

Behavior Prompting Policy: Demonstrations as Prompts for Manipulation

Austin Patel¹ Ben Pekarek¹ Joel Enrique Castro Hernandez² Shuran Song¹
¹Stanford University ²University of California, Berkeley
behavior-prompting.github.io

Abstract: We study behavior prompting, a paradigm that enables robots to perform new tasks at inference time given a single human demonstration, which we call a behavior prompt. To enable this capability, we present contributions in algorithm, data, and evaluation. For algorithm, we introduce Behavior Prompting Policy (BPP), an in-context visuomotor architecture that translates the behavior prompt and the current observation into robot actions. For data, we identify that task diversity is the primary driver of the prompting capability and introduce iPhUMI, a handheld manipulation interface for collecting diverse training data. For evaluation, we introduce DrawAnything and LIBERO-Gen to evaluate test-time adaptation to unseen drawing and tabletop manipulation tasks. We also demonstrate that iPhUMI serves as a practical interface for specifying behavior prompts at test time, enabling a human to command a robot via a single demonstration to complete known tasks or to define new robot capabilities. Altogether, behavior prompting provides a flexible and scalable way to teach robots new skills without the need for expensive fine-tuning.

Keywords: In-Context Learning, Visuomotor Policy, Manipulation

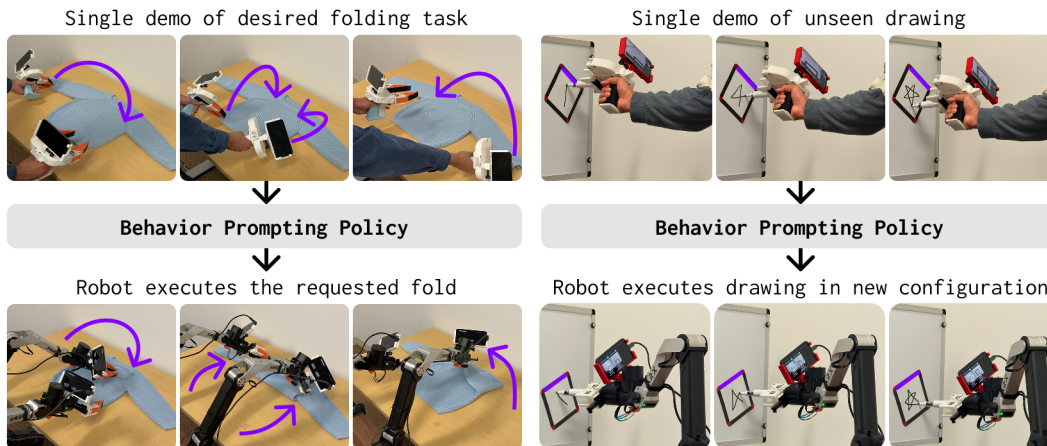


Figure 1: **Behavior prompting** conditions test-time execution on a single human demonstration. This enables a user to specify a task via demonstration (left) or define new robot capabilities (right).

1 Introduction

Teaching robots new skills typically requires exhaustive retraining or fine-tuning. In this paper, we propose *behavior prompting, a capability that enables robots to perform new tasks at test time given a single human demonstration* (Fig. 1). This human demonstration is called a behavior prompt, and consists of observations and actions in the sensorimotor space of the robot. It simultaneously defines *what* the intended task is along with one example of *how* to complete it.

This approach is inspired by the success of in-context learning in large language models, where models adapt to new tasks via few-shot examples provided at test time. Behavior prompting takes steps towards this capability in robotics, allowing users to rapidly deploy new skills “on the fly” without policy retraining. To realize this, we study three ingredients: algorithm, data, and evaluation.

Algorithm: How do we represent a behavior prompt? How do we use it in policy learning? While many prior works use language descriptions or goal image conditions, such methods often provide incomplete information about the task and desired behavior. In contrast, we show that a single task demonstration is an expressive prompt representation that is already accessible from existing demonstration data. However, leveraging it effectively requires the policy to simultaneously understand the *temporal correspondences* and *spatial differences* between the prompt demonstration and the robot’s current visual observations. To meet this requirement, we introduce **Behavior Prompting Policy** (BPP), an in-context visuomotor policy architecture that directly conditions on behavior prompts.

