Shared Autonomy for Robotic Manipulation with Language Corrections

Anonymous ACL submission

Abstract

Traditional end-to-end instruction following approaches for robotic manipulation are notoriously sample inefficient and *lack adaptivity*; for most single-turn methods, there is no way to provide additional language supervision to adapt robot behavior *online* – a property critical to deploying robots in collaborative, safety-critical environments. In this work, we present a method for incorporating language corrections, built on the insight that an initial instruction and subsequent corrections differ mainly in the amount of grounded context needed. To focus on manipulation domains where the sample efficiency of existing work is prohibitive, we incorporate our method into a *shared autonomy* system. Shared autonomy *splits agency* between the human and robot; rather than specifying a goal the robot needs to achieve alone, language informs the control space provided to the human. Splitting agency this way allows the robot to learn the coarse, high-level parts of a task, offloading more involved decisions – such as when to execute a grasp, or if a grasp is solid – to humans. Our user study on a Franka Emika Panda arm shows that our correction-aware system is sample-efficient and obtains significant gains over non-adaptive baselines.

1 Introduction

Research at the intersection of natural language and robotics has focused on *dyadic interactions* between humans and robots, often in the single-turn instruction following regime (Tellex et al., 2011; Artzi and Zettlemoyer, 2013; Thomason et al., 2015; Arumugam et al., 2017). In this paradigm, a human gives an instruction, and the robot executes behavior in the world, autonomously – simultaneously resolving the human's goal as well as planning a course of actions to execute in the environment. While impactful, building systems with this explicit division of agency between humans and robots is nontrivial; many existing systems either make strong assumptions about the environment in order to use motion planners (Matuszek et al., 2012; Kollar et al., 2013), or require extreme amounts of language-aligned data to learn general policies (Chevalier-Boisvert et al., 2019; Stepputtis et al., 2020; Lynch and Sermanet, 2020).

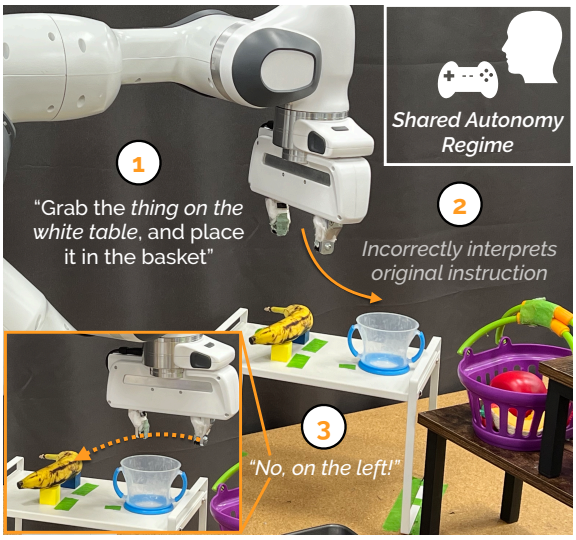

Figure 1: Our proposed system. Whereas prior work only allows for issuing a *single* language utterance held constant during execution (solid line), our approach allows users to provide *language corrections* during execution (left window – dashed line).

Coupled with the severe sample inefficiency of existing approaches is their *lack of adaptivity*. Consider the robot in Fig. 1, trying to execute "grab the thing on the white table and place it in the basket." This instruction is ambiguous and as a result, it is not clear what should happen. One natural option is for the human to provide a set of streaming corrections to the robot, changing its behavior on the fly. While recent work tries to get at the spirit of this idea by learning from dialogue (Thomason et al., 2019a,b), post-hoc corrections (Co-Reyes et al., 2019), or implicit feedback (Karamcheti et al., 2020), none of these approaches work *online*. Real systems for language-driven

