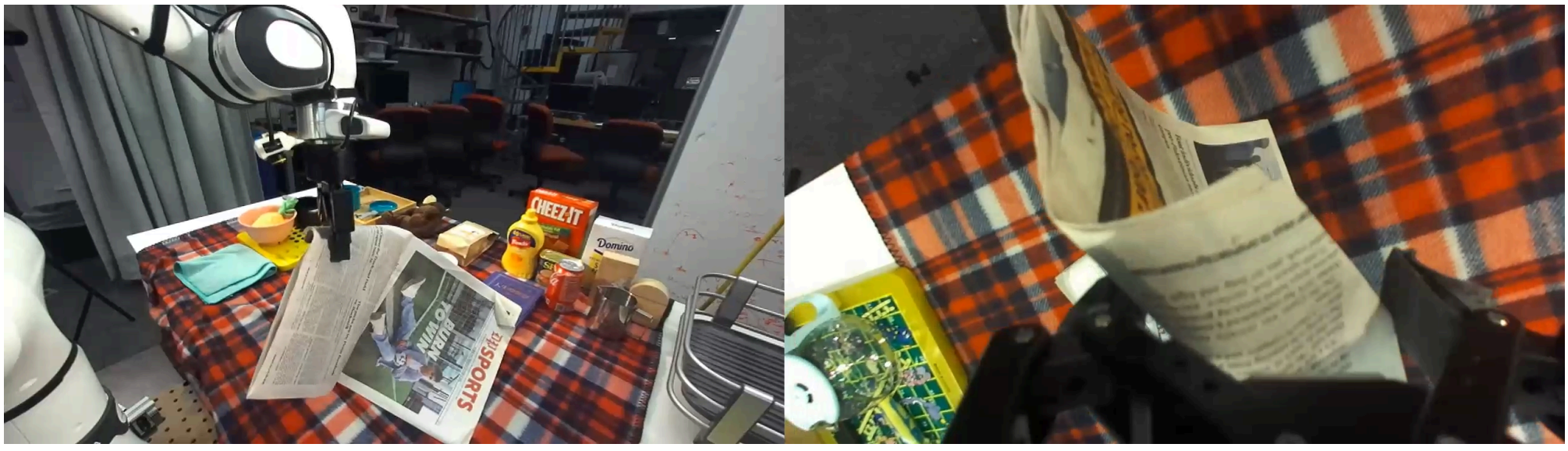
Evaluating Pi0 in the Wild @ GRASP Lab: Strengths, Problems, and the Future of Generalist Robot Policies

Jie Wang[1,*], Matthew Leonard[1], Edward S. Hu[1]

[1]University of Pennsylvania

Pi0-FAST Paper DROID Dataset Pi Website



"Fold up the newspaper"

Introduction

Pi0 is a state-of-the-art vision-language-action (VLA) model designed for general-purpose robotic manipulation. Built on internet-scale pre-training and the Open X-Embodiment dataset, Pi0 promises versatility across tasks like pick-and-place, articulation, dexterity and human robot interaction. But how well does it actually perform in the wild?

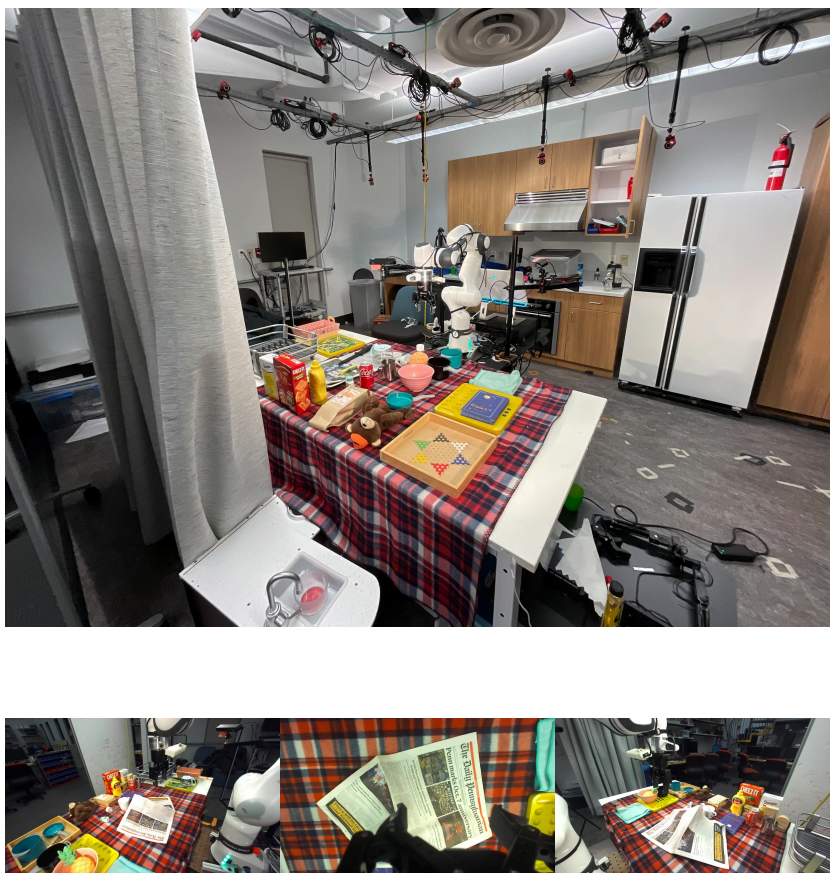
We are PAL Group and Daniilidis Group from GRASP Lab at the University of Pennsylvania. As contributors to the **Distributed RObot Interaction Dataset (DROID)** dataset, we are fortunate to have early access to the PI0 model. Our robot setup can be found here. With no training, no finetuning, we just deploy the *PI0-FAST-DROID* and use it zero-shot. [Edward's part] XXX, we largely test its performance on a variety of complex tasks, through pi0 into difficult scenes in the wild. The philosophy of our evaluation is: more in-the-wild, less controlled / lab setup. We conduct experiments in a vibe style, where we observe pi0 like biologists observing a new animal - conducting interesting tests, looking for broad, qualitative properties, to encourage others to try out pi0, this amazing foundation model for manipulation.

In this blog, we share results from **250+ trials** testing Pi0 on a Franka Research 3 robot. We explore its strengths, unexpected quirks, and limitations—and what this means for the future of imitation learning.

Key words: pi0, VLAs, Manipulation, Robot Learning, Imitation Learning.

