**Robotics: Science and Systems 2024**
**Delft, Netherlands, July 15-July 19, 2024**

# RT-H: Action Hierarchies Using Language

Suneel Belkhale[1,2], Tianli Ding[1], Ted Xiao[1], Pierre Sermanet[1], Quan Vuong[1], Jonathan Tompson[1], Yevgen Chebotar[*,1], Debidatta Dwibedi[*,1], Dorsa Sadigh[*,1,2]

[1]Google DeepMind, [2]Stanford University

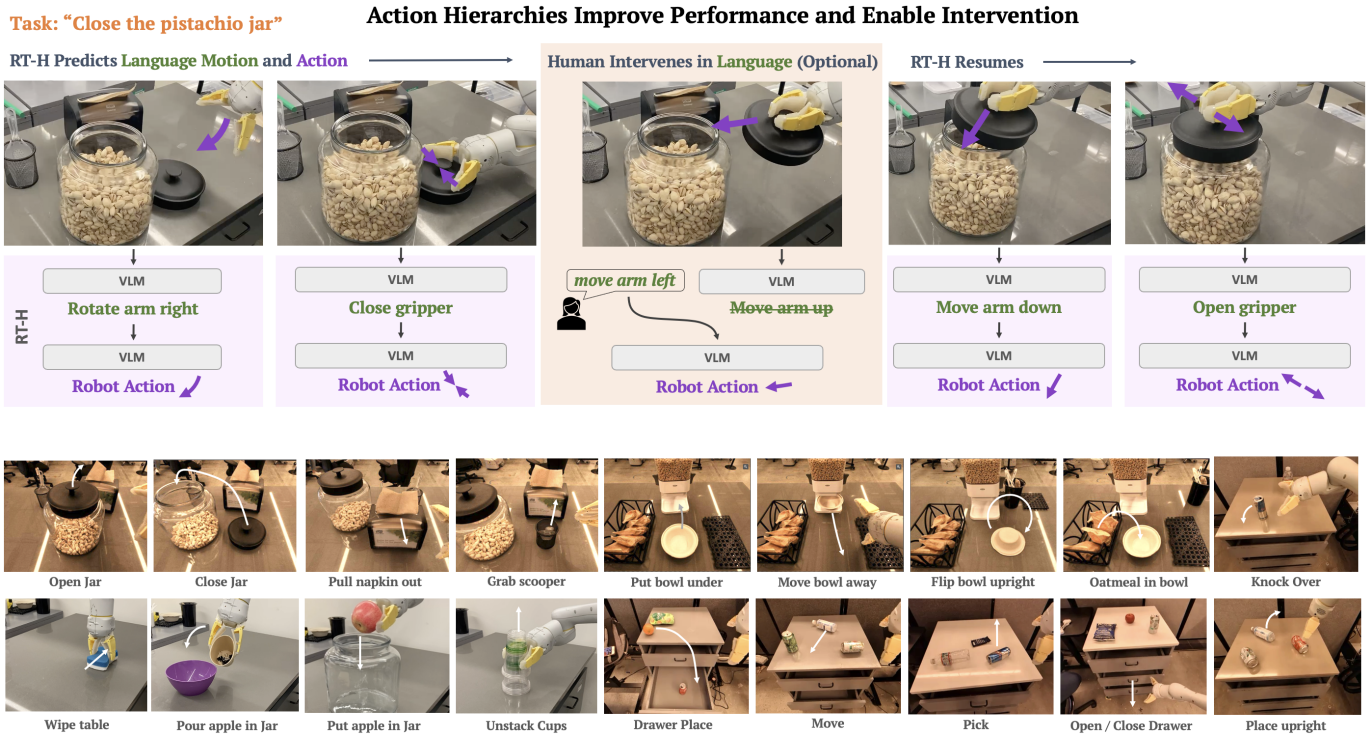



**Fig. 1:** Given a task in language like "close the pistachio jar" and an image of the scene, RT-H utilizes a Vision Language Model (VLM) to predict *language motions* like "move arm forward" and "rotate arm right", and then conditioned on these language motions, it predicts *actions* for the robot (purple box). This action hierarchy teaches the model the *shared structure* across tasks with many semantically different descriptions (see bottom task examples). These language motions enable better data sharing across diverse multi-task datasets as compared to mapping directly from the task to actions. This hierarchy also enables humans to optionally provide *language motion corrections* to the robot to prevent task failure, and then to use these new language motions to predict better actions (peach box). Once the human is done intervening, RT-H resumes predicting language motions like before.

***Abstract*—Language provides a way to break down complex concepts into digestible pieces. Recent works in robot imitation learning have proposed learning language-conditioned policies that predict actions given visual observations and the high-level task specified in language. These methods leverage the structure of natural language to share data between semantically similar tasks (e.g., "pick coke can" and "pick an apple") in multi-task datasets. However, as tasks become more semantically diverse (e.g., "pick coke can" and "pour cup"), sharing data between tasks becomes harder and thus learning to map high-level tasks to actions requires substantially more demonstration data. To bridge this divide between tasks and actions, our insight is to teach the robot the language of actions, describing low-level motions with more fine-grained phrases like "move arm forward" or "close gripper". Predicting these *language motions* as an intermediate step between high-level tasks and actions forces the policy to learn the shared structure of low-level motions across seemingly disparate tasks. Furthermore, a policy that is conditioned on language motions can easily be *corrected* during execution through human-specified language motions. This enables a new paradigm for flexible policies that can learn from human intervention in language. Our method RT-H builds an *action hierarchy* using language motions: it first learns to predict language motions, and conditioned on this along with the high-level task, it then predicts actions, using visual context at all stages. Experimentally we show that RT-H leverages this language-action hierarchy to learn policies that are more robust and flexible by effectively tapping into multi-task datasets. We show that these policies not only allow for responding to language interventions, but can also learn from such interventions and outperform methods that learn from teleoperated interventions. Our website and videos are found at rt-hierarchy.github.io.**

## I. Introduction

Language is the engine of human reasoning, empowering us to break complex concepts into simpler ones, to correct our misunderstandings, and to generalize concepts in new settings. In recent years, robots too have begun to leverage language's efficient, compositional structure for breaking down high-level

concepts [1], providing language corrections [2, 3], or enabling generalization to new settings [4]. These works often share a common paradigm: given a high-level *task* described in language like "pick coke can", they learn policies that map observations and task descriptions in language to low-level robot *actions* across large multi-task datasets. The advantage of language in these settings is to encode the shared structure between similar tasks (e.g., "pick coke can" vs. "pick an apple"), reducing the data needed to learn the mapping from tasks to actions. However as tasks become more diverse, so too does the language describing each task (e.g., "pick coke can" vs. "pour a cup"), making it harder to learn the shared structure between different tasks from only the high-level language.

To learn diverse tasks, our aim is to better capture the similarities between these tasks. We observe that language is capable of expressing much more than just the high-level task: we can also express *how* to do the task – a more fine-grained representation that lies closer to the low-level actions. For example, we can decompose the "pick coke can" task into a sequence of fine-grained behaviors, which we denote as *language motions*: "move arm forward", then "grasp the can", and then "move the arm up". Our key insight is to leverage language motions as an intermediate prediction layer between high-level task descriptions and low-level actions – thus building an *action hierarchy* via language motions. Creating such an action hierarchy leads to several benefits: (1) It enables much better data sharing between different tasks at the level of language motions, leading to better language motion composition and generalization in diverse multi-task datasets. For example, even though "pour a cup" and "pick up a coke can" are semantically different, they entirely overlap at the language motion level until the object is picked. (2) Language motions are not merely fixed primitives, but rather learned in the *context* of the current task and scene using the instruction and visual observation. For example, "move arm forward" alone does not convey how fast to move or in what exact direction vector; that depends on the task and the observation. The contextuality and flexibility of learned language motions introduce a new set of capabilities: it allows humans to provide their own *corrections* to language motions when the policy is not 100% successful (see center orange box in Fig. 1). Further, the robot can even learn from these human corrections, entirely in the realm of language motions. For example, with "pick coke can", if the robot closes its gripper early, we can instead tell it to "move arm forward" for longer, which RT-H interprets in context of the current scene. This slight change in language motions is not only easy for a human to provide, but also much easier to learn from compared to correcting individual robot actions.

Motivated by the benefits of language motions, we propose an end-to-end framework, RT-H (Robot Transformer with Action Hierarchies), for learning these action hierarchies: at each step, RT-H conditions on the observation and the high-level task description to predict the current language motion (language motion query), enabling the model to reason about how to do the task at a fine-grained level. Then RT-H uses the observation, the task, and the inferred language motion to predict the action for that step (action query), where the language motion provides additional context to improve the prediction of precise actions (see purple box in Fig. 1). To extract language motions from our data, we develop an automated approach to extract a simplified set of language motions from robot proprioception, yielding a rich library of over 2500 language motions without any manual annotation effort. We base our model architecture on RT-2, a large Vision-Language Model (VLM) which co-trains on internet-scale vision and language data to improve policy learning [4]. RT-H uses a single model for both language motion and action queries to leverage this broad internet-scale knowledge for all levels of the action hierarchy.

Experimentally, we find that using a language motion hierarchy yields substantial improvements when ingesting diverse multi-task datasets, outperforming RT-2 by 15% on a wide range of tasks. We also find that correcting language motions reaches near perfect success rates on the same tasks, demonstrating the flexibility and contextuality of learned language motions. Additionally, fine-tuning our model with language motion interventions outperforms state-of-the-art interactive imitation learning methods such as IWR [5] by 50%. Finally, we show that language motions in RT-H generalize to variations in scene and objects better than RT-2.

## II. Related Work

In this section, we discuss the role of language in policy learning, how hierarchy has been used in imitation learning, and previous approaches for providing and learning from human corrections on robot policies.

**Language-Conditioned Policies.** In recent years, language has emerged as a powerful goal representation for robotic tasks. In imitation learning (IL), many approaches encode tasks described in language into embeddings using pretrained language models, which are then inputted to a policy that is trained on multi-task robot datasets [6–11]. These pretrained language embeddings lack any visual understanding, so other works jointly train visual and language representations from the ground up, often using large internet-scale datasets and sometimes including robot data [12–15]. The resulting goal representations can then be inputted into a policy to provide both visual and semantic context. More recently, policies built on vision language model (VLM) backbones have become capable of learning actions directly from visual observations and language without the need for pretrained embeddings [4, 16]. All these approaches leverage language to represent the high-level task and often directly predict low-level actions – but as both language task descriptions become semantically diverse, sharing data between different tasks becomes challenging, so significantly more data is required.

**Hierarchical Action Representation in Imitation Learning.** An alternative approach to boost performance is to impose structure on the multi-task learning problem through the use of *hierarchical action representations*. Several works have explored learning general "skill" representations as parameterized primitives [17, 18] or embeddings to describe short sequences of actions or interactions with objects, often from

diverse multi-task datasets [10, 19–28]. While they generally improve performance, they are often quite computationally complex and sensitive to hyperparameters. Another line of work shows the benefits of separating coarse and fine action abstractions in IL, but requiring both coarse and fine action annotations [29, 30].

Language has also been used to create hierarchy in multi-task learning. When tackling long horizon instructions, many recent approaches use LLMs or VLMs to decompose long horizon instructions into a sequence of tasks specified in language [1, 16, 31–34]. Usually, scripted or individually trained policies are used to execute these tasks, limiting scalability. To learn long horizon policies end-to-end, Hu and Clune train a model to first predict language tasks and then predict actions conditioned on those tasks [35]. This approach is similar to RT-H but exists one level higher in the action hierarchy: they do not label or learn from fine-grained language motions. A few works explore the usage of more fine-grained language, for example predefined motion primitives or by predicting object dynamics from verbs [36, 37]. Sharma et al. use an LLM to decompose tasks into a sequence of motion primitives [36]. Yet, these motion primitives are hard-coded and lack the contextuality that is required in more complex settings. RT-H learns language motions in the context of both the task and the scene, enabling better policies and more contextual corrections.

**Interactive Imitation Learning and Correction.** Interactive IL methods learn from human feedback during robot execution [38]. Ross et al. proposed DAgger, which iteratively aggregates expert annotated actions for online rollouts [39]. While effective, providing such expert annotation is costly, so later works used more selective interventions, for example letting either the human decide when to intervene [5, 40] or letting the policy decide [41–44]. These methods all require intervention in the action space of the robot, i.e., robot teleoperation [5] or kinesthetic teaching [45, 46], which can be challenging for non-experts and hard to scale.

To make intervention more intuitive and scalable, several works have studied language as an intervention medium, for example intervening on incorrect task predictions with human guidance [32, 47]. Correcting language at a more fine-grained level is challenging. Several works define a fixed set of fine-grained language corrections and then map natural language utterances to these correction primitives [48]. Later work removes these brittle primitives and replaces them with composable cost functions, but they assume privileged environment knowledge [3]. Data driven approaches have also been explored, requiring large datasets of language corrections [49–52] or shared autonomy during deployment [2]. RT-H instead learns to *predict* language motions end-to-end with actions, enabling not just correction in the space of language motions but also efficient learning from those corrections.

