
ROBOTIC TASK GENERALIZATION VIA HINDSIGHT TRAJECTORY SKETCHES

Jiayuan Gu^{1,2}, Sean Kirmani¹, Paul Wohlhart¹, Yao Lu¹, Montserrat Gonzalez Arenas¹, Kanishka Rao¹, Wenhao Yu¹, Chuyuan Fu¹, Keerthana Gopalakrishnan¹, Zhuo Xu¹, Priya Sundaresan^{3,4}, Peng Xu¹, Hao Su², Karol Hausman¹, Chelsea Finn^{1,3}, Quan Vuong¹, Ted Xiao¹
¹Google DeepMind, ²University of California San Diego, ³Stanford University, ⁴Intrinsic

ABSTRACT

Generalization remains one of the most important desiderata for robust robot learning systems. While recently proposed approaches show promise in generalization to novel objects, semantic concepts, or visual distribution shifts, generalization to new tasks remains challenging. For example, a language-conditioned policy trained on pick-and-place tasks will not be able to generalize to a folding task, even if the arm trajectory of folding is similar to pick-and-place. Our key insight is that this kind of generalization becomes feasible if we represent the task through rough trajectory sketches. We propose a policy conditioning method using such rough trajectory sketches, which we call *RT-Trajectory*, that is practical, easy to specify, and allows the policy to effectively perform new tasks that would otherwise be challenging to perform. We find that trajectory sketches strike a balance between being detailed enough to express low-level motion-centric guidance while being coarse enough to allow the learned policy to interpret the trajectory sketch in the context of situational visual observations. In addition, we show how trajectory sketches can provide a useful interface to communicate with robotic policies – they can be specified through simple human inputs like drawings or videos, or through automated methods such as modern image-generating or waypoint-generating methods. We evaluate *RT-Trajectory* at scale on a variety of real-world robotic tasks, and find that *RT-Trajectory* is able to perform a wider range of tasks compared to language-conditioned and goal-conditioned policies, when provided the same training data. Evaluation videos can be found at <https://rt-trajectory.github.io/>.

1 INTRODUCTION

The pursuit of generalist robot policies has been a perennial challenge in robotics. The goal is to devise policies that not only perform well on known tasks but can also generalize to novel objects, scenes, and motions that are not represented in the training dataset. The generalization aspects of the policies are particularly important because of how impractical and prohibitive it is to compile a robotic dataset covering every conceivable object, scene, and motion. In this work we focus on the aspects of policy learning that, as we later show in the experiments, can have a large impact of their generalization capabilities: task specification and policy conditioning.

Traditional approaches to task specification include one-hot task conditioning (Kalashnikov et al., 2021), which has limited generalization abilities since one-hot vector does not capture the similarities between different tasks. Recently, language conditioning significantly improves generalization to new language commands (Brohan et al., 2023b), but it suffers from the lack of specificity, which makes it difficult to generalize to a new motion that can be hard to describe. Goal image or video conditioning (Lynch et al., 2019; Chane-Sane et al., 2023), two other alternatives, offer the promise of more robust generalization and can capture nuances hard to express verbally but easy to show visually. However, it has been shown to be hard to learn from (Jang et al., 2022) and requires more effort to provide at test time, making it less practical. Most importantly, policy conditioning not only impacts the practicality of task specification, but can have a large impact on generalization at inference time. If the representation of the task is similar to the one of the training tasks, the underlying model is more likely able to interpolate between these data points. This is often reflected with the type of generalization exhibited in different conditioning mechanisms – for example, if the policy is conditioned on natural language commands, it is likely to generalize to a new phrasing of the text command, whereas that same policy when trained on pick-and-place tasks will struggle with generalizing to

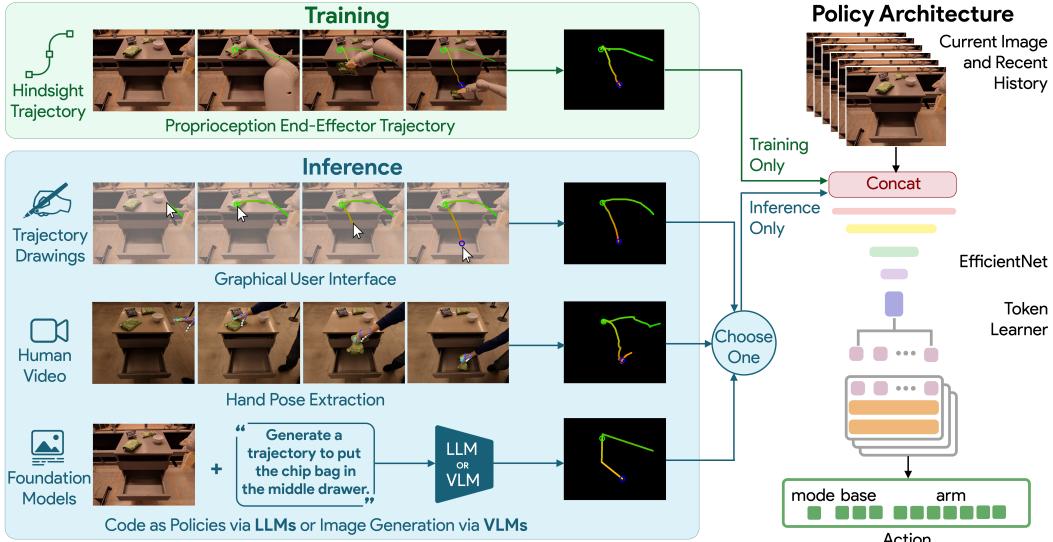


Figure 1: We propose *RT-Trajectory*, a framework for utilizing coarse trajectory sketches for policy conditioning. We train on hindsight trajectory sketches (top left) and evaluate on inference trajectories (bottom left) produced via *Trajectory Drawings*, *Human Videos*, or *Foundation Models*. These trajectory sketches are used as task specification for an RT-1 (Brohan et al., 2023b) policy backbone (right). The trajectories visually describe the end-effector motions (curves) and gripper interactions (circles).

a folding task, even if the arm trajectory of folding is similar to pick-and-place, because in language space, this new task is outside of the previously seen data. This begs a question: can we design a better conditioning modality that is expressive, practical and, at the same time, leads to better generalization to new tasks?

To this end, we propose to use a *coarse* trajectory as a middle-ground solution between expressiveness and ease of use. Specifically, we introduce the use of a 2D trajectory projected into the camera’s field of view, assuming a calibrated camera setup. This approach offers several advantages. For example, given a dataset of demonstrations, we can automatically extract hindsight 2D trajectory labels without the need for manual annotation. In addition, trajectory labels allow us to explicitly reflect similarities between different motions of the robot, which, as we show in the experiments, leads to better utilization of the training dataset resulting in a wider range of tasks compared to language- and goal-conditioned alternatives. Furthermore, humans or modern image-editing models can sketch these trajectories directly onto an image, making it a simple yet expressive policy interface.

The main contribution of this paper is a novel policy conditioning framework *RT-Trajectory* that fosters task generalization. This approach employs 2D trajectories as a human-interpretable yet richly expressive conditioning signal for robot policies. Our experimental setup involves a variety of object manipulation tasks with both known and novel objects. Our experiments show that *RT-Trajectory* outperforms existing policy conditioning techniques, particularly in terms of generalization to novel motions, an open challenge in robotics.

2 RELATED WORK

In this section, we discuss prior works studying generalization in robot learning as well as works proposing specific policy conditioning representations.

Generalization in Robot Learning Recent works have studied how learning-based robot policies may generalize robustly to novel situations beyond the exact data seen during training. Empirical studies have analyzed generalization challenges in robotic imitation learning, focusing on 2D control (Toyer et al., 2020), demonstration quality (Mandlekar et al., 2021), visual distribution shifts (Xie et al., 2023), and action consistency (Belkhale et al., 2023). In addition, prior works have proposed evaluation protocols explicitly testing policy generalization; these include generalizing to novel semantic attributes (Shridhar et al., 2021), holdout language templates (Jang et al., 2021), unseen object categories (Pinto & Gupta, 2016; Mahler et al., 2017; Shridhar et al., 2022; Stone et al., 2023), new backgrounds and distractors (Chen et al.,

2023; Yu et al., 2023), combinations of distribution shifts (Brohan et al., 2023b; Jiang et al., 2023), open-set language instructions (Xiao et al., 2023; Huang et al., 2023), and web-scale semantic concepts (Brohan et al., 2023a). While these prior works largely address semantic and visual generalization, we additionally study task generalization which include situations which require combining seen states and actions in new ways, or generalizing to wholly unseen states or motions altogether.