TL;DR

- **Overall Success:** 42.3% average task completion across 300+ trials
- **Key Finding:** Pi0 demonstrates impressive vision-language understanding but struggles with spatial reasoning and precise manipulation
- **Bottom Line:** Pi0 works impressively in the wild without fine-tuning, but success depends heavily on instruction phrasing and object familiarity



Our evaluation setup at GRASP Lab, showing the robot workspace, test objects, and typical manipulation scenes.

Key Findings

Through experiments of Pi0 across diverse manipulation tasks in the GRASP Lab, we observed a wide range of behaviors—remarkable performances, confusing failures, and many quirks. By testing Pi0 in varied environments (e.g., cluttered tables, articulated cabinets, human-involved settings), we concludes three critical insights:

1. **Pi0 Works (Mostly)**: It achieves **52% average progress** across tasks, even in unseen environments and objects. However, it can fail in some seemingly simple tasks.
2. **Prompt Engineering Matters**: While Pi0 is good at vision-action grounding, its **performance drops 20~100%** based on instruction phrasing. You need to carefully prompt it to make it work in some condition.
3. **Unexpected Quirks**: Pi0 can recover from failures, handle moving humans in the scene, but struggles with **mid-task freezing, collision avoidance, and fine-grained manipulation**.

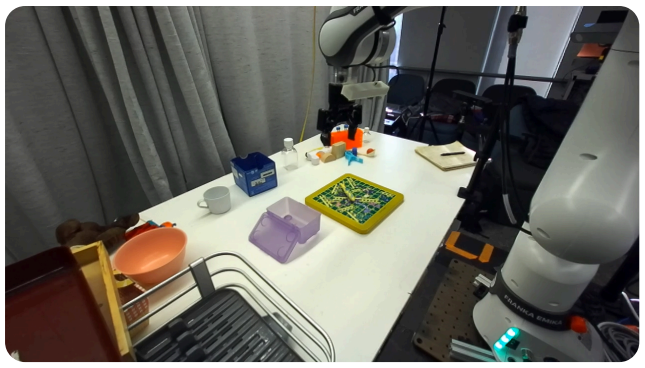
Below, we unpack each finding with video examples and analysis. For detailed metrics, see Section 4: Results.

Success Cases

Pick and Place

Articulation

Human Robot Interaction

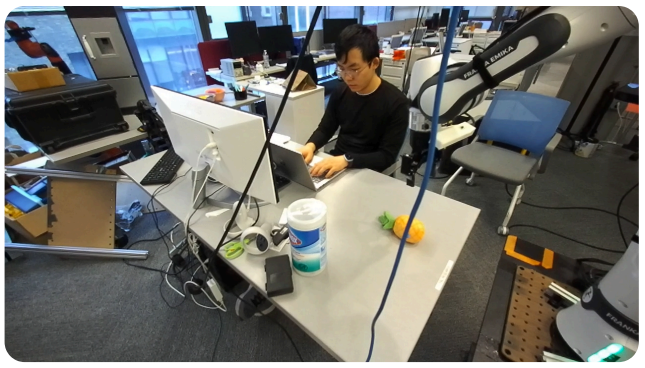


"Place the yellow fish into the purple box"

Precise placement of camouflage fish into the box

"Open the drawer"

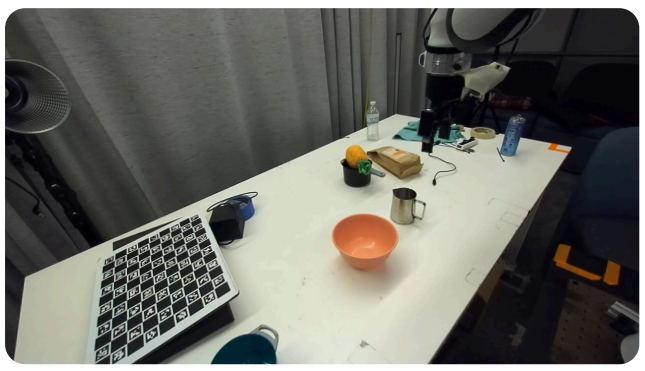
Opening drawer with multiple pull to make sure fully opened



"Hand the pineapple to the programmer"

Safe object handover to programmer, even there is wire occlusion from side view

Dexterity



"Pour water from the silver cup to the pink bowl"

Pour real water from Latte art vat into the target bowl

Multi-step Task

"Pick up all the objects into the basket"

Sequential bagging all the toys into basket

Novel Objects

"Fold up the newspaper"

Handling previously unseen items

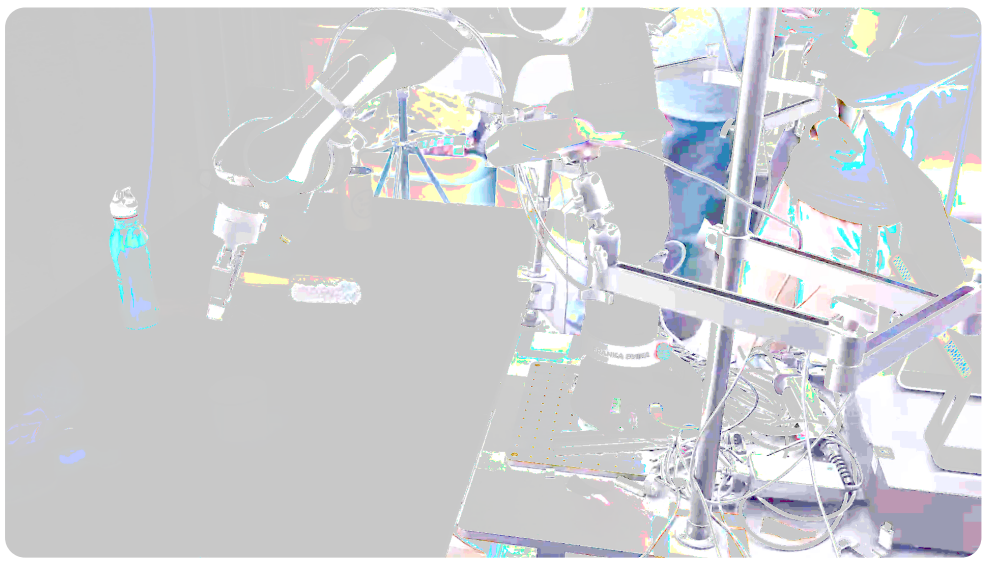
2.1 Remarkable Performance of PI0

2.1.1 Robust Vision-Language Understanding in Complex Scenes

Powered by **PaliGemma** (Google DeepMind's 3B VLM) as its vision encoder, Pi0 demonstrates robust scene comprehension and adaptability. Despite relying solely on uncalibrated monocular RGB inputs (224×224 pixels after compression), it can handle very challenging objects and environments like transparent, camouflaged and novel items. Showing the potential of End2End VLA in precision, closed loop control.