## III. RT-H: Action Hierarchies using Language

To effectively capture the shared structure across multi-task datasets – that is not represented by high-level task descriptions – our goal is to learn policies that explicitly leverage *action hierarchies*. Specifically, we introduce an intermediate *language motion* prediction layer into policy learning. The language motions describing fine-grained behavior of the robot can capture useful information from multi-task datasets and can lead to performant policies.

When the learned policies struggle to perform, language motions can again come to rescue: they enable an intuitive interface for online human corrections that are contextual to the given scene. A policy trained with language motions can naturally follow low-level human corrections and successfully achieve the task given the correction data. Additionally, the policy can even be trained on the language correction data and further improve its performance.

**RT-H Model Overview.** RT-H, shown in Fig. 2, has two key phases: It first predicts language motions from the task description and visual observation (*language motion query*, top left of Fig. 2), and then conditions on the predicted language motion, the task, and the observation to infer the precise actions (*action query*, bottom left of Fig. 2). We instantiate RT-H using a VLM backbone and following the training procedure from RT-2 [4]. Similar to RT-2, we leverage the immense prior knowledge in natural language and image processing in internet-scale data through co-training. To incorporate this prior knowledge into all levels of the action hierarchy, a single model learns both the language motion and action queries.

### A. Formalizing Action Hierarchies

We are given a dataset $\mathcal{D} = \{(\tau_1, g_1), \ldots, (\tau_N, g_N)\}$ of $N$ expert demonstrations ($\tau$) paired with task descriptions in natural language ($g \in \mathcal{G}$), where each $g$ describes exactly one task from a set of $m$ high-level tasks $\{T_i\}_{i=1}^m$. Each demonstration $\tau_i$ consists of a sequence of observations and *action hierarchies* of length $L_i$. We define an action hierarchy to consist of an intermediate action representation specified in natural language $z \in \mathcal{Z}$, and the low-level action $a \in \mathcal{A}$. Here, the intermediate action is more fine-grained than the high-level task, but more coarse-grained than the low-level action. Thus we write $\tau_i = \{(o_1, z, a_1), \ldots, (o_{L_i}, z_{L_i}, a_{L_i})\}$, with observations $o \in \mathcal{O}$. Our goal is to learn a sequence of policies: a high-level policy $\pi_h : \mathcal{O} \times \mathcal{G} \to \mathcal{Z}$ which maps observations and task descriptions to intermediate actions, and a low-level policy $\pi_l : \mathcal{O} \times \mathcal{G} \times \mathcal{Z} \to \mathcal{A}$ which maps observations, task descriptions, and the intermediate action to the low-level action. Then we define the action hierarchy policy as the composition of these two policies: $\pi(a, z|o, g) = \pi_h(z|o, g)\pi_l(a|o, g, z)$.

In this work, we model the intermediate action representation $z$ using language motions like "move arm forward" or "rotate arm right". Note that an action hierarchy can easily be extended to more than just a single level (i.e., $z^1 \ldots z^K$, in order of how fine-grained they are).

### B. RT-H: Model and Training Details

To model this action hierarchy and acquire the benefits of language motions, our method RT-H, shown in Fig. 2, learns $\pi_h$ and $\pi_l$ using a single VLM co-trained with internet-scale data. We instantiate this VLM with the same PaLI-X 55B [53] architecture as RT-2 [4] – RT-H uses a ViT encoder model to

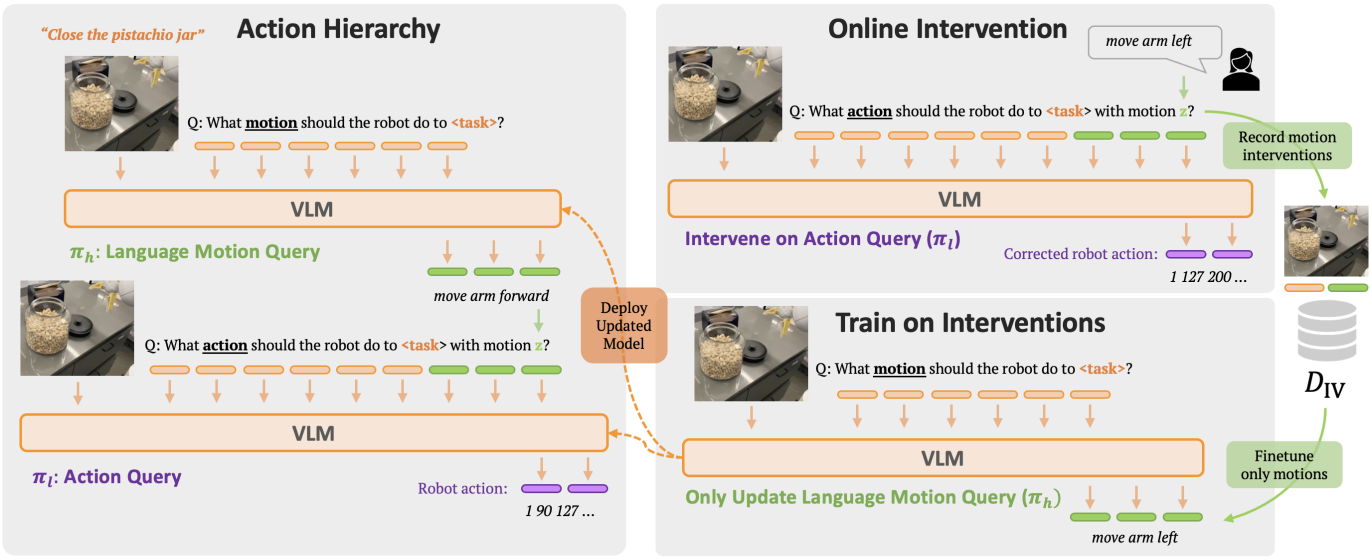



**Fig. 2:** RT-H Overview. **Left:** Our method leverages language to create an action hierarchy for policy learning. We separate the action prediction problem into a language motion query ($\pi_h$), which predicts a fine-grained language motion like "move arm forward" using the image tokens and task description tokens, and an action query ($\pi_l$), which flexibly decodes this language motion into actions using the context of the task and the scene. We leverage a single VLM for both queries based on RT-2 [4] that encapsulate the broad prior knowledge in internet-scale data at each level of the action hierarchy. **Right:** a user can intervene directly on the action query to provide language motion corrections to robot behavior, for example "move arm left" instead of "move arm forward" here (top). To learn from corrections, we can update only the language motion query with the newly labeled language motion corrections (bottom). Then we deploy the updated model back to the action hierarchy (orange block).

process images into tokens, and then uses an Encoder-Decoder transformer to convert streams of image and natural language tokens into action tokens. These action tokens are produced in the same fashion as RT-2, by discretizing each action dimension into 256 bins and encoding these bins as integer values. Each action $a$ is comprised of delta positions of the end effector, delta axis-angle rotations of the end effector, actions to close or open the gripper, and a termination flag. RT-H constructs two queries to the VLM. First, a **language motion query** models $\pi_h$, mapping tasks described in language $g$ and the image observations $o$ to language motions $z$ (Encoder sees $g$ and $o$, Decoder predicts $z$). This first stage teaches RT-H to first predict correct behavior (language motion) in a coarser and more compressed action space than the low-level robot actions, enabling better modeling of the structure of each task and thus better sharing of sub-trajectories across diverse tasks. Second, the **action query** models $\pi_l$, mapping the image $o$, task $g$, and the language motion $z$ to action tokens, which then get detokenized into robot actions $a$ (Encoder sees $g$, $o$, and $z$, Decoder predicts $a$). This second stage teaches RT-H to be contextual (both with the scene and the task description) in how it decodes language motion $z$ into precise actions to be executed. This extra context is often critical to complete the task successfully, and it is also important for performing and learning from correction, as we discuss in Section IV-A. Compared to training both language motion and action autoregressively in one query, using two queries enables (1) specialized prompts for each query, and (2) the language motion $z$ is passed into the Transformer Encoder rather than the Decoder when predicting actions.

RT-H is then co-trained using the same PaLI-X [53] training mixture that is used in RT-2, starting from a pre-trained checkpoint. The ViT encoder is frozen for this co-training. RT-H replaces the action prediction query in RT-2 with the language motion and action queries at equal sampling rates. Using a single model simplifies the training process, and enables both language motion and action queries to benefit from the broad prior knowledge in the PaLI-X training mixture. See Appendix A for model and training details.

### C. Extracting Language Motions

While in principle, humans can label the full spectrum of fine-grained language motions, we found that having humans provide these labels offline leads to language inconsistency across the dataset and even inaccuracy in the labeled skills. For example, humans would often mislabel the transitions between skills, or misjudge the direction of motion of the robot due to camera angles. Thus to cheaply extract reliable language motions $z$ at each time step in each episode, we develop an automated labeling scheme relying on robot proprioception information. First, we connect each dimension of the change in robot end effector pose to a spatial dimension (e.g., the z-axis of the position change maps to up and down). Doing this for all 9 action dimensions (3 dimensions for delta position, 3 dimensions for delta orientation, 2 dimensions for base movement, 1 dimension for gripper) we determine a list of the current *dominant* spatial movements of the robot, for example "move arm up and right", "close gripper", "rotate arm counterclockwise", or "turn base left". Then, we can filter out the dimensions that are below a chosen "small action" threshold, and then compose the resulting actions in order of the action magnitude. For example, if the robot is

predominantly moving the arm forward but also beginning to close gripper, we would extract "move arm forward and close gripper." In this manner, the combinatorial nature of language enables over 2500 language motions to be extracted from a simple set of known motions. Furthermore, since these language motions are derived directly from actions, they hold high predictive power for the actions themselves when running the action query in RT-H. Importantly, we fix the details of this procedure for all our experiments and datasets irrespective of the task, and so designing this procedure is a one-time fixed cost for the developer.

Of course, this procedure represents one simple way to define and extract language motions, and many others could exist. For example, one can imagine developing a higher-level object-referential language motion space, for example "reach the object" or "grasp the object handle", but this likely requires human annotation or robust object detection and tracking. One could also label at an even more fine-grained level, for example describing the rate of motion like "move arm forward slowly." However, these examples highlight a fundamental trade-off that exists in the chosen abstraction level for language motions: the more fine-grained they are, the harder they would be to predict for the language motion query, but the more guidance they provide to the action query, and vice versa. As we show in Section V, our choice of language motions strikes a good balance in both language motion and action query accuracy, while also being cheap to label.

## IV. RT-H: Inference & Correction

At test time, RT-H first runs the language motion query to infer the skill, and then uses this inferred skill in the action query to compute the action. However, this process doubles inference time since the two queries must be run sequentially at each time step. While not a problem for smaller models, with larger model sizes such as the 55B model used in RT-H, we will experience unavoidable querying lag. To handle this challenge, we discuss two modes of language motion inference: (1) **asynchronous querying**: we train just the language motion query in RT-H to predict the skill one step into the future. Then at test time, we query the action using the inferred language motion of the *previous* time step, while also predicting the language motion for the next time step. This enables us to batch the queries and thus achieve nearly identical querying lag as RT-2. (2) **fixed frequency**: we can evaluate skill queries once every $H$ steps, also reducing the amortized lag. In our experiments we opt for asynchronous querying, since skills often need to change at precise time steps that may not align with fixed frequencies.

### A. Correction via Language Motions

Even when an RT-H policy encounters new manipulation settings or fail at a given task, the action hierarchy in RT-H makes it possible to *correct* the policy: users can directly intervene on the learned language motions. While the policy might struggle at performing the high-level task, following the lower-level language motions is much easier.

Intervening on the model is simple in RT-H (see top right in Fig. 2), and since it is text-based, all you need is a keyboard or a microphone. Similar to prior interactive imitation learning approaches such as Intervention Weighted Regression (IWR) [5], we let the human operator decide when to intervene on the model, e.g., by pressing a key on the keyboard. Once they have entered the correction mode, they can type a new language motion correction on the keyboard or use hotkeys for common language motions. This new language motion will directly be passed into the action query in RT-H (see Fig. 2) to produce a contextual action that aligns with the user's intent.

We can also display the current predicted language motion for transparency and providing the user additional context, so they know what the robot was planning to do and can better choose their corrections. Then, at a fixed frequency, we requery the user to either enter a new language motion correction, keep running the previously entered language motion correction, or exit correction mode. Fixed frequency requerying gives the user time to update their correction or to decide to let the model take over once again.

