Contact-Anchored Policies: Contact Conditioning Creates Strong Robot Utility Models

Zichen Jeff Cui[1], Omar Rayyan[3], Haritheja Etukuru[2], Bowen Tan[1], Zavier Andrianarivo[1], Zicheng Teng[1], Yihang Zhou[1], Krish Mehta[6], Nicholas Wojno[1], Kevin Yuanbo Wu[1], Manan H Anjaria[1], Ziyuan Wu[1], Manrong Mao[1], Guangxun Zhang[1], Binit Shah[4], Yejin Kim[5], Soumith Chintala[1], Lerrel Pinto[1], Nur Muhammad Mahi Shafiullah[2]

[1]New York University, [2]University of California, Berkeley, [3]University of California, Los Angeles
[4]Hello Robot Inc., [5]Ai2, [6]University of Waterloo

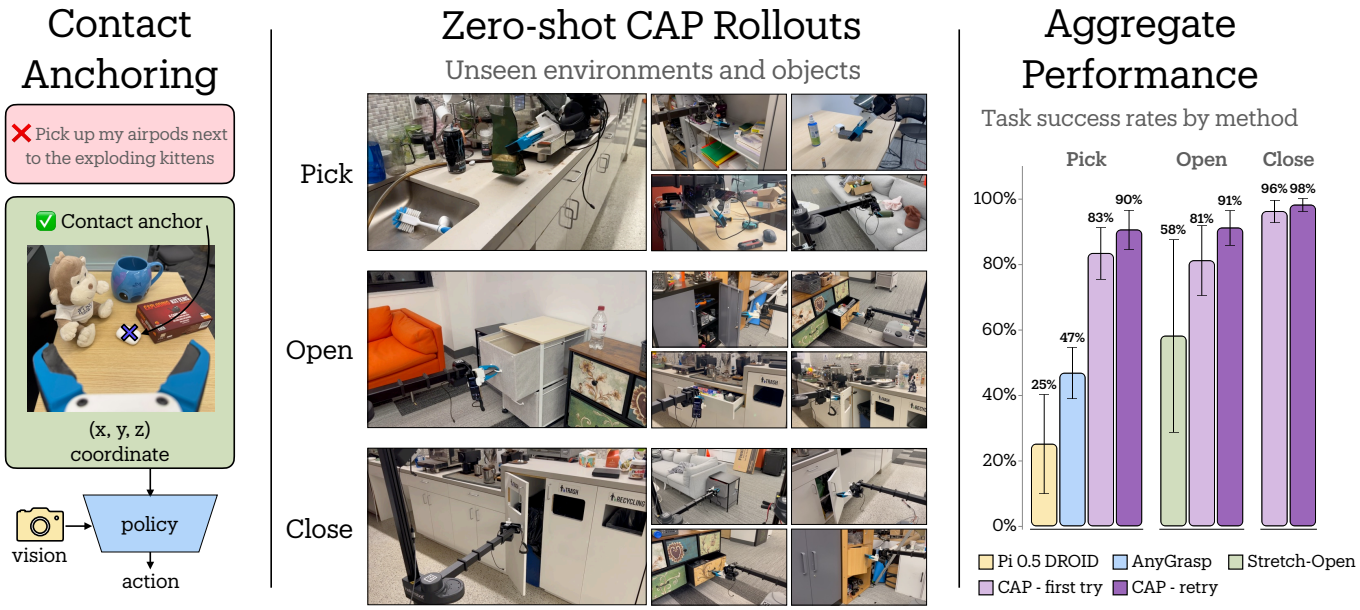


Fig. 1. We introduce Contact-Anchored Policies (CAP), a method to conditioning multimodal policies with physical contact information. Such policies are able to generalize zero-shot to novel objects and scenes with orders of magnitude less data, compute, and model parameters compared to frontier behavior model, while outperforming them on atomic skills trained with CAP.

***Abstract*—The prevalent paradigm in robot learning attempts to generalize across environments, embodiments, and tasks with language prompts at runtime. A fundamental tension limits this approach: language is often too abstract to guide the concrete physical understanding required for robust manipulation. In this work, we introduce *Contact-Anchored Policies* (CAP), which replace language conditioning with points of physical contact in space. Simultaneously, we structure CAP as a library of modular utility models rather than a monolithic generalist policy. This factorization allows us to implement a real-to-sim iteration cycle: we build EgoGym, a lightweight simulation benchmark, to rapidly identify failure modes and refine our models and datasets prior to real-world deployment. We show that by conditioning on contact and iterating via simulation, CAP generalizes to novel environments and embodiments out of the box on three fundamental manipulation skills while using only 23 hours of demonstration data, and outperforms large, state-of-the-art VLAs in zero-shot evaluations by 56%. All model checkpoints, codebase, hardware, simulation, and datasets will be open-sourced.**

I. Introduction

Now is the age for general robots. Yet, the resource requirements of training general policies is burgeoning constantly. Today it is measured in thousands: of human data collection hours, GPU cluster size, and numbers of real world evaluations. And even with all the resources, their generalization abilities remain more limited than a young child or household pet. What is the cause behind such a stark gap? One probable cause is that our current pipeline of learning general physical behavior may be backwards. Even though language is only a recent acquisition on our evolutionary ladder of skills, many of our current general robot policies are built on top of large language model bases. Folk wisdom holds that this internet-trained language backbone is necessary for generalization because language conditioning is how diverse behavior is elicited from the model. However, language as a medium for information for robot suffers from a few critical problems. First, language is imprecise: robotics needs precise spatial awareness which is not easy to convey in natural language abstractions. Second, language understanding comes at a cost of increasingly large model size leading to inefficient inference. These models are full of extraneous information, like distance between the earth and moon, that may be entirely unnecessary for a general robot.

In this work, we propose a simple fix: instead of natural

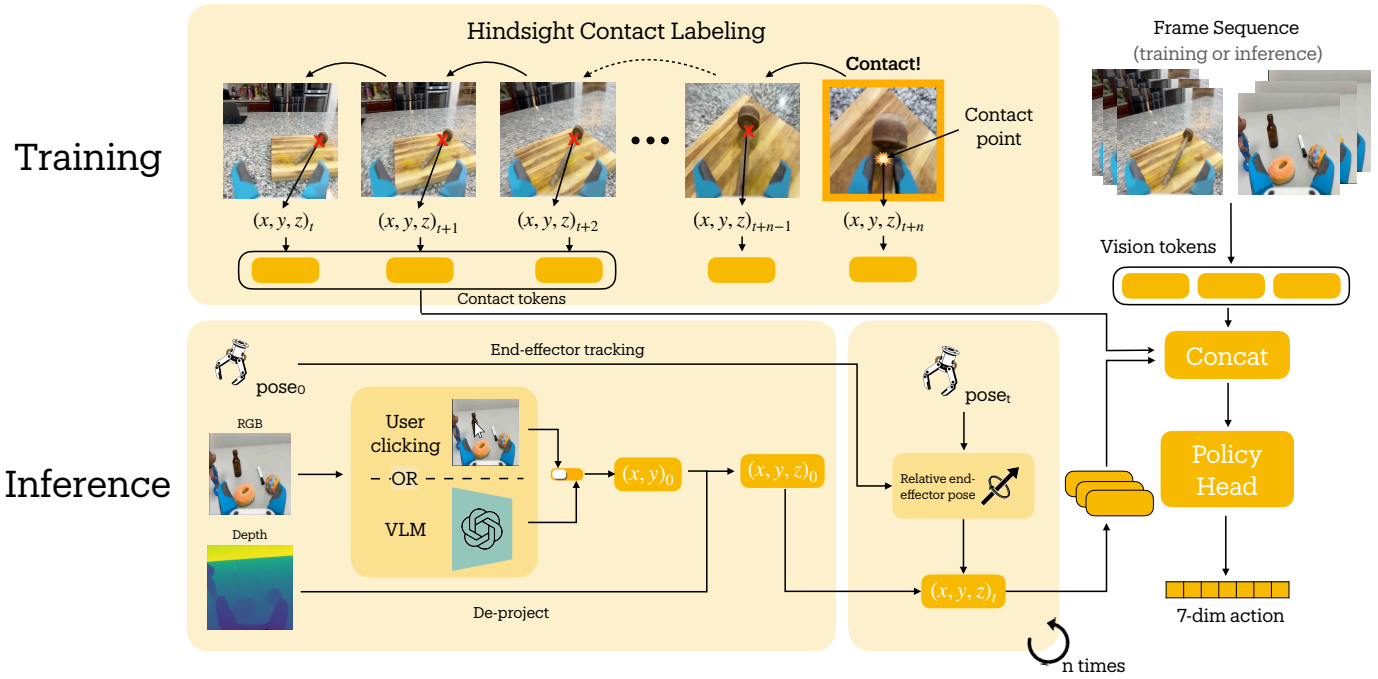


Fig. 2: The process of data labeling, training, and inference for Contact-Anchored Policies. (a) During training, we detect the contact point from the data and label the trajectory with hindsight relabeling. (b) During inference, we use a user click or VLM conditioned on user command to derive the contact condition. In both cases, the contact tokens and visual tokens get concatenated and passed to the model which uses them as input to predict the actions.

language, we propose *physical contacts* as the policy medium. Instead of modeling the robot observation and action together with an underspecified language description of the task, we model observation and action jointly with physical contact the robot makes with the environment. We call such policies Contact-Anchored Policies (CAP). With our simple change, we are able to train general policies for three set of common activities: picking up objects, and opening and closing doors and drawers; all from only 23 hours of human demonstrations. On zero-shot evaluations in fully novel scenes and objects, our models outperform state-of-the-art generalist vision-language-action models such as $\pi_{0.5}$ [43]. Moreover, since our policies are trained on handheld gripper data, we are able to deploy our policies on multiple robot embodiments out of the box.