Data: What kind of training data enables behavior prompting to perform new tasks at test-time? We find that *task diversity* is crucial to enable execution of unseen tasks. Under a fixed data budget, policies trained on more tasks with fewer demonstrations per task exhibit significantly stronger prompting ability. To meet this data requirement, we introduce **iPhUMI**, a handheld data collection interface that requires *minimal* setup and *zero* mapping time, unlike teleoperation or the original UMI system [1]. This interface enables fast collection of diverse *training* data across tasks and provides a real-time interface during *testing* to specify behavior prompts for new tasks.

Evaluation: How do we evaluate test-time adaptation? Existing benchmarks lack sufficient task diversity or emphasize semantic or visual adaptation (e.g., new object categories) rather than *action adaptation* (e.g., new low-level behaviors). Thus, we introduce two benchmarks: 1) **DrawAnything**, a drawing environment focused on continuous, fine-grained action adaptation and 2) **LIBERO-Gen**, an extension of the LIBERO [2] manipulation benchmark with significantly more tasks. These benchmarks are *conceptually simple* yet capture *core challenges* of behavior prompting: closed-loop visual control, high task diversity, and the ability to specify new tasks at test time.

In summary, our contributions are threefold: **1)** Behavior Prompting Policy, an in-context visuomotor policy for behavior prompting **2)** A systematic study of the data requirements for behavior prompting, supported by iPhUMI, a practical human demonstration interface. **3)** New benchmark suites for drawing (DrawAnything) and tabletop manipulation (LIBERO-Gen) that facilitate reproducible scientific study of behavior prompting without requiring industrial-scale data collection.

2 Related Work

Language Adaptation to New Tasks. Multi-task training can improve performance across training tasks [3] and can enable adaptation to new tasks at test-time [4, 5]. Leveraging pretrained language embeddings, models have achieved zero-shot adaptation to some unseen environments and object categories [4, 6]. To enhance adaptation to new language commands, one common approach is to finetune a pretrained vision-language model to predict actions (VLA models) [5, 7, 8, 9, 10]. However, despite attempts to align actions with language, the zero-shot VLA capabilities are largely restricted to novel environments and objects rather than to new low-level actions or skills [9, 10].

Few-shot Learning for Manipulation. Few-shot learning methods adapt to new tasks via a small set of demonstrations for the target task. The approach by Finn et al. [11] uses one demonstration to fine-tune their policy with gradient-based updates in a meta-learning approach. Task parameterized methods [12] enable spatial adaptation by transforming explicitly defined object or robot-centric reference frames from a demonstration into a new environment, potentially using learning [13, 14].

Many methods enable adaptation by conditioning on demonstrations in-context. These demonstrations are represented as robot trajectories [15], human hand trajectories or scene keypoints [16], multi-modal prompts with text and images [17], or human video demonstrations [18]. To reason over the demonstration, methods compute attention over the in-context demonstration [15, 19] or include the demonstration in the context of a transformer model [18, 20, 21]. Injecting more priors about the task structure can yield sample-efficient adaptation at the cost of generality [22].

Behavior Prompting. Behavior prompting fits in the class of in-context, few-shot learning methods. The primary distinction is the choice to use a sensorimotor task demonstration as the prompt representation. Most relevant to our work is ICRT [23], an autoregressive in-context visuomotor policy that also conditions on behavior prompts. The authors validate their model on a single-arm,

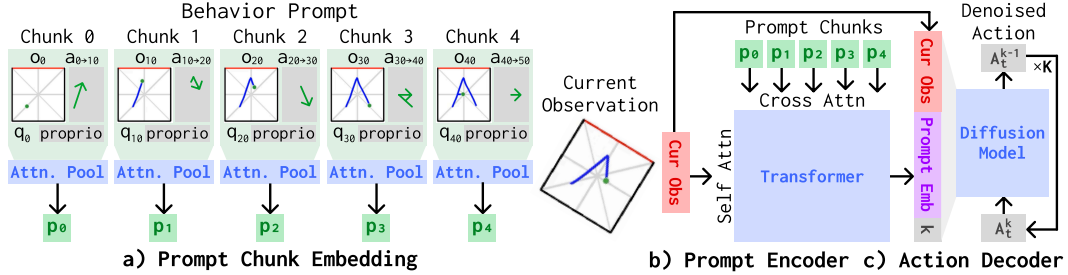


Figure 2: **Behavior Prompting Policy architecture.** a) Every Δt steps of the behavior prompt form a chunk that contains one step of observation and proprio along with Δt actions. Attention pooling merges $\{o, q, a\}$ into a chunk embedding p_i . The policy consists of: b) a prompt encoder, which extracts relevant prompt information given the current obs, and c) an action decoder, which reasons over the current obs and relevant prompt information to generate actions over K diffusion steps.

real-world tabletop manipulation dataset consisting of 1098 trajectories across 29 tasks and 6 motion primitives (picking, pick-and-place, stacking, pushing, poking, opening and closing drawers) [23].