**Learning from Correction Data.** To *learn* from these language motion corrections, we collect the language motion labels that were provided and the associated images for corrections from successful episodes. Then, we do not need to re-train the action query in RT-H, since the model already knows how to map the language motion correction to actions that succeed at the task; instead, we only need to update the language motion query to produce the correct higher level motions (see Fig. 2; bottom right). This significantly reduces the complexity of learning from corrections, since we only need to learn minor changes in the smaller language motion space rather than the large action space. Like IWR [5], we co-train with a mixture of the original dataset (both action and language motion queries) and the dataset of corrections (just language motion query). See Appendix B for more mixture details. Of course, since we use a single model for both language motion and action queries, updating only one will likely still update the other – co-training RT-H on both queries in the *original dataset* helps maintain action prediction performance while still learning the minor changes in language motion space through the intervention dataset.

## V. Experiments

To comprehensively evaluate the performance of RT-H, we study four key experimental questions:

- **Q1** (**Performance**): Do action hierarchies with language improve policy performance on diverse multi-task datasets?
- **Q2** (**Contextuality**): Are learned language motions in RT-H contextual to the task and scene?
- **Q3** (**Corrections**): Is training on language motion corrections better than teleoperated corrections?
- **Q4** (**Generalization**): Do action hierarchies improve robustness to out-of-distribution settings?

**Dataset**: We utilize a large multi-task dataset consisting of 100K demonstrations with randomized object poses and backgrounds. This dataset combines the following datasets:

- *Kitchen*: The dataset used in RT-1 [6] and RT-2 [4], consisting of 6 semantic task categories in 70K demonstrations.
- *Diverse*: A new dataset consisting of more complex range of tasks, with over 24 semantic task categories, but just 30K demonstrations (see Appendix C for more details).

We call this combined dataset the *Diverse+Kitchen* (*D+K*) dataset, and it is labeled with language motions using our automated procedure described in Section III-C. We evaluate our method trained on the full *Diverse+Kitchen* dataset on eight tasks that are a representative sample of its hardest tasks:

1) "flip bowl upright on the counter"
2) "open pistachio jar"
3) "close pistachio jar"
4) "move bowl away from cereal dispenser"
5) "put bowl under cereal dispenser"
6) "place oatmeal packet in the bowl"
7) "grab scooper from basket"
8) "pull napkin from dispenser"

These eight tasks, shown in Fig. 3, were chosen because they require complex sequences of motions and high precision.

**Methods**: We study and compare the following methods, including ablating a number of choices in RT-H:

- **RT-H** is our proposed method in this work, and we use the asynchronous querying variant for these experiments (see Section IV).
- **RT-H-Joint** is also our method but using a single autoregressive query to produce both language motion and action, rather than querying the VLM twice with two different prompts for each query. RT-H-Joint first outputs language motion then action (where action is still conditioned on language motion). While both RT-H and RT-H-Joint are autoregressive on the language motion, RT-H uses distinct queries for action and language motion ("What *motion* ..." vs. "What *action* ..., given motion ..."), whereas RT-H-Joint has just one query ("What *motion and action* ..."). More specifically, RT-H passes the language motion to the Encoder in the action query, while RT-H-Joint treats the language motion as a Decoder input when predicting the action. Thus, we expect this to perform comparably to RT-H.
- **RT-H-Cluster** is an ablation of the automated language motion labeling procedure, which instead clusters actions directly using K-means [55] into a set of classes with integer labels. These class labels are used in place of the automatically labeled language motions in RT-H.
- **RT-H-OneHot** ablates the use of language to represent motions in RT-H by replacing each unique language motion with an integer class label.
- **RT-2** is a flat model that does not use any action hierarchy [4].
- **RT-H + Human Intervention** involves having a human correct only the language motions during execution, but still using the action query from RT-H (top right of Fig. 2). RT-H + Human Intervention is a variant of our method that enables humans to intervene so we expect it to be an upperbound for the other non-intervention-based methods. We discuss this in Section V-C.
- **RT-H-Intervene** is an extension of RT-H method additionally trained on human intervention data using language motion corrections. We discuss this in Section V-C.
- **RT-H-InterveneAction** is an ablation of RT-H-Intervene method that trains on *both* the action corrections and language motion corrections from human intervention data. We discuss this in Section V-C.
- **RT-2-IWR** is the interactive version of RT-2, which is additionally trained with human interventions in the form of teleoperated demonstrations – in contrast to language motion corrections – and is compared to RT-H-Intervene in Section V-C.

Note that RT-H-Joint, RT-H-Cluster, and RT-H-OneHot are variants of RT-H that still utilize an action hierarchy. See Appendix A for exact queries and a deeper dive into each RT-H variant implementation.

In Section V-A, we first train and evaluate the performance of RT-H on a diverse multi-task dataset (**Q1**). In Section V-B, we qualitatively analyze the learned language motions across

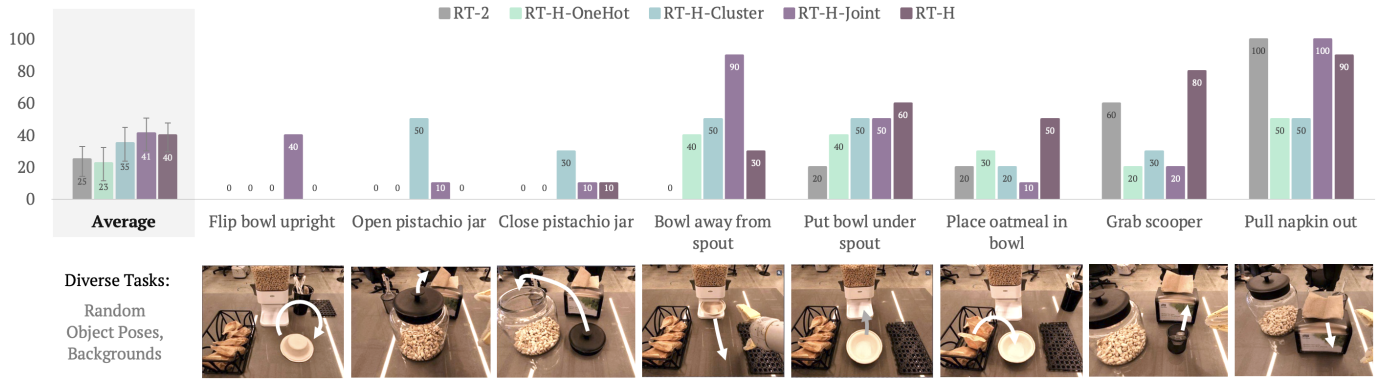



**Fig. 3:** Results on *Diverse+Kitchen* multi-task dataset, consisting of eight challenging evaluation tasks. 95% Wilson Score confidence intervals [54] are shown on the average success rates (left). RT-H outperforms RT-2 by 15% on average, getting higher performance on 6/8 of the tasks. Replacing language with class labels (RT-H-OneHot) drops performance significantly. Using action clusters via K-Means [55] instead of the automated motion labeling procedure leads to a minor drop in performance as well (RT-H-Cluster), demonstrating the utility of language motions as the intermediate action layer.

various tasks to see how language motions adapt to the context (**Q2**). In Section V-C, we collect and train on language motion corrections on top of RT-H, demonstrating that training on language motion corrections improves policy performance (**Q3**). Finally, in Section V-D we test the robustness of RT-H to variations in scenes, objects, and tasks (**Q4**).

### A. RT-H on Diverse Multi-Task Datasets

Here, we discuss how action hierarchies can improve policy performance addressing **Q1**. We will first discuss the online performance of RT-H and its variants when trained on *Diverse+Kitchen* dataset. We then present offline performance metrics to further analyze the role of language motions.

**On-Robot Performance**: Fig. 3 illustrates the performance of each method when trained on the *Diverse+Kitchen* dataset and evaluated on the 8 selected tasks within this dataset discussed earlier. Checkpoints are chosen using validation action MSE, and then run for 10 controlled trials for each task (80 total trials per method). RT-H outperforms RT-2 on most of the tasks, **surpassing RT-2 by 15% on average**, which strongly supports the benefit of action hierarchies (**Q1**), despite using no additional human annotation. See Appendix D for the success rates for each stage of each task, where we see that RT-H makes more progress towards success in 7/8 tasks. Furthermore, whereas RT-2 achieves nonzero performance on only 4/8 tasks, RT-H is nonzero on 6/8 tasks and RT-H-Joint is nonzero on all the tasks, suggesting that RT-H and RT-H-Joint are better at managing the diversity of tasks in the dataset.

**Ablations**: RT-H-Joint does comparably to RT-H, showing that RT-H is robust to the exact querying mechanism. RT-H-Cluster replaces the automating labeling procedure with action clustering, and without language it performs slightly worse than RT-H on average. Interestingly, RT-H-Cluster does better on the hardest tasks in the evaluation set (open and close pistachio jar). We hypothesize that since RT-H-Cluster uses clusters derived from the dataset, its clusters provide even more fine-grained action context than our labeling procedure, allowing it to outperform RT-H in precise tasks; however, the lack of language makes predicting clusters harder than predicting language motions when using broad datasets, leading to worse performance for RT-H-Cluster on the broader set of tasks. RT-H-OneHot replaces language motions with onehot class labels, and it performs much worse than RT-H despite being derived from the same underlying language motions. Thus, while action hierarchy itself gets us part of the way, the structure of language greatly improves language motion and action prediction.

**Offline Performance**: We investigate if language motions as an intermediate layer for action prediction has any noticeable effect by comparing the offline validation mean squared error (MSE) for end-to-end action prediction across RT-H and its joint variant RT-H-Joint vs. the flat RT-2 model (**Q1**). The end-to-end MSE reflects how well each model learns action prediction. For RT-H, we also study the action validation MSE when using the *ground truth* (GT) language motion that was labeled in the data as input to the action query ($\pi_l$), rather than inputting the inferred language motion from the language motion query ($\pi_h$). This ground truth MSE reflects how informative the true language motion is for predicting the actions. In Table I, we report the minimum MSE across training checkpoints for RT-H, RT-H-Joint, and RT-2 when trained on either the *Diverse+Kitchen* dataset or the *Kitchen* dataset. RT-H has roughly a **20% lower MSE than RT-2**, and RT-H-Joint has a **5-10% lower MSE than RT-2**, demonstrating that action hierarchies help improve action prediction offline in large multi-task datasets. Using two queries (RT-H) instead of one (RT-H-Joint) also seems to improve action prediction, which could stem from how the language motion gets passed into the model (through the encoder for RT-H vs. through the decoder for RT-H-Joint). RT-H (GT) uses the ground truth MSE metric, and we find the gap with the end to end MSE is 40%, illustrating that the correct labeled language motions are highly informative for predicting actions.

| Train Dataset | Eval Dataset | RT-2 | RT-H-Joint | RT-H | RT-H (GT) |
|---|---|---|---|---|---|
| *Kitchen* | *Kitchen* | 30.2 | 28.22 | **24.9** | 17.9 |
| *D+K* | *Diverse* | 27.7 | 25.44 | **23.6** | 17.8 |

**TABLE I:** Best checkpoint Mean Squared Error (MSE) for end-to-end action prediction on the validation set for models (columns) trained on different multi-task datasets (rows). *Kitchen* refers to the data used to train RT-1 [6] and RT-2 [4] (70K demonstrations), *Diverse+Kitchen* (*D+K*) refers to a combination of *Kitchen* and the more complex set of tasks (30K demonstrations). We also report the MSE of using the *ground truth* language motion (the labeled language motion) for the action query in RT-H (GT) rather than the inferred language motion from the language motion query. RT-H and RT-H-Joint achieve lower MSE on both datasets compared to RT-2, illustrating the benefits of action hierarchies for ingesting multi-task datasets compared to flat models like RT-2. Also, RT-H has lower MSE than RT-H-Joint.

### B. Contextual & Flexible Language Motions

In this section, we analyze (1) **contextuality**: how well the actions for a single in-distribution language motion adapt to the context of the scene and the task instruction, and (2) **flexibility**: how well RT-H responds to out-of-distribution language motions.

**Contextuality**: We illustrate several examples of contextual motions taken from online evaluations of RT-H in Fig. 4. We see that the same language motions often lead to subtle changes in actions to complete the task, while still respecting the higher level language motion (**Q2**). For example, for "move arm forward" in the top left example in Fig. 4, the arm moves generally forward but also towards the object of interest, the napkin dispenser. It also slightly tilts the arm to more easily grasp the napkin. In the top right, we see that the same command leads to a slight downward angle and a rotation of the gripper to avoid colliding with the cereal dispenser. For "move arm left" in the middle left of Fig. 4, we similarly see that left implies moving the oatmeal packet precisely above the bowl, while in the middle right, left implies precise motion of the lid to latch onto the jar. It would be immensely challenging

to design a single "move arm left" primitive to capture this contextuality. We see a similar behavior for "rotate arm right" in the bottom row of Fig. 4, where in the left case the arm rotates and stays up to sit on top of the closed jar, while in the right case the arm rotates and moves down to reach the lid on the table. See Appendix D for a quantitative analysis of language motion contextuality.