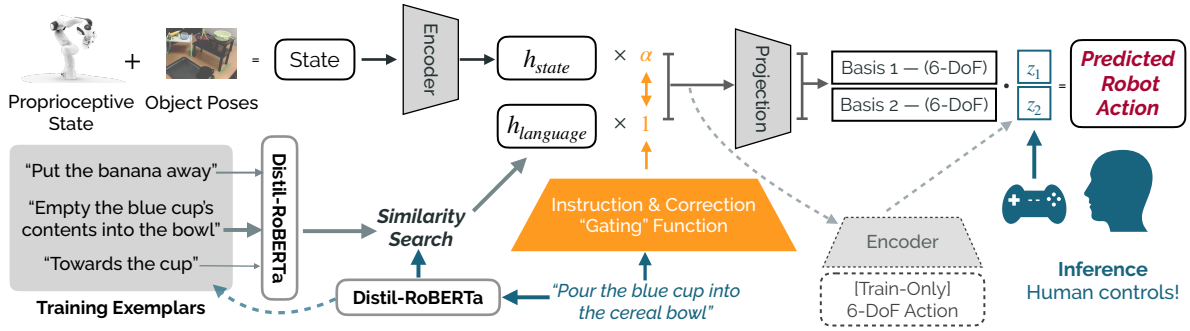

Figure 2: Our proposed LILAC model. Central to LILAC is the "gating" module (orange) which controls the amount of state-context for a given language input, allowing us to handle corrections. We use GPT-3 in lieu of a heuristic to provide α (see Appendix A for discussion), though plan to learn α from user feedback in the future. Solid lines represent the inference pipeline, while dashed lines indicate training-only steps.

human-robot interaction must be able to handle streaming corrections in a manner that is both *natural* and *sample efficient*.

One answer to sample efficiency is in leveraging existing methods in *shared autonomy* (Dragan and Srinivasa, 2013; Argall, 2018; Javdani et al., 2018). Unlike the explicit division above, approaches for shared autonomy *split agency* between the human and robot; during execution, both parties influence the ultimate actions of the robot, sharing the burden of reasoning over actions. The concrete instance of shared autonomy that we focus on in this work is assistive teleoperation (Herlant et al., 2016; Losey et al., 2021), where we learn low-dimensional control spaces that humans can use via joysticks, sip-and-puff devices, or other assistive tools to maneuver high-dimensional robots. These approaches operate at the scale of 5 - 10 examples per task, vs. the 10K - 100K reported by modern dyadic instruction following systems (Luketina et al., 2019).

A recent paradigm for building assistive teleoperation systems is with "learned latent actions" (Losey et al., 2020; Jeon et al., 2020; Karamcheti et al., 2021b), a set of approaches that uses small datasets of demonstrations to learn task-specific assistive controllers. While powerful, a key failing of these approaches is their ability to handle multiple objectives – and more importantly – provide a natural interface for users to specify their goals. Karamcheti et al. (2021a) introduce LILA (Language-Informed Latent Actions) to address this, by using language in a manner similar to the single-turn approaches above. While accumulating gains in sample efficiency, LILA lacks adaptivity – failing to handle online, streaming corrections.

In this work, we introduce **LILAC** (Language-Informed Latent Actions with Corrections), an adaptive system for real-world robotic manipulation, that – unlike LILA – can effectively interpret streaming language corrections. Critically, our key insight is in realizing language utterances vary in the amount of object state-dependence they require. An *instruction* like that in Fig. 1 – "grab the thing on the white table, and place it in the basket" – requires dense *grounded* information about object positioning while a *correction* – "no, on the left!" – can be expressed as a function of the robot's proprioceptive state, without object information. This insight allows us to decouple interpreting corrections from grounding: we can learn corrections from *even fewer examples*, boosting the robustness *and* generalization potential of our approach.

The following sections introduce LILAC, including how we identify the "state-dependence" of an utterance. We layout our experiments, and culminate with the results of a user study (small-scale, $n = 5$) conducted on a physical Franka Emika Panda (a 7-DoF fixed arm manipulator), with a discussion of future work on extending LILAC, and natural language supervision for shared autonomy.