Policy Conditioning Representations We examine a few approaches for policy conditioning. Broadly, there are 2 axes to consider: (1) over-specification and under-specification of goals, and (2) conditioning on all states in a trajectory versus only the end state. The most prolific recent body of work focuses on language-conditioned policies (Jang et al., 2021; Brohan et al., 2023b;a; Nair et al., 2021; Ahn et al., 2022; Hill et al., 2020; Lynch & Sermanet, 2021), which utilize templated or freeform language as task specification. Language-conditioned policies can be thought of as *under-specified on the end state* (e.g. there are many possible end-states for a policy that completes `pick can`). There are many image-conditioned policy representations with the most popular technique being goal-image conditioning: where a final goal image defines the desired task’s end-state (Bousmalis et al., 2023; Lynch et al., 2019). Goal image conditioned policies can be thought of as *over-specified on the end state* (i.e. “what to do”) because they define an entire configuration, some of which might not be relevant. For example, the background pixels of the goal image might not be pertinent to the task, and instead contain superfluous information. There are some examples of intermediate levels of specification that propose 2D and 3D object-centric representations (Stone et al., 2023; Shridhar et al., 2021; Huang et al., 2023), using a multimodal embedding that represents the task as a joint space of task-conditioned text and goal-conditioned image (Xiao et al., 2023; Jiang et al., 2023; Shridhar et al., 2021), and describing the policy as code (Liang et al., 2022) which constrains how to execute every state. An even more detailed type of state-specification would be conditioning on an entire RGB video which is equivalent to *over-specification over the entire trajectory of states* (i.e. “how to do it”) (Chane-Sane et al., 2023). However, encoding long videos in-context is challenging to scale, and learning from high-dimensional videos is a challenging learning problem (Jang et al., 2021). In contrast, our approach uses a lightweight coarse level of state-specification, which aims to strike a balance between sufficient state-specification capacity to capture salient state properties while still being tractable to learn from. We specifically compare against language-conditioning and goal-image conditioning baselines, and show the benefits of using a mid-level conditioning representation such as coarse trajectory sketches.

3 METHOD

3.1 OVERVIEW

Our goal is to learn a robotic control policy that is able to utilize a 2D coarse trajectory sketch image as its conditioning. A system diagram for our proposed approach can be seen in Fig 1. During policy training, we first perform hindsight trajectory labeling to obtain trajectory conditioning labels from the demonstration dataset (Section 3.2). This enables us to re-use existing demonstration dataset and ensures the scalability of our proposed approach to new datasets. We then train a transformer-based control policy that is conditioned on the 2D trajectory sketches using imitation learning (Section 3.3). During inference time, the user or a high-level planner is presented an initial image observation from the robot camera, and creates a rough 2D trajectory sketch that specifies the desired motion (Fig. 1 bottom left), which is then fed into the trained control policy to perform the designated manipulation task.

3.2 HINDSIGHT TRAJECTORY LABELS

In this section, we describe how we acquire training trajectory conditioning labels from the demonstration dataset. We introduce three basic elements for constructing the trajectory representation format: 2D Trajectories, Color Grading, and Interaction Markers.

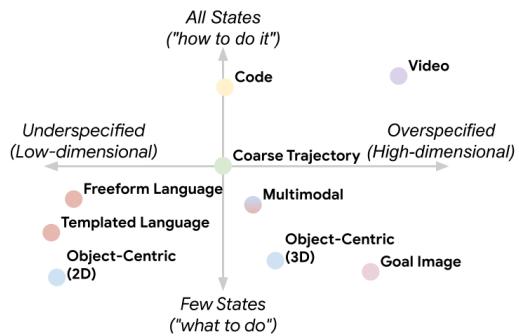


Figure 2: The choice of robot policy representation balances specification detail and focusing policies on “what to do” compared with “how to do it”.

(a) Example Trajectory

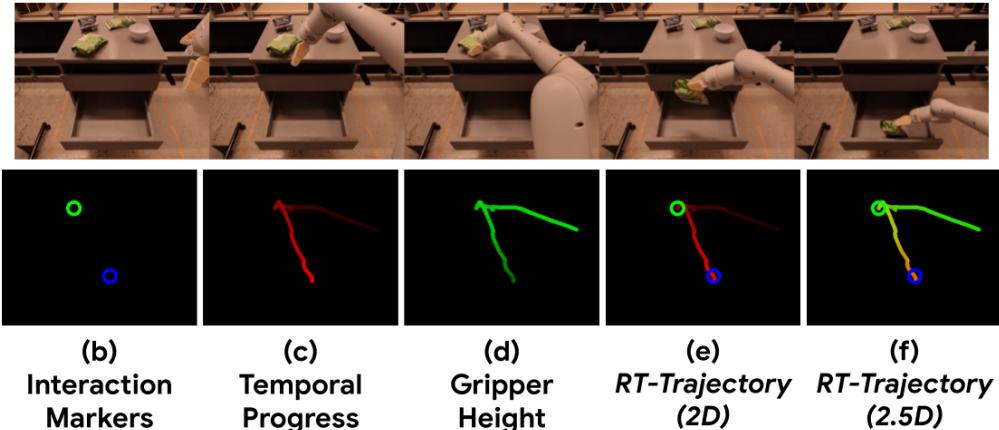


Figure 3: Visualization of the two hindsight trajectory sketch representations we study. Given (a) an example robot trajectory, we extract (b) gripper interaction markers, (c) temporal progress along the 2D end-effector waypoints, and (d) end-effector height. Combining (b) and (c) results in (e) **RT-Trajectory (2D)**, while combining (b), (c), and (d) results in (f) **RT-Trajectory (2.5D)**.

2D Trajectory For each episode in the demonstration dataset, we extract a 2D trajectory of robot end-effector center points. Concretely, given the proprioceptive information recorded in the episode, we obtain the 3D position of the robot end-effector center defined in the robot base frame at each time step, and project it to the camera space given the known camera extrinsic and intrinsic parameters. We assume that the robot base and camera do not move within the episode, which is common for stationary manipulation. Given a 2D trajectory (a sequence of pixel positions), we draw a curve on a blank image, by connecting 2D robot end-effector center points at adjacent time steps through straight lines.

Color Grading To express relative temporal motion, which encodes such as velocity and direction, we also explore using the red channel of the trajectory image to specify the normalized time step $\frac{t+1}{T}$, where t is the current time step and T is the total episode length. Additionally, we propose incorporating height information into the trajectory representation by utilizing the green channel of the trajectory image to encode normalized height relative to the robot base $\frac{h_{t+1} - h_{min}}{h_{max} - h_{min}}$.

Interaction Markers For robot manipulation tasks, time steps when the end-effector interacts with the environment are particularly important. Thus, we explore visual markers that explicitly highlight the time steps when the gripper begins to grasp and release objects. Concretely, we first compute whether the gripper has contact with objects by checking the difference $\delta_t = \hat{p}_t - p_t$ between the sensed (p_t) and target (\hat{p}_t) gripper joint positions. If the difference $\delta_t > 0$ and $\hat{p}_t > \epsilon$, where ϵ is a threshold of closing action (p_t increases as the gripper closes), it indicates that the gripper is closing and grasping certain object. If the status change, e.g., $\delta_t < 0 \vee \hat{p}_t \leq \epsilon$ but $\delta_{t+1} > 0 \wedge \hat{p}_{t+1} > \epsilon$, we consider the time step t as a key step for the closing action. Similarly, we can find the key time steps for the opening action. We draw green (or blue) circles at the 2D robot end-effector center points of all key time steps for closing (or opening) the gripper.

Trajectory Representations In this work, we propose two forms of trajectory representation from different combinations of the basic elements. In the first one, **RT-Trajectory (2D)**, we construct an RGB image containing the 2D Trajectory with temporal information and Interaction Markers to indicate particular robot interactions (Fig. 3 (e)). In the second representation, we introduce a more detailed trajectory representation **RT-Trajectory (2.5D)**, which includes the height information in the 2D trajectory (Fig. 3 (f)).

3.3 POLICY TRAINING

We leverage Imitation Learning due to its strong success in multitask robotic imitation learning settings (Jang et al., 2022; Bousmalis et al., 2023). More specifically, we assume access to a collection of successful robot demonstration episodes. Each episode τ contains a sequence of pairs of observations o_t and actions a_t : $\tau = \{(o_t, a_t)\}$. The observations include RGB images obtained from the head camera x_t and hindsight trajectory sketch c_{traj} . We then learn a policy π represented by a Transformer (Vaswani et al., 2017) using

Behavior Cloning (Pomerleau, 1988) following the RT-1 framework (Brohan et al., 2023b), by minimizing the log-likelihood of predicted actions a_t given the input image and trajectory sketch. To support trajectory conditioning, we modify the RT-1 architecture as follows. The trajectory sketch is concatenated with each RGB image along the feature dimension in the input sequence (a history of 6 images), which is processed by the image tokenizer (an ImageNet pretrained EfficientNet-B3). For the additional input channels to the image tokenizer, we initialize the new weights in the first convolution layer with all zeros. Since the language instruction is not used, we remove the FiLM layers used in the original RT-1.