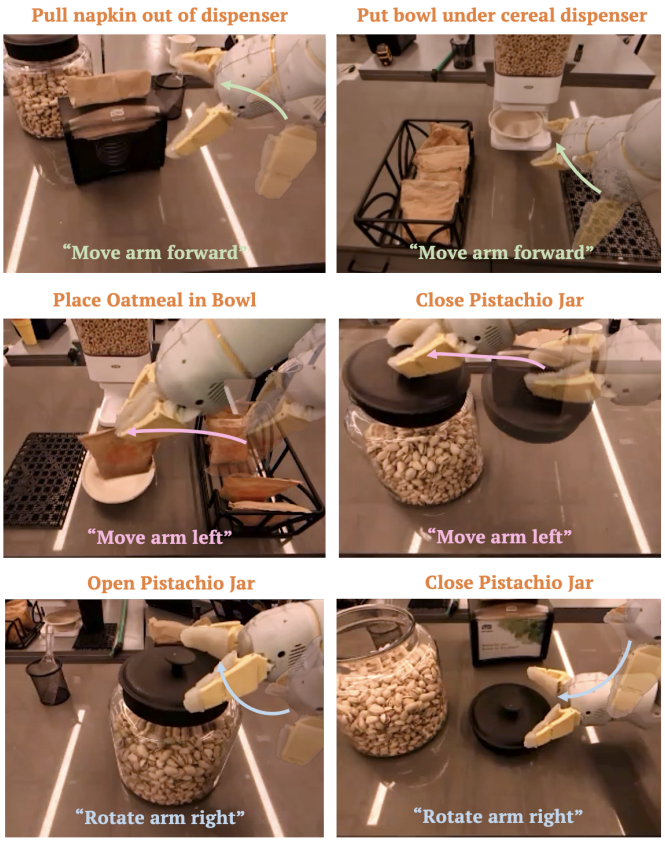
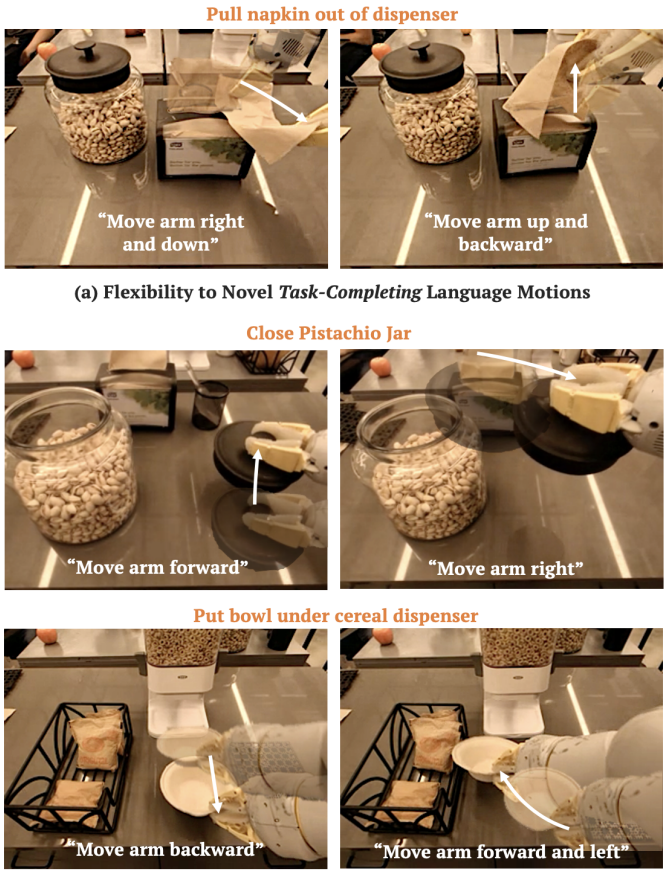



**Fig. 4:** Examples showing how language motions depend on the *context* of the scene and task, taken from online evaluations of RT-H trained on the *Diverse+Kitchen* dataset. For each row, the given language motions ("move arm forward", "move arm left", "rotate arm right") manifest with different variations (columns) depending on the task and observation, such as subtle changes in speed, non-dominant axes of movement, e.g., rotation for "move arm forward", and even gripper positions.

**Flexibility**: In Fig. 5, we demonstrate the flexibility of RT-H by intervening on language motions in RT-H online to instead perform out-of-distribution language motions for in-distribution tasks. In the first row (a), RT-H is tested with two valid ways of completing the "pull napkin" task, and we find it responds correctly to both. Despite each of the language motions demonstrated in Fig. 5 being out-of-distribution for the task, RT-H is capable of following these new language motions with ease (**Q2**). In the bottom two rows (b) of Fig. 5, we find that RT-H is also flexible to more general language motions that are not specific to the task and thus not seen in the training data. For example, in the middle right example, moving the arm away from the jar is not a common language motion for "close pistachio jar", but RT-H is still able to act correctly in response to this language motion. Being flexible




(a) Flexibility to Novel *Task-Completing* Language Motions

(b) Flexibility to *General* Language Motions

**Fig. 5:** Examples of the flexibility of learned language motions. In the top row (a) we correct RT-H using two different task-completing language motions for pulling the napkin out of the dispenser, either "right and down" or "up and backward", showing RT-H performs both correctly. For the bottom two rows (b), we still correct language motions but ask RT-H to perform a more general set of language motions for each task, demonstrating that RT-H is often flexible even to completely out-of-distribution language motions for a given task.

to more general language motions is critical for responding to a wide variety of language motion corrections, especially when the task or scene are out-of-distribution and require novel sequences of language motions.

Overall, we see that RT-H is able to maintain the flexibility and contextuality of actions while learning the high-level structure of each task through language motions (**Q2**). See Appendix D for quantitative analysis of contextuality and a qualitative look at language motion multimodality in RT-H, along with staged success rates for each method for each task. Next, we leverage these properties to collect and train on language motion corrections to improve RT-H.

### C. Training on Online Corrections

In this section we are interested in how well RT-H can learn from language motion corrections compared to methods without action hierarchy that use teleoperated correction data (**Q3**). We collect a multi-task language motion correction dataset and a teleoperated correction dataset for each of the eight tasks in Section V-A. As in prior interactive IL methods [5, 40], the

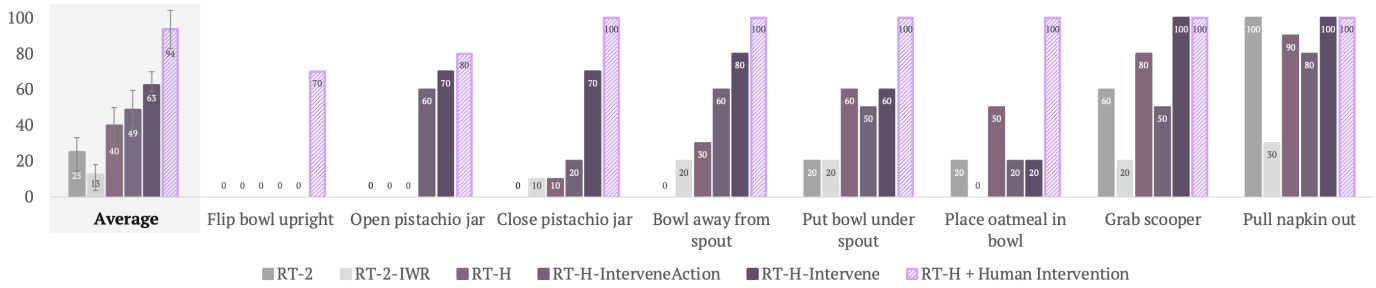



**Fig. 6:** Results for Corrections on models trained on the *Diverse+Kitchen* multi-task dataset, for the same eight evaluation tasks as in Fig. 3. 95% Wilson Score confidence intervals [54] are shown on the average success rates (left). RT-2-IWR is trained on teleoperation corrections from rolling out RT-2, while RT-H-Intervene is trained on language motion corrections from rolling out RT-H. RT-H-InterveneAction is trained on both language motion and action correction data. We see RT-H-Intervene both improves upon RT-H and substantially outperforms RT-2-IWR, suggesting that language motions are a much more sample efficient space to learn corrections than teleoperated actions. RT-H-InterveneAction performs better than RT-H, but fine-tuning actions sometimes leads to policy degeneration, since actions produced by RT-H during intervention can be suboptimal.

human decides when to correct in both datasets, usually either anticipating or responding to a task failure. Next we describe the collection and training pipelines for each method.

**RT-H-Intervene and RT-H-InterveneAction**: We collect 30 episodes (failed episodes filtered out) of language motion corrections for each of the eight tasks, using the correction procedure described in Section IV-A. The base policy used for collection is RT-H trained on the *Diverse+Kitchen* dataset (same as Section V-A). Then, we train RT-H on these on-policy corrections in the manner described in Section IV-A to produce RT-H-Intervene (only training the language motion query on the intervention data). To train RT-H-InterveneAction, we include both the language motion and action queries when training on the intervention data, at equal sampling rates.

**RT-2-IWR**: We collect 30 episodes (failed episodes filtered out) of teleoperated corrections for the same eight tasks, using VR-based teleoperation instead of language motion corrections. Since we only care about learning to correct the failure modes of RT-2, we must use RT-2 trained on the *Diverse+Kitchen* dataset (same as RT-H-Intervene) as the base policy for collection to ensure fair comparison to RT-H-Intervene. We then train RT-2 on these on-policy corrections using the Intervention Weighted Regression (IWR) method [5] to produce RT-2-IWR.

Of course, the base policy for RT-2 performs worse than *Diverse+Kitchen* on these tasks than RT-H, so to ensure a fair comparison we focus on the *change* in success rates before and after training on correction data for each method.

**Results**: We evaluate both methods in the same eight evaluation tasks as in Section V-A. In Fig. 6, we compare the performance of each method to the pre-correction models, RT-H and RT-2, respectively (duplicated from Fig. 3). We also compare RT-H + Human Intervention as an upperbound method that uses online human-in-the-loop language motion corrections when necessary, but still uses the action query in RT-H conditioned on these language motion corrections.

First, we see how amenable RT-H is to language motion corrections with RT-H + Human Intervention, which gets very high success rates even for the most precise tasks. This shows that RT-H actually does change its behavior in task-relevant ways with language motion corrections at test time. This further supports the claim in Section V-B that language motions are both flexible and contextual. In addition, this highlights that language motion prediction is often the bottleneck for performance, so we expect that refining language motion prediction through intervention will yield clear improvements.

RT-2-IWR, a state-of-the-art online imitation learning method, sees a degradation in performance from 25% to 13% on average, likely due to a combination of relatively small amounts of data per task and the use of only a single round of correction. In addition, we suspect teleoperation-based corrections are more likely to introduce action distributions that are too different from the training data (and thus the base policy). The language motion corrections in RT-H on the other hand are much more consistent with the training data because actions come from the base policy itself (under slight changes in language motion space) and thus easier to learn from.

RT-H-Intervene, on the other hand, substantially outperforms RT-2-IWR in this setting despite using the same amount of data, improving by 60-70% on the harder precise tasks (open and close pistachio jar). RT-H-Intervene regresses on just one task, "place oatmeal packet in bowl", where we observed that the robot would successfully grasp the packet very often, but would get stuck predicting "close gripper". We suspect there is a slight bias towards that language motion correction in the dataset, and it could be resolved by specifying "move arm up" as a follow up correction, or by running more rounds of correction. The oatmeal example also highlights how language motion corrections can make the policy's behavior *interpretable* and thus more intuitive to debug – more effectively allowing the designer to identify or correct the failure points.

RT-H-InterveneAction also improves upon RT-H, outperforming it by 9% on average. We suspect that compared to RT-H-Intervene, RT-H-InterveneAction suffers from policy degeneration, where new actions in the interventions bias the action distribution toward model generated actions (since we use the action query in RT-H to collect language motion correction data) rather than the true expert distribution. These model generated actions can be sub-optimal and thus can

impact performance: for example, we see that close pistachio jar task does not improve as much with RT-H-InterveneAction as with RT-H-Intervene, because the policy starts producing near-zero actions at states where the intervention-produced actions were small (e.g. after grasping the jar lid).

Overall, we see that language motion corrections bring the average success rates of RT-H **from 40% to 63% with just 30 episodes of correction** per task. By abstracting actions into the more condensed language motion space, RT-H can more quickly learn to improve itself from feedback from language motion corrections than from teleoperation corrections (**Q3**).

**Failure Modes**: RT-H demonstrates performance boosts on a wide variety of tasks, however the action hierarchy paradigm does lead to interesting failure modes. First, systemic language motion prediction errors can often confuse the action prediction, leading to oscillatory or incorrect behaviors where a flat model might not exhibit similar behaviors. For example, we occasionally noticed the robot would get stuck after closing the gripper by continuing to predict "close the gripper", likely due to subtle observation changes between "close the gripper" frames and future frames. Thankfully, these issues are easy to debug and resolve, either through adjusting the automated labeling procedure or through intervention as we have shown.

Second, when collecting language motion corrections, the human operator is limited by the performance of the underlying action prediction model. When language motion corrections do not result in the correct behavior, it might become frustrating for the operator. For example if the operator asks the robot to "move arm left" over the bowl in the middle left row of Fig. 4, but the robot overshoots the bowl, this can make the process slower than teleoperation. This failure mode rarely happens for in-distribution tasks, but as tasks diverge from the data distribution, it becomes more likely.