Our work addresses an open question: *how does behavior prompting scale with greater task diversity and what test-time capabilities does that enable?* To study this, we introduce benchmarks with an order of magnitude larger task distributions (up to 2000 tasks) spanning a wider range of manipulation problems: tasks requiring continuous instead of just discrete instruction following (drawing), tabletop manipulation (LIBERO-Gen), and high-precision bimanual tasks (laundry folding). These benchmarks enable reproducible, scientific study of behavior prompting in future work.

3 Method

We define behavior prompts §3.1, detail when to use them §3.2, and introduce Behavior Prompting Policy to condition on them §3.3. We then introduce iPhUMI, a handheld manipulation interface for collecting diverse training data and for specifying behavior prompts at test time §3.4.

3.1 What is a behavior prompt?

A behavior prompt is a single demonstration of a desired task (at a different environment configuration) consisting of a sequence of observations, proprioception, and actions in the same sensorimotor space as the robot’s execution. While language and goal images typically provide information about *what* task needs to be completed, behavior prompts additionally provide spatial and temporal information that inform the policy *how* to complete the task. During deployment, we use a training demonstration as the prompt or, for new tasks, a human operator provides a single demonstration.

3.2 When should I use behavior prompting?

Behavior prompting is useful in multi-task settings when a task demonstration clarifies task-relevant information. An example is when the task distribution involves following spatial-temporal information (ex: *step-by-step drawings*). It is also useful when the tasks are more easily described through example than language (ex: *a set of human preferences for loading a dishwasher or the desired way to fold a new piece of laundry*). It’s also useful when a demonstration clarifies the manipulation strategy (ex: *a particular way to grasp a bottle that makes it easier to put on a shelf*). Even when goal image or language unambiguously define the set of tasks, our empirical results in §4 find that using behavior prompting can improve adaptation to new tasks given one demonstration.

3.3 Behavior Prompting Policy (BPP)

Behavior Prompting Policy is an in-context visuomotor architecture that takes a behavior prompt and current observation as input and outputs closed-loop actions (Fig. 2). BPP consists of a prompt encoder and an action decoder, like in [18]. Model comparisons to ICRT [23] are in the appendix.

Prompt Encoder (Fig. 2b): To encode the prompt, we temporally downsample the observations and proprioception for computational efficiency (typically to 1Hz). Actions are not downsampled to retain the full behavior sequence. The prompt is a sequence of chunks each consisting of a single timestep of observation o , proprioception q , and the sequence of actions a leading up to the next chunk. We apply attention pooling per chunk to merge $\{o, q, a\}$ into a single chunk embedding p_i forming the sequence $P = [p_0, p_1, \dots, p_n]$ where n varies with prompt length. The pooling temporally associates information from the same timestep and reduces the prompt sequence length.

The prompt encoder is a transformer decoder that extracts relevant prompt information given the current observation. We tokenize the current observation by representing each observation entry with a single token per timestep of history and then perform cross-attention with the prompt chunk embeddings P . We use learned positional embeddings for the prompt and the current observation.

Action Decoder (Fig. 2c): Given the relevant prompt information extracted by the prompt encoder, the policy now needs to reason how to generate actions. We concatenate the current observation, the extracted prompt information, and diffusion timestep k and pass it to a diffusion model to iteratively denoise actions. We use the CNN action diffusion architecture with FiLM [24] from Chi et al. [25].

Training: We follow standard behavior cloning training practices for action diffusion policies with additional care taken to handle the behavior prompt. For each training step, we sample a single demonstration from the training data as the prompt. We then load a batch of receding horizon observations and future action chunks from other demonstrations of the *same task*. Each demonstration will have varying environment configurations, so the policy must reason about the temporal similarities and spatial differences between the prompt and the current observation to generate actions.

In this training paradigm *we require no explicit correspondence, spatially or temporally, between two demonstrations of the same task*. The prompt is simply provided as an additional model input and the model learns end-to-end how to best leverage the prompt. As behavior prompts consist of the same information as normal demonstrations, *BPP can be directly trained on existing multi-task imitation learning datasets with no additional data collection*. In general, the task groupings define the granularity at which behavior prompts can influence robot execution. For example, a pick-place task with two distinct object grasping strategies for one object would require two separate task groupings for the robot to adhere to the grasping strategy shown in the prompt.