3.4 TRAJECTORY CONDITIONING DURING INFERENCE

During inference, a trajectory sketch is required to condition *RT-Trajectory*. We study 4 different methods to generate trajectory sketches: *human drawings*, *human videos*, *prompting LLMs with Code as Policies*, and *image generation models*.

Human-drawn Sketches Human-drawn sketches are an intuitive and practical way for generating trajectory sketches. To scalably produce these sketches, we design a simple graphical user interface (GUI) for users to draw trajectory sketches given the robot’s initial camera image, as shown in App. B.1.

Human Demonstration Videos with Hand-object Interaction First-person human demonstration videos are an alternative input. We estimate the trajectory of human hand poses from the video, and convert it to a trajectory of robot end-effector poses, which can later be used to generate a trajectory sketch.

Prompting LLMs with Code as Policies Large Language Models have demonstrated the ability to write code to perform robotics tasks (Liang et al., 2022). We follow a similar recipe as described in (Gonzalez Arenas et al., 2023) to build a prompt which contains text descriptions about the objects in the scene detected by a VLM, the robot constraints, the gripper orientations and coordinate systems, as well as the task instruction. By using this prompt, the LLM writes code to generate a series of 3D poses - originally intended to be executed with a motion planner, which we can then re-purpose to draw the trajectory sketch on the initial image to condition *RT-Trajectory*.

Image Generation Models Since our trajectory conditioning is represented as an image, we can use text-guided image generation models to generate a trajectory sketch provided the initial image and language instruction which describes the task. In our work, we use a PaLM-E style (Driess et al., 2023) model that generates vector-quantized tokens derived from ViT-VQGAN (Yu et al., 2022) that represent the trajectory image. Once detokenized, the resulting image can be used to condition *RT-Trajectory*.

4 EXPERIMENTS

Our real robot experiments aim to study the following questions:

1. Can *RT-Trajectory* generalize to tasks beyond those contained in the training dataset?
2. Can *RT-Trajectory* trained on hindsight trajectory sketches generalize to diverse human-specified or automated trajectory generation methods at test time?
3. What emergent capabilities are enabled by *RT-Trajectory*?
4. Can we quantitatively measure how dissimilar evaluation trajectory motions are from training dataset motions?

4.1 EXPERIMENTAL SETUP

We use a mobile manipulator robot from Everyday Robots in our experiments, which has a 7 degree-of-freedom arm, a two-fingered gripper, and a mobile base.

Seen Skills We use the RT-1 (Brohan et al., 2023b) demonstration dataset for training. The language instructions consist of 8 different manipulation skills (e.g., Move Near) operating on a set of 17 household kitchen items; in total, the dataset consists of about 73K real robot demonstrations across 542 seen tasks, which were collected by manual teleoperation. A more detailed overview is shown in Table 2.

Unseen Skills We propose 7 new evaluation skills which include unseen objects and manipulation workspaces, as shown in Table 3 and Fig. 4. Both Upright and Move and Move within Drawer examine whether the policy can combine different seen skills to form a new one. For example, Move

within Drawer studies whether the policy is able to move objects within the drawer while the seen skill Move Near only covers those motions at height of the tabletop. Restock Drawer requires the robot to place snacks into the drawer at an empty slot. It studies whether the policy is able to place objects at target positions precisely. Place Fruit inspects whether the policy can place objects into unseen containers. Pick from Chair investigates whether the policy can pick objects at an unseen height in an unseen manipulation workspace. Fold Towel and Swivel Chair showcase the capability to manipulate a deformable object and interact with an underactuated system.

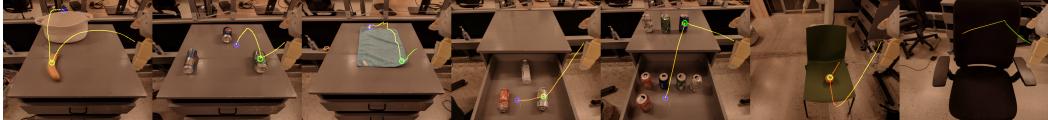


Figure 4: Visualization of trajectory sketches overlaid on the initial image for 7 unseen skills. From left to right: Place Fruit, Upright and Move, Fold Towel, Move within Drawer, Restock Drawer, Pick from Chair, Swivel Chair. See the rollouts in Fig. 15.

Evaluation Protocol Different trajectory sketches will prompt *RT-Trajectory* to behave differently. To make the quantitative comparison between different methods as fair as possible, we propose the following evaluation protocol. For each skill to evaluate, we collect a set of *scenes*. Each scene defines the initial state of the task, described by an RGB image taken by the robot head camera. During evaluation, we first align relevant objects to their original arrangements in the *scene*, and then run the policy. For conditioning *RT-Trajectory*, we use human drawn sketches for unseen tasks in Sec. 4.2. In Sec. 4.3, we evaluate other trajectory sketch generation methods described in Sec. 3.4.

4.2 UNSEEN TASK GENERALIZATION

In this section, we compare *RT-Trajectory* with other learning-based baselines on generalization to the unseen task scenarios introduced in Sec 4.1.

- RT-1 (Brohan et al., 2023b): language-conditioned policy trained on the same training data;
- RT-2 (Brohan et al., 2023a): language-conditioned policy trained on a mixture of our training data and internet-scale VQA data;
- RT-1-Goal: goal-conditioned policy trained on the same training data.

For *RT-Trajectory*, we manually generate trajectory sketches via the GUI (see Sec. B.1). Details about trajectory generation are described in App. B.2. For *RT-1-Goal*, implementation details and goal conditioning generation are presented in App. B.4. The results are shown in Fig. 5 and Table 4. The overall success rates of our methods, *RT-Trajectory* (2D) and *RT-Trajectory* (2.5D), are 50% and 67% respectively, which outperform our baselines by a large margin: RT-1 (16.7%), RT-2 (11.1%), RT-1-Goal (26%). Language-conditioned policies struggle to generalize to the new tasks with semantically unseen language instructions, even if motions to achieve these tasks were seen during training (see Sec. 4.5). *RT-1-Goal* shows better generalization than its language-conditioned counterparts. However, goal conditioning is much harder to acquire than trajectory sketches during inference in new scenes and is sensitive to task-irrelevant factors (e.g., backgrounds). *RT-Trajectory* (2.5D) outperforms *RT-Trajectory* (2D) on the tasks where height information helps reduce ambiguity. For example, with 2D trajectories only, it is difficult for *RT-Trajectory* (2D) to infer correct picking height, which is critical for Pick from Chair.

4.3 DIVERSE TRAJECTORY GENERATION METHODS

In this section, we aim to study whether *RT-Trajectory* is able to generalize to trajectories from more automated and general processes at inference time. Specifically, we evaluate quantitatively how *RT-Trajectory* performs when conditioned on coarse trajectory sketches generated by *human video demonstrations*, LLMs via *Prompting with Code as Policies*, and show qualitative results for *image generating VLMs*. Additionally, we compare *RT-Trajectory* against a non-learning baseline (*IK Planner*) to follow the generated trajectories: an inverse-kinematic (IK) solver is applied to convert the end-effector poses to joint positions, which are then executed by the robot.

Human Demonstration Videos We collect 18 and 4 first-person human demonstration videos with hand-object interaction for Pick (seen training skill) and Fold Towel. An example is shown in Fig. 6.

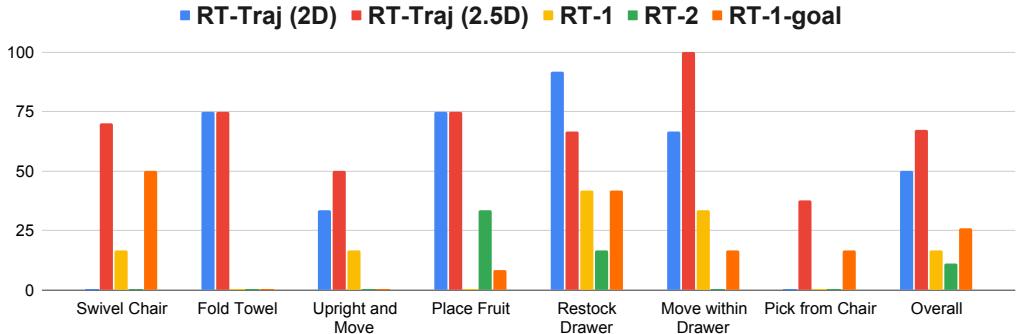


Figure 5: Success rates for unseen tasks when conditioning with human drawn sketches. Scenarios contain a variety of difficult settings which require combining seen motions in novel ways or generalizing to new motions. Each policy is evaluated for a total of 64 trials across 7 different scenarios.