### D. Generalization

To evaluate **Q4**, we study three types of generalization: generalization to new scenes (with similar objects but new backgrounds and lighting), to novel objects, and to novel tasks. We use RT-H trained on only the *Kitchen* dataset [6] unless otherwise noted (i.e., not including the *Diverse* data), which consists of the following training and evaluation tasks on various objects:

1) "knock over"
2) "drawer place"
3) "move"
4) "pick"
5) "open / close drawer"
6) "place upright"

**Generalization to New Scenes**: We evaluate each task in the *Kitchen* dataset in new environments, specifically in a new building consisting of varying lighting, and diverse backgrounds and floors. In Fig. 7, we see that RT-H and RT-H-Joint are more robust to changes in scenes, with especially large deltas for the hardest tasks of "place upright" task and "open / close drawer" tasks (**Q4**).

**Generalization to New Objects**: We evaluate "pick" and "move" under object generalization, using 50 evaluations of objects unseen during training such as pears, coconut water, and oreos. In Table II, we find that RT-H achieves 65% on these tasks, whereas RT-2 gets 55%. As shown in Appendix D, RT-H also progresses farther in each task (in terms of stages of each task) compared to RT-2 on average (**Q4**).

| | pick | move | **Average** |
|---|---|---|---|
| RT-2 | 60 | 50 | 55 |
| RT-H | 70 | 60 | 65 |

**TABLE II:** We evaluate RT-2 and RT-H trained on *Kitchen* data [6] on the "pick" and "move" tasks but under novel objects for 50 scenarios total. RT-H outperforms RT-2, demonstrating that action hierarchy helps the policy generalize to novel objects.

**Generalization to New Tasks with Limited Corrections**: While zero-shot success on out-of-distribution tasks is quite difficult, in Fig. 8, we qualitatively demonstrate that even for unseen tasks, RT-H requires just a few well-timed corrections to succeed at the task. For these examples, we use the version of RT-H trained on the *Diverse+Kitchen* dataset to provide RT-H with language motions demonstrated in a wide variety of contexts. Fig. 8 also shows the shared structure between seemingly diverse tasks: each of these tasks require some picking behavior to begin the task, and by learning the shared structure of language motions across many diverse tasks, RT-H can complete the picking stage without any correction (**Q4**). Even when RT-H is no longer able to generalize its language motion prediction, we see that language motion corrections often do generalize, allowing us to successfully complete the task with just a few corrections (**Q2**, **Q4**). This demonstrates the potential of language motions for scaling up data collection for novel tasks.

## VI. Conclusion

In this work, we introduce RT-H, which leverages language motions like "move arm forward" as an intermediate prediction layer between the high-level task and the low-level action. RT-H trains to map tasks described in language into language motions, and then uses the inferred language motion to predict the action, where both steps are conditioned on visual input and the task. We label language motions using an automated procedure that scales to a wide variety of tasks at no human labeling cost. We instantiate RT-H using a single transformer model like RT-2 [4], where both the action and language motion queries are co-trained with a vast amount of internet-scale data. RT-H (1) enables more data sharing between different tasks by learning the shared task structure across seemingly disparate tasks, and thus is more capable of ingesting multi-task datasets at scale, and (2) is amenable to language motion corrections that change the underlying behaviors within the context of the scene and task. In our experiments, we show that RT-H outperforms RT-2 and action hierarchy ablations on diverse multi-task data. Then we show that RT-2 is highly correctable in language motion space even for unseen language motions, and that learning from

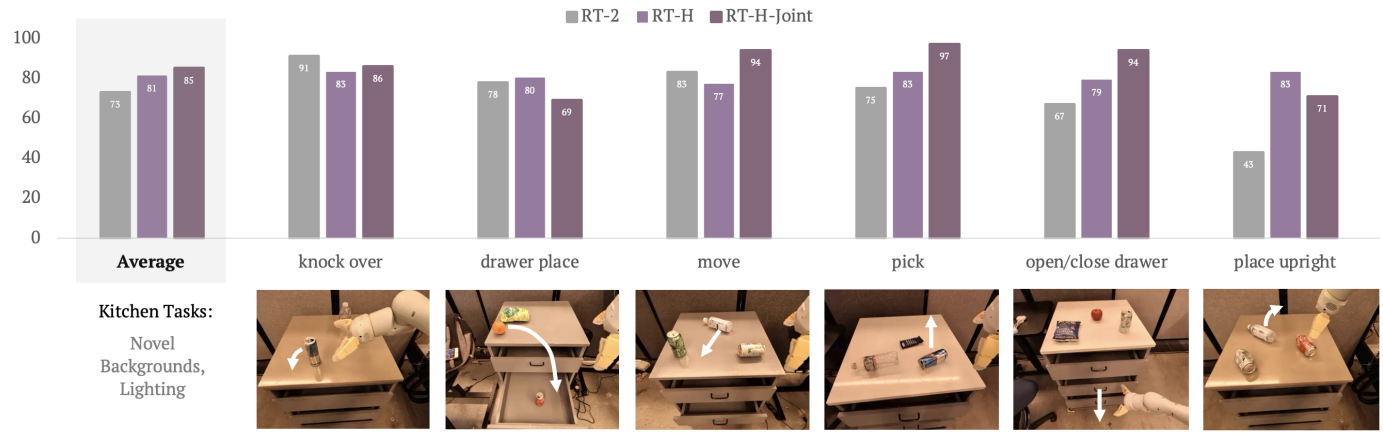



**Fig. 7:** Results when models trained on *Kitchen* data [6] are deployed on the same tasks, but in a new building with novel backgrounds, lighting, and flooring. RT-H and RT-H-Joint each outperform RT-2, suggesting that the use of action hierarchy helps the policy generalize to novel scenes. RT-2 struggles particularly with placing upright and opening and closing the drawers in these new scenes.

these language motion corrections outperforms learning from teleoperation-based corrections. Finally, we show that RT-H is more robust to scene and object variations compared to RT-2. These results show the promise of action hierarchies using language, and we believe RT-H provides a strong foundation on which to scale up data collection and robot learning.

**Limitations & Future Work**: RT-H opens several exciting avenues for future work. First, we test RT-H on a large and diverse datasets, achieving state-of-the-art performance, but the absolute success rates still leave room for improvement, even after training on corrections. We believe that, as evidenced by the more sample efficient language motion corrections of RT-H, future work should scale up both the offline datasets and correction pipeline – language motions could even be used to help bridge datasets with many different embodiments like OXE [56], or even to learn from human videos with actions described only in language.

Second, although we ablate different action hierarchies in Section V-A, future work is needed determine the best abstraction level for the intermediate layers (e.g., using object-referential language vs. our language motions). Additionally, we primarily test only one intermediate action layer in this work, or just one step of action reasoning, but future work might define multiple steps of action reasoning instead, where the user can intervene at any level of abstraction. For example, we might add a *task* prediction level to go from long horizon instructions like "clean the room" to individual tasks like "pick coke can", before mapping the task to language motions and then actions. To decompose corrections at any level of the hierarchy, one might even teach the model to automatically *locate* a correction in the hierarchy, and then autoregressively predict lower level actions until it deems it has reached the robot action level.

Third, language motions represent a contextual and compressed space in which to predict actions. One might leverage this motion contextuality in RT-H to greatly compress the action space for reinforcement learning methods and policy exploration, possibly leveraging language motion prediction as a prior. We suspect language motions will provide more meaningful degrees of exploration and more sample efficient policy learning, while also being highly interpretable and correctable to humans. Even in imitation learning, several works have shown the important of action consistency across demonstrators [57, 58], and we posit that using language motions as a compressed action space could lead to more consistent actions and thus more sample-efficient policy learning.

## Acknowledgments

We would like to thank Grecia Salazar, Deeksha Manjunath, Clayton Tan, Yansong Pang, Jornell Quimbao, Tran Pham, Utsav Malla, April Zitkovich, and Elio Prado for their help with dataset collection and on robot evaluation. We would also like to thank the greater Google DeepMind team for their feedback and contributions.

## References

[1] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

[2] Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. No, to the right: Online language corrections for robotic manipulation via shared autonomy. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '23, page 93–101, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399647. doi: 10.1145/3568162.3578623. URL https://doi.org/10.1145/3568162.3578623.

[3] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. Correcting robot plans with natural language feedback. *ArXiv*, abs/2204.05186, 2022. URL https://api.semanticscholar.org/CorpusID:248085271.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding,


**Fig. 8:** We show the generalization capabilities of RT-H with completely unseen tasks with minimal correction. By breaking down tasks into language motions, RT-H learns the shared structure between seemingly diverse tasks. This allows it to generalize language motions to new tasks, as shown in the first part of each task, where RT-H performs the picking phases easily. We also show that when RT-H cannot zero-shot generalize, language motion corrections often do generalize, allowing it to complete these tasks with just a few well-timed corrections.

Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

[5] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation. *CoRR*, abs/2012.06733, 2020. URL https://arxiv.org/abs/2012.06733.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yev-

gen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8kbp23tSGYv.

[8] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13139–13150. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9909794d52985cbc5d95c26e31125d1a-Paper.pdf.

[9] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[10] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

[11] Priya Sundaresan, Suneel Belkhale, Dorsa Sadigh, and Jeannette Bohg. KITE: Keypoint-conditioned policies for semantic manipulation. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=veGdf4L4Xz.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

[13] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, pages 892–909. PMLR, 2023.

[14] Siddharth Karamcheti, Suraj Nair, Annie S. Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. In *Robotics: Science and Systems (RSS)*, 2023.

[15] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2022.

[16] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023.

[17] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012. doi: 10.1177/0278364911428653. URL https://doi.org/10.1177/0278364911428653.

[18] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246, 2012. doi: 10.1109/IROS.2012.6386006.

[19] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on robot learning*, pages 418–437. PMLR, 2017.

[20] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020.

[21] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.

[22] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgHY0NYwr.

[23] Daniel Tanneberg, Kai Ploeger, Elmar Rueckert, and Jan Peters. Skid raw: Skill discovery from raw trajectories. *IEEE robotics and automation letters*, 6(3):4696–4703, 2021.

[24] Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

[25] Kourosh Hakhamaneshi, Ruihan Zhao, Albert Zhan, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. In *International Conference on Learning Representations*, 2021.

[26] Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

[27] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In Leslie Pack

Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1113–1132. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/lynch20a.html.

[28] Suneel Belkhale and Dorsa Sadigh. PLATO: Predicting latent affordances through object-centric play. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=UAA5bNospA0.

[29] Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 4613–4619. IEEE, 2021.

[30] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. In *Conference on Robot Learning*, pages 2113–2133. PMLR, 2023.

[31] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, pages 1769–1782. PMLR, 2023.

[32] Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J Joshi, Pete Florence, Wei Han, Robert Baruch, Yao Lu, Suvir Mirchandani, Peng Xu, Pannag Sanketi, Karol Hausman, Izhak Shafran, Brian Ichter, and Yuan Cao. Robovqa: Multimodal long-horizon reasoning for robotics. In *arXiv preprint arXiv:2311.00899*, 2023.

[33] Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. ELLA: Exploration through learned language abstraction. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=VvUldGZ3izR.

[34] Joey Hejna, Pieter Abbeel, and Lerrel Pinto. Improving long-horizon imitation through instruction prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7857–7865, 2023.

[35] Shengran Hu and Jeff Clune. Thought Cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 2023.

[36] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1713–1726, 2022.

[37] Rachel Ma, Lyndon Lam, Benjamin A Spiegel, Aditya Ganeshan, Roma Patel, Ben Abbatematteo, David Paulius, Stefanie Tellex, and George Konidaris. Skill generalization with verbs. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5844–5851. IEEE, 2023.

[38] Xinjie Liu. Interactive imitation learning in robotics based on simulations, 2022.

[39] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL http://arxiv.org/abs/1011.0686.

[40] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[41] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. In *Conference on Robot Learning*, pages 598–608. PMLR, 2022.

[42] Ryan Hoque, Ashwin Balakrishna, Carl Putterman, Michael Luo, Daniel S. Brown, Daniel Seita, Brijen Thananjeyan, Ellen R. Novoseller, and Ken Goldberg. Lazydagger: Reducing context switching in interactive imitation learning. In *CASE*, pages 502–509, 2021. URL https://doi.org/10.1109/CASE49439.2021.9551469.

[43] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2891–2897. AAAI Press, 2017.

[44] Kunal Menda, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048, 2019. doi: 10.1109/IROS40897.2019.8968287.

[45] Mengxi Li, Alper Canberk, Dylan P Losey, and Dorsa Sadigh. Learning human objectives from sequences of physical corrections. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2877–2883. IEEE, 2021.

[46] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. Physical interaction as communication: Learning robot objectives online from human corrections. *The International Journal of Robotics Research*, 41(1): 20–44, 2022.