Inference: During inference, we select a single prompt per rollout. This means that we can generate and cache the prompt chunk embeddings once per rollout and reuse them for each inference step. The action decoder is also decoupled from the prompt encoder, meaning we handle the extraction of relevant prompt information once per inference step. After that, we can do many action diffusion steps without having to reference the entire prompt each denoising step.

3.4 iPhUMI

A practical behavior prompting system requires an intuitive interface to collect diverse training data and to specify behavior prompts for new tasks at deployment. To meet both needs, we introduce iPhUMI, a handheld data collection interface for behavior prompting. Adapted from the UMI gripper [1], iPhUMI retains the core design but replaces the GoPro with an iPhone 15 Pro. This provides two concrete benefits: 1) *Instant localization*: we leverage on-device ARKit for real-time SLAM, bypassing the tedious environment mapping step. 2) *Wireless prompting*: our iPhUMI app can wirelessly transmit behavior prompts to a workstation to immediately condition the policy at test time.

4 Evaluation

Behavior prompting is a fundamentally different paradigm than single-task policies in that we need to focus on *high task diversity, rather than single task complexity*. We introduce DrawAnything and LIBERO-Gen as two new benchmarks (Fig. 3). Both benchmarks have high task diversity, the ability to procedurally generate demonstration data, and facilitate specifying new tasks at test time. *These properties enable study of behavior prompting without requiring industrial-scale data collection*.

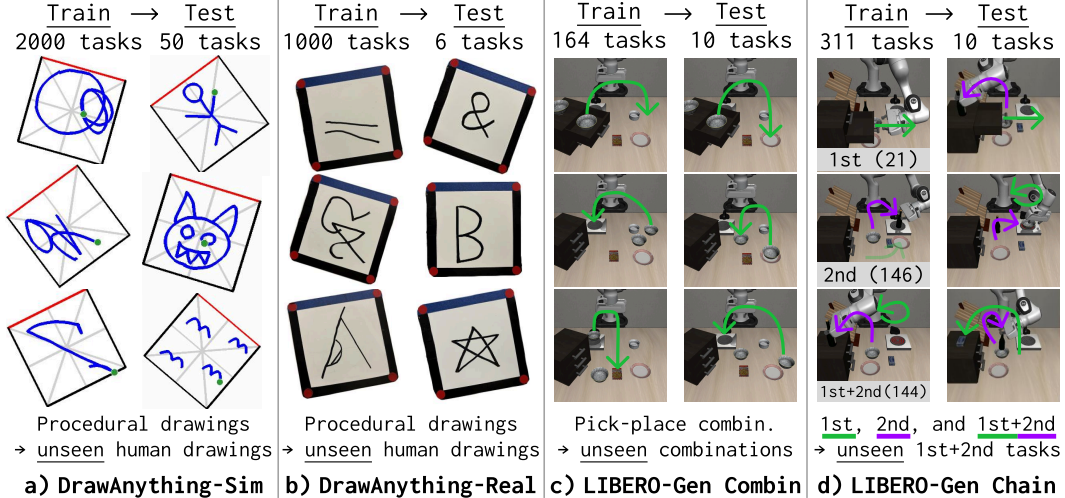


Figure 3: **Benchmarking Suite.** In DrawAnything (a, b), we evaluate whether a policy can recreate a previously unseen drawing at varying board poses given a single human demo. In LIBERO-Gen Combination (c) two identical bowls are randomly positioned, and the robot is given instructions for which one to grasp and where to place it. In LIBERO-Gen Chain (d) we explore the set of two step interactions. We have: 1) *first step tasks* include open middle/top drawer, push plate, turn on stove, and pick-place, 2) *second step tasks* just do second action (pick-place only) with first action already done, and 3) *chained tasks* where we do a first step task then a second step task in succession.

