

A APPENDIX

A.1 TETHER POLICY

Correspondence Filtering. As explained in Section 3.1, our primary method of filtering out invalid demo correspondences is triangulation: if the backprojected correspondences do not intersect within a threshold of 10 centimeters, then we deem the demo as an *infeasible* match for the observation o .

However, we find in practice that incorrect correspondences may have a valid triangulation purely by coincidence, thereby passing the feasibility check. In the worst case, this can cause dangerous collisions. As our robot setup lacks built-in collision avoidance, we opt to build a simple custom safety layer via the fast multi-view correspondence algorithm MAST3R (Leroy et al., 2024). For both the demo and observation o , we match each keypoint from its view to the same scene in the other camera view (left to right camera, and vice versa). We then compute the distance of the 3-D waypoint to the ray projected by this match, and if the distance differs between the demo and observation by over 10 centimeters, we deem this correspondence as infeasible.

Post-Warp Speed Adjustment. Another key consideration in implementing trajectory warping is the difference in relative waypoint positions w_t, w_{t+1} between the demo and warped trajectory: if there is a large position difference (e.g., objects closer in demo but farther apart in observation) and we keep the number of intermediate timesteps constant, then the resulting warped trajectory has a large velocity difference as well—which can be dangerous.

Thus, after the warping process described in Section 3.1, we recompute intermediate timesteps such that velocity remains the same between demo and warped trajectories. Specifically, we scale the number of intermediate keypoints T_{segment} by the arc length \tilde{L} of the new action plan segment $\tilde{\mathbf{a}}_t$ relative to the arc length L of the original segment \mathbf{a}_t , $\tilde{T}_{\text{segment}} = \frac{\tilde{L}}{L} T_{\text{segment}}$. Then, we compute new intermediate actions following the relative distribution of the original intermediate actions: for original intermediate action a_t , we map timestep t/T_{segment} to s/L where s is the arc length from w_t to a_t . Then, we interpolate along this function to map new timesteps $t/\tilde{T}_{\text{segment}}$ to \tilde{s}/\tilde{L} , and compute the resulting intermediate action as the position with \tilde{s} arc length to \tilde{w}_t .

A.2 AUTONOMOUS PLAY

Algorithm 1 Autonomous Play

```

1: Require: Tether Policy  $\pi$ , Task Library  $T$ , Demos  $\{\mathcal{D}_t\}_{t \in T}$ , Success Detector  $\mathcal{S}$ , Task Planner
   VLMplanner, Success Evaluator VLMevaluator
2: Hyperparameters: Demo subsample size  $k$ 
3: // Initialize generated trajectory set  $\mathcal{G}_t$  for each task  $t$ 
4:  $\{\mathcal{G}_t \leftarrow \emptyset\}_{t \in T}$ 
5: while TRUE do
6:   Receive observation  $o$ 
7:   // Select a task  $t_{\text{targ}}$  to target
8:    $t_{\text{targ}} \sim \text{Softmax}(-|\mathcal{G}_t|)$ 
9:   // Generate a plan for  $t_{\text{targ}}$  and attempt first task  $t$ 
10:   $\{t, \dots\} = \text{VLM}_{\text{planner}}(o, t_{\text{targ}})$ 
11:  // Select  $k$  demos for  $t$  using UCB
12:   $\mathcal{D} \leftarrow \text{TopK}_k(\text{UCB}(\kappa_i) \mid \kappa_i \in \mathcal{D}_t)$ 
13:  // Run Tether to select demo  $\kappa^*$  and produce rollout  $\tau$ 
14:   $\tau, \kappa^* = \pi(\mathcal{D}, o)$ 
15:  // Check success and update UCB
16:  if  $\mathcal{S}(\tau, \kappa^*, \text{VLM}_{\text{evaluator}})$  then
17:    // If successful, add  $\tau$  to generation set  $\mathcal{G}_t$ 
18:     $\mathcal{G}_t \leftarrow \mathcal{G}_t \cup \{\tau\}$ 
19:     $r \leftarrow 1$ 
20:  else
21:     $r \leftarrow 0$ 
22:  end if
23:   $\text{UCB}(\kappa^*) \leftarrow \text{UpdateUCB}(\text{UCB}(\kappa^*), r)$ 
24: end while
25: Output: Generated trajectories  $\{\mathcal{G}_t\}_{t \in T}$ 

```

Pseudocode. In Algorithm 1 we describe our play procedure in detail.