2 Motivating Example

To gain a picture of LILAC's latent actions pipeline, we present a motivating example in Fig. 1. A user first gives an instruction to "grab the thing on the white table and place it in the basket," which presents them a 2-DoF control space they can use via the joystick in their hands. With this new control space, pushing `up` on the joystick may bring the arm closer to the table, while `right` might twist the gripper to align with the objects on the table. Unfortunately, the initial model's control space prediction is not perfect, and the arm skews towards the wrong object! With LILAC, the *user is able*

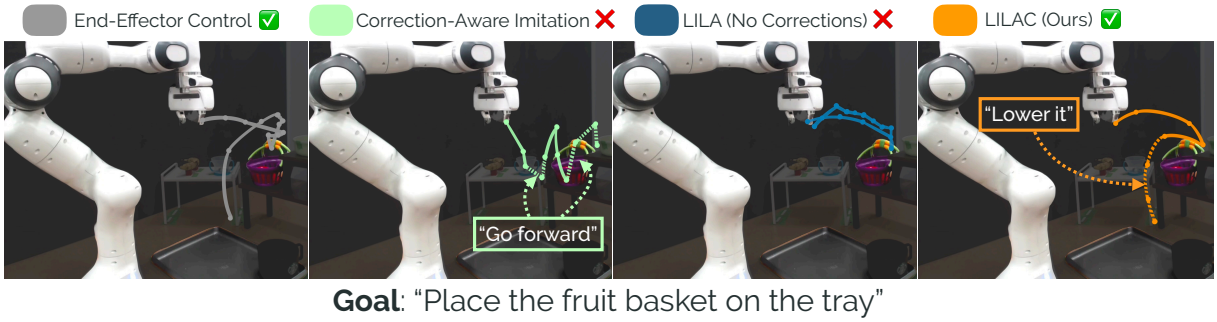

Figure 3: Qualitative trajectories from 4 different control strategies, operated by an "oracle." Note the dashed lines indicate when language corrections were used. In general, both LILAC and the End-Effector control methods are able to solve tasks, with LILAC able to do so more efficiently. Imitation Learning (green) fails to fit the tasks – even with corrections – and LILA is unable to progress without finer-grained correction information.

to issue real-time language corrections, updating their control space. Now, pushing left on the joystick brings the arm left, allowing the user to grasp the banana, as they intended. LILAC learns from less than 2 hours of demonstration data, and provides a strong interface for solving complex tasks.

3 LILAC: Natural Language Corrections

LILAC builds off of LILA as introduced by Karamcheti et al. (2021a) by adding a *gating* module to handle streaming corrections. The architecture is depicted in Fig. 2; solid lines denote the inference logic, while dashed lines denote training logic.

Overview. LILAC is a conditional autoencoder with some extra structural elements. The encoder takes in (language, state, action) pairs, and factors modeling into predicting a set of (language, state)-conditioned *action basis vectors*, and the latent actions z, which weight those vectors. Decoding involves taking the linear combination of the action bases and the predicted z values, with the objective of reconstructing the original high-DoF action.

At train time, we assume a dataset of language instructions (or corrections), paired with robot trajectories comprised of (state, action) pairs. For this work, we assume the action space is the 6-DoF end-effector velocity of the robot (obtained via forward kinematics), and the state space is the combination of the robot's proprioceptive state, containing information about its joint states and end-effector pose (also in 6D), as well as the coordinates of each object in the scene. Exactly as in LILA, we use a pretrained Distil-Roberta model from Sentence-BERT (Reimers and Gurevych, 2019) to encode language utterances, in tandem with an "unnatural-language processing" nearest neighbors index (Marzoev et al., 2020); because we're in the small data regime (2 hours of demonstrations), projecting inference-time utterances onto existing training exemplars prevents the LILAC model from generalizing poorly as language embeddings drift, which could lead to practical issues of user safety.

Gating Instructions vs. Corrections. The key insight of this work is that language instructions and corrections differ in their amount of object state-dependence – but what does this mean? From a linguistic perspective, one might categorize different utterances based on the number of *referents* present; an utterance like "grab the thing on the white table and place it in the basket" as in Fig. 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 without object information, solely as a function of the robot's pose.