Figure 6: Trajectory from human demonstration video to fold a towel. From left to right, the first 4 images show the human demonstration, and the last image shows the derived trajectory sketch.

Method	Pick	Fold Towel	Method	Pick	Open Drawer
IK Planner	42%	25%	IK Planner	83%	71%
Ours (2D)	94%	75%	Ours (2D)	89%	60%
Ours (2.5D)	100%	75%	Ours (2.5D)	89%	60%

(a) Trajectory from human video demonstrations.

(b) Trajectory from LLM prompting.

Table 1: Success rate of different trajectory generation approaches across tasks.

Details about video collection and how trajectory sketches are derived from videos are described in App. B.3. The resulting trajectory sketches are more squiggly than the ones for training. Results are shown in Table 1a.

Prompting with Code as Policies We prompt an LLM (OpenAI, 2023) to write code to generate trajectories given the task instructions and object labels for two seen skills, `Pick` and `Open Drawer`. After executing the code written by the LLM, we get a sequence of target robot waypoints which can then be processed into a trajectory sketch. In contrast with human-specified trajectories, LLM-generated trajectories are designed to be executed by an IK planner and are therefore precise and linear as seen in Fig. 20. While they are also different from the hindsight trajectories in the training data, *RT-Trajectory* is able to execute them correctly and outperform the IK planner in diverse pick tasks due to its ability to adapt motion to the scene nuances like object orientation. Results are shown in Table 1b.

Image Generation Models We condition the VLM with a language instruction and an initial image to output trajectory tokens which are de-tokenized into 2D pixel coordinates for drawing the trajectory. Qualitative examples are shown in Fig 7. Although we see that generated trajectory sketches are noisy and quite different from the training hindsight trajectory sketches, we find promising signs that *RT-Trajectory* still performs reasonably. As image-generating VLMs rapidly improve, we expect that their trajectory sketch generating capabilities will improve naturally in the future and be usable by *RT-Trajectory*.

4.4 EMERGENT CAPABILITIES

Prompt Engineering for Robot Policies Similar to how LLMs respond differently in response to language prompt engineering, *RT-Trajectory* enables *visual* prompt engineering, where a trajectory-conditioned



Figure 7: Example trajectory from image generation models. From left to right, the first image shows the overlaid trajectory sketch, and the next 4 images show the rollout conditioned on it. The language instruction is: pick orange can from top drawer and place on counter.

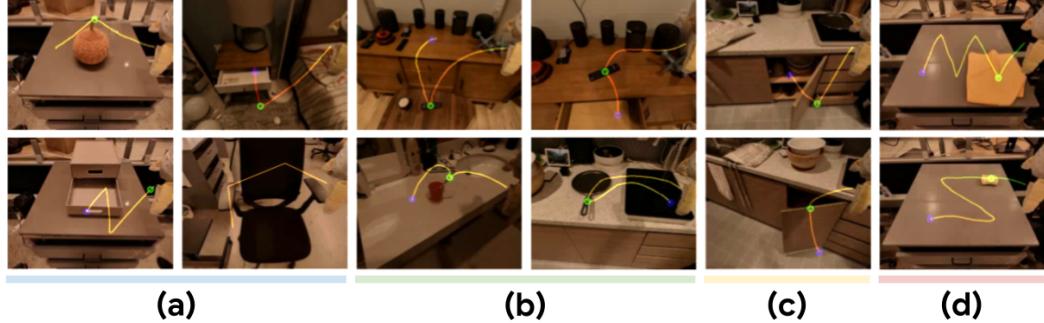


Figure 8: Example *RT-Trajectory* evaluations in realistic scenarios involving (a) novel articulated objects requiring new motions, (b) manipulation on new surfaces in new buildings in new heights, (c) interacting with a pivot-hinge cabinet despite training only on sliding-hinge drawers, and (d) circuitous tabletop patterns extending beyond direct paths in the training dataset. Full rollouts are shown in Fig. 17 and the supplemental video at <https://rt-trajectory.github.io/>.

policy may exhibit better performance when the initial scene is fixed but the coarse trajectory prompts are improved. We find that changing trajectory sketches induces *RT-Trajectory* to change behavior modes in a reproducible manner, which suggests an intriguing opportunity: if a trajectory-conditioned robot policy fails in some scenario, a practitioner may just need to “query the robot” with a different trajectory prompt, as opposed to re-training the policy or collecting more data. Qualitatively, this is quite different from standard development practices with language-conditioned robot policies, and may be viewed as an early exploration into zero-shot instruction tuning for robotic manipulation, similar to capabilities seen in language modeling (Brown et al., 2020). See App. E.1 for examples.

Generalizing to Realistic Settings Prior works studying robotic generalization often evaluate only a few distribution shifts at once, since generalizing to simultaneous physical and visual variations is challenging; however, these types of simultaneous distribution shifts are widely prevalent in real world settings. As a qualitative case study, we evaluate *RT-Trajectory* in 2 new buildings in 4 realistic novel rooms which contain entirely new backgrounds, lighting conditions, objects, layouts, and furniture geometries. With little to moderate trajectory prompt engineering, we find that *RT-Trajectory* is able to successfully perform a variety of tasks requiring novel motion generalization and robustness to out-of-distribution visual distribution shifts. These tasks are visualized in Fig. 8 and rollouts are shown fully in Fig. 17.

4.5 MEASURING MOTION GENERALIZATION

We wish to explicitly measure motion similarity in order to better understand how *RT-Trajectory* is able to generalize to unseen scenarios and how well it can tackle the challenges of novel motion generalization. Towards this, we intend to compare evaluation trajectories to the most similar trajectories seen during training. To accomplish this, we propose to measure trajectory similarity by utilizing the discrete Fréchet distance (Fréchet, 1906) (details in App. C.1). By computing the distance between a query trajectory and all trajectories in our training dataset, we can retrieve the most similar trajectories our policy has been trained on. We perform this lookup for trajectories from the rollouts for the unseen task evaluations in Sec. 3.4. Fig. 9 showcases the 10 most similar training trajectories for a selection of query trajectories.

Fig. 10, 11, and 12 furthermore show statistics of the most similar training samples, such as the distribution of skill semantics. We find that the trajectories for unseen tasks show varying levels of similarity to training

Top-10 Most Similar Training Trajectories to Query Trajectories

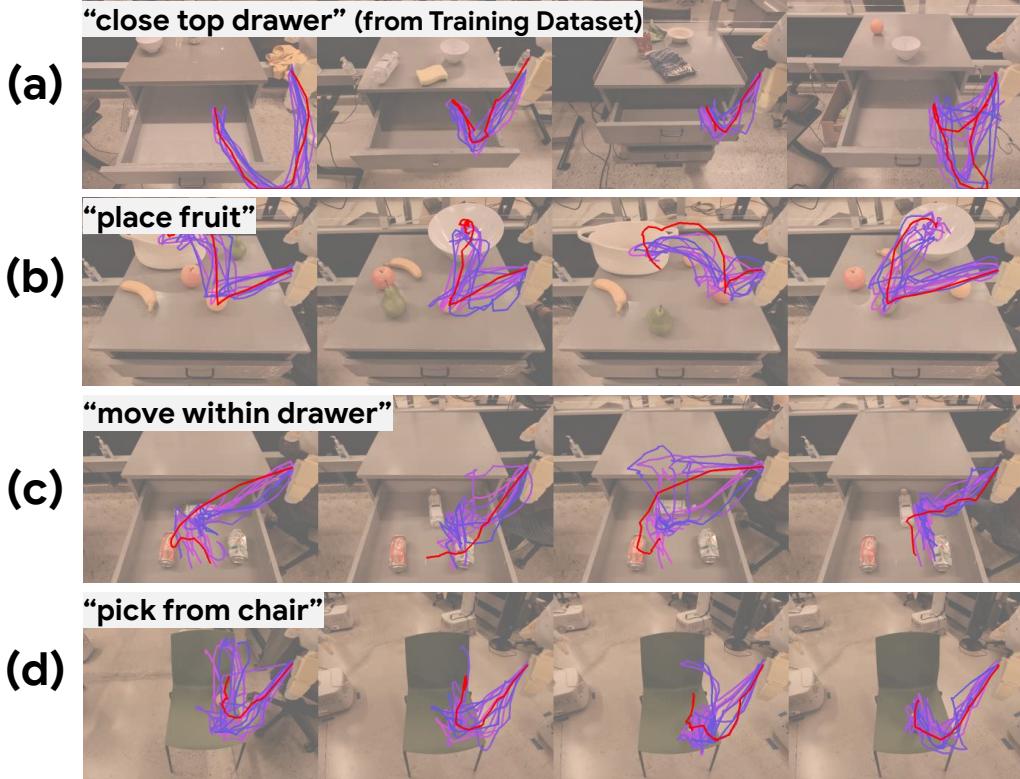


Figure 9: Each row contains 4 instances of an initial image of an evaluation rollout super-imposed with the executed evaluation trajectory (red) compared with the 10 most similar trajectories (purple) in the training dataset. Row (a) shows query trajectories of the in-distribution `close top drawer` skill seen in the training data. Rows (b,c,d) show query trajectories of unseen evaluation skills.

trajectories. For example, the motion for `place a fruit into a tall bowl` may be surprisingly similar to the motion for particular seen instances of the `move X near Y`. However, for many unseen skills, the most similar examples in the training data are still significantly more different than for examples within the training set. In addition, even for evaluation trajectories that seem close in shape to the most similar training trajectories, we find differences in precision-critical factors like the z-height of gripper interactions (picks that are just a few centimeter incorrect will not succeed) or semantic relevance (the most similar training trajectories describe different skills than the target trajectory). Thus, we expect that the proposed new skills for evaluation indeed require a mix of interpolating seen motions along with generalizing to novel motions altogether.