Multi-Task Design. Our 6 tasks for autonomous play, as plotted in Figure 6 are as follows:

1. Place pineapple from table to shelf.
2. Place pineapple from shelf to table.
3. Place pineapple from table to bowl.
4. Place pineapple from bowl to table.
5. Place bowl from table to shelf.
6. Place bowl from shelf to table.

This set of tasks possesses the approximately indefinitely composable structure as described in Section 3.2: regardless of execution success or failure, the pineapple will be at the table, shelf, or bowl, and the bowl will be at the table or shelf. Then, for any of these combinations as a start state, there is at least one task from the list above that is valid.

VLM Queries. Below, we present the VLM prompts and example responses from our play procedure.

```
Analyze the image depicting a robot environment, containing views from left and right cameras
captured at the same timestamp. What do you see in the scene, focusing especially on the
objects below and their spatial relationships?

The objects in the scene are provided here along with their descriptions and quantities.
Please use their exact names if you refer to them in your response:
<objects>

We want the robot to perform the toy task: <task>. However, the current scene might not be
suitable for executing the task directly. Based on your description of the scene and
logical reasoning over actions, can you create a plan to execute the toy task?

The following atomic actions are available, in no particular order:
<actions>

Important Notes:
- Each atomic action involves only the specified verbs, objects, and locations. Interpret the
  verbs, objects, and locations literally, without making additional assumptions (eg, "
  object on table" means directly on the table, not in a bowl on the table). There are no
  additional assumptions or preparatory actions, and the action will be executed directly
  and literally. Also, pay close attention to the locations specified in the actions (eg "
  from", "to", "on", etc).
- Your plan should be a chronologically ordered sequence of actions, including the task at the
  end. Please ensure that each of your actions are from the list above, verbatim and
  exactly as they are provided. Do not paraphrase or reinterpret the actions.
- An action is valid if its initial conditions, such as the placement or state of objects, are
  completely satisfied (eg, to "place object in drawer" requires the drawer to be open
  beforehand). For subsequent actions, assume that the previous actions were executed
  successfully.
- Your initial action must be valid given the depicted scene, and the subsequent ones must be
  valid for your reasoning of the scene after the previous action. Make sure you do not
  include invalid actions in your plan.
- Your plan should be as concise as possible and should not include extraneous actions that do
  not contribute to the completion of the task.
- If the task is already achieved in the scene, your plan should disrupt the scene so the task
  is not achieved anymore, and then plan to achieve the task. Do not include invalid
  actions that do not change the state of the scene (eg "place into bowl" if the object is
  already in the bowl).

Please think step by step, then filter out invalid actions and provide your final answer in
the following format:
```json
{
 "description": "<description of the scene>",
 "actions": [
 {
 "action": "<initial action to execute>",
 "action_letter": "<list letter>",
 "reasoning": "<reasoning for action>"
 },
 {
 "action": "<next action to execute>",
 "action_letter": "<list letter>",
 "reasoning": "<reasoning for action>"
 },
 ...
]
}
```

```

{
 "action": "<last action to execute>",
 "action_letter": "<list letter>",
 "reasoning": "<reasoning for action>"
}
]
}
...

```

Listing 1: Task Plan Generation.