To operationalize this idea whilst remaining sample efficient[1] we use a *gating* function (orange, in Fig. 2) that given language, predicts a value α from 0–1. A value of 0 signifies a *correction*. Appropriately, in our architecture, this zeroes out any state-dependent information (see the α term in Fig. 2), and predicts an action solely based on the provided language. In this work, we construct a prompt harness with GPT-3 (Brown et al., 2020) to output $\alpha = 0, 0.5, 1$ – the prompt text and motivation for this decision can be found in Appendix A.

[1] One question is why not treat all utterances as equal; 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 the LILAC approach, we realistically only need one!

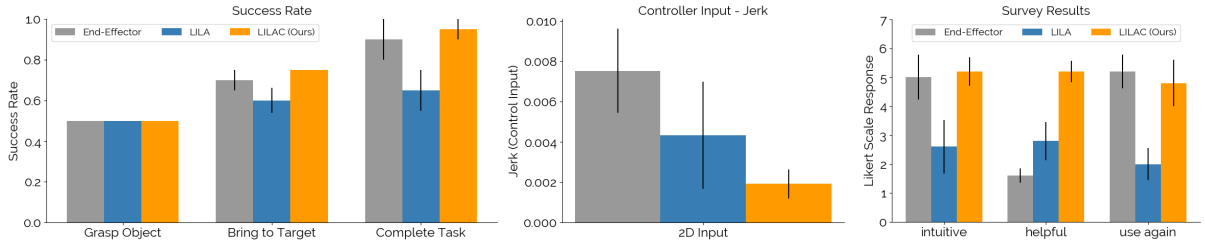

Figure 4: User study results ($n = 5$) with a Franka Emika Robotic arm. LILAC outperforms the non-adaptive LILA model across the board, obtaining higher success rates, and is preferred by users. While LILAC and End-Effector control obtain similar success rates (though LILAC is slightly higher), plotting the "jerk" (2nd derivative of input velocity) paints a different picture: controlling the end-effector is very jerky, requiring high user load.

4 Experiments & User Study

Our experiments consist of a targeted evaluation with an expert "oracle" of LILAC and 3 different baselines (results in Fig. 3), as well as a real-world user study ($n = 5$) on a complex manipulation workspace with a Franka Emika Panda arm (results in Fig. 4). We reuse all the publicly released data from Karamcheti et al. (2021a), and for correction data, collect a handful of corrections for moving in the 6 cardinal directions, as well as two mixed corrections that require some level of state grounding ("lower the bowl" and "tilt the cup").

Targeted Expert Evaluation. The goal of our expert-controlled evaluation was to evaluate the corrections-module in LILAC, and get a point of reference to LILA and traditional methods for robot control – specifically, using a control scheme that uses inverse kinematics to control the end-effector of the robot, two axes at a given time (e.g. `[X, Y]`, `[Z, Roll]`, `[Pitch, Yaw]`). Fig. 3 shows the results; LILAC and End-Effector control are both able to accomplish the two tasks, but LILA struggles to recover from overshooting problems. We also evaluate one other strategy – Language & Correction-Aware Imitation, which doesn't have enough data to fit a reasonable policy (see discussion in Karamcheti et al. (2021a) for more information). We additionally evaluate a "no-language" variant of latent actions in Appendix C.

User Study. Given the results from the automatic evaluation, we run a within-subjects user study ($n = 5$, 3 male, 2 female, 3 users with prior teleoperation experience), with the three operation schemes – LILAC, LILA, and End-Effector control – as the three conditions. Each user was randomly assigned one of the five original tasks from the LILA work, and asked to complete it with each control strategy.[2] Fig. 4 shows the general quantitative (left) and subjective (right) study results.

In general, LILAC strictly dominates LILA, showing the benefit of adding even simple real-time correction handling. However, compared to End-Effector control, the success is less clear. While obtaining a slightly higher success rate, the qualitative results are mixed. A possible confound is the limited pool of participants, many who already had prior teleoperation experience (due to COVID restrictions, we could not widen the pool). For additional insight, we plotted the 2nd-derivative of acceleration – jerk – of the input controls; we see here that End-Effector approaches require significantly more user load and fast movement compared to LILAC. Not only is this more taxing on the user, but is potentially unsafe depending on the application – an axis we will explore in future work.