1. It can grasp transparent object

PI0 is capable of identifying and manipulating transparent objects, as shown below. It picks up the bottle with a stable grasp, aligns it to the small cup, and precisely drops it in. Many traditional grasp detection techniques require an accurate 2D or 3D reconstruction of the scene, and transparent objects can cause issues in reconstruction accuracy. This makes it all the more impressive that the model can detect transparent objects solely from uncalibrated, mono-RGB views from the wrist camera and one side view camera.

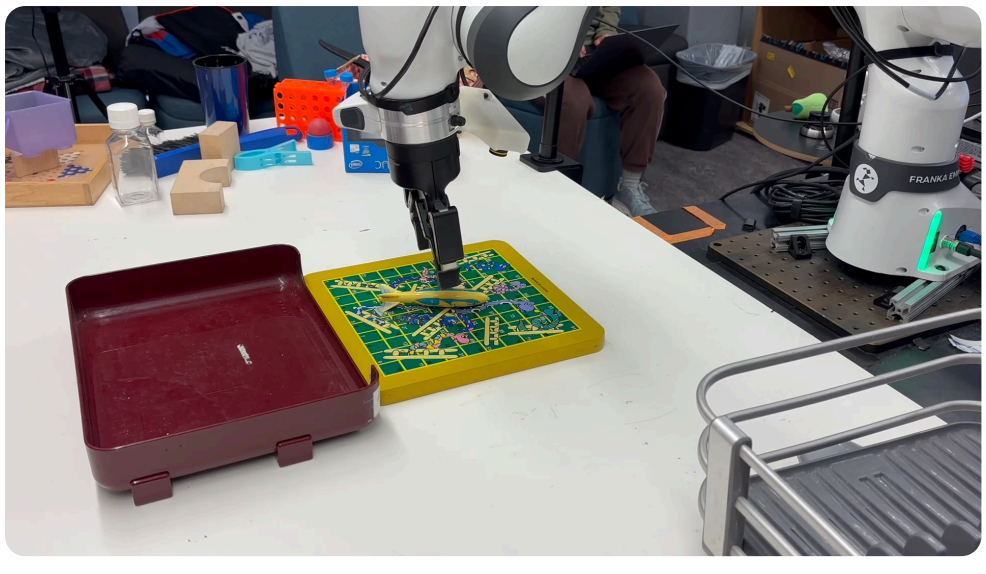


"Place the plastic bottle into the white cup."

"Place the plastic bottle into the white cup."

2. It can grasp an object even when it is camouflaged into a colorful background

PI0 can identify the 'yellow fish' here even when it is placed on top of a colorful board game. This object has an unusual and difficult shape, and it blends in well with the background, but pi0 detects well with its position, and its shape, well enough to grasp it up.



"Place the fish into the red box"

"Place the fish into the purple box"

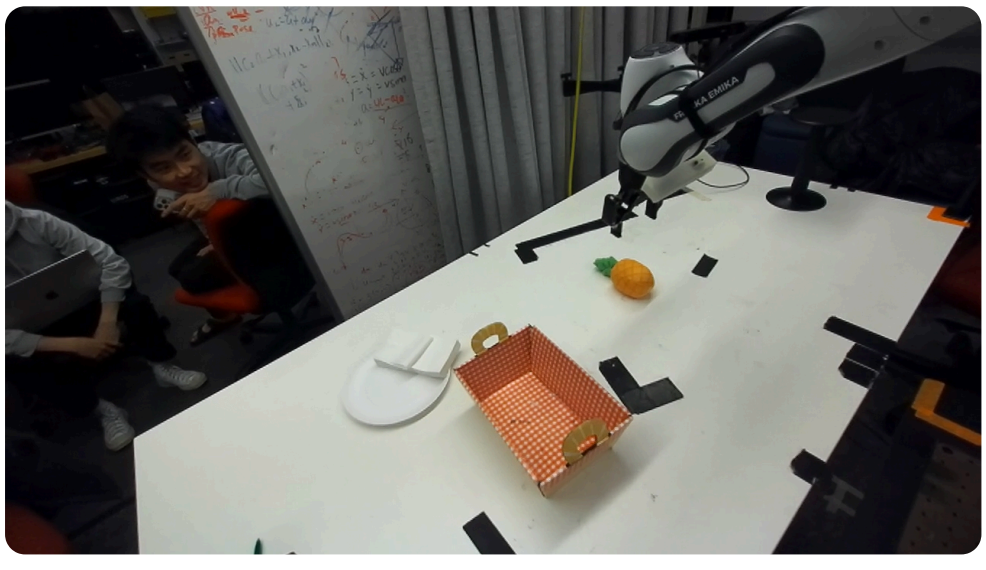
3. It is robust to human activity in the input

During experiments, there were many scenarios where the side-view camera captured humans moving around in the background. However, pi0 was still focused on the task, keeping the robotic arm's movements focused on object manipulation.

We believe there are two reasons for pi0's robustness to human movement. First, the pre-trained VLM backbone of Pi0 is trained on human-involved images, so humans are in-distribution. Next, from our occlusion experiments in **Section 2.3.1**, the policy seems to prioritize the wrist camera's images during pick-and-place tasks, so distractors in the side-view camera seem to minimally affect the policy.

Here are two side-view videos involving humans in the scene. Please refer to **Section 4.6** for more results on **human-robot interaction**.

(All videos featuring humans were uploaded with permission from the individuals involved.)



"Pick the pineapple and place it into the basket"

"Place the black knife into the basket"

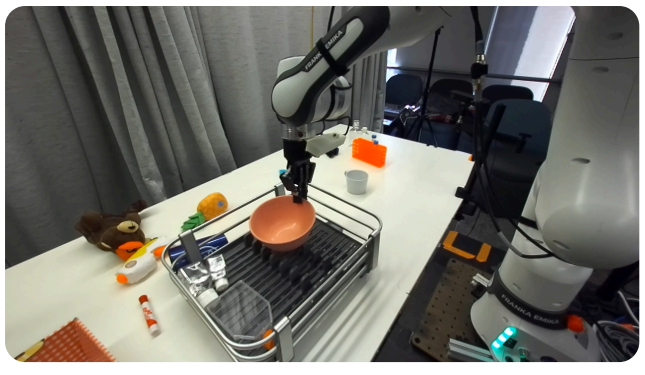
There's plenty of work in the past in CV, Robotics that does transparent object detection and manipulation. But the nice part here is that we have a data-driven system that does it, without any special logic or care for transparent objects.

"Pi0's ability to handle transparency and clutter hints at a future where robots see the world as humans do—through semantics, not just pixels."