- **DrawAnything-Sim (Fig. 3a):** Drawing is well suited for behavior prompting as it requires the policy to continuously reference low-level drawing instructions in the prompt. We train on 2000 procedurally generated drawing tasks with 5 demos per task at randomized board orientations. We evaluate on 50 unseen drawing tasks collected by a human to ensure significant variation from the training tasks. During evaluation, the policy must reconstruct a previously unseen drawing given a single human demo. The rollout occurs at a different board orientation than the prompt, meaning the policy must understand the spatial differences between the prompt and rollout.
- **DrawAnything-Real (Fig. 3b):** We create an analogous real-world drawing setup using an ARX robot arm with an iPhone wrist camera and marker attached to the end effector (Fig. 1). Within a large whiteboard we have a square drawing region that can be placed at any position and in a 90° rotation range. We collect 1000 training tasks (200 tasks at 5 demos/task from a human using iPhUMI and 800 tasks at 6 demos/task with a scripted policy). Evaluation is done on 10 tasks (4 training, 6 unseen) collected by a human using iPhUMI. This benchmark requires full 6DoF action (compared to 2D in sim) and robustness to visual occlusions caused by the marker.
- **LIBERO-Gen Combination (Fig. 3c):** We introduce LIBERO-Gen, a procedural generation framework for the LIBERO [2] tabletop manipulation benchmark. Compared to other LIBERO extensions that focus on visual or semantic robustness [26, 27], LIBERO-Gen helps evaluate *test-time instruction-following capability on unseen tasks* by generating new environments, tasks, and demonstrations. Using this tool, we create LIBERO-Gen Combination which extends the 10 tasks in LIBERO Spatial with 164 more tasks in the same environment. All tasks instruct the policy to pick one of the two identical bowls and place it on one of nine locations. For evaluation, we hold out 10 “combinations” of pick-place locations seen individually during training, but never jointly.
- **LIBERO-Gen Chain (Fig. 3d):** Using LIBERO-Gen, we create LIBERO-Gen Chain, which extends the 10 original tasks in LIBERO Goal with 311 additional tasks in the same environment. The tasks involve sequentially executing two single-step tasks in a “chain.” They extend beyond just pick-and-place (open middle/top drawer, push plate, turn on stove) and involve several objects. To understand whether behavior prompting can perform unseen, long-horizon manipulation skills at test time, we hold out 10 two-step tasks consisting of individually seen manipulation skills.

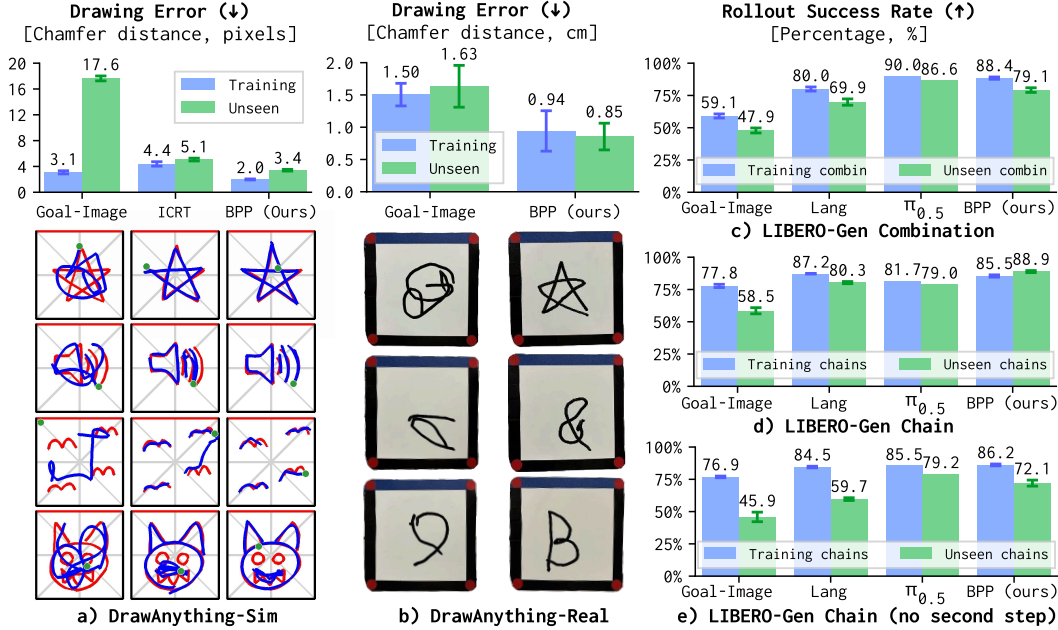


Figure 4: **Benchmark Results.** For DrawAnything (a,b) we find Goal-Image performs well only on training drawings, while BPP (ours) and ICRT [23] perform well on unseen drawings. Side-by-side qualitative results in (a,b) are for *unseen* drawings. For unseen manipulation tasks in LIBERO-Gen (c,d,e), BPP outperforms baselines and rivals $\pi_{0.5}$ despite not having foundation pretraining. We report the ± 1 stdev error bar across three seeds for sim and across tasks for a single seed for the real-world drawing. The $\pi_{0.5}$ [5] results are for one seed after 100K LoRA [28] finetuning steps.