```

```json
{
  "description": "The scene shows a robot in front of a table covered with a checkered cloth
. To the left of the robot is a cabinet, and to the right is a wooden bookshelf. The
wooden bookshelf has three shelves. On the middle shelf of the wooden bookshelf is a
pineapple toy. On the table, in front of the robot, is a pink bowl. The task is to
pick up the pineapple from the bowl and place it on the table, however the pineapple
is currently on the bookshelf, not in the bowl.",
  "actions": [
    {
      "action": "pick up pineapple from shelf and place on table",
      "action_letter": "e",
      "reasoning": "The pineapple is currently on the middle shelf of the bookshelf, not
in the bowl. To perform the task, the pineapple must first be on the table.
This action moves the pineapple from the shelf to the table, which is a
necessary intermediate step for the task."
    },
    {
      "action": "pick up pineapple from table and place in bowl on table",
      "action_letter": "c",
      "reasoning": "The pineapple is now on the table. To perform the task of picking up
the pineapple from the bowl, the pineapple must be in the bowl. This action
places the pineapple into the pink bowl, which is also on the table."
    },
    {
      "action": "pick up pineapple from bowl and place on table",
      "action_letter": "d",
      "reasoning": "The pineapple is now in the bowl on the table, and the task is to
pick up the pineapple from the bowl and place it on the table. This action
directly executes the desired task."
    }
  ]
}
]
}
...

```

Listing 2: Task Plan Generation Example Response.

```

Analyze the chronologically ordered keyframes depicting a robot attempting to perform the toy
task: <task>. Determine if the task was completed succesfully by the keyframes.

Each keyframe set contains views from the left, right, and hand cameras, all captured at the
same timestamp. The top-left corner of each keyframe indicates its order, with the
smallest number representing the earliest frame.

The objects in the scene are provided here along with their descriptions. Please use their
exact names if you refer to them in your response:
<object_names>

Important Notes:
- Carefully cross-reference all three camera views in each keyframe to determine object states
, gripper actions, spatial relationships, and to resolve depth ambiguities.
- Any unintentional object slip, tip-over, or drop should result in the task being marked as
incomplete.
- Assume smooth transition between the keyframes and nothing special happens between the
keyfraems.

Please think step by step and provide your reasoning. Then, provide your final answer in the
following format:
```json
{
 "reasoning": <reasoning for evaluation>,
 "completed": <true or false>
}
...

```

Listing 3: Success Evaluation.

```

'''json
{
 "reasoning": "Step by step analysis of the keyframes:
- Keyframe 00: The pink bowl is on the table in front of the wooden bookshelf. The robot's
gripper is open and positioned above the table, not yet interacting with the bowl.
- Keyframe 01: The robot's gripper is moving closer to the pink bowl, preparing to grasp it.
The bowl remains upright and undisturbed on the table.
- Keyframe 02: The gripper is now very close to or making contact with the pink bowl, as seen
from the hand camera. The bowl is still upright and in its original position.
- Keyframe 03: The robot has successfully grasped the pink bowl and lifted it off the table.
The bowl is held securely and remains upright in the gripper, as seen from all camera
views.
- Keyframe 04: The robot is moving the pink bowl towards the bookshelf middle shelf. The bowl
is still held securely and upright in the gripper, and is now positioned above the middle
shelf.
- Keyframe 05: The pink bowl is now placed on the bookshelf middle shelf. The gripper is open
and has released the bowl, which is upright and stable on the shelf. There is no evidence
of the bowl slipping, tipping over, or being dropped at any point.