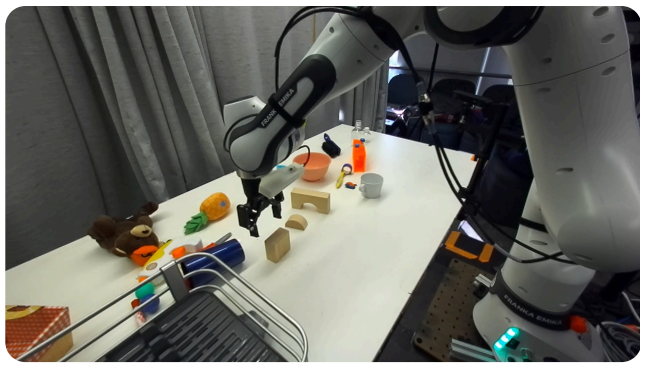
2.1.2 High frequency dexterity of robot policy

Even though pretrained on massive datasets, Pi0 can perform high-frequency, closed-loop control up to 50Hz. Despite a **90ms-300ms delay** introduced by HTTP connections between our laptop and the GPU server, Pi0 achieves real-time responsiveness. It can do precise manipulation of complex objects like newspapers, T-shirts, and articulated tools.

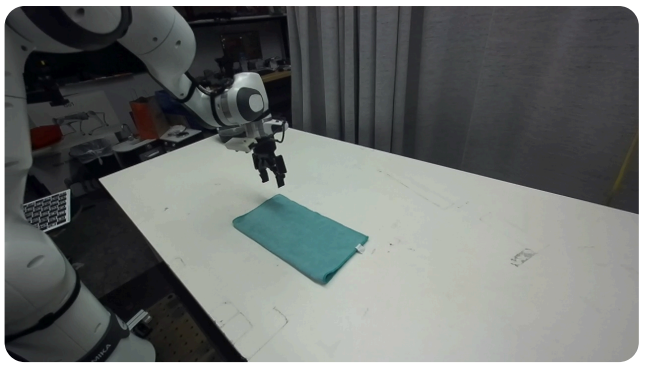
Here are some impressive cases demonstrating the dexterity and precision of pi0. Notice all videos in this blog are **uncut and unedited** from the ZED 2 camera at **1x speed**.



"Remove the pink bowl from the tray"



"Stack the wooden blocks"



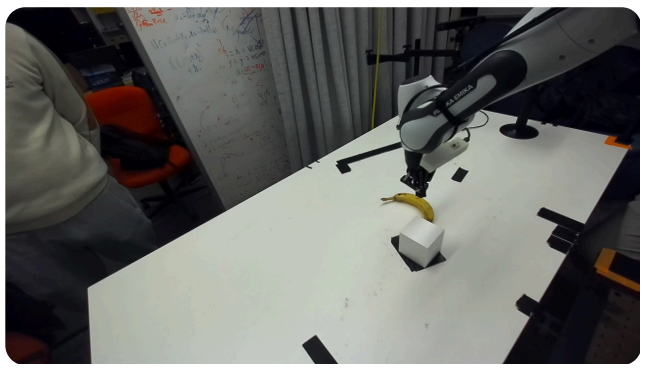
"Fold the cloth from left to right"

According to the PI0 paper and pi0-FAST, we think the dexterity stems from its flow matching architecture. Recently there are some efforts to fasten VLA with 7B VLM encoder. We look forward to more strong models running at a higher frequency!

2.1.3 Failure recovery and generalization

One key difference between the pi0 and other models is: it can work in the wild, without online data collection, without calibration, without much position tuning on the policy, we can let it run on our franka robot directly.

More interestingly, the policy behaves similar to diffusion policy, as shown below, it could recover from failures and retry tasks that initially failed.



"Pick up the banana"

Failture Case

Pi0 demonstrates very impressive robustness across different lights, locations and tasks. However, it do exist some weakness, here is some examples on its cons:

OOD Objects

"Place the beaker into the pink bowl"

Cannot find the transparent glass beaker

OOD Background

"Pick the black box on the white box"

Can not handle well with unseen background

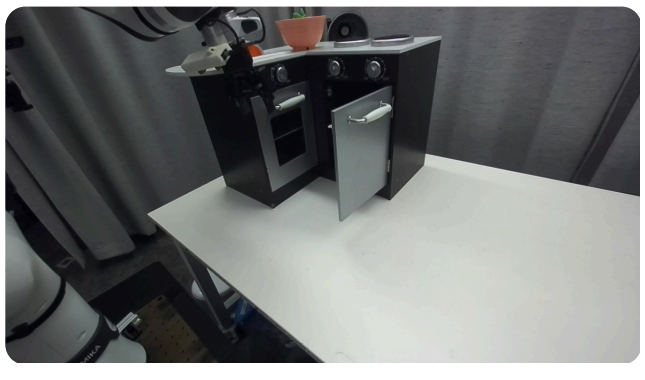
Task Misunderstanding

"place the yellow fish into the basket"

Pick up the wrong object in cluttered scene (25% progress)

Spatial Reasoning

General Articulation



Coffee Making

"Place the can into the tray"

Misjudges object position relative to container (19% error)

"Close the right cabinet door"

Fails to open toy kitchen cabinet on the table (0% progress)

"Pour coffee bean into the grinder"

Cannot work with espresso machine (10% progress)

2.2 Problem with PI0

Early stopping

A common failure case is the robot freezing in the middle of execution, due to the policy outputting stop commands prematurely. For example, in the opening drawer test, the robot stops after its gripper grasped drawer handle. It seems the action is to stop and wait. However, the scene is static, so the robot will just keep stopping until human intervention. If we move the side view camera, or we move the robot base a bit, the policy will start to get out of 'local optimum', keep trying to complete the given task.

There are two type of reasons causing stopping:

1. Stop by semantic understanding

The robot's language capabilities are limited. When it does not understand a command, it gets stuck. In some experiments, we found that the object we wanted to test was out of the vocabulary of the robot, causing it to stop moving for the rest of the episode. We directly evaluated the robot's ability to choose between two objects based solely on the text description, and found that it often failed to pick between the two objects. The robot would move in between the two objects, and then fail to move any further.

"Pick up the black knife"

2. Stop by frame prediction

Stuck is caused by 'Autoregressive' fashion of PI0 --- a core problem with this framework. The model falls into 'local optimum' for that frame, i.e. stop and wait. Also, when pi0 receives a command it doesn't understand, like out of distribution(OOD) objects. It will also halt. Take the toy gun pick-and-place task as example, it stops because it can't recognize which object is 'gun', and thus treats the scene as complicated.