5 Discussion

The union of natural language and shared autonomy for real-world robot manipulation is a rich and vibrant research area. Moving beyond strict dyadic interactions towards the shared autonomy setting opens the door to rich work in language supervision for robotic manipulation – work that has so far been limited by the steep data requirements of training language-conditioned policies (Stepputtis et al., 2020; Shridhar et al., 2021).

Future work in language and shared autonomy will allow for learning new corrections (and tasks) online, from user feedback. More broadly, we hope to generalize our correction module to other versions of language-informed robotics – for example, to policy blending (Dragan and Srinivasa, 2013), guided planning, and interactive imitation learning (Kelly et al., 2019).

[2] More information about the structure of the study can be found in Appendix B

References

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

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)*, 1:49–62.

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)*.

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*.

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)*.

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)*.

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

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)*, pages 35–42.

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)*, 37:717–742.

Hong Jun Jeon, Dylan P. Losey, and Dorsa Sadigh. 2020. Shared autonomy with learned latent actions. In *Robotics: Science and Systems (RSS)*.

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)*.

Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. 2021a. LILA: Language-informed latent actions. In *Conference on Robot Learning (CoRL)*.

Siddharth Karamcheti, A. Zhai, Dylan P. Losey, and Dorsa Sadigh. 2021b. Learning visually guided latent actions for assistive teleoperation. In *Learning for Dynamics & Control Conference (L4DC)*.

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)*, pages 8077–8083.

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

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)*, pages 1–33.

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)*, pages 378–384.

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)*.

Corey Lynch and Pierre Sermanet. 2020. Grounding language in play. *arXiv preprint arXiv:2005.07648*.

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*.

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)*, pages 1671–1678.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. 2021. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning (CoRL)*.

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)*.

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)*.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2019a. Vision-and-dialog navigation. In *Conference on Robot Learning (CoRL)*.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin W. Hart, Peter Stone, and Raymond J. Mooney. 2019b. Improving grounded natural language understanding through human-robot dialog. In *International Conference on Robotics and Automation (ICRA)*.

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)*.

A Using GPT-3 to Identify Corrections

A core component of LILAC is the choice of gating function, for producing the object state dependence weight α for a given language utterance. Critically, α dictates whether an utterance is an *instruction* $(\alpha = 1)$ that depends on the current environment context, a *correction* $(\alpha = 0)$ that does not, and can be interpreted without additional grounded information, or something *in-between* $(\alpha = 0.5)$.

We made the realization early on that identifying whether a language utterance fell into one of the above categories could (at least heuristically) be decoupled from any environment information; that is to say, we could predict α directly from the language utterance alone. Given this, and given we were not sure whether the α-gating would even work in our small-data regime, we found it difficult to defend the choice to collect data to learn α upfront, prior to running through the whole system. Given this, our choices were to either hardcode a series of α values for a small, fixed set of correction language, or come up with some heuristic (e.g., referent-counting) that could break or generalize poorly to "in-between" utterances. Instead, we chose to try a prompt construction based approach, leveraging GPT-3 (Brown et al., 2020).

The prompt we specified is shown in Fig. 5; we did minimal prompt-tuning, only reordering the examples shown, and turning the temperature down to 0 (as we wanted this to be deterministic). We found GPT-3 to work incredibly well out of the box – a phenomenon that *cannot be overstated*. With a straightforward procedure, we were able to use GPT-3 as a drop-in replacement for what otherwise would have been a brittle heuristic, or a limited set of language corrections. It is incredibly exciting to be able to prototype these systems via GPT-3 quickly, and we hope that this type of usage becomes prevalent throughout not only human-robot interaction, but widespread NLP pipelines as a whole. Given the results with the "drop-in" GPT-3 model, we have a good idea as to where we need to focus future work with respect to *learning* α – specifically, for handling the nuanced utterances that are "in-between" corrections and instructions – and are excited to tackle this moving forward.