Throughout all keyframes, the pink bowl is handled carefully, remains upright, and is
successfully transferred from the table to the bookshelf middle shelf without any mishaps
. The task is completed as specified."
,
 "completed": true
}
'''

```

Listing 4: Success Evaluation Example Response.

### A.3 EXPERIMENTAL SETUP

**Diffusion Policy (DP) Setup.** For training and deploying all diffusion policies, we use the codebase provided by DROID (Khazatsky et al., 2024), following the original hyperparameters with a few modifications. Specifically, we set the batch size to 32, the shuffle buffer size to 1000, and training epochs to 600. For each task, we ablate the choice of camera views given to the policy: "Pineapple to Shelf" and "Pineapple from Shelf" use the left and right external cameras, while the rest use the wrist and right external cameras.

**Keypoint Action Tokens (KAT) Setup.** Keypoint Action Tokens (KAT) takes a series of teleoperated robot demonstrations, encodes the initial visual observation and actions into a pair of input and output tokens, and feeds the demonstration tokens and current visual observation to generate a trajectory in a new scene. We find that KAT failed to achieve any successes on our tasks, and attempted to make it succeed by changing the three main components of the method: the keypoint tokens, the action tokens, and the large language model used for experiments. We review the attempted fixes for each of the components below.

In the original paper, keypoint tokens are extracted by extracting visual descriptors using DINO-ViT, extracting correspondences using Best-Buddy Nearest Neighbor matching between the descriptors of two initial observations in the input set of demonstrations, and then finding the corresponding location of each selected visual descriptor in the new scene. We find that when using this method to encode observations in our scene, the visual tokens often fall on parts of the scene that are constant