One thing interesting is how pi0 learns 'stop/terminate state' from demonstration, some of which ending may be some tasks' middle point. In the ablation study below, we test how pi0 behaves when we move the side view camera during policy inference. It seems the side view camera greatly affects the initial observation and termination condition. When the robot executes a task and goes back to its initial position to stop, if the side view camera is moved, then the robot arm will move to set itself into another ending position.

In our understanding, STOPPING phenomena could be potentially because of pi0's autoregressive prediction behavior. PI0-FAST-DROID generates an action trunk at 15 Hz, it keeps requesting the 3B VLA with 2 current images (wrist + side view) and text instruction. For each query, the robot is in fact open-loop control; it doesn't encode history / past observations / past action into current reasoning. But the query is fast enough for following action to recover from failure, reaching a dynamic, adaptive planning over spatial and temporal closed-loop control. But this nature also makes long-horizon manipulation, like tasks involving history particularly hard for pi0 to learn, which leaves much space for research.

Imprecise spatial reasoning

Pi0 often struggles with spatial reasoning over height. For example, when asked to pick up an object and place it into a container, the policy can not lift the object high enough to clear the height of the container. This suggests the drawback of vision-based autoregressive VLA: it does not have a metrically accurate method to estimate the distance between gripper and surrounding environment. As shown in the video below, the robot seems to think the gripper is high enough, and so it pushes the destination

object. If the robot were able to accurately estimate the size of the object that it holds, and that of the gap between the height of the object and the bowl, and understand that it must increase the gap, it could complete the task successfully.

We also tried to prompt pi0 to raise the gripper higher. But it didn't learn on this, making it easy to collide with the container. When pi0 is asked to operate with an articulation object, it becomes even harder for it to estimate the distance from the side view camera, causing frequent collisions. This is particularly worth noting when the robot is intersecting with humans. As it doesn't have an explicit constraint map, it will tend to hit / grasp the user's hand, which is unsafe and hurt per force.

Please refer to section 4.6 for more information.

What's more, when Pi0 is required to manipulate a new, household appliance, it will tend to collide the device or stop during inference trials. As shown below, it seems not compilable with the Coffee Machine in our lab. For more videos, please refer to section 4.7 result for more details on it.

"Take out the cup under the coffee machine" -> collide top of coffee machine

"Take out the silver cup from the coffee machine" success - grasp the handle beautifully

One possible solution is to involve concepts like Vox map and planning constraint. Depth camera is also very helpful to implement collision avoidance.

Sensitivity to Distribution Shift

1. Orientation of objects significantly affects performance

Initial condition of both objects & robot could cause success rate change a lot. As shown below, when the spam can is parallel to the viewer, the robot fails consistently. When perpendicular, it succeeds consistently. This orientation bias likely stems from the distribution of demonstrations in the training data, where certain viewpoints were overrepresented.

We test in which angle could robot open the cabinet door. It seems 45 degree is a sweet spot.

2. Grasps at awkward spots on some objects, resulting in drops

There is no built-in, geometry-aware grasp detection system. This highlights a fundamental limitation of behavior cloning approaches like Pi0.

- **Demonstration Bias**: The policy shows clear preferences for grasping objects in ways that mirror the training demonstrations, even when those aren't optimal for the current scenario. For example, we observed pi0 attempts to pick up cups from their edges rather than their handle. And the policy tends to grasp marker pens even if they are not requested to. The policy actually is trying to "pick up the most familiar object" it saw in the dataset. (Like in DROID, marker pen takes 16.1% of all the objects.)
- **Distribution Mismatch**: When objects appear in positions or orientations underrepresented in the training data, performance drops dramatically. This covariate shift is a well-known weakness of behavior cloning - the policy hasn't learned to recover from states outside its training distribution.

The model doesn't actually know where & how to manipulate objects semantically - it can only reproduce patterns from demonstrations without true understanding of object affordances or functional parts. Though scaling up data may solve some scenarios, there are countless edge cases that need to be labeled. We need some more effective representation like autonomous driving met.

Potential solutions might involve augmenting behavior cloning with:

- Explicit object-centric representations to identify functional parts
- Adversarial training to improve robustness to distribution shifts
- Incorporating physical priors about stable grasps and object affordances

2.3 Quirk: Some interesting behavior of pi0

2.3.1 Good quirk:

Quirk: Active Perception & Viewpoint Robustness

One of the most frequently asked questions on Pi0 is, how robust it is when the sensory inputs are disrupted? We did several tests about blocking the camera, the object and moving the side view camera. Here's what we learned.

Camera Blocking Experiments

Setup:

- **Task**: "Pick up the pink object and place it into the bowl."
- **Cameras**: Side-view (primary) + wrist-mounted (secondary).
- **Blocking Scenarios**: Partial/full occlusion of one or both cameras.

Blocking Type	Success Rate	Behavior
No Block	100%	Flawless execution. Picks pink object, attempts secondary objects.
Block Side Camera Mid-Trial	50%	Relies on wrist cam. Gripper misalignment → partial success.
Block Wrist Camera Entirely	0%	Frozen—no recovery.
Block Both Cameras Initially → Unblock Mid-Trial	70%	Chaotic exploration → stabilizes after unblocking.

Key Observations

1. **Active Perception**:
   - When the side camera is blocked, Pi0 uses its wrist camera to retry grasps ("hand-eye coordination").
   - Example: In Trial 3 (side camera blocked), Pi0 adjusted its gripper position twice to finally secure the pink object.
2. **Viewpoint Robustness**:
   - Pi0 tolerates shifted camera angles mid-task.
   - Blocking then unblocking cameras triggered exploratory movements (e.g., sweeping the arm to re-localize objects).
3. **Failure Modes**:
   - Total wrist camera blockage → robot freezes (no fallback perception).
   - Over-reliance on initial frames: If objects shift after blocking, Pi0 struggles to update its spatial model.

Object Blocking Experiment

Setup:

- **Task**: "Pick up the red box."
- **Occlusion Levels**: None (fully visible), 50% occluded, 100% occluded.
- **Behavior Metrics**: Success rate, recovery attempts, and failure modes.

We sought to determine how the robot would respond when the target of its task was occluded. In order to do this, we asked it to grasp a box, varying the occlusion level from fully visible, to 50%, to 100% occluded. When the box was fully visible, the robot would succeed. When it was 50% occluded, the robot would generally succeed, sometimes moving the occluding object in order to get a better view or grasp of the target. When it was fully occluded, though, the robot would become confused and stop. During one fully occluded trial, the robot knocked over the occluder and the red box, revealing more information about the location of the target but making it impossible for the robot to proceed with a good grasp.

