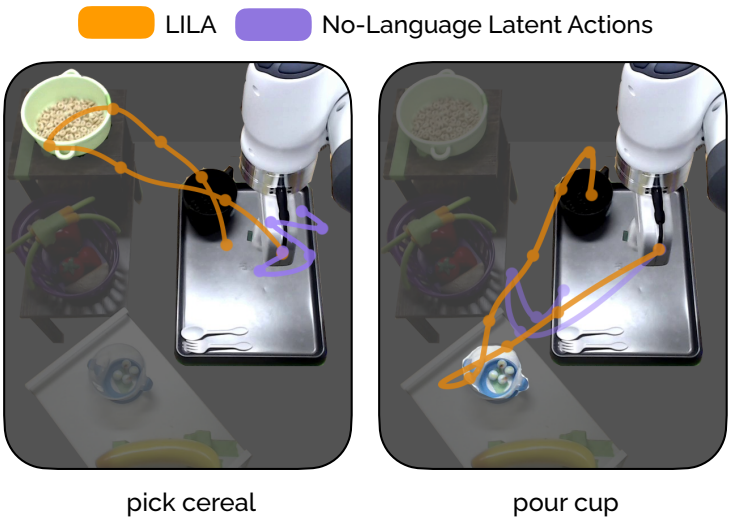

Figure 8: Visualized Trajectories for a No-Language Latent Actions model vs. LILA (with language) trained on the same set of demonstrations, operated by an expert user. With the No-Language Latent Actions model, the user tries their best to complete the task with the provided controls. On the left are trajectories for the instruction *"grab the cereal bowl"* and on the right, *"pour the blue cup into the black coffee mug"* – these two examples are from our training language set. We observe that the No-Language Latent Actions is unhelpful for completing the task, and unable to even completely reach the target object for either task, demonstrating the importance of incorporating language to help condition the learned latent actions.

We mitigate this in two ways; first, as a side effect of doubling the number of collected demonstrations for Imitation Learning – recall that, in the main paper, to get IL to show semantically meaningful behavior, we needed to collect 30 demonstrations per task instead of the 15 per task LILA was given – we found that Imitation Learning sees 40% *more* (s, a) pairs than LILA (even more if you account for the fact that we tripled the data augmentation for Imitation Learning!). Second, Appendix E presents a more concrete experiment where we train the Imitation Learning model *on the LILA demonstrations*, and show that the resulting model performs worse than when trained on the standard IL dataset.

E Additional Experiments: Imitation Learning Baseline

A critical question from our user study has to do with the poor performance of the imitation learning baseline relative to both LILA and End-Effector control. This section explores additional ablation experiments as well as a technical argument for why imitation learning performs poorly – namely due to sample inefficiency, exacerbated by our real-robot setting.

We also address the point raised in Appendix D – how Imitation Learning performs when trained with the "latent actions" style demonstrations ("sweeping" motions) rather than the "pure," straight-through demonstrations.

Ablation Experiments. The following URL contains several sets of ablation experiments, with corresponding text annotations: `https://sites.google.com/view/lila-corl21/home/il-ablation`. Many of these experiments show qualitative behavior, and are best viewed via the linked URL; however, we summarize the main findings here.

First, we put LILA and Imitation Learning on an equal footing, picking 3 of the 5 original tasks we used in the user study – namely, **`Pick Cereal`**, **`Pick Fruit Basket`**, **`Pour Cup`**. We collect 10 demonstrations for each task, and trained a base model with the noise-based data augmentation (add noise once for each state), for both LILA and Imitation Learning. We show that LILA is able to fully succeed at all three of these tasks with *only 10 demonstrations*, whereas Imitation Learning completely fails. However, we note that even at 10 demonstrations, Imitation Learning is starting