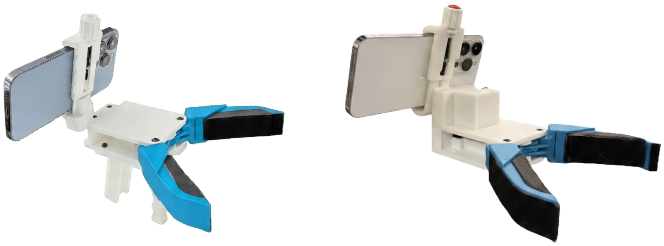
Developing such policies efficiently also requires multiple iterations on modeling and dataset curation. Typically, such iterations require training and evaluating models on the target set of tasks. To take advantage of the facts that (a) our target tasks are factored and (b) our primary goal is zero-shot environment and object generalization, we develop a lightweight simulation benchmark, EgoGym, as our key iteration metric. This benchmark focuses primarily on object and scene diversity and trades off photorealism for speed. As our primary goal is generalization, we find that success in these simulation environments under distribution shift is a great metric for capturing the emergence of general behavior.

II. Background

a) Behavior Cloning: Behavior cloning (BC) is one of the primary ways of teaching robots intelligent behavior from humans. BC casts the problem of learning a robotic behavior policy π that maps robotic observations $o \in \mathcal{O}$ to robotic actions $a \in \mathcal{A}$ as a supervised learning problem. Given a dataset $\mathcal{D} \subset \mathcal{O} \times \mathcal{A}$ of human demonstrations trajectories, BC defines a policy class Π and an loss function $\mathcal{L}$ and trains a policy $\pi \in \Pi$ that minimizes the loss function $\mathcal{L}(\mathcal{D})$. There has been significant research on finding the best learning objective [20, 48, 11, 34] as well as methods for collecting BC datasets, such as leader-follower teleoperation [61, 60], VR teleop [27, 10], and handheld tools [54, 50, 12].

b) Vector Quantized Behavior Transformer (VQ-BeT): VQ-BeT [34] is a behavior cloning algorithm designed to learn robotic behaviors from large, multi-modal behavior datasets. VQ-BeT is a two-stage algorithm. The first stage finds a self-supervised discrete action representation using the actions from the dataset by training a Residual Vector Quantized Variational Autoencoder (VQ-VAE). Then, second stage trains an autoregressive transformer to predict the tokenized actions given the observation sequence. In this work, we use VQ-BeT for our behavior modeling as autoregressive architectures are more straightfoward to condition compared to diffusion models [11, 1], and they lead to smaller and faster models.

c) Robot Utility Models: The advent of large robotic datasets [50, 42, 31] has training policies that can generalize to novel environments, objects, or tasks. Many large, proprietary models focus on task-level generalization [5, 43, 4] as the primary generalization axis. Our work is inspired by Etukuru et al. [18] which showed that performing a single task generally in diverse scenes and robots can be done efficiently–with a few hours of demonstrations–with the diverse, high quality



Handheld gripper for data collection

Robot gripper for policy deployment

Fig. 3: Our data collection tool and matching robot deployment gripper.

data and a multi-modal behavior cloning policy. This work also introduces verifier-guided retrying for robotics, where with guidance from an automated verified, a robot gets to retry a task until it is stuck or successful.

III. Contact-Anchored Policies

A. Data Collection and Contact Annotations

1) Gripper Hardware Design: To minimize the embodiment gap between data collection and robot inference, we design a low-cost, 3D-printable gripper compatible with both handheld operation and robot mounting. The handheld gripper is designed to be lightweight, ergonomic, convenient, and strong. Apart from the iPhone, the gripper itself consists almost entirely of 3D printed parts. We've designed the trigger handle such that the closing mechanism feels natural, allowing for prolonged use. Its small form factor makes it easy to throw into a backpack, and collect data in any location. The gripper design comprises an angular jaw 2-fingered mechanism, that allows for greater force and pinching of small objects. Similarly, the robot end-effector features compliant, back-drivable fingers with deformable foam padding for stable grasping across a diverse range of rigid and deformable objects. We use an iPhone 13 Pro as the primary sensor suite for both data collection and inference. The phone is rigidly mounted to the gripper chassis. For data collection, the gripper fingers are actuated by a handle. For inference, the same gripper modules are driven by a Dynamixel XL430 servo. This unified design ensures that the observation space remains consistent between the expert demonstrations and the robot's policy execution.

2) Data collection: We collect expert demonstrations for three primary tasks: Pick, Open, and Close. Data is collected with the *AnySense* iOS application [3]. The app records synchronized RGB-D streams and 6-DoF camera poses via ARKit visual-inertial odometry at 30Hz. Following Etukuru et al. [18], we prioritize collecting data in diverse environments with varying lighting conditions, background clutter, and task object form. Trajectories with tracking loss, significant jitter, or failed task completions are discarded. The final dataset contains 20,365 demonstrations (23.1 hours) across 424 environments, consisting of:

- **Pick:** general object pickup; 14,606 demonstrations (16.3 hours) across 289 environments.
- **Open:** opening drawers and cabinet doors; 3,690 demonstrations (4.7 hours) across 87 environments.
- **Close:** closing drawers and cabinet doors; 2,069 demonstrations (2.0 hours) across 48 environments.

3) Data preprocessing: **Observation**: we resize RGB and depth images to 224×224, and augment the data with horizontal flip on the RGB-D observations and the corresponding camera odometry. We find that this helps the policy generalize to left-right symmetries such as cabinet doors. **Action**: the action space consists of the delta end-effector (EE) pose and the gripper aperture. Since we mount the iPhone rigidly onto the gripper, we can extract the delta EE pose directly from the iPhone camera pose trajectory. **Visual gripper state estimation**: we extract gripper action labels directly from the visual observations using SAM2 [45]. For each frame, we segment the left and right fingers and compute the centroid of their respective masks. The distance between these centroids is linearly mapped to a scalar $\in [0, 1]$, representing fully closed to fully open. Further details are in App. Section A.

4) Hindsight contact labeling: We define the *Contact Anchor* as a 3D coordinate p where the policy is expected to interact with the object. To generate these labels for training, we do the following steps (illustrated in Fig. 2):

- **Contact Detection:** We first identify the timestep of contact, $t = c$. For Pick and Open tasks, this is naturally defined as the frame where the gripper aperture ceases to decrease, signaling that the fingers have made physical contact and halted against the object geometry. For the Close task, we label the contact frame during data collection by closing the grippers upon contact with the door.
- **Anchor Definition:** We define the contact frame as when the gripper has stopped closing further, which corresponds to a completed grasp. This works for Pick/Open since they require grasping; for Close, we label the trajectory during collection by manually closing the gripper upon contact. At $t = c$, we instantiate the contact anchor p_c as the 3D coordinate centered between the gripper fingers in the camera frame.
- **Anchor Propagation:** For all previous timesteps $t < t_c$, we generate contact anchors with hindsight relabeling by back-projecting p_c using the recorded camera odometry. Let $A_t \in \mathrm{SE}(3)$ denote the camera pose in the world frame at timestep t. The contact anchor in the camera frame for all previous timesteps $t < t_c$ is then simply given by $p_t = A_t^{-1} A_c p_c$ as the gripper approaches the contact. For the remainder of the episode $t > t_c$, in tasks such as Pick or Open, the object moves rigidly with the gripper as the gripper establishes contact. Post-grasp, the contact anchor is frozen: we simply repeat p_c until the end of the episode.

B. Policy Learning

We formulate the policy learning as a conditional imitation learning problem, where the policy $\pi(a_{t:t+h} | o_{t-k:t}, p_{t-k:t})$ predicts actions given a history of visual inputs and contact anchors. We implement this using a Vector-Quantized Behavior Transformer (VQ-BeT) [34] with an observation context length of $k = 3$. For the visual (RGB) observation,

we pretrain a ResNet-50 backbone with MoCo [9] on our dataset. For each timestep, we embed the 224×224 RGB input into a feature vector $z_v \in \mathbb{R}^{256}$. Separately, the contact anchor $p_t \in \mathbb{R}^3$, expressed in the current camera frame, is linearly projected to a contact embedding $z_c \in \mathbb{R}^{256}$. We concatenate these embeddings to form the observation token $s_t = [z_v, z_c]$. We feed a context window of observation tokens $s_{t-k:t}$ into VQ-BeT to predict the demonstrated actions. Each action a_t consists of the delta end-effector pose and the continuous gripper position command. By conditioning the action distribution jointly on the RGB observation and the contact anchor, the policy adapts to diverse object geometry while anchoring its manipulation trajectory to the intended interaction point.