5 CONCLUSION AND LIMITATIONS

In this work, we propose a novel policy-conditioning method for training robot manipulation policies capable of generalizing to tasks and motions that are significantly beyond the training data. Key to our proposed approach is a 2D trajectory sketch representation for specifying manipulation tasks. Our trained trajectory sketch-conditioned policy enjoys controllability from visual trajectory sketch guidance, while retaining the flexibility of learning-based policies in handling ambiguous scenes and generalization to novel semantics. We evaluate our proposed approach on 7 diverse manipulation skills that were never seen during training and benchmark against three baseline methods. Our proposed method achieves a success rate of 67%, significantly outperforming the best prior state-of-the-art methods, which achieved 26%.

Though we demonstrate that our proposed approach achieves encouraging generalization capabilities for novel manipulation tasks, there are a few remaining limitations. First, we currently assume that the robot

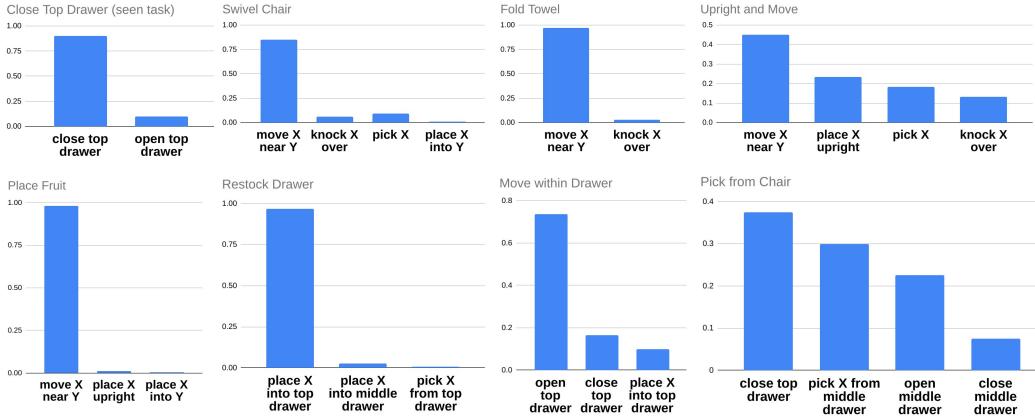


Figure 10: Semantic relevance measures how the semantic skills of rollout trajectories compare to the semantic skills of the most similar training trajectories, as measured by motion similarity. For the seen skill (`close top drawer`), the most similar training trajectories are largely of the same semantic skill. For the unseen skills, the most similar training trajectories are composed of semantically different tasks.

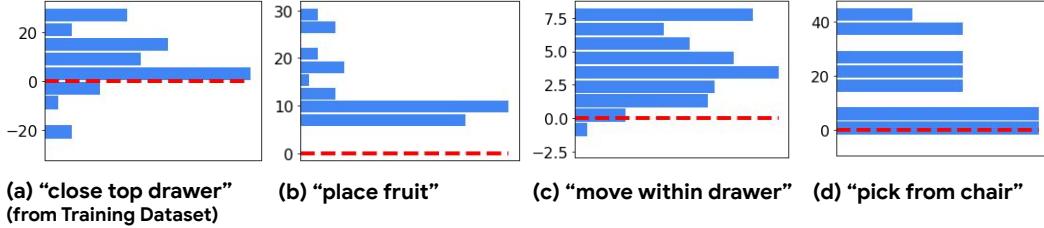


Figure 11: First-interaction height alignment compares the relative difference between the z-height of the first gripper interactions of query trajectories to the first gripper interactions of the most similar training trajectories, as measured by motion similarity. The red line represents the baseline relative difference of the query trajectory compared with itself, which would be a difference of 0.0. The unseen skills in general see large variance in the difference of first-interaction heights of the query trajectories compared to the most similar training trajectories.

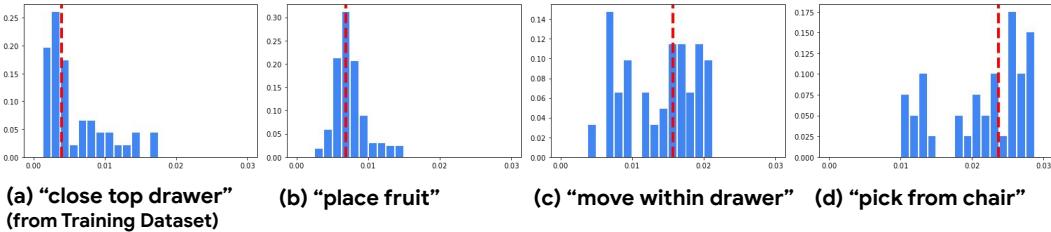


Figure 12: We visualize the distribution of Fréchet distances of query trajectories to the most similar training trajectories, as measured by motion similarity. The red line represents the median of the average distance between evaluation trajectories and the most similar training trajectories. Query trajectories of unseen skills in general see larger Fréchet distances to the most similar training trajectories, compared to query trajectories from training skills.

remains stationary and only uses the end-effector for performing useful manipulation motions. Extending the idea to mobile-manipulation scenarios that allow the robot to manipulate with whole-body control is a promising direction to explore. Second, our trained policy makes its best effort in following the trajectory sketch guidance. However, a user may want to specify spatial regions where the guidance is more strictly enforced, such as when to avoid fragile objects during movement. Thus, an interesting future direction is to enable systems to use trajectory sketches to handle different types of constraints.

ACKNOWLEDGMENTS

The authors would like to thank Wenxuan Zhou for help with the human hand pose tracking infrastructure. Also, we would like to thank Cheri Tran, Emily Perez, Grecia Salazar, Jaspiar Singh, and Jodilyn Peralta for their immense contributions to evaluations.

REFERENCES

- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.
- Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. *arXiv preprint arXiv:2306.02437*, 2023.
- Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer vision, graphics, and image processing*, 27(3):321–345, 1984.
- Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Peveciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. Robocat: A self-improving foundation agent for robotic manipulation, 2023.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander 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 *Conference on Robot Learning (CoRL)*, 2023a.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. *Proceedings of Robotics: Science and Systems (RSS)*, 2023b.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Learning video-conditioned policies for unseen manipulation tasks, 2023.
- Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

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 Serbanet, 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, 2023.

Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. 1994.

M Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo* (1884-1940), 22(1):1–72, 1906.

Montserrat Gonzalez Arenas, Ted Xiao, Sumeet Singh, Vidhi Jain, Allen Ren, Quan Vuong, Jake Varley, Alexander Herzog, Isabel Leal, Sean Kirmani, Mario Prats, Dorsa Sadigh, Vikas Sindhwani, Kanishka Rao, Jacky Liang, and Andy Zeng. How to prompt your robot: A promptbook for manipulation skills with code as policies. 2023.

Felix Hill, Sona Mokrá, Nathaniel Wong, and Tim Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *CoRR*, abs/2005.09382, 2020. URL <https://arxiv.org/abs/2005.09382>.

Rachel M Holladay and Siddhartha S Srinivasa. Distance metrics and algorithms for task space path optimization. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5533–5540. IEEE, 2016.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023.

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.

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, 2022.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

Dmitry Kalashnikov, Jake Varley, Yevgen Chebotar, Ben Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *Conference on Robot Learning (CoRL)*, 2021.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

Camillo Lugaesi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Ubweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019.

Corey Lynch and Pierre Serbanet. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Serbanet. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019.

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

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. *Conference on Robot Learning (CoRL)*, 2021.