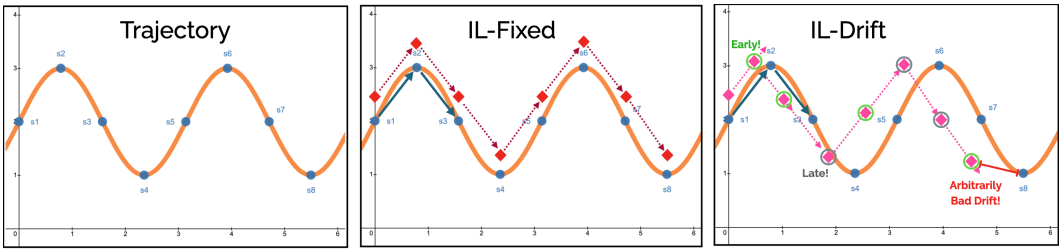

Figure 9: We consider a sinusoidal trajectory of a simplified end-effector through 2D space, where a robot's motion is continuous. If we can sample states when collecting this "demonstration" at a fixed interval, then the state-error of an imitation learning agent trained with behavior cloning grows minimally over time. However, the fixed interval assumption may not be true when collecting data on a real robot, where any noise can lead to compounding errors resulting in arbitrarily bad drift. This example supports our observation of poor IL performance on the 7-DoF Franka Emika Panda robot in our user study.

to show semantically meaningful behavior – for the **`Pick Cereal`** the end-effector clearly moves toward the cereal bowl (though not close enough to grasp), and then down to the tray (though not close enough to execute a successful drop).

We then experiment with the impact of data augmentation, first looking at 3x the amount of augmentation (noising each state 3 times in the dataset following the procedure detailed above), and then 5x. We show that Imitation Learning performance is slightly better than the reference with 3x data augmentation, but that 5x doesn't additionally help.

We then vary the number of demonstrations from 10 to 20, then 30 demonstrations per task, with 3x data augmentation. Critically, we show that Imitation Learning *improves as we add more demonstrations, even though it still isn't able to solve the tasks.* As an interesting data point, at 20 demonstrations, the Imitation Learning agent is able to execute a successful grasp of the cup (see the video on the webpage), but cannot finish the task.

Unfortunately, after collecting 90 demonstrations (30 for each task), we felt we'd need close to 50-100 demonstrations per task to actually get imitation learning to solve these tasks – almost *two orders of magnitude* more data than LILA needed. This would have been prohibitive to collect so we stopped, and instead decided to analyze the possible cause of this extreme sample inefficiency. Again, videos and annotations depicting these ablations visually, in an easy-to-follow format can be found here: https://sites.google.com/view/lila-corl21/home/il-ablation.

Why is Imitation Learning Sample Inefficient? The above experiments point to an extreme sample inefficiency in Imitation Learning. Imitation Learning is generally subject to problems of cascading errors and the general noise problems of real robotics (imprecise resets, inherent noise in robot joints). However, we will now present an argument that illustrates why Imitation Learning and LILA may be even worse than expected in our real-robot setting, due to specific implementation details in our publish/subscribe based methodology. Notably, there is imperfect communication between the top-level Python process (housing the learned models) and the low-level C++ robot controller that exacerbates the cascading errors imitation learning has to deal with. A graphical walkthrough of this argument can be found at the bottom of the webpage for this section here: https://sites.google.com/view/lila-corl21/home/il-ablation.

Consider the sinusoidal trajectory of a simplified end-effector through 2D space shown in Figure 9. The robot's motion is continuous, but we can sample states when collecting this "demonstration" at fixed intervals. Notably, this is an assumption present implicitly in most simulators (fixed frame rate, or fixed control iterations/sec) – this is also usually true when operating with discrete action spaces (move forward/right/left). However, this fixed interval assumption may not be true when collecting data on a real robot, depending on the implementation. More on this in a bit, but for now, let's assume our demonstration data consists of evenly spaced (state, action) pairs with this fixed interval between samples.