C. Contact Prompting during Inference

Unlike training, where contact anchors are derived from hindsight, inference requires an initial anchor p_0 before execution. Given the initial RGB-D observation, a pixel coordinate (u, v) is selected on the object of interest. This selection can be performed manually, or by querying an off-the-shelf VLM (e.g. Gemini Robotics-ER 1.5 [56]) with a text prompt ("point to the red mug"). Then, we deproject the 2D pixel (u, v) using the depth map value $d_{u,v}$ and camera intrinsics K to obtain the initial contact anchor in the camera frame, $p_0 = d_{u,v} K^{-1}[u, v, 1]^T$. As the robot executes the policy, the camera frame moves with the gripper. We track the anchor in the camera frame using the robot's forward kinematics, which provides higher accuracy than visual-inertial odometry. Let $A_t \in \mathrm{SE}(3)$ be the camera pose in the world frame at time t, derived from the robot's kinematic chain; the anchor p_t is simply updated via $p_t = A_t^{-1} A_0 p_0$, so that the policy sees a consistent contact anchor in the world frame (see Fig. 2). After the gripper closes, we freeze the contact anchor to match the training data distribution.

D. Simulation-in-the-loop Development

To support rapid iteration and evaluation of CAPs, we develop EgoGym, a lightweight simulation suite used during policy training and development. It is designed with the motivation of (i) providing training signals beyond validation loss, which poorly correlates with real world performance, and (ii) accelerating CAP refinement by detecting failure modes prior to real-world deployment, and (iii) enabling the quick calibration of grasping thresholds we set for our policies.

EgoGym is implemented in MuJoCo [58] and trades off visual realism in favor of scene diversity and execution speed. This allows it to run sufficiently fast to be included directly in the training loop of all 3 of our policies, enabling frequent checkpoint evaluation to detect overfitting. We induce diversity through task-specific procedural scene generation. For our pick task, objects are sampled from a pool of 915 Objaverse [15] assets and spawned with varying poses and arrangements. For our opening and closing tasks, we procedurally generate articulated doors and drawers with randomized geometrical parameters at run-time. Across all three tasks, additional diversity is introduced by randomizing surface textures and adding distractor objects. Additional details on the assets and the randomization employed are provided in App. Section D.

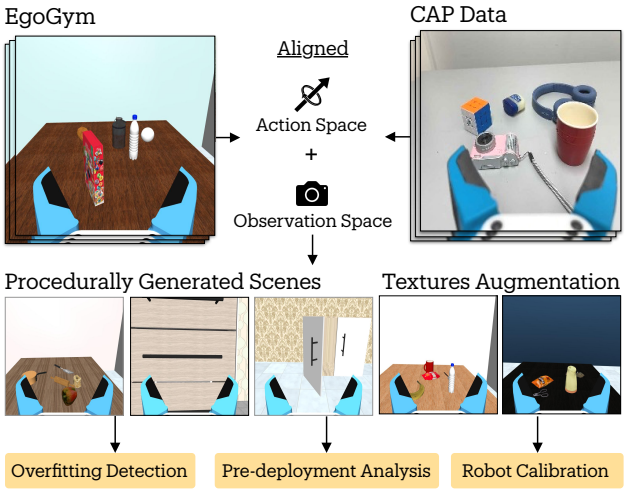


Fig. 4: EgoGym: a lightweight simulation-in-the-loop environment used for quick development and evaluation of Contact-Anchored Policies (CAPs). EgoGym enables fast checkpoint evaluation and failure mode discovery across Pick, Open, and Close tasks using procedurally generated scenes.

IV. Evaluating CAP

We evaluate the zero-shot generalization performance of Contact-Anchored Policies in diverse real and simulated environments, across multiple robot embodiments, including independent third-party evaluations at three external institutions. Our experiments are designed to answer the following questions:

1) How well does CAP generalize zero-shot to unseen environments and objects?
2) Can CAP work across robot embodiments out of the box?
3) Can CAP be used to compose long-horizon manipulation behavior with tool calling?
4) Do simulated evaluations of CAP reflect its real-world performance, and if so, can we use it to get detailed understanding of our policy performance?

A. Zero-shot Environment Generalization

We evaluate CAP on the three core manipulation tasks represented in our dataset: Pick, Open, and Close. All evaluations are zero-shot on unseen environments with no additional fine-tuning. We manually provide the policy with oracle contact prompts to isolate the generalization performance of CAP.

1) Pick: We evaluate CAP in five unseen scenes (kitchen, couch, meeting room, storage cabinet, work area) on the Stretch 3 platform. For each scene, we present the policy with a set of five objects not seen during training, for total 25 objects (Fig. 5). The policy attempts to pick up each of the objects for 10 trials, randomizing the robot's initial position by 16×11cm (horizontal $\times$ vertical), for 250 total trials.

Fig. 5: Evaluation environments for CAP. Each scene and object combination has 10 trials, so Pick checkpoints are evaluated for 250 episodes and Open or Close checkpoints are evaluated for 100 episodes in total.

2) Open, Close: We evaluate CAP on five cabinet doors and five drawers unseen during training (Fig. 5), on Stretch 3. The policy attempts to open and close each door and drawer for 10 trials, randomizing the robot's initial position by 16×11cm (horizontal $\times$ vertical), for 100 total trials. With oracle contact prompts, CAP generalizes zero-shot to unseen environments, achieving a single-try performance of 83% in Pick, 81% in Open, and 96% in Close (Table I).

TABLE I: Evaluation results of CAP and baselines on our three different tasks and four different robot embodiments.

Task	Robot	Model	Success rate
Pick	Stretch	CAP + Retry	$90.4\% \pm 6.0\%$
Pick	Stretch	CAP	$83.2\% \pm 7.9\%$
Pick	Stretch	AnyGrasp	$46.7\% \pm 7.9\%$
Pick	Franka	CAP	$79.0\% \pm 10.9\%$
Pick	Franka	CAP-VLM	$81.0\% \pm 9.2\%$
Pick	Franka	$\pi_{0.5\text{-DROID}}$	$25.0\% \pm 15.2\%$
Pick	XArm	CAP	$83.0\% \pm 17.9\%$
Pick	UR3e	CAP	$70.0\% \pm 15.2\%$
Open	Stretch	CAP + Retry	$91.0\% \pm 5.3\%$
Open	Stretch	CAP	$81.0\% \pm 10.7\%$
Open	Stretch	Stretch-Open	$58.0\% \pm 29.3\%$
Close	Stretch	CAP + Retry	$98.0\% \pm 3.0\%$
Close	Stretch	CAP	$96.0\% \pm 3.5\%$
EgoGym-Pick	Sim Gripper	CAP	$79.88\% \pm 1.1\%$
EgoGym-Pick	Sim Franka	$\pi_{0.5\text{-DROID}}$	$20.9\% \pm 1.1\%$

Next, to set up fully autonomous rollouts, we evaluate whether we can autonomously generate contact prompts comparably to a human oracle. To propose contact anchors autonomously, we use Gemini Robotics-ER 1.5 [56]. We provide the VLM with the initial observation, and prompt it to point to the task object for the contact anchor (see App. Section C for details). We evaluate CAP with the same procedure described above. With VLM-generated contact anchors, CAP achieves a comparable single-try performance of 81% in Pick, 80% in Open, and 97% in Close, validating that automatic, VLM-generated contact prompts work as well as the human oracle.

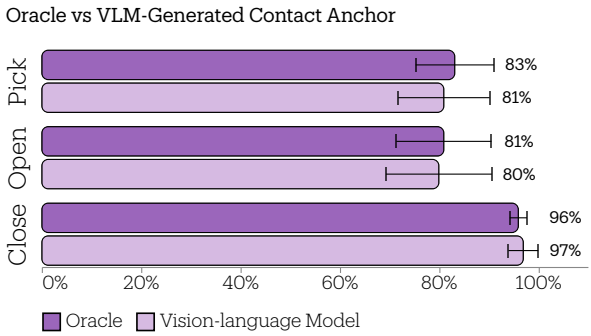


Fig. 6: Comparison of contact instructions generated by human oracles and vision-language models for the CAPs. Downstream CAP performance is comparable on all tasks.

As we establish in Fig. 6 that CAP does not require a human in the loop, we add a VLM verifier for automatic retry upon failure. Following Etukuru et al. [18], we use GPT-4o to verify whether the policy has successfully completed the task, using VLM-generated contact anchors for retries. We evaluate CAP with the same criteria above, allowing up to 10 retries per trial. With automatic retries, CAP achieves 90% in Pick, 91% in Open, and 98% in Close (Table I, Fig. 1). The vast majority of failures are from verifier false positives, i.e. classifying a failed task as successfully completed.

B. Zero-shot Embodiment Generalization

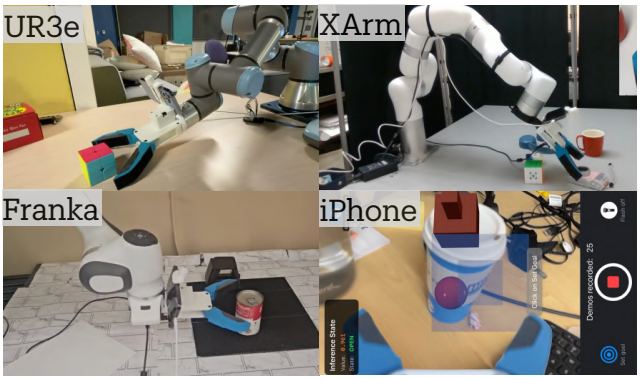


Fig. 7: Cross-embodiment deployment of CAP on a Franka FR3, XArm 6, Universal Robotics UR3e, and an iPhone app. For the robots, the same CAP checkpoint generates EE-space motion that we translate to joint position control with IK. For the iPhone, the user is prompted to move the iPhone to where the robot should go next.