-
- Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation, 2021.
- OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL <https://api.semanticscholar.org/CorpusID:257532815>.
- Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *IEEE international conference on robotics and automation (ICRA)*, 2016.
- Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.
- Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation, 2021.
- Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.
- Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Brianna Zitkovich, Fei Xia, Chelsea Finn, and Karol Hausman. Open-world object manipulation using pre-trained vision-language models, 2023.
- Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. The magical benchmark for robust imitation. *Advances in Neural Information Processing Systems*, 33:18284–18295, 2020.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, Sergey Levine, and Jonathan Tompson. Robotic skill acquisition via instruction augmentation with vision-language models. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *arXiv preprint arXiv:2307.03659*, 2023.
- Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.
- Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspia Singh, Clayton Tan, Dee M, Jodilyn Peralta, Brian Ichter, Karol Hausman, and Fei Xia. Scaling robot learning with semantically imagined experience. In *arXiv preprint arXiv:2302.11550*, 2023.

A EXPERIMENT DETAILS

A.1 SEEN SKILLS

Skill	Count	Description	Example Instruction
Pick Object	17	Lift the object off the surface	pick coke can
Move Object Near Object	337	Move the first object near the second	move pepsi can near rxbar blueberry
Place Object Upright	8	Place an elongated object upright	place water bottle upright
Knock Object Over	8	Knock an elongated object over	knock redbull can over
Open Drawer	3	Open any of the cabinet drawers	open the top drawer
Close Drawer	3	Close any of the cabinet drawers	close the middle drawer
Place Object into Receptacle	84	Place an object into a receptacle	place brown chip bag into white bowl
Pick Object from Receptacle and Place on the Counter	82	Pick an object up from a location and then place it on the counter	pick green jalapeno chip bag from paper bowl and place on counter
Total	542		

Table 2: The list of seen training tasks with their descriptions and example language instructions. Language instructions are only used for language-conditioned baselines. “Count” refers to the number of distinct tasks per skill (e.g., Pick coke can and Pick apple are two different tasks).

A.2 UNSEEN SKILLS

Skill	Count	Description	Example instruction
Place Fruit	12	Place fruit into the container	place orange into basket
Upright and Move	6	Place an object upright <i>and</i> move it near another	place green can upright near pepsi can
Move within Drawer	6	Move one object near another <i>within</i> the drawer	move coke can near 7up can at top drawer
Restock Drawer	12	Place objects into the desired position in the drawer	place coke can into the top right of top drawer
Pick from Chair	8	Pick an object placed on the chair	pick apple from chair
Fold Towel	4	Fold the towel by moving one corner to another	fold towel from bottom right
Swivel Chair	10	Swivel the office chair	push the chair

Table 3: The list of unseen evaluation tasks with their descriptions and example language instructions. Language instructions are only used for language-conditioned baselines. “Count” refers to the number of scenes collected for evaluation.

A.3 QUANTITATIVE RESULTS FOR UNSEEN TASKS

Task	RT-Traj (2D)	RT-Traj (2.5D)	RT-1	RT-2	RT-1-goal
Place Fruit	75%	75%	0%	33%	8%
Upright and Move	33%	50%	17%	0%	0%
Move within Drawer	67%	100%	33%	0%	17%
Restock Drawer	92%	67%	42%	17%	42%
Pick from Chair	0%	38%	0%	0%	17%
Fold Towel	75%	75%	0%	0%	0%
Swivel Chair	0%	70%	17%	0%	50%
Overall	50%	67%	17%	11%	26%

Table 4: Success rates for unseen tasks when conditioning with human drawn sketches.

B IMPLEMENTATION DETAILS FOR DIFFERENT INPUT MODALITIES

B.1 GUI FOR HUMAN-DRAWN TRAJECTORY SKETCHES

As the main trajectory generation method we study is user-specified trajectory drawings, we develop a graphical user interface (GUI) for users to draw trajectory sketches. See Fig. 13 for example. Given the current robot camera image, a user can drag and move the mouse to draw curves on the canvas. Then, they

can click on the canvas to add markers to indicate gripper closing or opening actions. Additionally, the UI interface also supports simple height annotation. Users can specify the desired height values for pixels they select on the canvas. This height value will be assigned to the closest point on the drawn 2D trajectory. For unannotated points on the 2D trajectory, we interpolate their height values according to annotated ones.



Figure 13: Left: The GUI for users to draw trajectory sketches given the robot’s current camera image. The 2D trajectory is directly drawn by manual input, which can then be annotated with interaction markers or waypoints corresponding to user-specified heights. Right: The resulting height-aware trajectory sketch generated according to the output of the UI.

B.2 COLLECTING HUMAN-DRAWN TRAJECTORY SKETCHES

For each scene, we use a held-out *RT-Trajectory* (2.5D) policy to explore different trajectory “prompts” given a budget of trials, and save the trajectory sketch of the first successful episode. We refer to such process as “prompt engineering” (Sec. 4.4). If all attempts fail, we just save the trajectory sketch from the last episode. *RT-Trajectory* policies used for evaluation are trained with different random seeds and evaluated with the saved trajectory sketches as conditioning. Note that we observe that our evaluated policies can have non-zero success rates on the scenes where we fail to find a successful episode during “prompt engineering”.

B.3 HUMAN HAND POSE ESTIMATION

We employ Mediapipe ([Lugaresi et al., 2019](#)) to detect the human hand pose represented as 21 landmarks from the 2D image at each video frame. The two landmarks on the thumb and another two landmarks on the index finger are used to represent a parallel gripper. The 2D landmarks are lifted to 3D given the depth map. We then interpolate the end-effector pose from these four points. We manually annotate the key frames at which the hand begins to grasp and release the target object. Given estimated end-effector poses and key frames for interaction, we can generate a trajectory sketch per video.

B.4 IMPLEMENTATION DETAILS FOR *RT-1-Goal*

The network architecture of *RT-1-Goal* is the same as *RT-Trajectory*, except a goal image is used instead of a trajectory sketch. To acquire goal conditioning for training, we use the last observation of each episode as the goal image for all frames in the episode. For the quantitative comparison in Sec. 4.2, the image of the last step of the episode (App. B.2) used to generate the trajectory sketch for each scene is saved as the goal conditioning for evaluation.

C MOTION DIVERSITY ANALYSIS

C.1 COMPUTATION OF TRAJECTORY SIMILARITY

To measure the distance between two end-effector motion trajectories, we employ the Fréchet distance ([Fréchet, 1906](#); [Eiter & Mannila, 1994](#)), a measure that quantifies the similarity between two curves by finding the minimum “leash length” required for two agents traversing each curve simultaneously while maintaining their respective temporal order. As a well-adopted similarity measure in computer vision

and vehicle tracking (Borgefors, 1984), Fréchet distance may be a reasonable choice for comparing 3D robot end-effector waypoint trajectories since it is order-preserving and parameterization independent (Holladay & Srinivasa, 2016).

Specifically, consider two trajectories τ and τ' where each trajectory contains n waypoints $\tau = \{\rho_0, \rho_1, \dots, \rho_m\}$ and $\tau' = \{\rho'_0, \rho'_1, \dots, \rho'_n\}$, and $d(\rho_i, \rho'_i)$ is a distance measure like Euclidean distance. Then, using the notation $\tau[1:]$ to denote removing the first element and returning the rest of the sequence τ , the Fréchet distance F_D is recursively defined as:

$$F_D(\tau, \tau') = \max(d(\rho_0, \rho'_0), \min\{F_D(\tau[1:], \tau'[1:]), F_D(\tau, \tau'[1:]), F_D(\tau[1:], \tau')\})$$

In this work, each waypoint is the sensed end-effector center position and the distance measure is Euclidean distance. Note that the orientation and interaction (closing/opening action) are not taken into consideration.

C.2 ADDITIONAL SAMPLES OF TRAJECTORY SIMILARITIES

Figure 14 shows additional examples of evaluation trajectories and their most similar trajectories in the training dataset.

D ADDITIONAL VISUALIZATION

Fig. 15 visualizes the example rollouts of unseen skills. Fig. 16 demonstrates more example trajectories from image generation models and corresponding rollouts.

Fig 17 shows the rollouts of example evaluations in realistic scenarios mentioned in Sec. 4.4. We showcase additional evaluations in Fig. 18. Notably, we find that *RT-Trajectory* is quite robust to various simultaneous visual distribution shifts including new buildings, new backgrounds, new distractors, new lighting conditions, new objects, and new furniture textures. In addition, these realistic “in the wild” evaluations were not ran in controlled laboratory environments, so the evaluations often required generalization to new heights or furniture geometries (different shaped drawers, cabinets, or tables).

E CASE STUDIES IN EMERGENT CAPABILITIES AND BEHAVIORS

E.1 PROMPT ENGINEERING

Fig. 19 illustrates two examples of prompt engineering mentioned in Sec. 4.4. For instance, if the user wants to prompt *RT-Trajectory* to place an object at a high position, it is better to draw a trajectory that first reaches a higher peak, and then move downward to the target.