Why This Matters

Pi0's ability to **improvise with partial observations** hints at emergent active perception—a critical feature for real-world deployment where lighting, occlusions, or camera failures are inevitable.

Left: Blocking the side camera forces Pi0 to rely on wrist-view. Right: Success rates across blocking scenarios.

This quirk underscores Pi0's potential as a resilient generalist—though it's no substitute for dedicated SLAM or depth sensing... yet.

Camera blocking experiment

2.3.2 Bad quirk:

Quirk 1: Opening > Closing

Pi0 exhibits unexpected behaviors when interacting with articulated objects (e.g., drawers, cabinets), Pi0 achieves higher success rates when **opening** articulated objects (e.g., drawers) compared to **closing** them.

Examples:

- **Task**: "Close the drawer."
  - **Behavior**: Pi0 grasps the handle, attempts complex maneuvers (e.g., twisting), and often fails to align with the articulation axis.
  - **Result**: **60% success rate** for opening vs. **35% for closing** (see video).
- **Toy Cabinet or Real Drawer on table Challenge**:
  - **Task**: "Close the toy kitchen cabinet."
  - **Behavior**: Pi0 freezes or pushes randomly when it is required to manipulate with articulated objects on table.
  - **Result**: **0% success rate** (see video).

Possible Cause:

- Pi0's training data prioritizes **opening actions** (common in datasets like DROID).
- Closing requires precise backward articulation, which the model struggles to infer from RGB-only inputs.

In general, closing things is much easier than opening. While pi0 tends to execute some more fancy actions to "close one thing". Like the robot will firstly grasp the drawer handle, then try to close it. While the closing task could be generally easy to move along the articulation axis.

Accelerated video of closing the drawer

Accelerated video of closing the right cabinet door

For unfamiliar, toy-like cabinet doors, the closing becomes even harder. As pi0 is never trained on this articulation object, and it may tend to stop as it never learns how to close this type of object.

Quirk 2: Language Specificity Matters

Pi0's performance can be improved after tuning its instruction. Like this examples:

Instruction	Success Rate	Behavior
"Close the toilet"	0%	Wanders aimlessly, unable to localize the target.
"Close the white lid of the toilet"	100%	Precise alignment and execution.

Why This Happens:

- Vague instructions (e.g., *"toilet"*) force Pi0 to guess the target sub-component.
- Specificity (e.g., *"white lid"*) aligns with PaliGemma's object-part grounding capability.

Video Examples:

Close the white lid for the toilet (Success)

Close the toilet (Failure)

On the contrary, Pi0 freezes or fails when instructions contain **typos, grammatical errors, or ambiguous phrasing**.

Example:

- **Instruction**: *"Close the tiolet"* (misspelled).
- **Behavior**: Robot hesitates, then picks up the nearest familiar object (e.g., a marker pen).
- **Implication**: Pi0 lacks the linguistic robustness of LLMs like GPT-4, failing to infer intent from context.

Quirk 3: What will pi0 do if there is no specific language goal?

With no language goal, the robot will try to pick up the most familiar object within dataset based on RGB image.

Examples:

- Pick up the mark pen and move back and forth
- Reach the block back and forth

Here, mark pen takes 16.67% of the DROID dataset. Fine tuning on DROID may increase the probability of associating image with action to pick-and-place it.

Why This Matters

These quirks highlight critical limitations in Pi0's **spatial reasoning** and **language grounding**, emphasizing the need for:

1. **Expanded Articulation Datasets**: Covering closing actions and OOD objects.
2. **Language Model Upgrades**: Integrating LLMs for better instruction parsing.
3. **Force Feedback Integration**: Enabling tactile-based correction during manipulation.

3. Robot & Model Setup

The following are details of our experiment set up.

3.1 Hardware:

- **Franka Research 3 Arm**: 7-DOF force-sensitive robot with a 3 kg payload.

- **Robotiq 2F-85 gripper:** two-finger gripper with 5mm stroke and adjustable force control.
- **Cameras**:
  - **Side-view:** ZED 2 stereo camera for global scene understanding
  - **Wrist-mounted:** ZED Mini for close-range object manipulation
  - **Perception Mode: Pure RGB** (no depth calibration)

Robot setup

3.2 Computing

GPU Server:

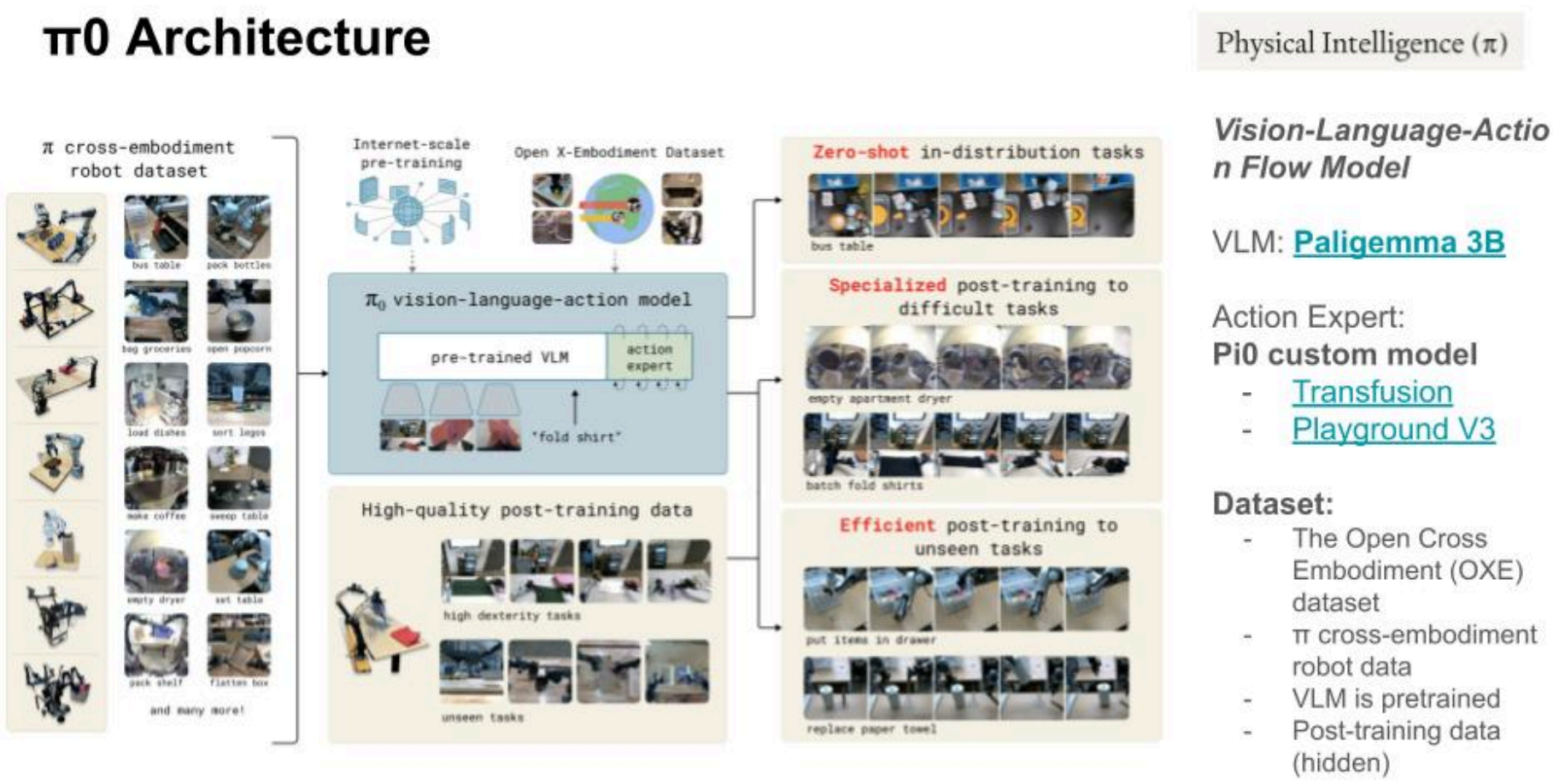
- **GPUs:** 1× NVIDIA RTX A6000 (48GB VRAM)
- **CUDA Version:** 12.3
- **Usage:** VLA model inference.