Since Contact-Anchored Policies are trained on handheld gripper data, they are theoretically compatible with any robot arm with six or more degrees of freedom. However, often because of idiosyncrasies in human behavior, policies trained with handheld tools have difficulty transferring to real robots by taking actions that violate robot kinematics. To validate

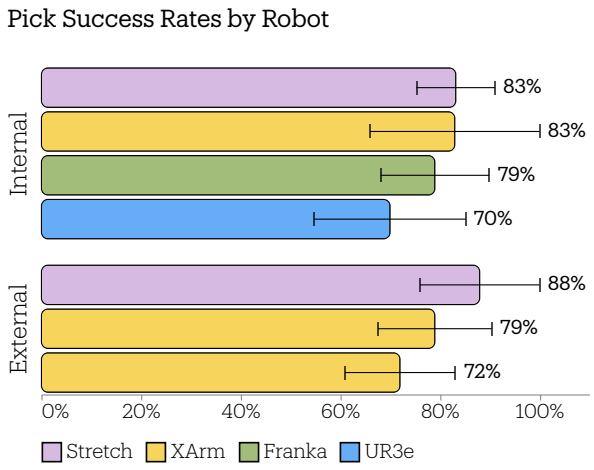


Fig. 8: Evaluation of CAP zero-shot on diverse embodiments: each bar is a different set of evaluations in a unique site. To evaluate system robustness, we share checkpoints, setup instructions, and evaluation methodology to external collaborators and get performance numbers from them.

cross-embodiment performance, beyond our primary embodiment in Stretch, we also evaluate our Pick policy on Franka FR3, XArm 6, and Universal Robotics UR3e (Fig. 7). For these evaluations, the same policy checkpoint is evaluated everywhere, and we only adapt our robot gripper mount and the inverse kinematic controller to the specific embodiments. Since these robots are mounted, we evaluate in the fixed environment with two sets of five unseen objects for a total of 10 objects, randomizing the object positions. The policy attempts to pick up each object for 10 trials, for a total of 100 trials. We see from Fig. 8 that their success rates are comparable, showing the versatility of our policy. Out of the arms, UR3e achieves the lowest success because of its particularly short reach forward.

1) External evaluations: Cross-embodiment evaluations of this kind are a test of system integration as much as they are of policy capabilities. We send our checkpoints and evaluation process to external collaborators at three institutions for independent evaluations. As we see in Fig. 8, result of such evaluations also broadly line up with internal evaluations.

2) iPhone evaluations: Even before running a policy on the robot, it can be useful to understand if a policy will behave sensibly in a scene. Since our model has only 52 million parameters, we can deploy and infer from it in real time on modern iPhones. We develop an iPhone app that uses the camera as input stream, the ARKit API for pose tracking, and neural engine chips for inference. We rely on user taps for contact conditioning, and emulate a dummy robot gripper on screen to match observations. When a user taps on a target object, the app shows the next target position of the phone, as well as the predicted gripper motions. The app is able to show the user how CAP would navigate an in-the-wild scene and execute different grasps matching the target object affordance.

C. Baseline Comparisons

We compare CAP with state-of-the-art task-specific and generalist baselines:

- $\pi_{0.5\text{-DROID}}$ [43] (Pick): a generalist VLA model trained on a large proprietary dataset and fine-tuned on DROID [31]. We evaluate it on the DROID embodiment with a Robotiq 2F85 gripper mounted on a Franka FR3, a ZED Mini wrist camera, and a ZED 2i external camera. We use the prompt "pick up the {object}" and run the policy for 400 steps.
- AnyGrasp [19] (Pick): an RGB-D grasp pose prediction model for object pickup that uses an external depth camera to generate grasps and then uses a planner to execute them. We evaluate it on the Hello Robot Stretch 3 robot platform.
- `stretch-open` [22] (Open): a modular pipeline for opening doors and drawers. The pipeline uses an RGB-D head camera to locate the handle, and generates and executes a motion plan to open it. We evaluate it on the Hello Robot Stretch 3 robot platform with the stock gripper to exactly match the embodiment in [22].

We evaluate AnyGrasp with the same Stretch 3 evaluation procedure as described in Section IV-A1 for three trials per object, for a total of 75 trials. We evaluate $\pi_{0.5\text{-DROID}}$ with the Franka FR3 evaluation procedure as described in Section IV-B. For a fair comparison, we also evaluate CAP with VLM-generated contact anchors with the same object description given to $\pi_{0.5\text{-DROID}}$, denoted as CAP-VLM in Table I. AnyGrasp achieves a 47% success rate, and $\pi_{0.5\text{-DROID}}$ achieves a 25% success rate in our evaluations. `stretch-open` achieves a 58% success rate. As shown in Table I, CAP outperforms comparable baselines by 23% to 56%.

D. Eliciting Complex Behaviors with Tool Calling

A standard argument for using large, end-to-end behavior policies is that theoretically they can exhibit long-horizon behavior that is unavailable to atomic utility models. However, because they are atomic, we posit that utility models can be chained together as a sequence of *tool calls* [47] by a larger model specializing in System 2 intelligence.

In Fig. 6, we verify that such models are able to generate contact conditions almost as well as a human for all three of our tasks. As a proof of concept, we take CAPs with verifier guided retrying and add a supervising high-level controller to get our robots to complete complex, long-horizon tasks as seen in Fig. 9. We perform two long horizon tasks: fetching some coffee beans from a kitchen cabinet, and cleaning up a table. In the first task, the robot's goal is to retrieve a yellow bag of coffee beans from within the white kitchen cabinets. The robot needs to open the cabinet door, pick up the bag, drop it on the table, and close the door. In the second task, there are multiple objects on a table in front of the robot, and it has to move all of them into a bin next to it. The robot needs to perform a sequence of picks and drops until the table is clear. We use our trained Pick, Open, and Close policies, as well as scripts for executing "Drop" and "Move the mobile base". We run 10 experiments for each task.

Our compositional policy succeeds 6/10 trials on the coffee beans task, and 10/10 trials on the table cleaning task. Table II

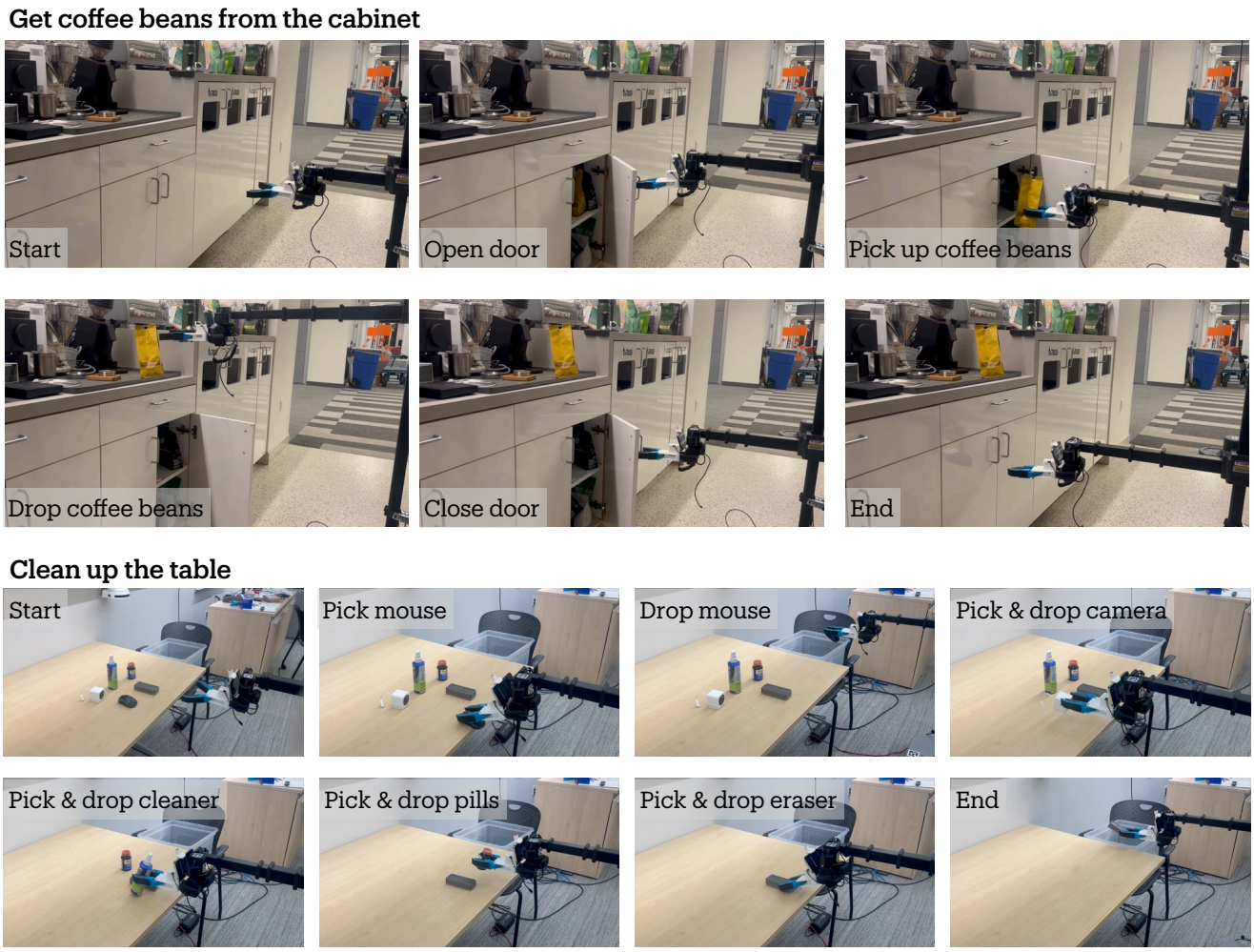


Fig. 9: Performing long-horizon manipulations with Contact-Anchored Policies controlled by a high-level VLM controller via tool-calling. On the top, to retrieve coffee beans from cabinet, a controller combines Pick, Open, and Close CAPs, while on the bottom, a table cleanup is performed with Pick CAP.