E.2 RETRY BEHAVIOR

Compared to non-learning methods, *RT-Trajectory* is able to recover from execution failures. Fig. 20 illustrates the retry behavior emerged when *RT-Trajectory* is opening the drawer given the trajectory sketch generated by prompting LLMs with Code as Policies (CaP) mentioned in Sec. 3.4. After a failure attempt to open the drawer by its handle, the robot retried to grasp the edge of the drawer, and managed to pull the drawer.

E.3 HEIGHT-AWARE DISAMBIGUATION FOR *RT-Trajectory* (2.5D)

2D trajectories (without depth information) are visually ambiguous for distinguishing whether the robot should move its arm to a deeper or higher. We find that height-aware color grading for *RT-Trajectory* (2.5D) can effectively help reduce such ambiguity, as shown in Fig. 21.

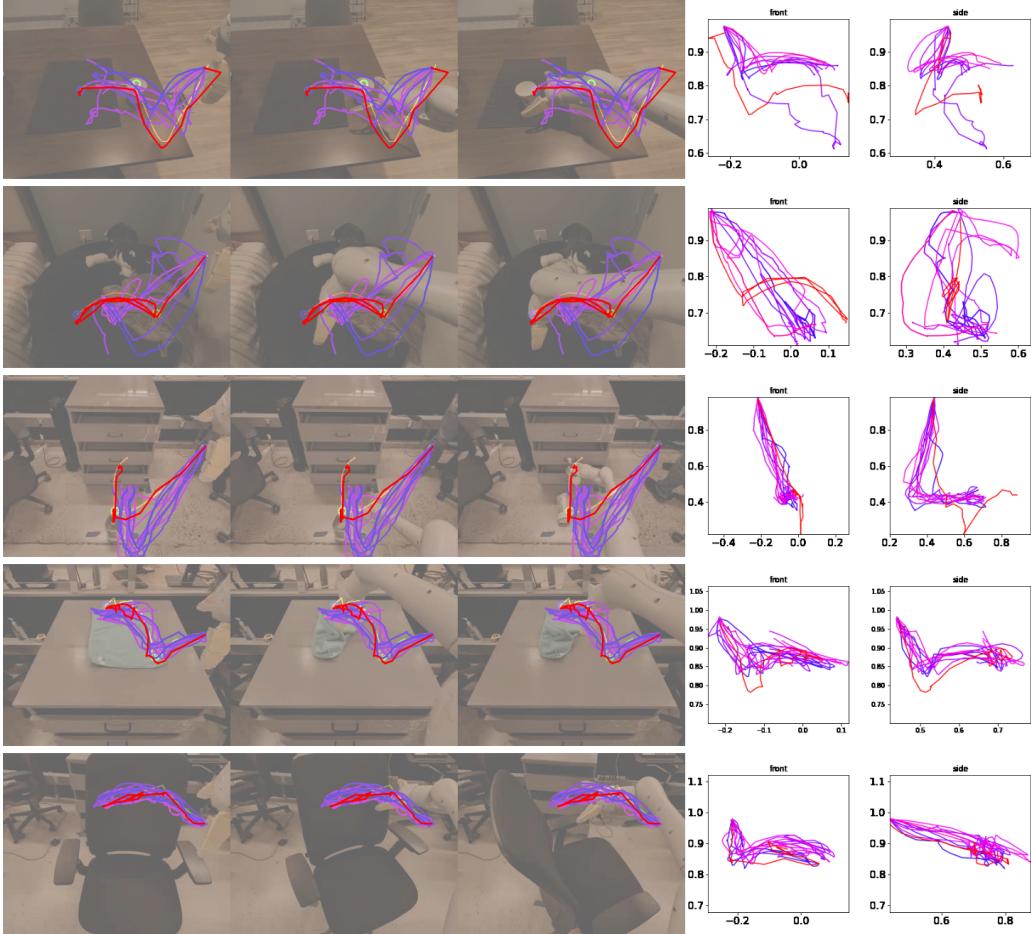


Figure 14: Evaluation trajectories for new skills and their 10 closest trajectories from the training set. Each row shows three frames of a skill evaluation rollout, with the executed trajectory and similar training set trajectories overlaid, as well as depicting the trajectories in an orthographic projection in robot base frame looking at the robot from the front and the side. As can be seen, the policy is able to follow the desired trajectories closely and achieve the tasks. While in many cases, in particular in image space, some of the similar trajectories from the training set look very close to the executed trajectory, the front and side view in rows 1 to 4 reveal that the policy at some crucial point has to - and successfully does - deviate from what it has seen during training. E.g., in row 3 the prompt says to go all the way down to pick up a bottle, while all nearest training trajectories are from `close middle drawer`, which doesn't move the gripper down far enough. Additionally, row 5 is an example where for a `swivel chair` prompt trajectory there coincidentally are many very closely matching `move X near Y` training trajectories. However, the prompt here specifies to not close the gripper at the first contact point, which the policy is able to respect.

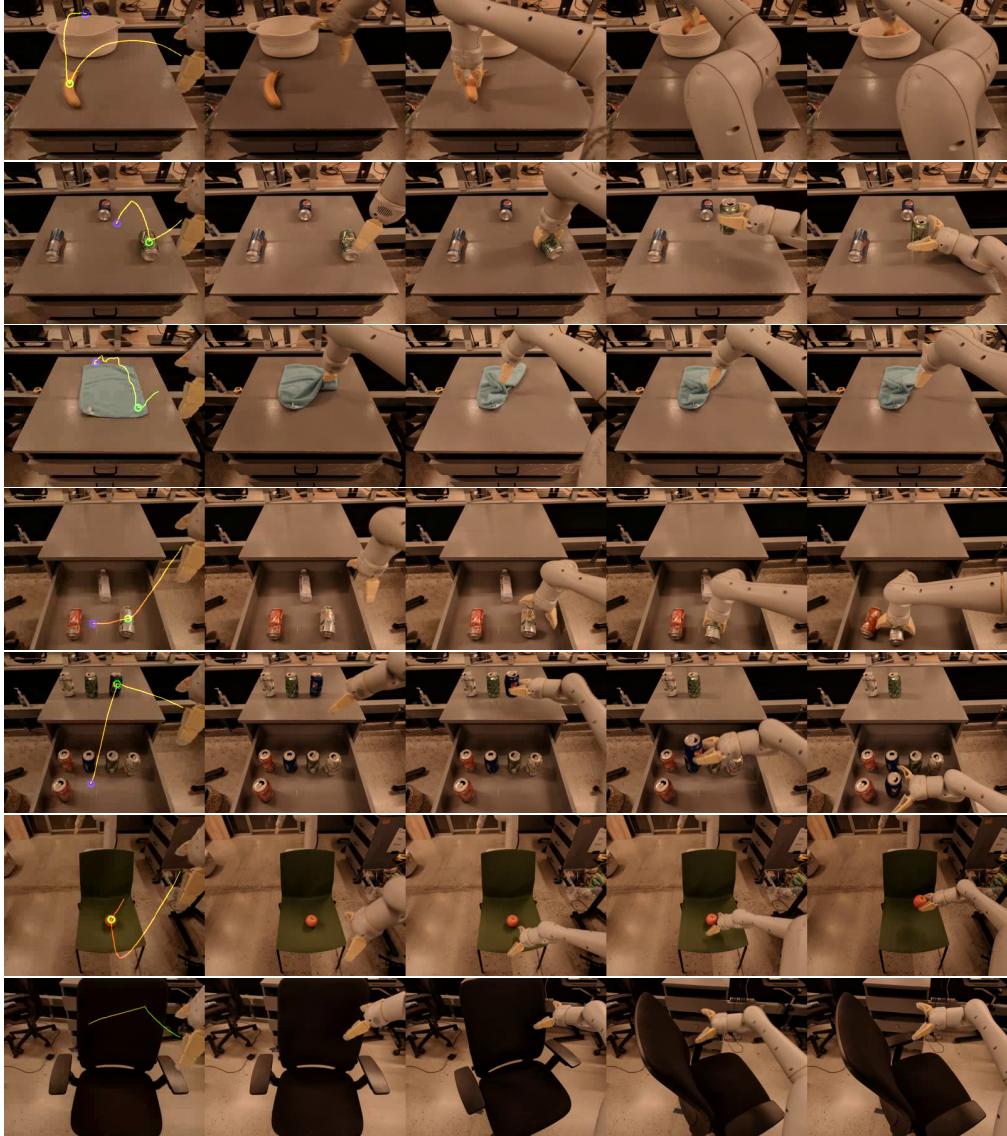


Figure 15: Example rollouts of 7 unseen skills. The trajectory sketch overlaid on the initial image is visualized. From top to bottom: Place Fruit, Upright and Move, Fold Towel, Move within Drawer, Restock Drawer, Pick from Chair, Swivel Chair.