Workstation

- **GPU:** NVIDIA GeForce RTX 3080 (16GB VRAM)
- **CUDA Version:** 12.6
- **Usage:** DROID low level control.

3.3 Pi0-FAST-DROID:

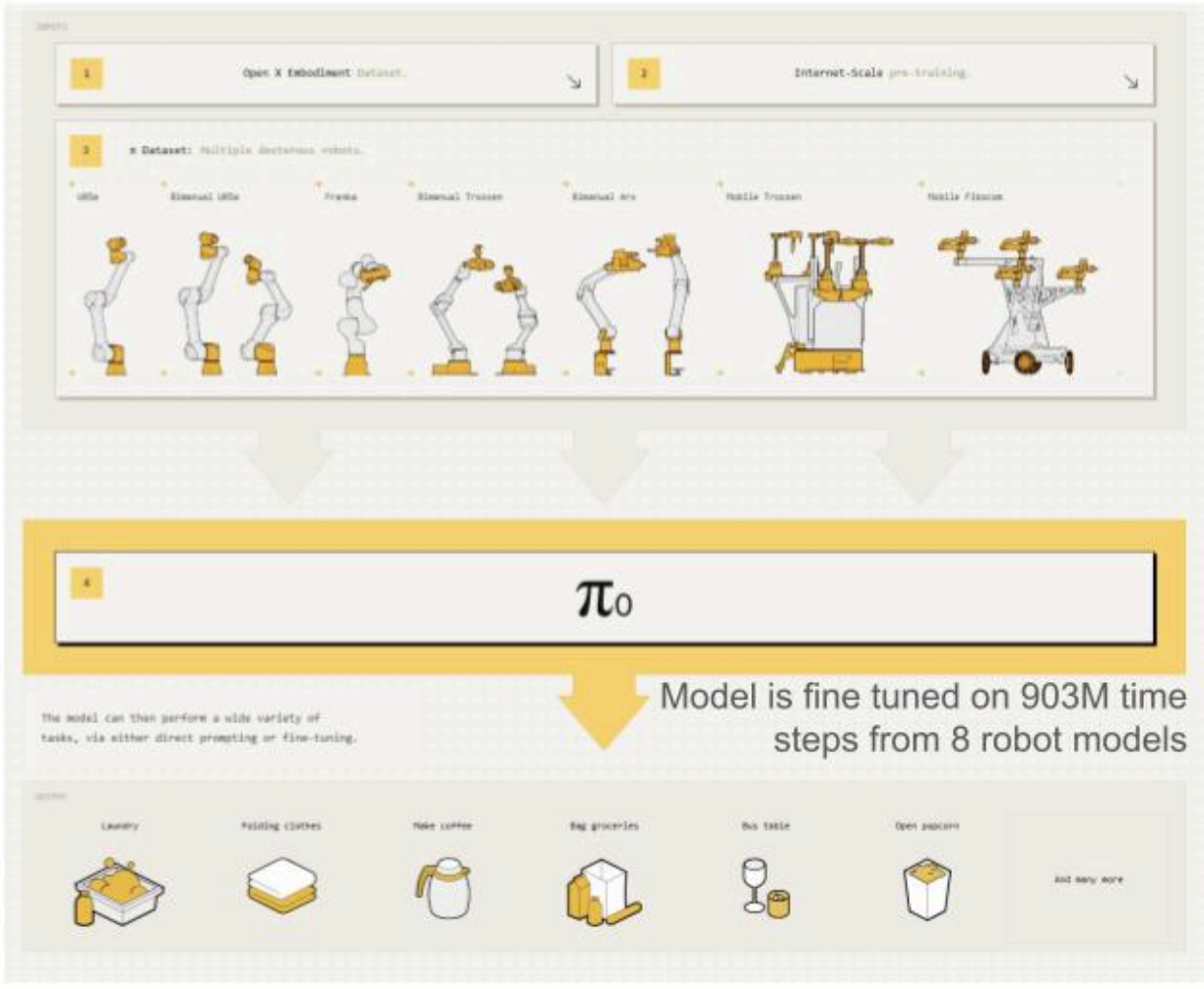
- **Vision-Language Model**: *Paligemma 3B* for spatial and semantic understanding.
- **Action Expert**: Build upon *Transfusion* + *Playground V3* for high-frequency control.
- **Training Data**: Pretrained on π cross-embodiment robot dataset & Open X-Embodiment, fine tuned on DROID dataset.



Dataset:

- The **Open Cross Embodiment** (OXE) dataset
- **π cross-embodiment** (10k hour+)
- 3B VLM is pretrained by Google
- Post-training data for finetune

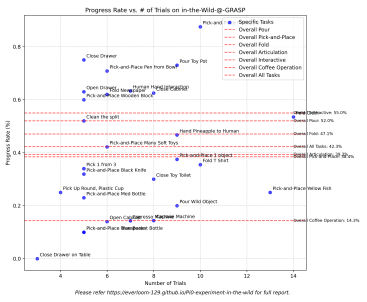
Most common tasks are **pick and place,** more complex kitchen tasks, stacking, opening and closing doors.