across demos but are not relevant to the demonstrated task, such as the shelf and cabinet in the background. In contrast, we find that locations that are crucial to track for the current task, like the location of the pineapple when the task is picking up a pineapple and putting it in a bowl, are not tracked by any of the descriptors. We hypothesize that this is because KAT does not support inputting demonstrations with visual distractors. While they provide experiments showing their method can adapt to scenes with visual distractors at test time, this experiment removes the distractors from the scene when the demonstrations are being collected, meaning that the distractors are added after the relevant descriptors have been selected.

To attempt to support inputting demonstrations taken in scenes with visual distractors, we test manually selecting the relevant keypoints to track the task-relevant locations in an image. For example, for the task of picking up a pineapple and putting it into a bowl, we annotate keypoints on the location of the pineapple and bowl. However, we ultimately find that even with only the task-relevant keypoints marked, the output trajectory is not accurate.

KAT records actions for its demonstrations at 4 Hz. We tested recording the demonstrations with various frequencies in the range between 3 Hz and 15 Hz. We find that after the frequency increases beyond 10 Hz, the model frequently outputs the same trajectory, regardless of the keypoint locations for the input visual observation.

KAT uses GPT-4-Turbo to generate the new trajectory. We tested GPT-4-Turbo, but found that it often outputs invalid action sequences or inaccurate actions. Since we use Gemini-2.5-Flash and GPT-4.1 in our method, we also test these two models' ability to output accurate trajectories. We find that while both models output validly formatted action sequences, the output action sequences are not accurate behaviors. We believe that this is likely due to the failure of the LLMs to handle the multi-dimensional numerical pattern that this task requires it to learn due to the orientation changes and non-linear velocities needed to complete our target tasks.