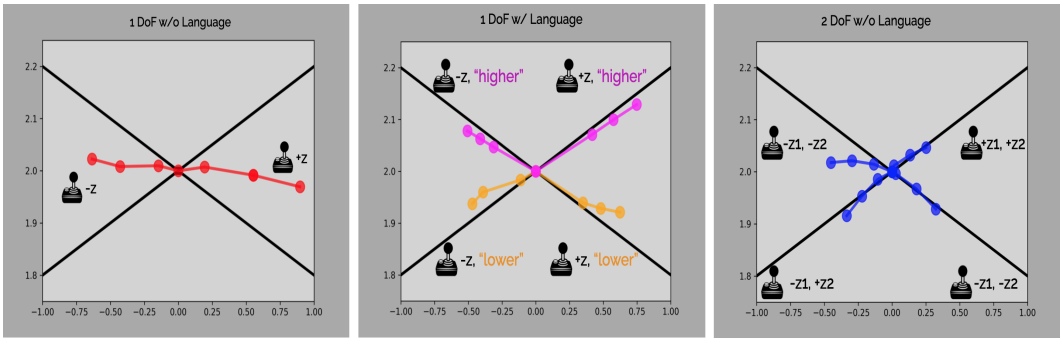

Figure 10: We train 3 different latent action models for the cross disambiguation task from Figure 2 of the main paper, which showed a simple "cross" example, where starting from the mid-point, the goal is to be able to navigate towards all 4 possible directions. As the standard LA framework only takes in the controller inputs, it is impossible for a user to succeed with only 1 degree-of-freedom (DoF). To solve this task, we need an additional axis to condition on (at least 2-DoF). Our results show that, as expected, without any additional input such as language, a 1-DoF controller (left) is incapable of task disambiguation, with a clear 0% task success rate for our simple cross setting.

Consider training an imitation learning agent via behavioral cloning, where the policy is parameterized as a neural network. It's not clear what a NN will do given a state outside it's training distribution (could be arbitrarily bad) , but to simplify, *let's assume given a new state, this policy will predict an action based on retrieving the "nearest-neighbor" from it's training demo.* Assume there's some noise in the reset (this is representative – there's always *some* noise in the initial joint states - this is represented in simulators like Mujoco and PyBullet). If you roll out the imitation learning policy, you get behavior like that shown in the middle of Figure 9 – *critically, assuming the same constant sampling rate, the state-error grows minimally over time.* However, for continuous state and continuous action robotics grounded in a real-world robot, this assumption does not exist, which leads to the following point.

With our implementation on a 7-DoF Franka Emika Panda robot, we noticed that despite our best efforts, *we are not able to ensure states/actions are dispatched at a constant sampling rate*. The result (based on our straw man nearest-neighbors NN argument from above – though note the real world behavior, especially in higher dimensions will be much much worse!) is shown in Figure 9 on the right. With slippage in the read/publish times of states and actions, we can read states too early or too late, execute a "bad" action, and *cascade to arbitrarily bad final states over the course of execution* (even completely breaking in the middle of execution).

Imitation Learning with Latent Action Demonstrations. Finally, as raised in Appendix D, there is a question about the fairness of training on the "LILA-style" demonstrations with the "sweeping" motion vs. the more pure, forward-only imitation learning demonstrations. To address this, we train two Imitation Learning models (identical architecture, augmentation), with one model trained on the original "pure" data (the model from the main paper, used in the user studies) and a model trained on the "LILA-style" demonstrations with the sweeping behavior.

Fig. 7 shows visualizations of the trajectories for the models rolled out on two instructions from the training set. We see that the Imitation Learning model trained on the LILA demonstrations is worse than the "standard" Imitation Learning model. Whereas the pure model is able to at least closely reach the target objects, the other model attempts to reach for an object, but wastes time moving around the object rather than focusing on the task trajectory – intuitively this makes sense, as the "sweeping" motions present in the LILA data are confusing for imitation learning; given two opposing actions in the same state, what should it do?

These experiments, coupled with the experiments in the main paper provide ample evidence that the comparison between LILA and Imitation Learning is not only fair, but highlights the sample efficiency of LILA as well.

Algorithm 1 Filtering Language Utterances

for task $= 1, 2, \ldots, T$ **do**
  Initialize list E_{task}.
  for user $= 1, 2, \ldots, N$ **do**
    Append embedding of utterance `embed(user, task)` to E_{task}
  end for
end for
for user $= 1, 2, \ldots, N$ **do**
  Initialize list D_{user}
  for task $= 1, 2, \ldots, T$ **do**
    Append `cosine distance` d between `embed(user, task)` and $\text{avg}(E_{\text{task}})$ to D_{user}
  end for
end for
Filter out all utterances from users with K highest $\text{avg}(D_{\text{user}})$

F Additional Experiments: No-Language Ablation

Another possible question is how latent actions performs *without language* – in other words, is LILA necessary, or are prior latent actions models expressive enough to solve the tasks?

We answer this in two ways. First, we train a latent actions model, completely ablating the language encoding pipeline (keeping architecture the same, but removing the FiLM components described in Appendix C). The corresponding latent action decoder only takes in the latent action z and state s as an input to predict high-DoF actions a. Fig. 8 shows visualizations of trajectories for this No-Language model as well as LILA operated by an expert user, trying to accomplish two specific tasks. While LILA performs as expected, the No-Language model is unable to make progress at all. As soon as control begins and the user moves the robot into a state close to any object, the control space loses meaning, and the user is unable to recover, instead generating random behavior. Again this makes sense; without language to condition on, the No-Language model has no idea what task to perform. It lacks the ability to *disambiguate* tasks, and as such, cannot make progress. We present additional videos and experiments to the above here: https://sites.google.com/view/lila-corl21/home/no-lang-baseline.