4.1 Key Findings

Eval Q: Does behavior prompting work? A: Yes, it can improve test-time adaptation.

We compare to Goal-Image and Language that match BPP but replace prompting with ViT goal image encoding [30] and finetuned CLIP language encoding [31], respectively. We also compare to ICRT [23], a prior behavior prompt model and finetuned $\pi_{0.5}$ [5], a VLA with foundation pretraining.

DrawAnything: We observe BPP can reconstruct new drawings from a single human demonstration via mouse in sim or iPhUMI in real (Fig. 4a,b). For unseen drawings in sim, BPP achieves a substantial 80.7% error reduction compared to Goal-Image and a 33.3% error reduction compared to ICRT [23]. A single goal image only indicates *what* drawing to do, while a behavior prompt describes a step-by-step example for *how* to complete a drawing. Unlike BPP, ICRT retains the entire rollout history in the model context, making it susceptible to OOD due to spurious correlations [32].

LIBERO-Gen: Across both LIBERO-Gen benchmarks, BPP improves test-time adaptation to unseen manipulation tasks. For LIBERO-Gen Combination there are two decisions (which bowl to pick and where to place it) and for LIBERO-Gen Chain there are up to four (ex1: *pick, place, pick, place*, ex2: *turn stove knob on, pick, place*). The language command explicitly indicates these steps, while the goal image only provides the final state. In Fig. 4e, we reduce the LIBERO-Gen Chain training task distribution from 1st,2nd,1st+2nd \rightarrow 1st,1st+2nd as an ablation. Removing 2nd step tasks makes adaptation to unseen chained tasks (1st+2nd) more challenging as the policy will have never seen the 2nd step after the 1st step has completed (ex: *never seen how to put the wine on the cabinet after a cabinet drawer is opened*). In this ablation, BPP achieves a 20.8% gain over Language (Fig. 4e) compared to a 10.7% gain over Language in Fig. 4d. Additionally, despite having no pretraining, BPP rivals the $\pi_{0.5}$ [5] foundation VLA model finetuned on LIBERO-Gen.

Summary: We find the benefits of more temporally-rich task descriptors (*goal image \rightarrow language \rightarrow behavior prompt*) are more pronounced as the temporal task complexity increases (*single step pick-place \rightarrow two-step complex chained manipulation \rightarrow dense drawing*).