TABLE II: Success rate by stages for long-horizon tasks

Get coffee beans	
Stage	Success
Open cabinet	10/10
Pick bag	7/10
Drop bag	7/10
Close cabinet	6/10

Clear table	
Stage	Success
1st Object	10/10
2nd Object	10/10
3rd Object	10/10
4th Object	10/10
5th Object	10/10

describes the success rate by stages. For the coffee fetching task, most failures are from the policy opening the door partially, which the VLM verifier counts as a success, leading to hardware collisions with the door during the Pick stage. For the table cleaning task, the verifier can classify a successful grasp as a failure, leading to unnecessary retries, but the policy was able to move all objects into the bin for all 10 runs.

E. Understanding Real Performance via Simulation

a) Establishing Sim-and-real Correlation: To confirm whether CAPs behavior in EgoGym is indicative of real-world performance, we carry out a single-blind correlation study in which an evaluator, unaware of EgoGym results, is sent four Pick CAP checkpoints to evaluate in the real world. The evaluations includes 250 real-world runs and 5 000 EgoGym episodes, which include texture and objects augmentation as well as four random distractor objects per episode. While the real evaluation numbers are limited, the results in Fig. 10 (left) suggest strong alignment between EgoGym and real-world performance across the checkpoints. For the rest of this section, we describe these four checkpoints with letters A through D in order of success rates.

b) Iterating CAP via failure analysis: To better understand how policy behavior evolves across checkpoints, we analyze failure modes using EgoGym rollouts. Details of the failure categorization are provided in App. Section D. Figure 10 (left) shows the distribution of outcomes over 5 000 simulated episodes for each analyzed checkpoint.

These observed failure modes motivated refinements to the CAP data processing pipeline. For example, the low-lift failures at checkpoint B revealed the existence of many post-grasp transitions with little or no end-effector motion, motivating us to introduce static-frame filtering which resulted

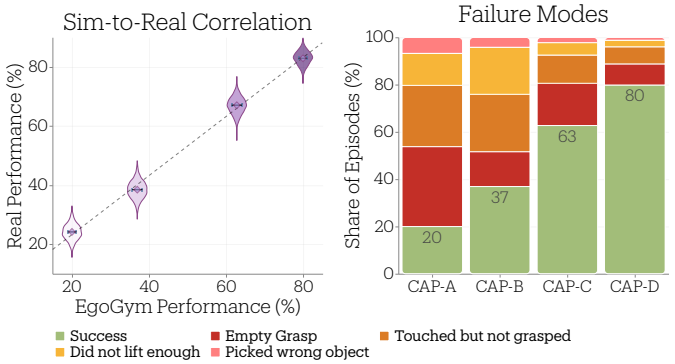


Fig. 10: **Left:** Sim-to-real correlation for single-blind EgoGym-Pick evaluations. **Right:** Analysis of failure modes of four iterations of CAP Pick checkpoints in EgoGym. With feedback from each iteration, our pipeline changes allowed better policy quality in real and sim.

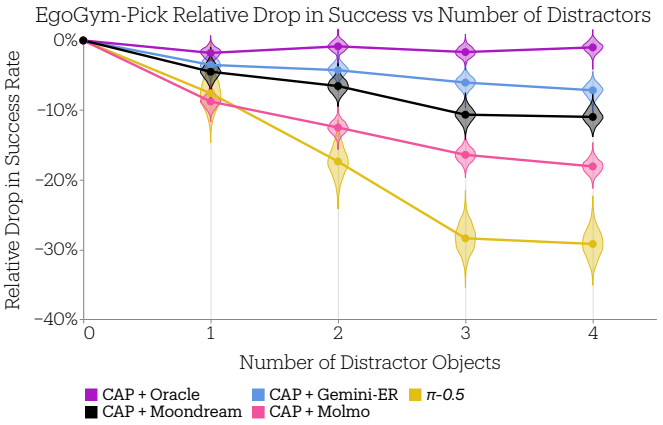


Fig. 11: Relative success rate as a function of visual distractors for CAP and $\pi_{0.5}$ models on EgoGym-Pick. Success rates are normalized to each model's performance with zero distractors.

in the following C checkpoint.

F. Ablations

We run ablation experiments to answer three questions related to Contact-Anchored Policies' performance:

- How important are the different modalities (RGB, anchor point) in CAP?
- How robust is our policy performance to distractor objects or noise in the contact prompt?
- Does CAP learn to use the visual input meaningfully?

TABLE III: Ablation results of CAP on three different task.

Model	Pick	Open	Close
CAP	83%	81%	96%
CAP – RGB Only	14%	8%	58%
CAP – Anchor Only	63%	6%	28%

1) Modality ablation: To understand the role of different modalities, such as visual input and contact anchor, we run some baselines without including each modality on the same data and architecture. We train an RGB-only ablation of CAP, an anchor-only version without the visual input, and run the same evaluation procedure as described in Section IV-A2. While the vanilla CAP achieved $> 80\%$ success on every task, the anchor-only ablation performed much worse at 32% average, while the RGB only is worst, at 27% average (Table III), validating our hypothesis that having a contact anchor alongside RGB input improves manipulation performance.

2) Distractor objects and policy performance: We evaluate CAP with various VLMs for contact prompt generation, as well as $\pi_{0.5\text{-DROID}}$ in the EgoGym-Pick environment. The baseline CAP with privileged oracle contact anchor information remains stable in performance regardless of distractor objects. We see a decrease in performance for CAP + VLMs and $\pi_{0.5}$, picking up more wrong objects as we increase the number of distractors in the scene.

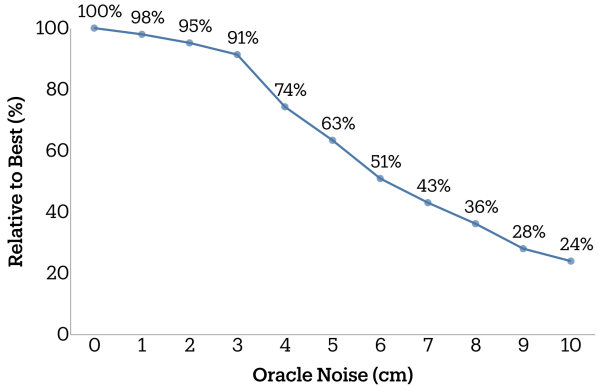


Fig. 12: Relative success rate as a function of noise added to the ground truth anchor for CAP on EgoGym-Pick. Success rates are normalized to the model's performance with zero noise.

3) Robustness to anchor noise: To understand how any potential noise may impact CAP, we evaluate the Pick model in EgoGym with extra noise introduced in the contact-point prompt. In Fig. 12, we can see that as we add higher noise to the ground truth object position, the performance of CAP declines cleanly, with 91% of performance retained even at 4 cm of added noise level.

4) Ablating visual input: A natural question is whether CAP's visual stream contributes anything beyond localizing the anchor, or whether a scripted approach to the deprojected contact point would suffice. We test this from two angles.

Point-based visual servoing. We compare against PBVS, which deprojects the contact anchor to 3D and executes a scripted approach-and-grasp at that point. On the Stretch 3 Pick evaluation (25 objects, 250 trials), PBVS achieves 62.4%, comparable to the anchor-only ablation (63%) and well below CAP (83%), showing that the RGB stream contributes substantially beyond anchor localization.

Rotation sensitivity. If CAP were simply goal-seeking toward the anchor, grasp orientation would be independent of object pose. To check this, we evaluate Pick on five unseen

elongated objects laid on the table, each run with five trials rotated $45°$ clockwise from a vertical reference and five rotated $45°$ counterclockwise (50 trials total). CAP achieves 88% success, aligning the gripper to the principal axis of each object rather than defaulting to a fixed approach. The Open task tests the same property implicitly, requiring the policy to distinguish horizontal versus vertical handles and rotate the wrist accordingly.

V. Related Works

A. Generalist Behavior Models

Current progress in learning-based robotic systems and advent of large-scale data in robotics has enabled development of general robot manipulation policies: models that can be applied to novel scenes, objects, tasks, or robots without having trained on the same [18, 5, 43, 25]. In most cases, the standard recipe requires creating and training on a large training set which contains diversity in the intended axes of generalization. Note that this paradigm is different than many large robotic models in literature [57, 1, 4, 33, 53] which do not claim zero-shot performance and requires some post-training in the particular evaluation setup (task, scene, or robot) to show reasonable behavior in that setup.

Current understanding on the necessary and sufficient amount of scale in data, model size, or compute for such generalization is still in its infancy. Existing multi-task generalist models, mostly proprietary, uses at least 1,000 to 10,000 hours of data, while single-task general policies such as [18, 25, 12] has been trained on as little as 1,000 demonstrations per task.