[47] Lihan Zha, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[48] Alexander Broad, Jacob Arkin, Nathan Ratliff, Thomas Howard, and Brenna Argall. Real-time natural language corrections for assistive robotic manipulators. *The International Journal of Robotics Research*, 36(5-7):684–698, 2017.

[49] Arthur Bucker, Luis Figueredo, Sami Haddadinl, Ashish Kapoor, Shuang Ma, and Rogerio Bonatti. Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 978–984. IEEE,

2022.

[50] Arthur Bucker, Luis Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, Sai Vemprala, and Rogerio Bonatti. Latte: Language trajectory transformer, 2022.

[51] John D Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, Jacob Andreas, John DeNero, Pieter Abbeel, and Sergey Levine. Guiding policies with language via meta-learning. In *International Conference on Learning Representations*, 2018.

[52] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, pages 1–8, 2023. doi: 10.1109/LRA.2023.3295255.

[53] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.

[54] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. ISSN 01621459. URL http://www.jstor.org/stable/2276774.

[55] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489.

[56] Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[57] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=JrsfBJtDFdI.

[58] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=FwmvbuDiMk.

[59] Gpt-4v(ision) system card. 2023. URL https://api.semanticscholar.org/CorpusID:263218031.

[60] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

## APPENDIX

We first outline the implementation of RT-H and ablations in Appendix A, along with the training recipes. Then we discuss implementations and training protocols for methods using corrections in Appendix B. After, we detail each of the datasets used in Appendix C. Then, we show more detailed results in Appendix D, including success rates for different stages of each task for each method, quantitative analysis of contextuality in RT-H, qualitative analysis of the language motion multimodality of RT-H. Finally we cover some frequently asked questions about RT-H.

### A. Method Implementations

As described in Section III, RT-H and RT-2 [4] are implemented using a Pali-X 55B Multimodal Encoder Decoder Transformer architecture [53]. Images are encoded using a 22B ViT architecture, which is learned during the Pali-X pre-training phase but fixed during robot demonstration data co-training. The encoded images and the prompt are passed through the Encoder, and the output for each query is autoregressively decoded with the Decoder. Next we describe each method (offline only) in detail.

**RT-H**: RT-H first predicts language motions from the task using the following **language motion query**: *Q: What skill should the robot do to [task]? A:* , where the task is specified in language, and the resulting language motion (skill) is returned in language. Then, it uses the predicted language motion to better inform action prediction using the following **action query**: *Q: What action should the robot do to [task], with current skill: [motion]? A:* , where the output is the tokenized action string. RT-H trains in the same fashion as RT-2 [4]. First, we pretrain RT-H on the same large vision language dataset as RT-2, and we then co-train the language motion and action queries using 50% of the overall training samples (25% each), along with 50% using the original pre-training mixture. Similar to RT-2, we use a learning rate of 1e-3 and with a constant linear warmup and square root normalize decay, and a batch size of 1024.

**RT-H-Joint**: Unlike RT-H, RT-H-Joint uses a single query to predict both language motion and action. While both methods are autoregressive on the language motion, RT-H has two queries which using different wordings to indicate language motion or action prediction, and RT-H also passes in the language motion to the encoder for the action query since it is part of the prompt string. The prompt for RT-H-Joint is as follows: *Q: What skill and action should the robot do to [task]? A:* . Then the output is a concatenation of first language motion (skill) in language, and then the tokenized action string, in the form *skill: [skill], action: [action]*. RT-H-Joint is trained identically to RT-2 and RT-H as well, but with the joint language motion and action query instead of action prediction from RT-2.

**RT-H-Cluster**: RT-H-Cluster follows the same two query procedure and training implementation as RT-H. In order to determine the action clusters, we first normalize the actions in the dataset using dataset statistics. Then, we cluster the actions using K-means [55] using 256 cluster centers. We chose this number to be on par with the number of actively used language motions in the dataset from our automated labeling procedure. Then, the cluster centers are replaced with integers from 0 to 255, and used in place of language motions in the action hierarchy. This ablation tests the utility of language motions compared to embeddings tuned to the specific actions in the datasets.

**RT-H-OneHot**: RT-H-OneHot also follows the same two query procedure and training implementation as RT-H. The only change is to replace unique language motions with integers. We first enumerate skills in order of how common they are, and then assign a unique integer value to each. Importantly, this formulation does not capture the inherent structure of language: for example, "move arm forward" and "move arm forward and left" are similar in many ways and should be treated as such, but their replaced one-hot labels will likely be as equidistant as any other two random language motions. Thus, RT-H-OneHot tests the importance of the structure of language when predicting language motions.

### B. Corrections

As described in Section IV-A, RT-H enables humans to intervene with new language motions, and then these corrections can be deployed on robot. To train RT-H on corrections, we can directly type or say language motion corrections that will be passed directly into the action query in place of the inferred language motion from the language motion query. This shifts the burden of correction up one level in the action hierarchy, from actions to language motions. We collect the dataset of language motion corrections, recording the observations, task, and language motion corrections, and then co-train our model with the original pre-training dataset, the robot demonstration dataset, and upweighted language motion corrections. For such a large demonstration dataset, we aim for each correction sample to be seen 50x as often as a corresponding demonstration dataset example. Thus the sampling weights during training on the *Diverse+Kitchen* dataset are as follows:

- Pre-training Queries: 50%
- Demonstration Data language motion Query: 23%
- Demonstration Data Action Query: 23%
- Correction Data language motion Query: 4%

Given that the ratio of the demonstration dataset size to the language motion correction dataset size is roughly 300:1, this corresponds to upweighting each language motion correction sample by 50:1.

We use the same recipe for training from teleoperated corrections with RT-2-IWR. The only difference is that the language motion training queries are replaced with action queries like in IWR.

### C. Datasets

We use two datasets in this work, the *Kitchen* dataset from RT-1 [6] and RT-2 [4], and the new *Diverse+Kitchen* dataset (which is an extended version of *Kitchen*). *Kitchen* consists of the **6 semantic tasks used for evaluation in 70K demonstrations**, across several common object categories like

cans, bottles, and fruits, forming 542 unique instructions. The semantic task instructions are as follows:

- **knock over**: Knock `Object` Over.
- **drawer place**: Pick `Object` from drawer and place it on counter.
- **move**: Move `Object` Near `Object`.
- **pick**: Pick `Object`.
- **open / close drawer**: [Open / Close] [Top / Middle / Bottom] Drawer.
- **place upright**: Place `Object` Upright.

*Diverse+Kitchen* consists of all the demonstrations from *Kitchen*, but with **24 more semantic tasks in only 30K additional demonstrations**, with 165 unique instructions. The new task instructions in the dataset are as follows, sorted by frequency (most to least):

- pull napkin out of dispenser and place napkin flat on counter
- **pull napkin out of dispenser**
- **pick a bowl and place the bowl upright on counter**
- **close the large glass jar containing pistachios using the lid on counter**
- pick a cup and place the cup upright on counter
- **open the large glass jar with pistachios**
- **grab a scooper**
- pick up the scoop from the basket
- open the large glass jar with pistachios and place the lid on counter
- place the scoop inside the basket
- **put a bowl under the cereal dispenser spout**
- **move the bowl away from underneath the spout**
- **pick an oatmeal packet and place the oatmeal packet in the bowl**
- pick up spoon and place spoon in bowl with cereal
- swivel the cereal dispenser until the bowl is half full
- pick up the tong from the basket
- place the tong inside the basket
- pour the snack from the scoop into the cup
- scoop the snack from the jar
- pick `object`
- move `object` near `object`
- knock `object` over
- place `object` upright
- squeeze honey into the bowl

This represents a diverse range of behaviors for the robot, often with huge data imbalances between tasks. Note that there are additional demonstrations for the knock over, move, pick, and place tasks in this dataset, although these comprise a small fraction of the overall data. The tasks used for evaluation in Section V-A and Section V-C are bolded.

### *D. Detailed Results*

**Diverse Evaluations**: Next we show the cumulative success rates for different stages of each task in the *Diverse+Kitchen* evaluations from Section V-A. Fig. 9 shows the "Place Bowl Upright" task, and RT-H and RT-H-Joint are able to pick up the bowl 50% of the time (compared to 20% for RT-2), but RT-H struggles to rotate the bowl afterwards. Fig. 10 shows the "Open Pistachio Jar" task, where we see that methods with action hierarchy get substantially farther than RT-2 on this task. Fig. 11 shows the "Close Pistachio Jar" task, where once again RT-2 rarely exhibits the correct behavior compared to methods with action hierarchy. Thus even though success rates for all methods are fairly low on the open and close jar tasks, we see that RT-H and its variants are able to progress much farther. Fig. 12 shows the "Move Bowl Away" task, where we see once again that methods with action hierarchy get much farther in the task than RT-2. Here, we can see that RT-H struggles to grasp the thin rim of the bowl, compared to RT-H-Joint which has high success with grasping. Fig. 13 shows the "Put Bowl Under" task, where once again RT-H and other action hierarchy methods do better on each stage of the task, with RT-H getting the highest final success rate. Fig. 14 shows the "Place oatmeal in bowl" task, and RT-H and RT-H-Joint get much farther in the task compared to RT-2, RT-H-OneHot, and Fig. 15 shows the "Grab Scooper" task, and it is one of the few tasks where RT-2 does better than some action hierarchy methods (RT-H-Cluster and RT-H-OneHot), but RT-H outperforms RT-2 on all stages of the task. In Fig. 16, we show the "Pull napkin out" task, and RT-H, RT-H-Joint, and RT-2 all get very high success rates.

Overall, we see that in many cases, there are only one or two stages of the task that require correction. This often only requires a few language corrections, which provides insight as to why RT-H-Intervene can improve task performance with so little new data.

**Generalization**: Next we show the staged cumulative success rates for RT-H generalizing to novel objects, as shown in Section V-D and Table II. Fig. 17 shows the pick task and Fig. 18 shows the move task, and in both tasks we see that RT-H does better not just in final success rate but also in each individual stage of each task.

**Contextuality**: To highlight the contextuality of RT-H quantitatively, we compute in Table III the mean (and standard deviation) of each action dimension for actions that belong to the same language motion group. We use the validation set of the *Diverse+Kitchen* dataset (using the automated language motion labeling procedure from Section III-C) to compute these statistics. We find that even though the dominant action dimension for each language motion has the largest mean and action variance (bold in Table III), other action dimensions also have nontrivial variance, suggesting that the interpretation of each language motion changes with the scene and the task. In other words, translating language motion to action (action query) is a contextual process. Sometimes, the mean of the non-dominant action axis is also nontrivial (e.g., for rotate arm right, the arm has some arm (x) and (y) bias), which is likely due to bias from the chosen set of tasks in the dataset.

**Multimodality**: Next, we study if the language motion abstraction has enabled RT-H to learn not just the correct language motion at each step, but also diverse ways of accomplishing the same task. To analyze this qualitatively, we run the language motion query on offline validation data

with beam search to output the top three language motions for images in the dataset. We show four examples of this in Fig. 19. In the first row (examples (a) and (b)), RT-H predicts language motions that differ slightly from each other but in task-contextual ways (e.g., move arm forward vs. move arm down and forward, both are accurate for the task). In the second row (examples (c) and (d)), RT-H predicts language motions that are quite different from each other (e.g., move arm left vs. close gripper), but despite the variety, each language motion is reasonable given the context of the scene and task. This shows that RT-H can capture the multimodality of behaviors for tasks from the data. In fact, this ability to represent multiple high level behaviors could be how RT-H is so efficient in learning from language motion intervention data – an intervened language motion might be quite likely already in the model, and so updating the model to predict the new language motion might be a trivial change to the model. Additionally, this language motion multimodality could be quite useful for exploration in a reinforcement learning context (e.g., sampling language motions from the model instead of actions from some uniform distribution).

### *E. Frequently Asked Questions*

**Why does RT-H outperform RT-2 conceptually, despite using the same observations and low-level actions?**
Predicting an intermediate action abstraction *structures* the action prediction problem into two stages, each simpler to learn than the combined objective. Learning the relationship between each image, task and each low-level action is a very high-dimensional mapping to learn, especially in multi-task datasets. Language motions provide a bottleneck that reduces that dimensionality for each stage of action prediction. Predicting language motions from the task is much simpler than predicting each action for each dimension directly, and thus leaves less room for overfitting. In addition, language motions learned from multi-task datasets enable the possibility of transfer across the different datasets. For example, we might not expect any transfer between two distinct tasks such as "picking up a cup" vs "pouring to a cup", but both tasks would potentially share a common language motion of "moving forward" enabling transfer of these low-level motions across tasks/datasets, and potentially allowing for more generalization beyond flat models such as RT-2. Finally, predicting actions from language motions and the task is also much simpler, since the language motion narrows down the space of valid actions for the model to predict. This action "reasoning" through multiple steps is analogous to why LLMs do better at step-by-step reasoning compared to direct prediction.