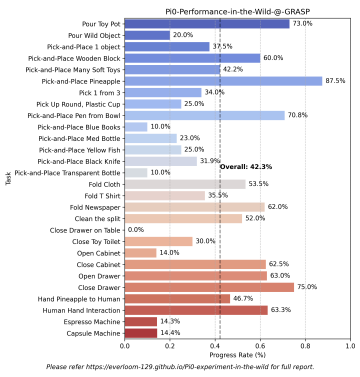


Dataset used for pretraining pi0

Results & Insights

Overall Performance





Across our 240+ test trials, Pi0 achieved varying degrees of success:

- **Complete Success:** 38%
- **Partial Success:** 28%
- **Complete Failure:** 34%

We observed that performance varied significantly based on task type, environmental conditions, and most importantly, the phrasing of instructions.

Performance across 240+ trials (52% average progress)

Task-Specific Performance

Pick-and-Place (44% Success)

Strengths:

- Familiar objects (e.g., pineapple toy, markers) - 90% success
- Clear spatial targets ("in the pink bowl") - 85% success

Weaknesses:

- Large objects (black cube) - 35% success
- Vague targets ("inside the bowl") - 40% success
- Multi-object tasks ("Put all objects into the basket") - 33% success

Chart showing pick and place performance

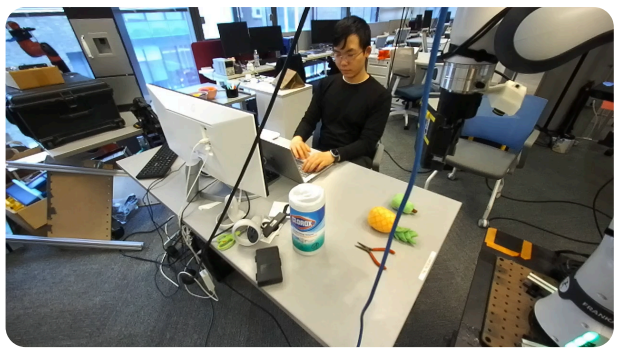
Human Interaction (53.5% Success)

Strengths:

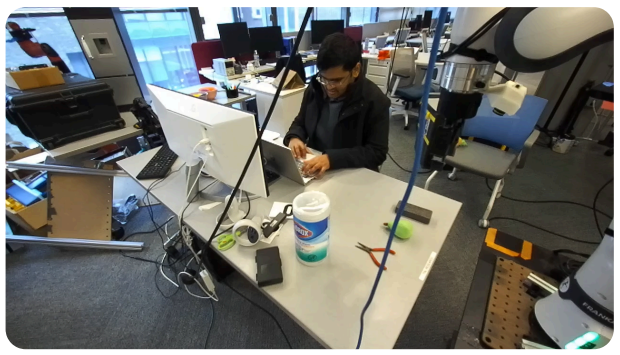
- Object handovers - 46.67% success
- Following a moving human - 65% success

Weaknesses:

- Precision interactions ("Shake hands") - 30% success
- Recovery after interruption - 40% success



"Give the pineapple to the programmer"



"Give the whiteboard eraser to the programmer"

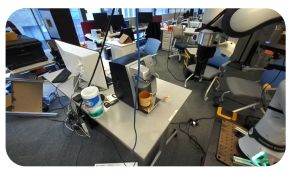
Coffee Machine (8.00% Success)

Capsule Coffee Machine:

- Close the capsule lid of coffee machine - 50% success
- Pick up the capsule from the coffee machine - 0% success
- Place the capsule into the coffee machine - 0% success

Espresso Coffee Machine:

- Pick up the coffee portafilter - 0% success
- Pour the coffee into the cup - 0% success
- Pick up the silver milk frothing pitcher- 33% success



"Place the capsule into the coffee machine"

"Close the capsule lid of the coffee machine"

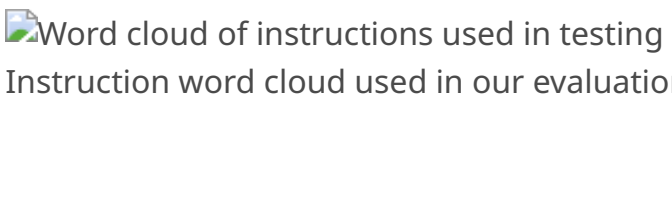
The Power of Prompts

Pi0's 3B language model is very sensitive to phrasing:

- **Good:** "Put A into B" → Clear spatial logic → 80% success
- **Bad:** "Fill B with A" → Ambiguous → 20% lower success

Example:

- "Close the toilet" → 0% success
- "Close the white lid of the toilet" → 100% success



Instruction word cloud used in our evaluations

The Bigger Picture

Why Pi0 Matters

Generalization

Scalability

Open Source Impact

Pi0 shows decent zero-shot performance across a wide range of tasks. For centuries, humans have had a strong desire to build a generalist agent system that could work in the wild, going out of the robotics lab and assisting our daily life. With the rapid development of VLAs model, the existence of Pi0 marks a milestone for robotics research.

FAST tokenizer enables 5x faster training, which is an impressive milestone and turning point for VLA research. The effective utilization of data implies that what matters to robots is not only the scaling size of dataset, but also the representation's expressiveness.

The open-sourcing of Pi0 is very intuitive and warm-hearted for the whole robotics industry. The impact not only lies in that every lab gets a very powerful manipulation baseline, but it will also push and extend the horizon of research in manipulation problems.

Future Directions

Better Language Grounding

As discussed in their new work "Hi Robot", VLA's language understanding ability could be enhanced via a two-tiered inference framework. However, as we saw in our examples, the 3B VLM backbone may still suffer from instruction following & generalization across language axes.

Force Feedback Integration

Current models rely primarily on visual input. Incorporating force feedback could help with tasks requiring delicate pressure control and improve interaction with varied surfaces.

Memory Modules

Currently, Pi0's autoregressive paradigm has achieved great success, but lacking history/memory capabilities is a significant limitation. Future models will likely incorporate explicit memory mechanisms to maintain task state across interruptions.

Conclusion

Pi0 represents a promising step toward generalist robots, but significant challenges remain. Instruction following and fine-grained tasks still pose considerable difficulties. As foundation models evolve and incorporate more modalities and memory capabilities, we're optimistic about the future—soon, every robotics lab might have a Pi0-like baseline!

Explore Full Results Watch Full Videos

Pi0 Evaluation by PAL Research Group, GRASP Lab, University of Pennsylvania.

The website template is adapted from Nerfies.