B. Conditioning Multi-modal Behavior Models

A general policy capable of multiple possible behaviors in a particular environment needs to be conditioned by user intent or goal to elicit useful behavior. Earliest forms of conditioning behavior policies relied on communicating a future state or image [40, 13, 6] to the robot. With the rise of capable language models, language became a popular mode for conditioning language starting with [39, 29, 7, 8]. In these works, language is used directly as an input modality to the model. However, advent of multimodal grounding models such as [44] allowed [51, 49, 52] to ground language in some spatial conditioning as an input to the robot.

Other low-dimensional policy conditioning include [21, 55] where robot trajectory and a sketch of the goal are used to condition the model, and more recent work conditions on image-space waypoint traces as an intermediate reasoning step [33]. The most related to our work is RT-Trajectory [21], where gripper motion and hindsight relabeling are used to condition a model. These approaches assume that the image-space conditioning is drawn on a calibrated, exocentric camera view from which the trajectory remains well-defined throughout the rollout. Since our egocentric wrist camera moves during the rollout, our approach uses a single 3D contact anchor that is camera-frame-equivariant: forward kinematics propagate it through camera motion, leaving the conditioning signal valid throughout execution. Our method also simplifies the premise to focus only on the contacts between the robot and the environment as the minimal interface, and shows that it already leads to strong general behavior models.

Another line of work uses keypoints extracted by pretrained models to give robots generalization abilities [23, 35, 26, 2]. However, they forgo the pixel-to-action paradigm by using only keypoints as observation or using a planner to generate robot actions.

C. Evaluating Real Manipulation Policies in Simulation

One of the biggest bottlenecks in training policies for real robot is evaluation [62]: training or test losses are unindicative of policy success, and getting statistically significant effect sizes require onerous evaluation schedules [1]. Therefore, there has been significant interest in using simulated setups to evalute real robot performance [1, 28, 37, 30]. A shared focus of these works is to start from some ground-truth real world environments and then model them as closely as possible in simulations. This focus on simulation fidelity purportedly serves to ensure transfer of simulated evaluation results to real world. But even minor differences between sim and real can make it difficult to hill-climb a simulation metric that will lead to real world improvement [1]. Note that, these simulation benchmarks have a different goal than [38, 36, 41] where the goal is to compare different learning algorithms trained on a fixed set of data and not general task competence.

In this work, we instead look at factored, single task simulation with many procedurally generated scenes, which makes overfitting to this metric difficult. Indoor navigation has successfully used this approach [17] through simulations providing a large number of household environments [32, 46, 14].

VI. Conclusion

In this work, we introduce Contact-Anchored Policies (CAP), a principled way of conditioning general policies that achieves superior zero-shot performance on single tasks with a modestly sized dataset, model size, and compute budget. We additionally demonstrate how we can develop such CAPs with simulation in the loop, and also chain together CAPs via tool calling to perform long-horizon manipulations. With CAP, we hope to provide the right framework for researchers with limited resources, such as those in academia, to study the emergence of general behavior in robotics. While CAP makes advances in the problem of learning general manipulation policies, we are only able to scratch the surface and there are many aspects of CAP that we think should be studied in future work. Firstly, extending CAP to tasks with multiple contact anchors or for bimanual tasks would be useful, and will require extending the system to predict and ingest multiple contacts or even a distribution of them. Secondly, CAPs rely on two input modalities, and studying the process through which they decide on their relative weights in the decision making may elucidate fundamental factors about the dynamics of supervised policy learning itself. The contact point modality depends on a human or VLM during the test time, which may also make them less portable and autonomous. The 3D contact anchor also relies on having reliable depth maps as opposed to pure RGB input. Finally, as a practical matter, exploring

how we could roll the verifier guided retrying process into the end-to-end part of the policy via either real world or simulated reinforcement learning would go a long way into making CAPs practical for high-stakes applications.

Acknowledgments

We thank Shenglong Wang and the NYU HPC team for helping us with compute, Kanad Patel and Dhawal Kabra for helping with the robot hardware, Gaoyue Zhou, Raunaq Bhirangi, Irmak Güzey, and Nikhil Bhattasali for insightful discussions, and Andrew Liao, Lin Yang, Peng Jiang, and Santosh Srinivas for contributions to the dataset. NYU authors are supported by grants from Honda, Hyundai, NSF award 2339096 and ONR awards N00014-21-1-2758 and N00014-22-1-2773. LP is supported by the Packard Fellowship and Sloan Fellowship. Hello Robot authors are supported by NIH NIA R43AG072982.

References

[1] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025. URL https://doi.org/10.48550/arXiv.2507.05331.

[2] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pages 306–324. Springer, 2024. URL https://doi.org/10.1007/978-3-031-73116-7_18.

[3] Raunaq Bhirangi, Zeyu Bian, Venkatesh Pattabiraman, Haritheja Etukuru, Mehmet Enes Erciyes, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Anysense: An iphone app for multi-sensory data collection and learning, 2024. URL https://github.com/NYU-robot-learning/AnySense.

[4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

[6] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. RoboCat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023. URL https://arxiv.org/abs/2306.11706.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://doi.org/10.48550/arXiv.2212.06817.

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://doi.org/10.48550/arXiv.2307.15818.

[9] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

[10] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback, 2024. URL https://arxiv.org/abs/2407.01512.

[11] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. URL https://doi.org/10.1177/02783649241273668.

[12] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024. URL https://doi.org/10.48550/arXiv.2402.10329.

[13] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022. URL https://doi.org/10.48550/arXiv.2210.10047.

[14] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, Ali Farhadi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://doi.org/10.48550/arXiv.2206.06994.

[15] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. URL https://doi.org/10.1109/CVPR52729.2023.01263.

[16] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini,

et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv e-prints*, pages arXiv–2409, 2024.

[17] Ainaz Eftekhar, Rose Hendrix, Luca Weihs, Jiafei Duan, Ege Caglar, Jordi Salvador, Alvaro Herrasti, Winson Han, Eli VanderBil, Aniruddha Kembhavi, et al. The one ring: a robotic indoor navigation generalist. *arXiv preprint arXiv:2412.14401*, 2024. URL https://doi.org/10.48550/arXiv.2412.14401.

[18] Haritheja Etukuru, Norihito Naka, Zijin Hu, Seungjae Lee, Julian Mehu, Aaron Edsinger, Chris Paxton, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8275–8283, 2025. doi: 10.1109/ICRA55743.2025.11127857. URL https://doi.org/10.1109/ICRA55743.2025.11127857.

[19] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 2023.

[20] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021. URL https://arxiv.org/abs/2109.00137.

[21] Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023. URL https://doi.org/10.48550/arXiv.2311.01977.

[22] Arjun Gupta, Michelle Zhang, Rishik Sathua, and Saurabh Gupta. Opening cabinets and drawers in the real world using a commodity mobile manipulator. *arXiv preprint arXiv:2402.17767*, 2024.

[23] Siddhant Haldar and Lerrel Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. *arXiv preprint arXiv:2502.20391*, 2025. URL https://doi.org/10.48550/arXiv.2502.20391.

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. IEEE, 2020. doi: 10.1109/CVPR42600.2020.00975.

[25] Yingdong Hu, Fanqi Lin, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024. URL https://doi.org/10.48550/arXiv.2410.18647.

[26] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024. URL https://doi.org/10.48550/arXiv.2409.01652.

[27] Aadhithya Iyer, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024. URL https://doi.org/10.48550/arXiv.2403.07870.

[28] Arhan Jain, Mingtong Zhang, Kanav Arora, William Chen, Marcel Torne, Muhammad Zubair Irshad, Sergey Zakharov, Yue Wang, Sergey Levine, Chelsea Finn, et al. Polaris: Scalable real-to-sim evaluations for generalist robot policies. *arXiv preprint arXiv:2512.16881*, 2025. URL https://doi.org/10.48550/arXiv.2512.16881.

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

[30] Yash Jangir, Yidi Zhang, Kashu Yamazaki, Chenyu Zhang, Kuan-Hsun Tu, Tsung-Wei Ke, Lei Ke, Yonatan Bisk, and Katerina Fragkiadaki. Robotarena ∞: Scalable robot benchmarking via real-to-sim translation. *arXiv preprint arXiv:2510.23571*, 2025. URL https://doi.org/10.48550/arXiv.2510.23571.

[31] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. URL https://doi.org/10.48550/arXiv.2403.12945.

[32] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. In *CVPR*, 2017. URL https://arxiv.org/abs/1712.05474.

[33] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025. URL https://arxiv.org/abs/2508.07917.

[34] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024. URL https://doi.org/10.48550/arXiv.2403.03181.

[35] Mara Levy, Siddhant Haldar, Lerrel Pinto, and Abhinav Shirivastava. P3-po: Prescriptive point priors for visuo-spatial generalization of robot policies. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4167–4174. IEEE, 2025. URL https://doi.org/10.1109/ICRA55743.2025.11128755.

[36] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez,

et al. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024. URL https://doi.org/10.48550/arXiv.2403.09227.

[37] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024. URL https://doi.org/10.48550/arXiv.2405.05941.

[38] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

[39] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020. URL https://doi.org/10.15607/RSS.2021.XVII.047.

[40] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020. URL https://arxiv.org/abs/1903.01973.

[41] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024. URL https://robocasa.ai/.

[42] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. URL https://doi.org/10.1109/ICRA57147.2024.10611477.