**How does the action hierarchy in RT-H compare to LLM planning methods like SayCan [1]?**
The action hierarchies in SayCan and other LLM planning frameworks address a fundamentally different problem: long-horizon instruction following. They often start with a long-horizon **instruction** (e.g. "bring me a cold drink") that breaks down to medium horizon **tasks** (e.g. "open the fridge" + "pick up a coke can" + "close the fridge" + "bring the coke can to the table") via an LLM or task planner, and the medium horizon task, e.g. "pick up a coke can" relies on existing pretrained primitives and often an affordance value function to shape the LLM predictions. In RT-H, we instead learn action hierarchies from medium horizon **tasks** (e.g. "pick up a coke can") to short horizon **motions** (e.g. "move arm forward").

Therefore, the closest analog to an approach like SayCan in our setting would be using predefined language motion primitives (e.g., hardcoded versions of "move arm forward" or "close gripper") We did try to implement this idea, but we found it was impractical for a few reasons:

1) There is significant *contextuality* of language motions required when solving precise manipulation tasks (see Fig. 4, e.g., the speed or direction variety for a single language motion) – there was no single predefined primitive for many language motions that could safely and efficiently progress at the task. See Appendix D for a quantitative analysis of the contextuality of each major language motion in the dataset. We find that each language motion has nontrivial variation in multiple action dimensions, not just the major action dimension.
2) LLMs would inherently struggle to predict language motions because they are not grounded in the visual context of the scene. Therefore we would not expect these models to understand directions like "left" and "up" or to know when to close the gripper with just a textual description of the scene (as provided in SayCan). Thus VLMs are much better suited for this task.

**Can large VLMs be used to directly predict language motions without the need for automated labeling?**
We explored this idea with state of the art VLMs like GPT-4V [59], but were unable to get reasonable language motion outputs. This is likely because current VLMs do not have strong spatial priors on robot behaviors (i.e., they are not grounded well in concepts like "left" or "forward"). More recent work like PIVOT [60] show that VLMs do better with visual spatial information (i.e., arrows drawn on the image), but there are still long term questions about how to incorporate rotations or gripper actions under these frameworks. We show that our automated labeling procedure provides a robust and scalable way to teach VLMs like RT-H these spatial concepts. As VLMs gain more and more knowledge of the physical world, we are excited to see how they improve at predicting language motions zero-shot.

**Have we considered scenarios where the task involves tool use, and how language motions can evolve to describe them?**
As we noted in the future work section, we hope to expand language motions to include object-referential language like "rotate the screw" or "grasp the pot handle". We believe this will unlock a whole new set of capabilities and types of corrections. However the main challenge is how to get these types of language motions without expensive human annotation. One idea is to label a fraction of the data with object-referential language, and label the rest of the data using

a model trained on this dataset or through self-supervised techniques. As VLMs become more powerful, it might even be easier for VLMs to zero-shot provide these object-referential language motions than the spatial language motions we use in RT-H.

**Does co-training on intervention data lead to model collapse, where the model worsens at tasks not present in the intervention data?**
We did not notice any model collapse in practice – specifically, we noticed that offline metrics on different splits of the data remained stable after adding in the intervention data. Likely due to the immense parameter counts, RT-H and other large VLMs seem very capable of integrating new data under co-training schemes.

**Does the asynchronous language motion inference in RT-H hurt policy performance?**
Inference time greatly increases for synchronous inference, and unfortunately this can make online evaluation of the synchronous procedure quite difficult. Instead, we can turn to the offline metrics in Table I, where we see RT-H has a lower MSE (this uses the asynchronous inference procedure but offline) than RT-H-Joint and RT-2. As we note in the paper, the comparison between RT-H and RT-H-Joint can also be seen as a comparison between asynchronous and synchronous inference, and there we see fairly minor performance differences between the two (if any).