B User Study Details & Tasks

As mentioned in §4, we ran a within-subjects study with a small number of participants $(n = 5)$. The study consisted of each user using the following

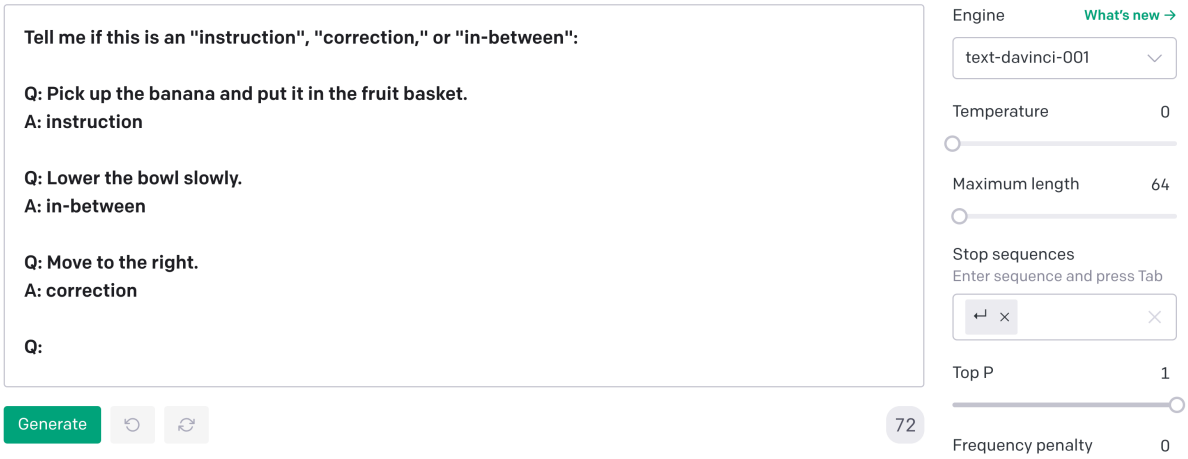

Figure 5: The concrete prompt used for GPT-3 `davinci-instruct`, visualized in the OpenAI API Playground. We primed GPT-3 with 3 handcrafted examples, without much other thought, and used the corresponding outputs as our α gating values ("instruction" = 1, "in-between" = 0.5, "correction" = 0).

strategies to solve a given task: End-Effector control, LILA (no corrections), and LILAC (our proposed approach). The order of strategies was randomized across users. The tasks were as follows:

1. `Pick & Insert Banana` – Grasp the banana, and place it in the plastic fruit basket, turning the gripper appropriately to insert the banana.

2. `Pick & Place Basket` – Grasp the basket by the handles, and lower it onto the tray.

3. `Pick & Place Cereal Bowl` – Grasp the green cereal bowl by the lip of the bowl, lift it off the pedestal, and lower it onto the tray without collision.

4. `Pick & Pour into Cereal Bowl` – Grasp the blue cup with marbles by either the lip of the cup or the handle on the side, then lift it over the cereal bowl, tilting the cup to pour the marbles into the bowl.

5. `Pick & Pour into Mug` – Grasp the blue cup with marbles by either the lip of the cup or the handle on the side, then lift it over the black mug – avoiding collisions – then pour the contents into the mug.

Prior to executing the task in a given control mode, each user was given 3 minutes to practice using that control mode. During practice, the user could experiment with various patterns of joystick input to better understand how they translated to movement of the robot arm. We found that the amount of "naturalization" time was critical in getting users to adapt to the interfaces provided by LILA, and LILAC. Furthermore, for users not already experienced in robot teleoperation, we found this practice period important as well. However, we found that it *takes longer to naturalize to LILAC vs. End-Effector control* – this also explains the slight difference in results between LILAC and End-Effector control. Future work will explore the impact of this "naturalization period" under various conditions.