[43] Physical Intelligence Team et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, volume 139, pages 8748–8763, 2021. URL https://arxiv.org/abs/2103.00020.

[45] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[46] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavika Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. URL https://doi.org/10.1109/ICCV.2019.00943.

[47] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[48] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022. URL https://doi.org/10.48550/arXiv.2206.11251.

[49] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022. URL https://doi.org/10.15607/RSS.2023.XIX.074.

[50] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023. URL https://doi.org/10.48550/arXiv.2311.16098.

[51] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022. URL https://arxiv.org/abs/2109.12098.

[52] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *Conference on Robot Learning (CoRL)*, 2022. URL https://doi.org/10.48550/arXiv.2209.05451.

[53] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://doi.org/10.48550/arXiv.2506.01844.

[54] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020. URL https://doi.org/10.1109/LRA.2020.3004787.

[55] Priya Sundaresan, Quan Vuong, Jiayuan Gu, Peng Xu, Ted Xiao, Sean Kirmani, Tianhe Yu, Michael Stark, Ajinkya Jain, Karol Hausman, et al. Rt-sketch: Goal-conditioned imitation learning from hand-drawn sketches. In *8th Annual Conference on Robot Learning*, 2024. URL https://doi.org/10.48550/arXiv.2403.02709.

[56] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and

motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[57] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. URL https://doi.org/10.48550/arXiv.2405.12213.

[58] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012. URL https://doi.org/10.1109/IROS.2012.6386109.

[59] vik. moondream2 (revision 92d3d73), 2024. URL https://huggingface.co/vikhyatk/moondream2.

[60] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023. URL https://doi.org/10.1109/IROS58592.2024.10801581.

[61] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. URL https://doi.org/10.48550/arXiv.2304.13705.

[62] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, Chelsea Finn, and Abhinav Gupta. Train offline, test online: A real robot learning benchmark, 2023.

APPENDIX

A. Data collection details

1) Visual gripper state estimation by SAM2: We resize the video observation to 256×256. At the start of each video, the gripper is fully open. We prompt SAM2 with positive points (green) belonging to the gripper, and negative point (red) outside the gripper, shown in 13, to generate a gripper segmentation mask for all frames. We compute the center of mass of the left and right gripper from the binary mask. The horizontal pixel distance between these two centers represents the gripper aperture, linearly mapped to 0 (fully closed) to 1 (fully open).

2) Data Filtering and Augmentation:

a) Static Frame Filtering: We filter out frames where the gripper does not significantly move. Starting from the first frame, we scan forward and select a new frame when the cumulative movement exceeds 0.3cm in translation, 0.1 radians in rotation, or 0.05 in gripper aperture.

b) Trajectory Mirroring: We apply horizontal flips on each recorded demo for both Open and Close tasks. We keep the original trajectory, and a horizontally mirrored copy of the trajectory with flipped visual observations and end effector poses.

c) Data Cleaning: We discard a trajectory if the tracked pose displacement exceeds 5cm between adjacent frames at 30Hz. For point tracking during inference, we find camera odometry acceptable but prefer forward kinematics since it is robust and readily available onboard. We avoid straight up/down poses during data collection, and limit the wrist pitch on Stretch to -1.5 rad to avoid gimbal lock.

B. Policy training details

1) Visual encoder pretraining: We pretrain ResNet-50 backbones with MoCo [24] on recorded RGB observations. The Pick encoder is trained on frames from 14,606 demonstrations (16.3 hours) across 289 environments, while the Open/Close encoder is trained on frames from 5,759 demonstrations (6.7 hours) across 135 environments. The linear projector for contact is shared across all frames.

TABLE IV: Hyperparameters used for Pick, Open, and Close tasks

Hyperparameter	Pick	Open	Close
Obs window size	3	3	3
Training Steps	308,565	364,811	277,522
Batch Size	256	200	200
Learning Rate	3e-4	1e-4	2.7e-4
Transformer Depth	8	8	8
Attn Heads	8	8	8
Embedding Dim	512	512	256
VQ-VAE codebook size	16	32	32
VQ-VAE embedding dim	512	512	512

C. Policy deployment details

For policy deployment on Stretch, we run inference at up to 2Hz directly on CPU on the onboard Intel NUC. For deployment on xArm, Franka, and other fixed embodiments, we run inference on an NVIDIA RTX A4000 GPU.

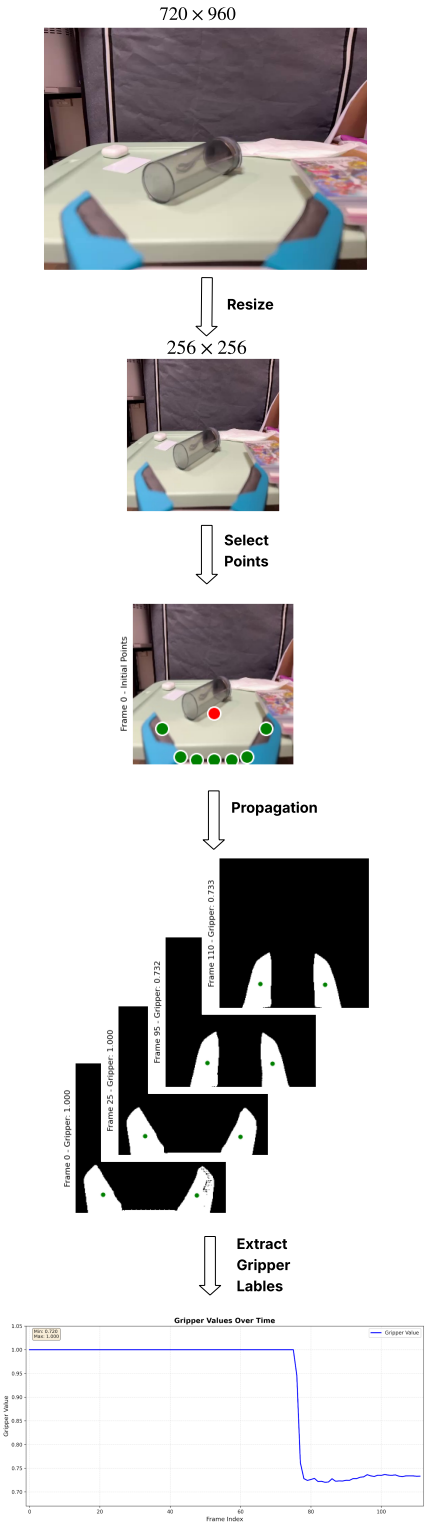


Fig. 13: Pipeline for Extracting Gripper Label By Using SAM2

All evaluation objects and environments used in Pick, Open, and Close are unseen during training. See Figure Fig. 19 for the 25 objects used Pick evaluation.

For each object used in the Pick and each door and drawer

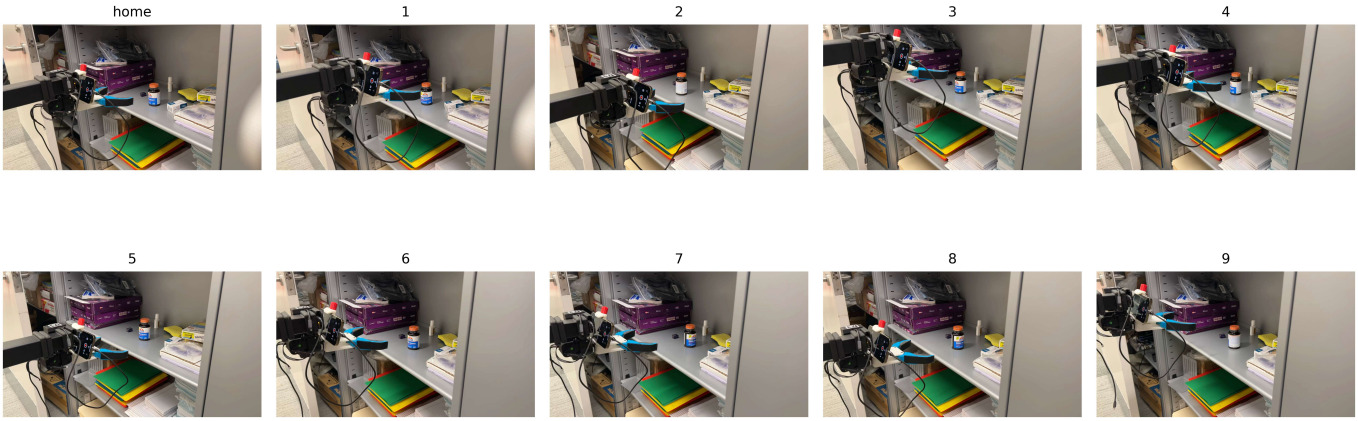


Fig. 14: Real robot configurations corresponding to each starting pose

used in the Open and Close, we have 10 trials. Correspondingly, we start the robot at 10 initial positions depicted in Fig. 14, Fig. 15.

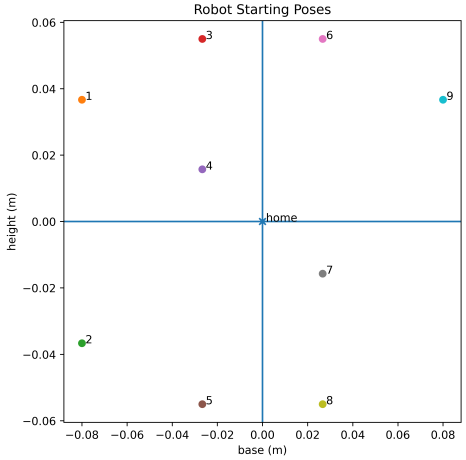


Fig. 15: Robot starting poses in the base–height parameter space

1) Evaluation object details: For the Pick task, see Fig. 19 for a display of the 25 objects used for evaluation. For the Open and Close tasks, see Fig. 5 for a display of the five doors and five drawers used for evaluation.