Second, we train 3 different latent action models for the cross disambiguation task seen in Figure 2 of the main paper: A 1 DoF without language baseline, LILA (our proposed approach) with 1 DoF, and a 2 DoF without language baseline. Each model is trained on a dataset of 100 demonstrations collected across the 4 tasks. We then visualize movement trajectories (see Fig. 10) by controlling the latent action (z for 1 DoF, z_1 and z_2 for 2 DoF). As expected, without any additional input such as language, a 1-DoF controller is incapable of task disambiguation, with a clear 0% task success rate for our simple cross setting. With both our 1-DoF controller w/ language input and a 2 DoF controller, task disambiguation is possible – highlighting the necessity of additional information of any modality. However, note that in large multi-task environments with dozens of tasks, re-designing a controller can be difficult – language is a much more flexible and natural way to add this information.

Together, these results motivate why no-language latent action baselines are incapable of being useful for multi-task environments, as they are limited by total degrees of freedom of the controller. Because the goal of our user study is to compare methods that could be useful for successful task completion, If included, such a baseline would be uninformative as it would be impossible for any user to achieve above a 0% task completion rate.

G Crowdsourcing & User Study

As described in the paper, to train LILA with language utterances, we hire crowdworkers on Amazon Mechanical Turk to provide language utterances give video demonstrations of a human assisting the robot arms. We paid crowdworkers 1.20 dollars to provide short utterances for *seven* videos. Notably, crowdworkers were not given any information about the possible tasks, or names of the objects in the scene. An image of the interface provided to crowdworkers is included in Fig. 11.

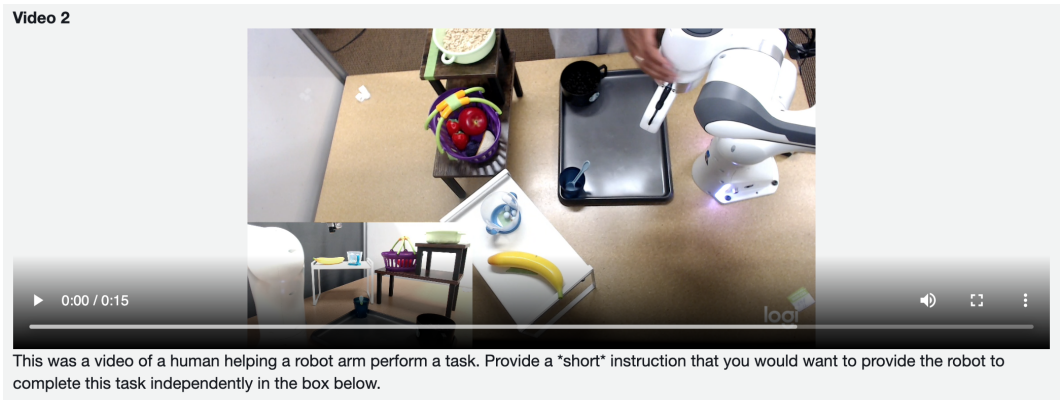

Figure 11: Interface shown to crowdworkers for collecting language utterances.

Filtering Crowdsourced Language Annotations. As described in the main paper, one issue with crowdsourcing language utterances from Amazon Mechanical Turk is the presence of "spam", noise, or extremely out-of-domain annotations. We accept and pay all crowdworkers for their responses, but adopt the following filtering algorithm before training on the collected utterances:

We set the threshold K to use utterances from 15 different crowdworkers for each of the 5 tasks. Example utterances from crowdworkers who we filtered out include *"move your arm outwards towards the left about 10 inches, lower your arm until you can't anymore then close your claw."* and *"the human the robot to take bowl"*, highlighting the challenges of eliciting high quality language utterances from demonstrations, and the necessity of such filtering methods.

All 15 utterances for each of the 5 tasks are listed in the file `'full-annotations.txt'` (under `attachments/` in the code repository), constituting the entirety of our training data. The file `'filtered.txt'` contains the utterances that were filtered out by our procedure.

User Study Instructions. As described in the main paper, we conducted a user study with 10 participants. The attached PDF file `'user-study.pdf'` shows the example instructions provided to each participant in our study.