C Additional Visualizations & Baselines

To supplement our experiments from §4, we present two additional sets of trajectory visualizations in Fig. 6 and Fig. 7. Fig. 6 shows the same 4 strategies as in Fig. 3, except for a more complex pouring task; in general, the pattern of behavior is the same – both LILAC and End-Effector control are able to solve the task, whereas LILA stalls out due to a lack of a corrective signal, with Correction-Aware Imitation Learning failing to fit a decent policy given limited data.

One other baseline we ran was the *no-language variant of latent actions*; we ran this to show the impact that language has on providing intuitive, useful control spaces. Fig. 7 shows the results of this baseline vs. LILAC – at a glance, the no-language variant completely fails, reducing to random, oscillatory behavior, because it cannot learn to provide a good control space for all tasks, without an extra conditioning signal. This is the same observation made in Karamcheti et al. (2021a).

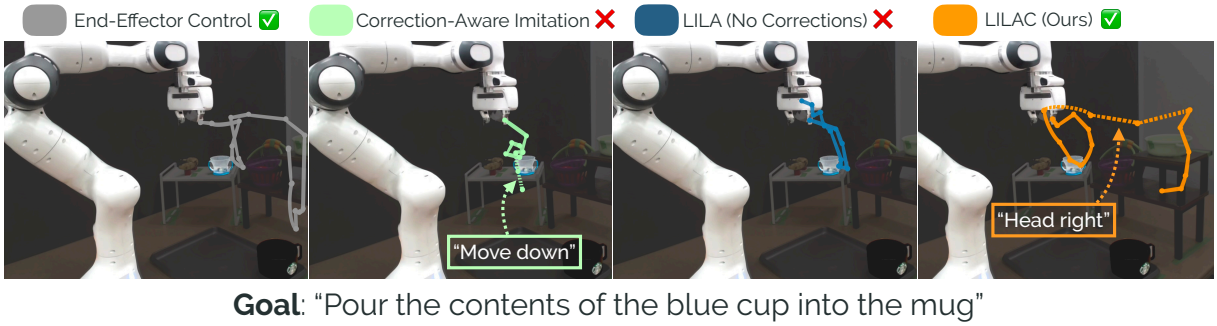

Figure 6: Additional trajectory visualizations for the four strategies from §4, this time for a more complex "pouring" task; we see similar results as in Fig. 3.

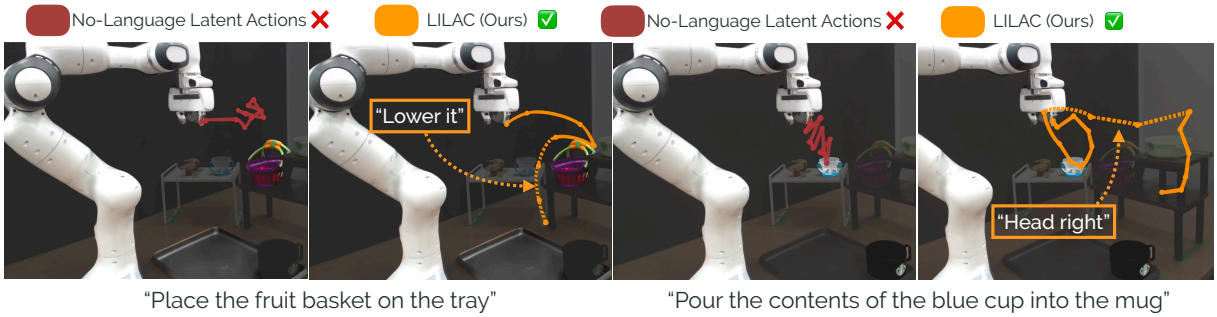

Figure 7: Results visualizing trajectories for LILAC and an additional baseline strategy – "No-Language Latent Actions" – the *language-free* variant of our approach. Note that this approach trivially fails, as its overloaded, unable to find a satisfying control scheme that would allow users to perform all 5 tasks without extra conditioning information. This translates to aimless, oscillatory behavior during execution.