Figure 16: Example trajectories from image generation models. Each row shows the trajectory sketch overlaid on the first frame and the rollout. The language instructions are: open middle drawer, place orange into middle drawer, move 7up can near blue plastic bottle.

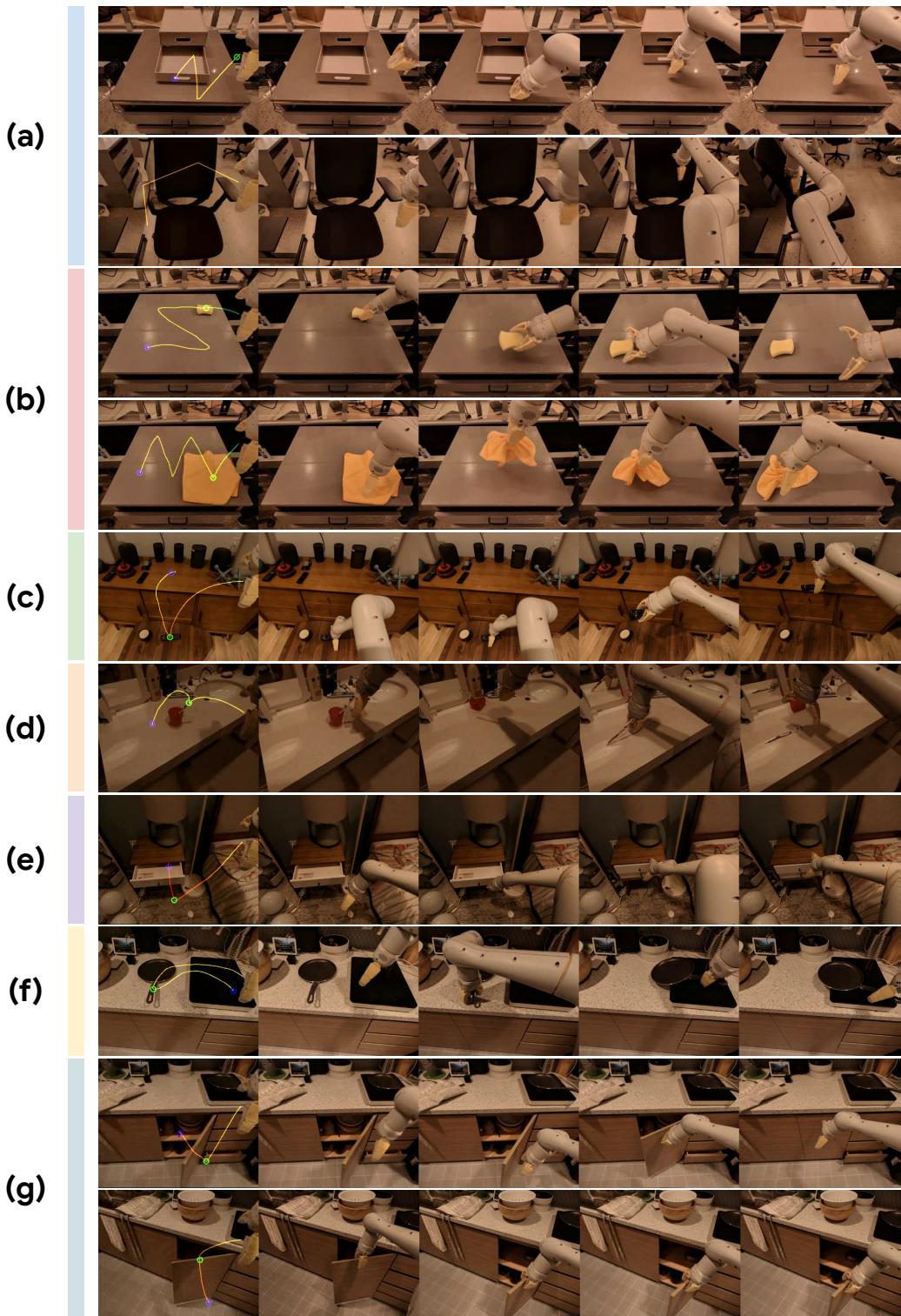
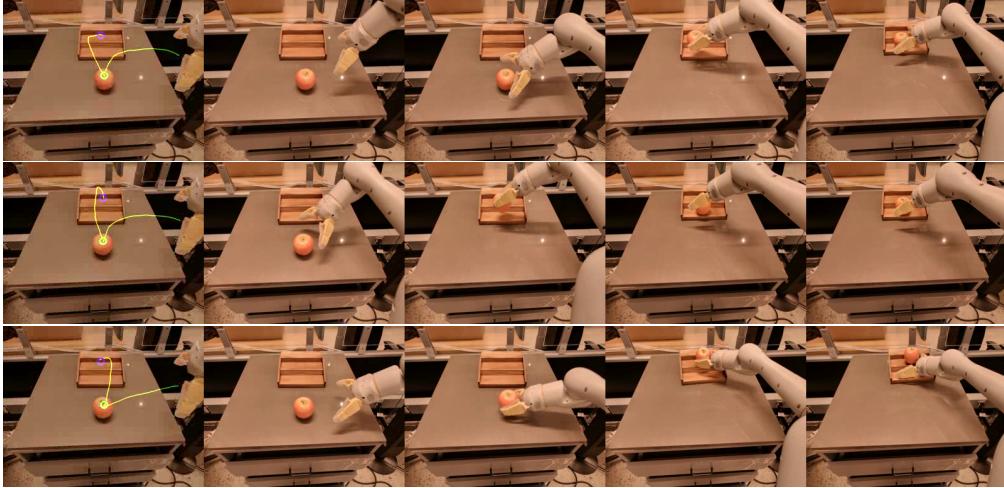


Figure 17: Qualitative examples of emergent capabilities of *RT-Trajectory* in realistic scenarios beyond the training settings: (a) new articulated objects requiring novel motion strategies, (b) new circuitous motions requiring multiple turns, (c) new living room setting with a new height, object, and background, (d) new bathroom setting with precise picking from a cup, (e) new bedroom setting with a drawer at a new height, (f) new kitchen setting with an unseen pan requiring placement onto a new stove, and (g) new kitchen setting with a new pivot hinge that requires a new motion for opening and closing.



Figure 18: Visualizing additional interesting examples of *RT-Trajectory*'s generalization performance in new scenarios. These include a novel kitchen room setting with an unseen cup and unseen placemat, a new living room room setting with new manipulation objects with new furniture pieces in new heights, and a bathroom setting with harsh lighting and different table height.



(a) The objective is to place the apple onto the *middle* stage.



(b) The objective is to place the apple onto the *top* stage.

Figure 19: Case studies in prompt engineering. Each row shows the trajectory sketch overlaid on the first frame and the corresponding rollout. As seen in the first two rows, suboptimal trajectory prompts result in failures. However, by keeping the initial scene conditions identical but simply improving the trajectory prompt, the policy is able to succeed.



Figure 20: Example of retry behavior. The first image is the trajectory sketch generated from the CaP overlaid on the initial observation. The remaining images show the rollout. The robot first attempts to open the drawer by grasping its handle, but fails (2nd image). Then, it retries to open the drawer by grasping the edge instead.

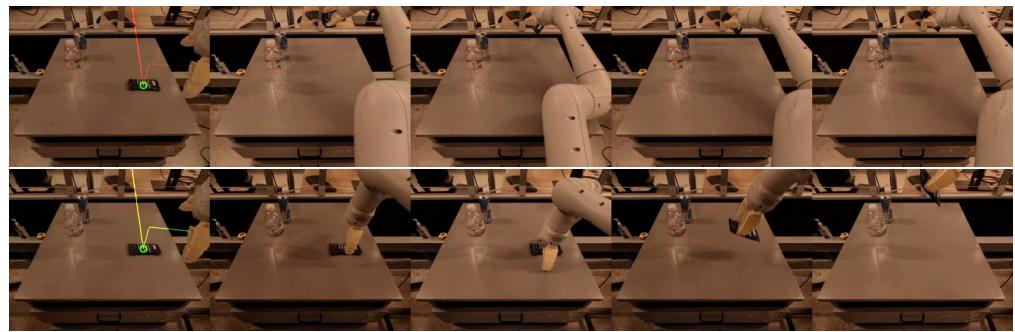


Figure 21: Comparison between *RT-Trajectory* (2D) and *RT-Trajectory* (2.5D). Given the same 2D trajectory generated by the CaP, *RT-Trajectory* (2.5D) lifts the object while *RT-Trajectory* (2D) moves the object to a deeper position due to the ambiguity of a 2D trajectory.