#### A.4 EXPERIMENTAL RESULTS

Method	Pineapple from Shelf	Pineapple to Bowl	Bowl to Shelf	Bowl from Shelf
Tether (10 Demos)	$1.0 \pm 0.0$	$1.0 \pm 0.0$	$0.9 \pm 0.09$	$1.0 \pm 0.0$
Tether (5 Demos)	$0.8 \pm 0.13$	$0.8 \pm 0.13$	$1.0 \pm 0.0$	$1.0 \pm 0.0$
Tether (1 Demo)	$0.8 \pm 0.13$	$0.9 \pm 0.09$	$0.7 \pm 0.14$	$1.0 \pm 0.0$
$\pi_0$ (Black et al., 2024b)	$0.5 \pm 0.16$	$0.7 \pm 0.14$	$0.0 \pm 0.0$	$0.4 \pm 0.15$
DP (Chi et al., 2023)	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.1 \pm 0.09$
Method	Apple to Bowl	Strawberry to Bowl	Pineapple to Basket	Pineapple to Cup
Tether (10 Demos)	$0.9 \pm 0.09$	$0.9 \pm 0.09$	$0.9 \pm 0.09$	$0.9 \pm 0.09$
Tether (5 Demos)	$1.0 \pm 0.0$	$0.9 \pm 0.09$	$0.9 \pm 0.09$	$0.9 \pm 0.09$
Tether (1 Demo)	$0.9 \pm 0.09$	$0.9 \pm 0.09$	$0.9 \pm 0.09$	$0.5 \pm 0.16$
$\pi_0$ (Black et al., 2024b)	$0.6 \pm 0.15$	$0.7 \pm 0.14$	$0.4 \pm 0.15$	$0.1 \pm 0.09$
DP (Chi et al., 2023)	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$
Method	Wiping with Cloth	Opening Cabinet	Hanging Tape	Inserting Coffee
Tether (10 Demos)	$0.6 \pm 0.15$	$0.6 \pm 0.15$	$0.7 \pm 0.14$	$0.4 \pm 0.15$
Tether (5 Demos)	$0.0 \pm 0.0$	$0.3 \pm 0.14$	$0.3 \pm 0.14$	$0.2 \pm 0.13$
Tether (1 Demo)	$0.1 \pm 0.09$	$0.1 \pm 0.09$	$0.1 \pm 0.09$	$0.2 \pm 0.13$
$\pi_0$ (Black et al., 2024b)	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$
DP (Chi et al., 2023)	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$	$0.0 \pm 0.0$

Table 1: Success rates and standard error measured over 10 trials.