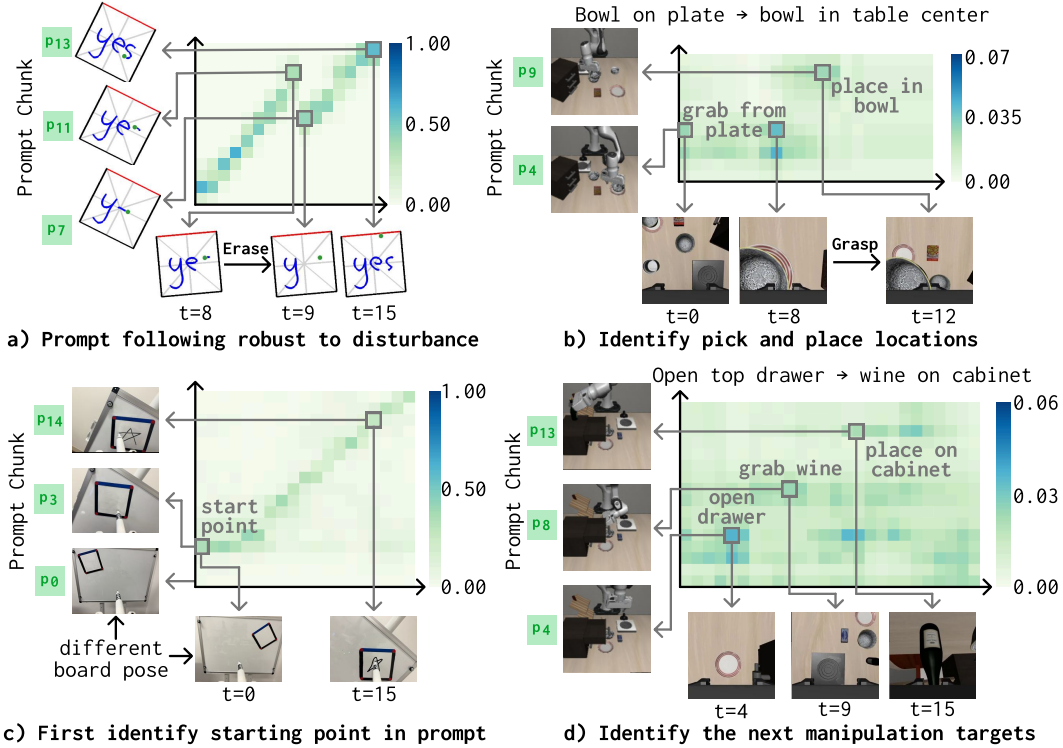


Figure 5: **Prompt encoder attention (unseen tasks)**. We visualize the normalized attention scores in the BPP prompt encoder on *unseen* tasks during rollout (x-axis) to see how the model attends to the prompt (y-axis). For DrawAnything (a,c) the policy attention continuously tracks the portion of the prompt closest to the current observation, while in LIBERO-Gen (b: Combination, d: Chain) the attention tracks discrete “milestones” in the task. The lower attention magnitude in LIBERO-Gen is due to the use of learned attention sink tokens [29] added to the start of the prompt (not shown).

Alg Q: How is the behavior prompt used? A: The prompt provides dense sub-goals.

In Fig. 5, we visualize the attention in the prompt encoder. For DrawAnything, the attention closely follows the task progression, indicating that BPP identifies *temporal similarities* through a prompt lookup operation to find the section most similar to the current observation. As a result, the policy can extract upcoming states and actions that inform action generation after accounting for *spatial differences* between the prompt and rollout board pose. This *step-by-step guidance largely simplifies the learning complexity* compared to Goal-Image, which must reconstruct complex tasks from just the final state. For LIBERO-Gen, we also find that the attention follows task progression, though in a more discrete fashion to identify the next key event, such as a transition between tasks, the next object to interact with, or where to place an object. ***In short, BPP achieves test-time adaptation through dense sub-goal conditioning on the prompt.***

All upcoming ablations are specific to DrawAnything-Sim and findings may vary for other domains.

Alg Q: What is a good representation for a behavior prompt? A1: Including multi-modal sensorimotor information (Fig. 6a). For DrawAnything-Sim, we find that including observations in the prompt is necessary to anchor the prompt lookup and that actions provide useful temporal transitions between the downsampled observations. Proprioception (i.e., cursor position) is not useful as the cursor is already visually shown in the observation image. **A2: Including observations at a sufficiently high frequency** (Fig. 6b). Aggressive downsampling of the prompt observations (less than 1Hz) makes BPP struggle to follow demonstrations for unseen drawings. **A3: Using attention pooling to aggregate the modalities** (Fig. 6c). Attention pooling helps temporally associate modalities from the same prompt chunk. It also reduces the prompt sequence length by merging each prompt chunk into a single embedding, rather than having separate tokens per modality.

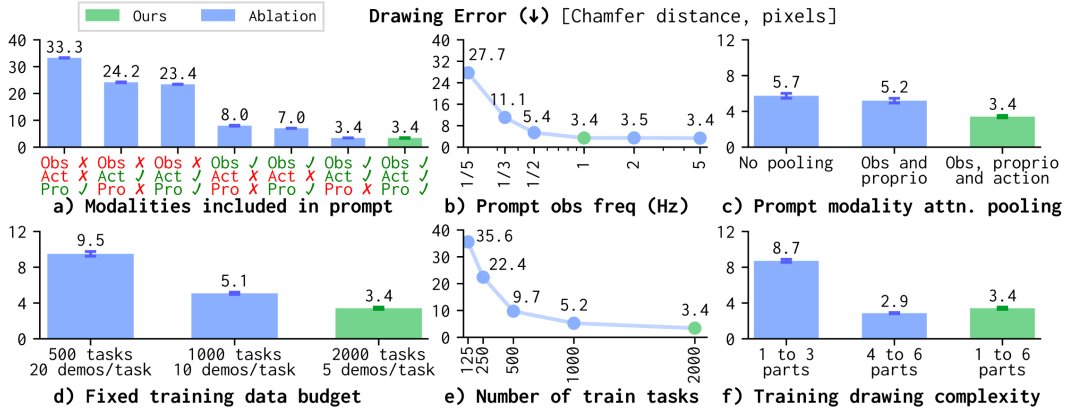


Figure 6: **BPP ablations on DrawAnything-Sim.** We report performance on *unseen* drawing tasks (3 seeds). [Top row] We ablate what information is included in the prompt and the impact of applying attention pooling to the prompt tokens. [Bottom row] We ablate the composition of our training data.

Data Q: What type of training data enables behavior prompting to execute unseen tasks?