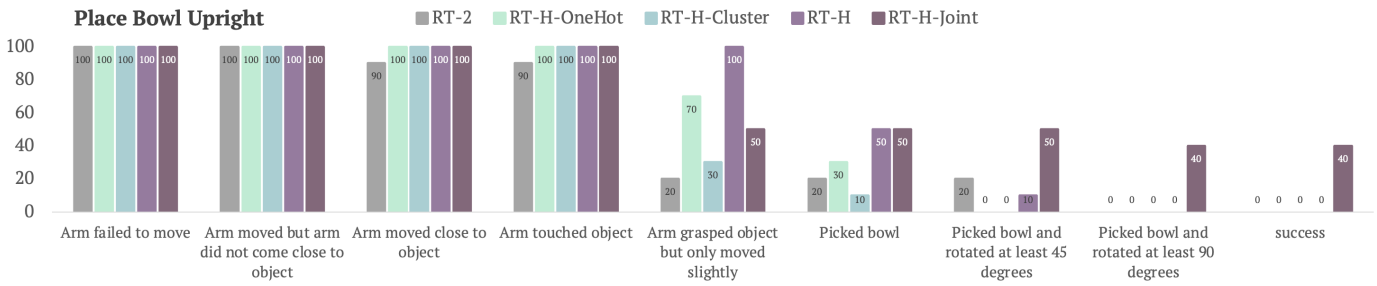



**Fig. 9: Place Bowl Upright on Counter**: Cumulative success rates for each method.

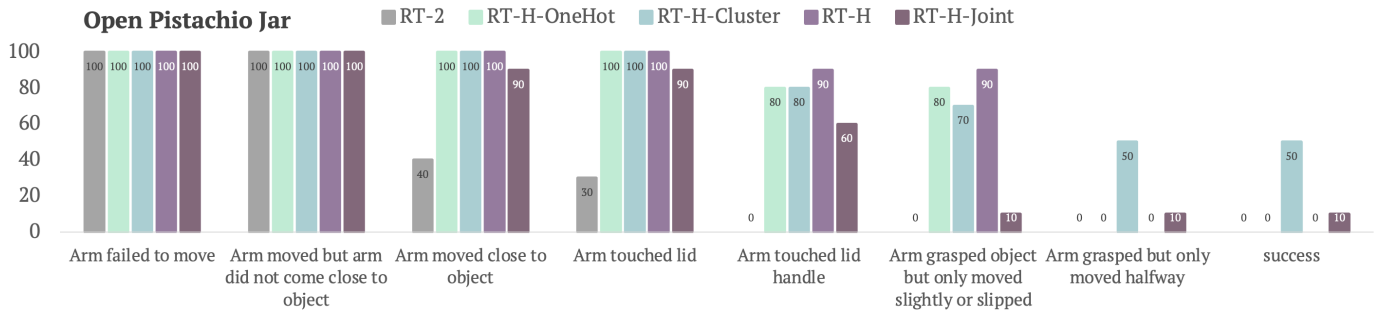



**Fig. 10: Open Pistachio Jar**: Cumulative success rates for each method.

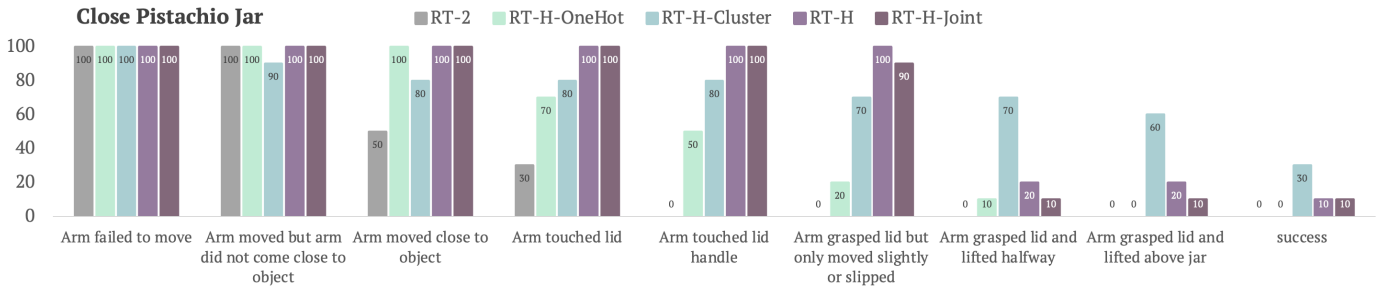



**Fig. 11: Close Pistachio Jar**: Cumulative success rates for each method.

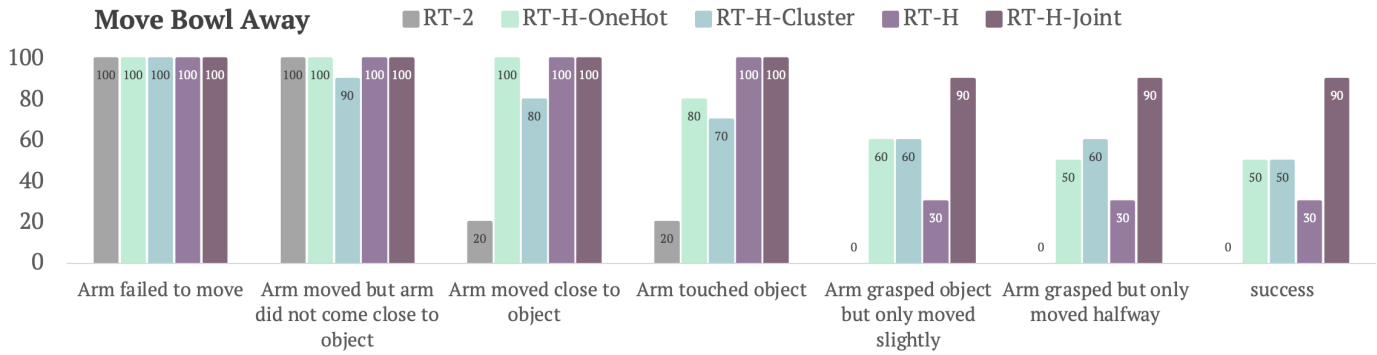



**Fig. 12: Move Bowl Away from Cereal Dispenser**: Cumulative success rates for each method.

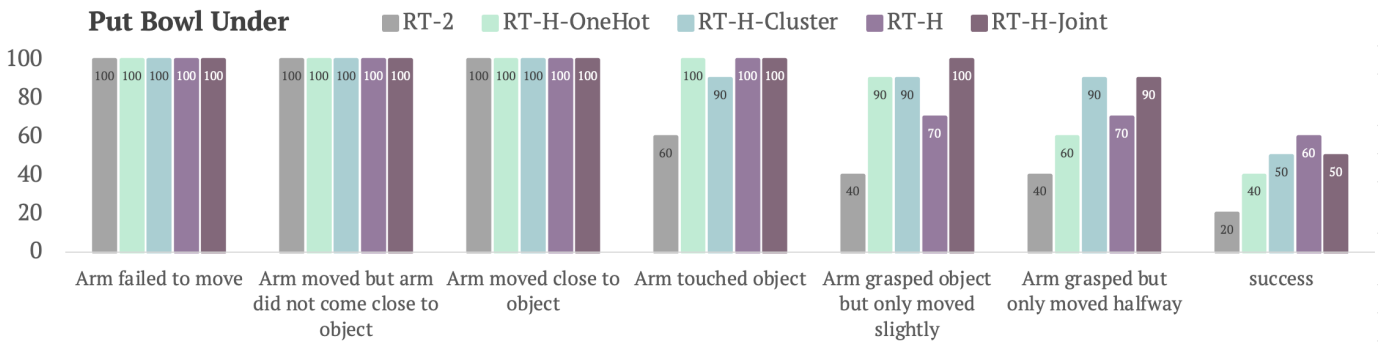



**Fig. 13: Put Bowl Under Cereal Dispenser**: Cumulative success rates for each method.

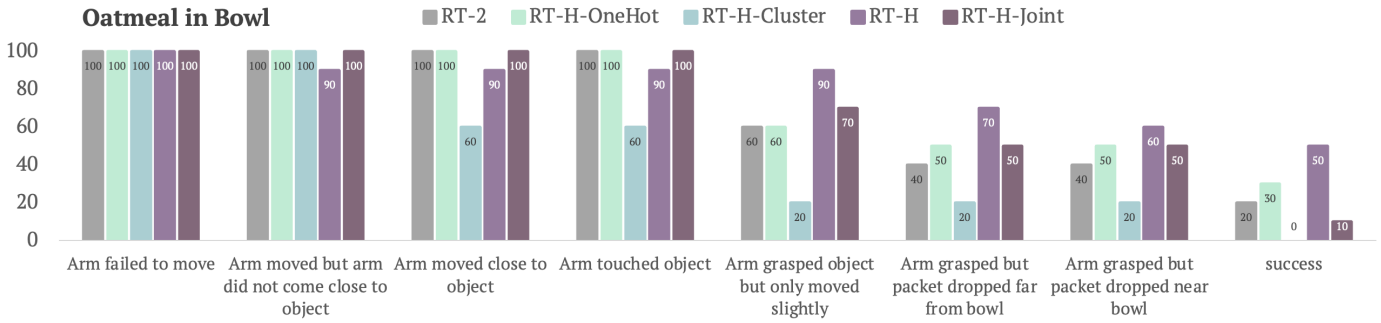



**Fig. 14: Place Oatmeal Packet in Bowl**: Cumulative success rates for each method.

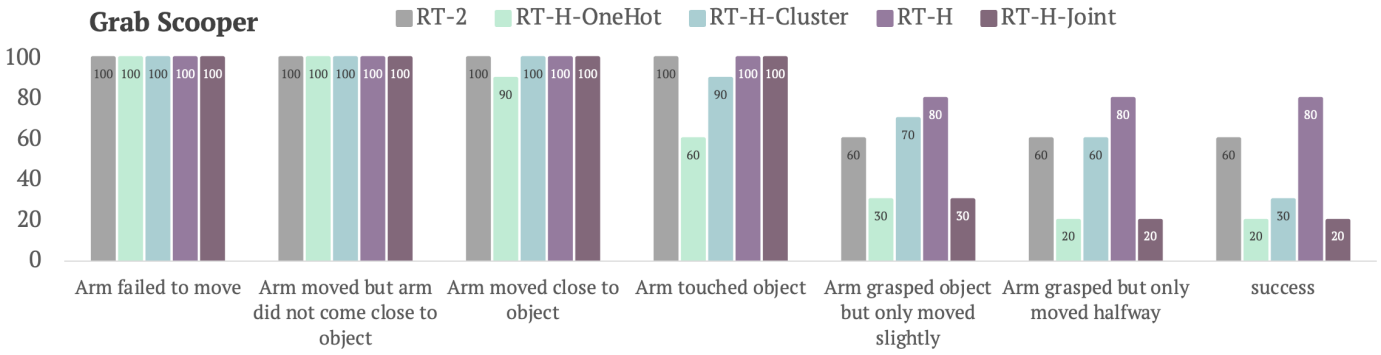



**Fig. 15: Grab a Scooper**: Cumulative success rates for each method.

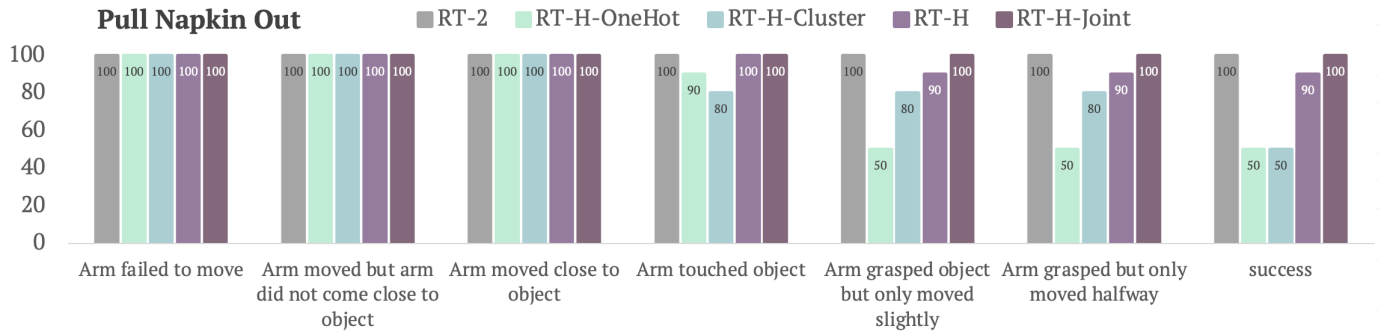



**Fig. 16: Pull Napkin out of Dispenser**: Cumulative success rates for each method.

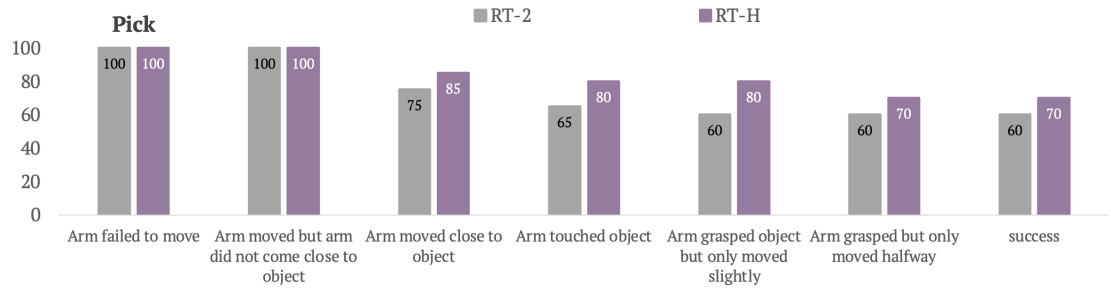



**Fig. 17: Pick (novel objects)**: Cumulative success rates for RT-2 and RT-H. RT-H not only has higher final success rates compared to RT-2, but also success at each stage of the task.

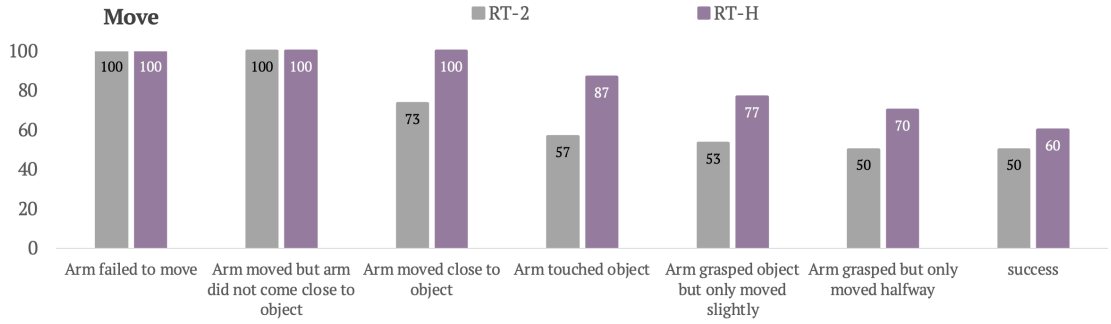
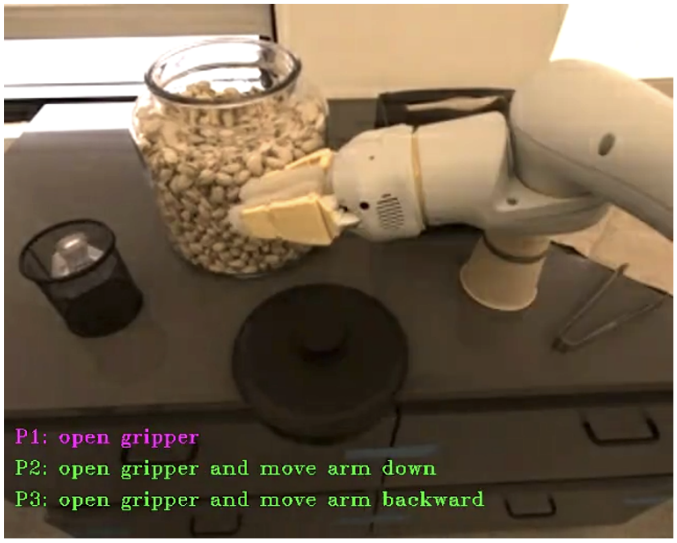


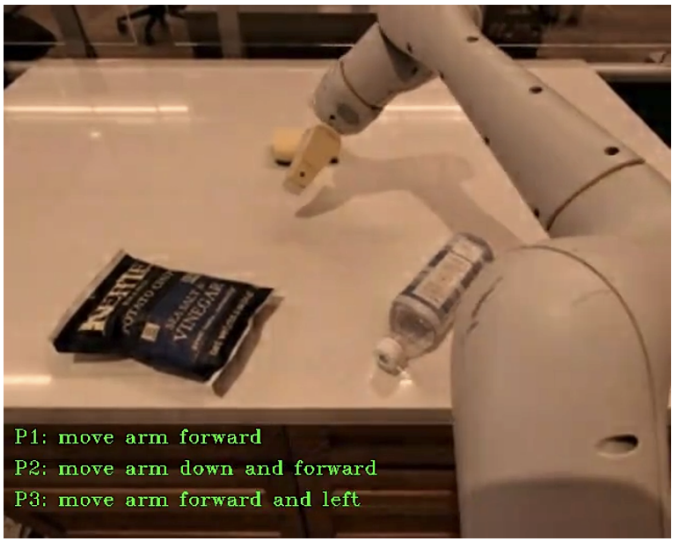

**Fig. 18: Move (novel objects)**: Cumulative success rates for RT-2 and RT-H. RT-H not only has higher final success rates compared to RT-2, but also success at each stage of the task.

| language motion | arm (x) | arm (y) | arm (z) | arm (rx) | arm (ry) | arm (rz) | gripper |
|---|---|---|---|---|---|---|---|
| *move arm forward* | **7.7 (6.2)** | -0.5 (3.3) | -1.4 (3.2) | 3.4 (10.9) | -1.1 (8.9) | -4.0 (10.3) | 1.0 (3.3) |
| *move arm backward* | **-10.9 (10.1)** | -2.4 (4.6) | -2.3 (5.2) | 0.5 (9.2) | 7.8 (11.6) | 0.0 (13.1) | 0.5 (6.6) |
| *move arm left* | -0.4 (3.2) | **8.2 (6.3)** | -2.1 (3.4) | 6.1 (12.0) | 2.5 (7.7) | 7.9 (11.5) | 1.0 (2.1) |
| *move arm right* | -1.4 (4.1) | **-6.9 (7.3)** | -1.5 (4.0) | -1.9 (11.7) | 2.2 (7.7) | -6.6 (11.6) | 1.1 (0.9) |
| *move arm up* | -1.7 (4.6) | -2.1 (4.9) | **11.1 (8.6)** | 1.7 (11.7) | -7.4 (10.9) | 0.5 (10.5) | 0.9 (4.6) |
| *move arm down* | -0.6 (4.0) | -0.3 (3.1) | **-8.1 (9.0)** | 1.4 (10.0) | 5.6 (10.5) | -1.0 (8.2) | 1.1 (3.5) |
| *rotate arm right* | 2.1 (4.6) | 3.0 (5.5) | 0.3 (5.1) | **29.3 (24.5)** | -2.1 (12.2) | 0.0 (13.0) | 1.0 (1.5) |
| *rotate arm left* | 0.4 (4.2) | -0.4 (4.2) | -0.5 (3.9) | **-24.3 (21.9)** | -4.1 (14.8) | 0.9 (10.6) | 1.2 (1.6) |
| *rotate arm up* | 0.0 (4.5) | 0.3 (3.3) | -2.3 (4.4) | 2.5 (12.4) | **20.5 (18.2)** | 1.7 (8.8) | 1.0 (1.4) |
| *rotate arm down* | 0.3 (4.5) | 0.8 (3.8) | 1.9 (5.0) | -7.4 (15.1) | **-28.7 (26.6)** | -0.1 (13.6) | 1.0 (1.6) |
| *rotate arm counterclockwise* | 3.3 (4.3) | -1.1 (4.8) | -0.3 (4.5) | 4.2 (12.7) | -3.1 (12.0) | **-24.1 (21.5)** | 1.1 (3.4) |
| *rotate arm clockwise* | -0.5 (5.4) | 3.1 (5.1) | 0.0 (4.8) | -0.9 (12.3) | -0.1 (11.8) | **24.0 (20.8)** | 0.9 (5.2) |
| *open gripper* | 0.6 (0.9) | 0.7 (1.3) | 0.7 (1.3) | 0.7 (2.7) | 0.6 (2.0) | 0.8 (2.3) | **-67.6 (42.9)** |
| *close gripper* | 0.7 (1.8) | 0.8 (1.3) | 0.8 (2.0) | 0.9 (4.3) | 0.5 (3.8) | 0.9 (3.6) | **65.0 (43.5)** |

**TABLE III:** Action Means (and Standard Deviations) for basic language motions (cardinal directions) for each action dimension (arm delta x,y,z; rotate arm delta x,y,z; and gripper) in the *Diverse+Kitchen* dataset, computed over the validation set. The bolded numbers correspond to the dominant axis which the language motion refers to. Note that positions and rotations are not scaled to match each other. We find that while the dominant axis has the largest mean and variance for each skill (bolded numbers), other axes also have nontrivial variation (but often close to zero mean). This demonstrates that a given skill is not merely a fixed primitive, but maps to a wide variety of potential actions depending on the actual state and the task (i.e., context).




(a) Close Pistachio Jar




(b) Move the sponge toward the chip bag


(c) Open Pistachio Jar


(d) Grab the Scooper

**Fig. 19:** Four examples of multimodal language motion prediction in RT-H using beam search on the language motion query. We show the top three language motion predictions (P1, P2, and P3) for each example. In (a) and (b) (top row), we see that RT-H is capable of representing multiple valid ways of doing the task that are all very contextual. The language motions outputted in (a) and (b) are quite similar to each other, but differ in subtle but task-relevant ways. In (c) and (d), we see that RT-H predicts even more diverse ways to accomplish the task, once again contextual to the task and scene.