D. Simulation: EgoGym details

a) Available environments: EgoGym provides three Gymnasium environments:

- EgoGym-Pick-v0: a tabletop pick-and-lift task.
- EgoGym-Open-v0: opening an articulated object.
- EgoGym-Close-v0: closing an articulated object.

Figures 19 and 17 show visualizations of the EgoGym Pick and Open/Close environments.

b) Environment initialization arguments: Each environment is configurable through a set of arguments. The robot embodiment can be either *CAP* or *DROID*, and actions can be (*relative*) or (*absolute*). The scene is populated by sampling from a specified object set.

Optionally, environments can be wrapped with a VLM to provide unprivileged perception. Supported models include Moondream [59], Gemini-Robotics-ER-1.5 [56], and Molmo [16]. When no wrapper is used, the policy has privileged access to object identity.

c) Observations: All tasks expose a shared visual and proprioceptive observations. For the DROID embodiment, joint positions are additionally provided. In the Pick task, the observation includes the pose of the target object. In the Open and Close tasks, the observation instead includes the pose of the object's handle.

d) Rewards: Each environment provides a simple dense reward signal. In the Pick task, reward is given by the vertical displacement of the target object relative to its initial placement. In the Open task, reward corresponds to the fraction of the articulated object that has been opened. In the Close task, reward is computed as a residual-to-closed score from the object's opening percentage.

e) VLM Distractor evaluation protocol: For the Pick task, we evaluate robustness to distractors by varying the number of objects in the scene from one to five, across different VLMs. Evaluation is performed over 5,000 episodes. Episodes are considered successful if the maximum reward exceeds a threshold of 0.03. All experiments use a maximum horizon of 80 steps.

f) $\pi_{0.5}$ baseline (Pick): We additionally evaluate the $\pi_{0.5}$ baseline on EgoGym-Pick-v0 using the `pi05_droid_jointpos` checkpoint hosted at `gs://openpi-assets/checkpoints`. These experiments use a task horizon of 350 steps. Evaluation is performed over 5,000 episodes with the same success threshold as above.

g) Failure mode classification (Pick).: We assign each episode to exactly one outcome using an ordered decision tree (first match wins):

1) **Success**: Maximum lift > 0.03m.
2) **Did not lift enough**: The gripper fingers contacted the target object at least once and maximum lift ≥ 0.005m (≥ 0.5cm), but not successful.
3) **Object touched but not grasped**: The gripper fingers contacted the target object at least once, but did not lift.
4) **Picked wrong object**: The gripper fingers contacted a

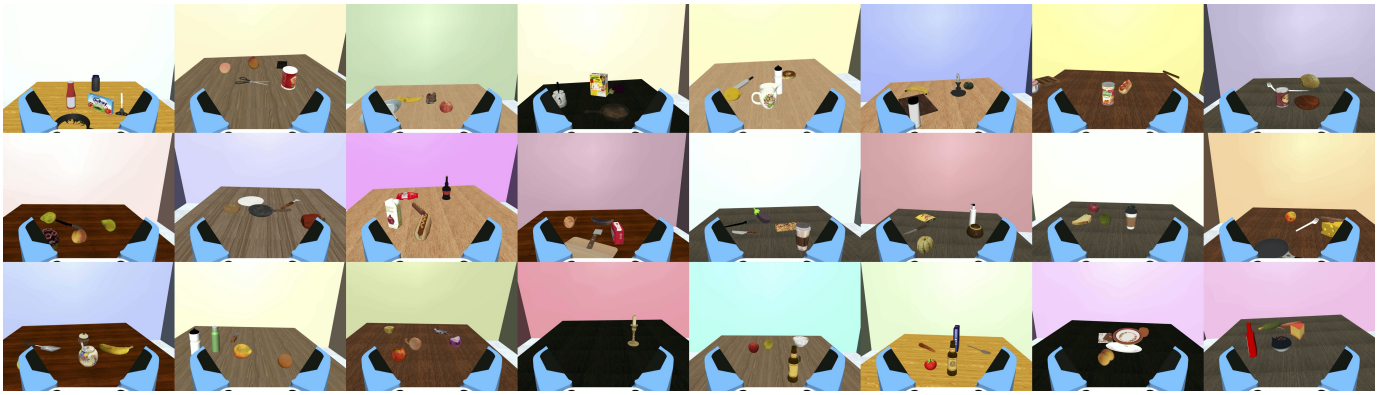


Fig. 16: EgoGym Pick environment visualizations

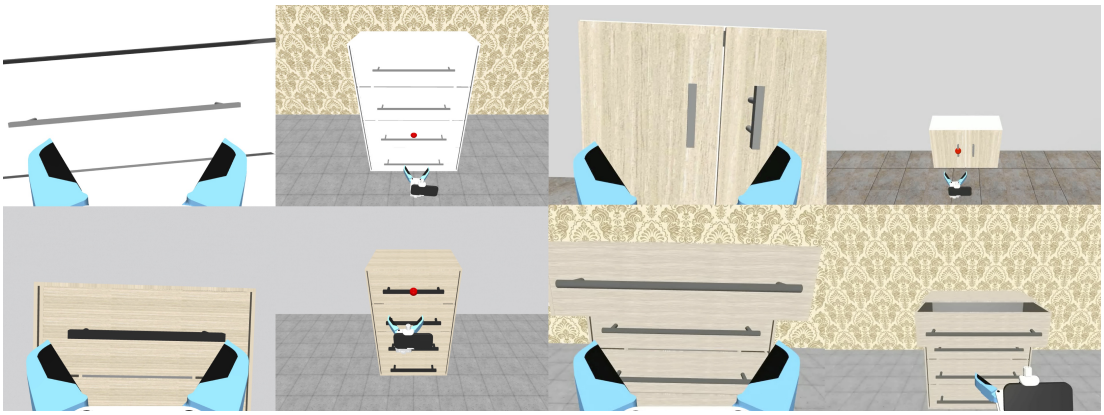
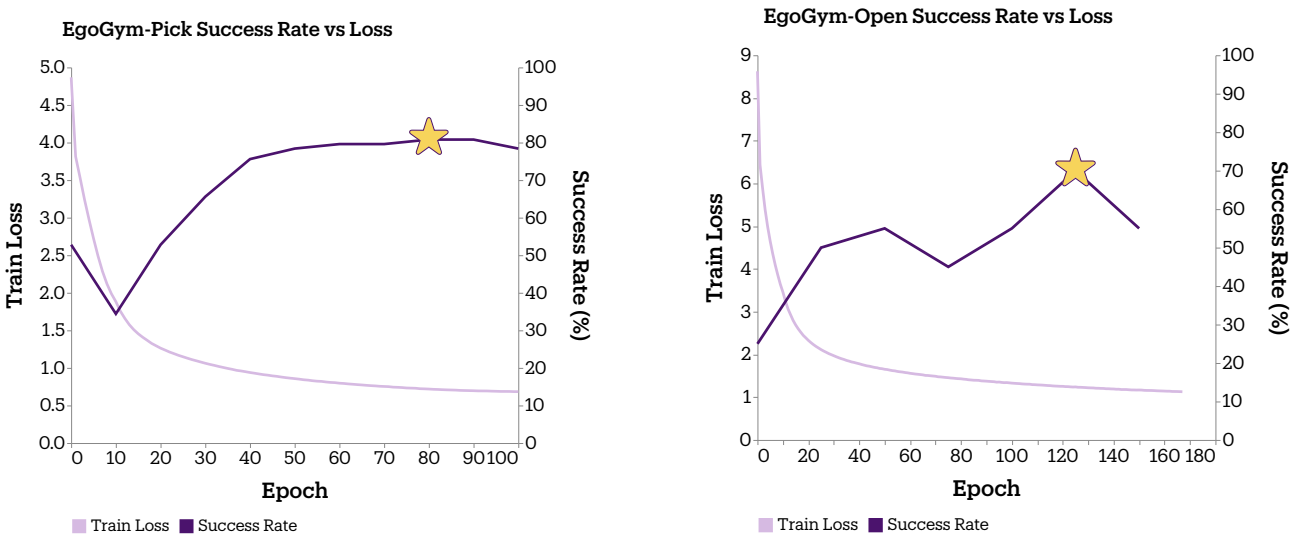


Fig. 17: EgoGym Open/Close environment visualizations

non-target object at least once, but never contacted the target object (i.e., only distractor contact).

5) **Empty Grasp**: None of the rules above triggered, but the episode ends with the gripper in a closed state or the gripper fingers made contact with each other, indicating a close-on-nothing.
6) **Did not grasp**: None of the above (no meaningful contacts).

h) Integration into the training loop: EgoGym is integrated into the training loop to periodically evaluate task success rates, since loss alone is not a reliable indicator of policy performance. Figures 18a and 18b show example CAP training runs, reporting simulation success rate alongside loss for the Pick and Open tasks, respectively.



(a) Pick task.

(b) Open task.

Fig. 18: Loss and simulation success rate over time during CAP training runs.

Fig. 19: Evaluation objects used for the Pick evaluations