**Policy Comparison.** In Table 1 we report the exact figures from Figure 5 along with standard errors.

**Intermediate Tether Visualizations.** We visualize intermediate steps of our Tether policy from the evaluations in Section 4.2. Specifically, we include annotations for semantic correspondence, where the red dot indicates the correspondence match between source (demo) and target (scene), and the blue dot is the MAST3R match between viewpoints. Additionally, we include a few visualizations of our trajectory warping. Note that the demonstration depicted is the one selected by our method as the most similar to the scene.

In Figures 8 and 9, we visualize examples of successful semantic generalization: from the pineapple to strawberry and bowl to cup. These demonstrate the key ability of semantic correspondence to generalize beyond the demonstration, enabling interactions with out-of-distribution scenes.



Figure 8: Computed correspondence for the Pineapple to Cup task.

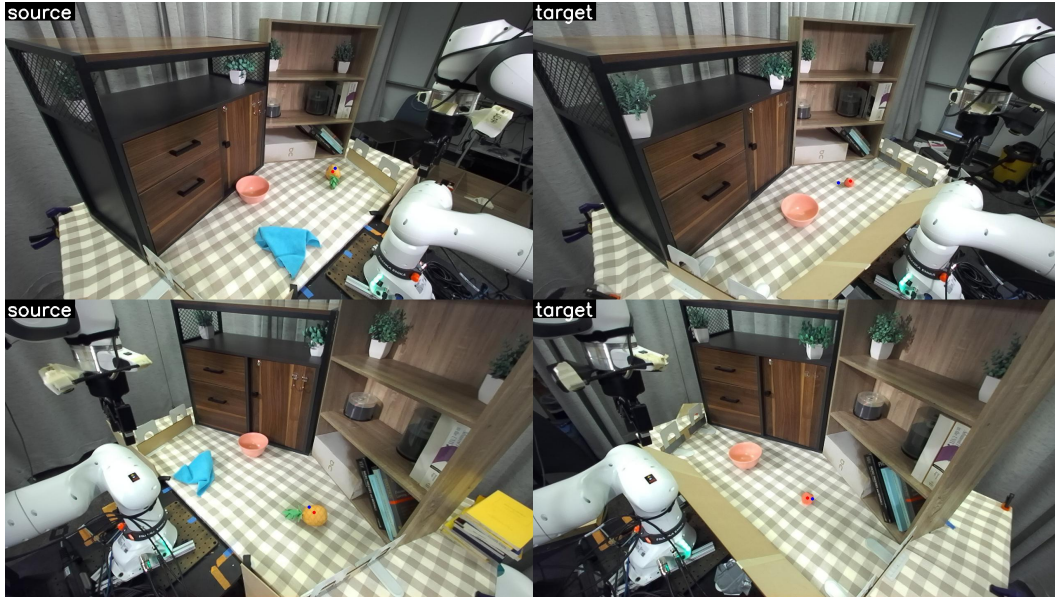


Figure 9: Computed correspondence for the Strawberry to Bowl task.

In Figures [10](#) and [11](#), we visualize examples of successes in hanging tape and wiping the whiteboard with cloth. The former shows the accuracy of correspondence even with small objects like the silver hook, and the latter demonstrates correspondence with deformable, non-rigid objects like the cloth.

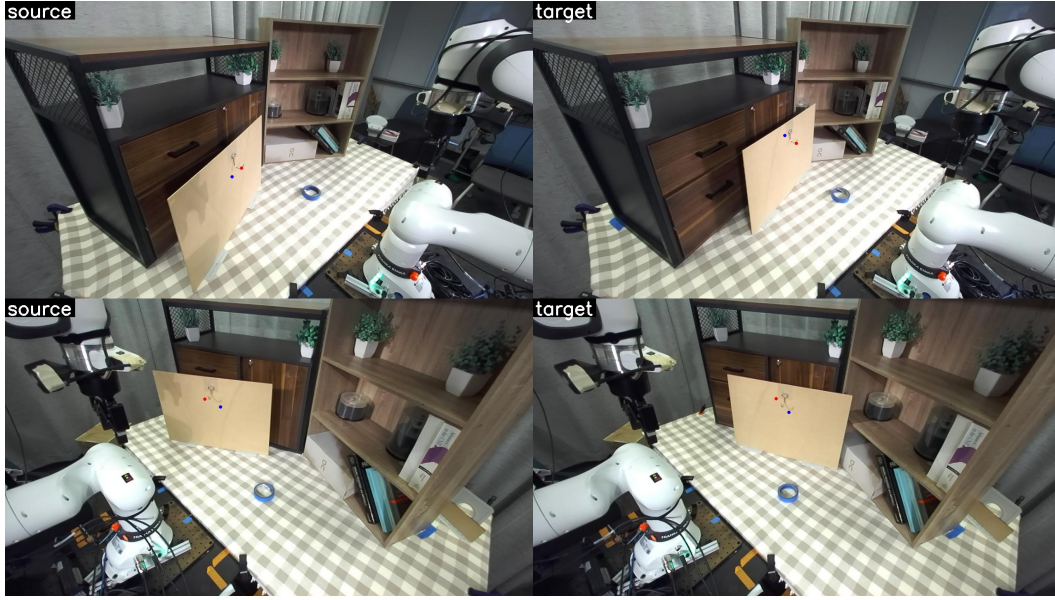


Figure 10: Computed correspondence for the Hanging Tape task.

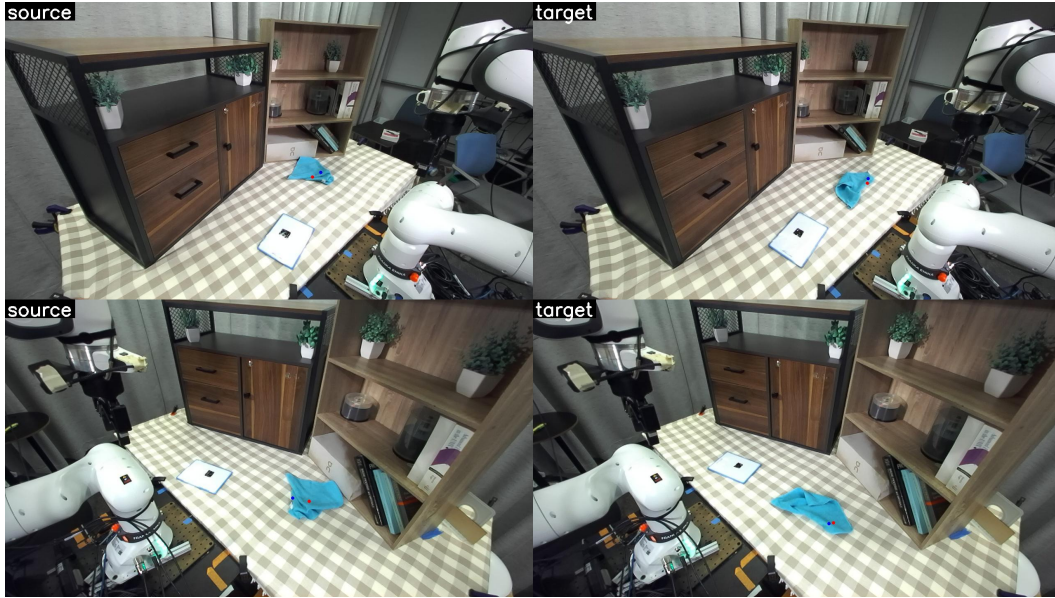


Figure 11: Computed correspondence for the Wiping with Cloth task.

Finally, in Figures [12](#) and [13](#), we visualize examples of the full warped trajectory, with the source demo on the left, the warped result on the right, and annotated waypoints along the trajectory.

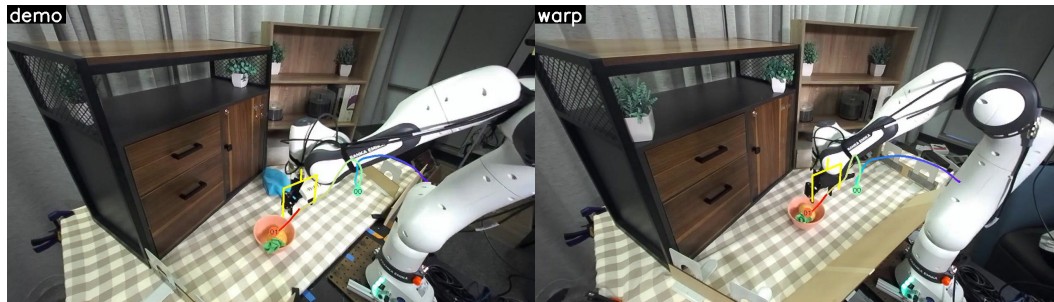


Figure 12: Trajectory warping for the Pineapple to Bowl task.

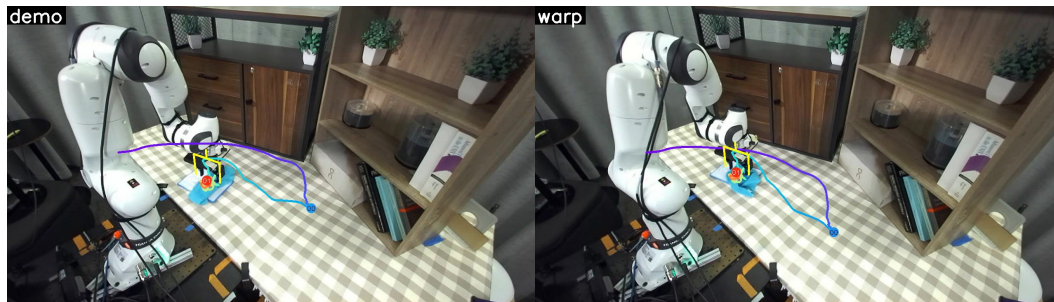


Figure 13: Trajectory warping for the Wiping with Cloth task.

**Play Timelapse.** We record a subsection of our 26-hour long autonomous play experiment. The timelapse, sped up by 100x, can be viewed here: <https://iclr-2026-15740.github.io/supplemental/>