A1: Task diversity is more important than quantity per task. For a fixed demonstration budget, collecting a few demonstrations per task for many different drawing tasks enables better test-time adaptation compared to many demonstrations per task for few tasks (Fig. 6d). We also find that increasing the number of training tasks improves performance on unseen tasks with just 5 demos per task (Fig. 6e). **A2: Including complex training tasks.** We ablate the number of parts (line, circle, oval, Bézier curve) that are stitched together in the procedurally generated drawings that form the training set (Fig. 6f). We find that training only on low complexity drawings (1 to 3 parts) yields poor adaptation to complex, unseen drawings. Training on only complex drawings (4 to 6 parts) yields the best performance, but at the expense of a substantially larger demonstration dataset.

4.2 Case Study: Laundry Folding

We apply behavior prompting to three real-world laundry folding tasks (Fig. 1). A human is able to command a bimanual robot to complete a folding task from the training set by prompting BPP with a single iPhUMI demonstration. However, we observe that BPP exhibits weaker task conditioning compared to language conditioning in this low task diversity setting. See Appendix §A for details.

5 Limitations

In many cases, goal images or language may be sufficient to achieve the desired level of test-time adaptation and do not require collecting a full behavior prompt. Thus we envision extending BPP to flexibly use different task descriptors. Another limitation is the substantial training data diversity needed to enable test-time adaptation. For tabletop manipulation, we do not yet find evidence that BPP can adapt to entirely new action primitives given a single behavior prompt. We also find weaker task conditioning in settings with low training task diversity compared to language conditioning. For our experiments, the behavior prompts occur at different object configurations than deployment, but are still within the same environment; future work can study performance when the prompt and execution environment substantially differ. Future work could also investigate hardware interfaces for behavior prompting with dexterous hands, prompting real-world execution with behavior prompts from simulation, and applying behavior prompting to models with foundation-level pretraining.

6 Conclusion

We hope that behavior prompting broadly enables a pathway for specifying *human preferences* to robots at test time via demonstration, such as a preferred way to fold a piece of clothing. We also envision behavior prompting as a means to adapt pretraining knowledge (i.e., large foundation behavior prompting model) to a new environment (e.g., a person’s home) through a single demonstration for a target task; a demonstration provides rich information about the target environment and desired manipulation strategy that *simplifies* the adaptation problem.

Acknowledgments

We would like to thank the REAL lab for continuous project support, in particular Huy Ha for their insightful project guidance, Yihuai Gao for their help with the robot deployment setup, and Max Du for help with data collection. Regarding iPhUMI development, Yihuai Gao contributed USB streaming support and Xiaomeng Xu extended the bimanual data collection feature to support head-mounted iPhone data collection. We would also like to thank Chen Chen, Benoit Landry, Walter Talbott, and Jian Zhang for their feedback and discussions. This work was supported in part by the NSF Graduate Fellowship, NSF Award #2143601, #2037101, and #2132519, and Apple. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

- [1] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. URL <https://arxiv.org/abs/2402.10329>.
- [2] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.
- [3] T. L. Team, J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, N. Kuppaswamy, K.-H. Lee, K. Liu, D. McConachie, I. McMahon, H. Nishimura, C. Phillips-Grafflin, C. Richter, P. Shah, K. Srinivasan, B. Wulfe, C. Xu, M. Zhang, A. Alspach, M. Angeles, K. Arora, V. C. Guizilini, A. Castro, D. Chen, T.-S. Chu, S. Creasey, S. Curtis, R. Denitto, E. Dixon, E. Dusel, M. Ferreira, A. Goncalves, G. Gould, D. Guoy, S. Gupta, X. Han, K. Hatch, B. Hathaway, A. Henry, H. Hochsztein, P. Horgan, S. Iwase, D. Jackson, S. Karamcheti, S. Keh, J. Masterjohn, J. Mercat, P. Miller, P. Mitiguy, T. Nguyen, J. Nimmer, Y. Noguchi, R. Ong, A. Onol, O. Pfannenstiehl, R. Poyner, L. P. M. Rocha, G. Richardson, C. Rodriguez, D. Seale, M. Sherman, M. Smith-Jones, D. Tago, P. Tokmakov, M. Tran, B. V. Hoorick, I. Vasiljevic, S. Zakharov, M. Zolotas, R. Ambrus, K. Fetzer-Borelli, B. Burchfiel, H. Kress-Gazit, S. Feng, S. Ford, and R. Tedrake. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026. doi:10.1126/scirobotics.eaea6201.
- [4] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. 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>.
- [5] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.
- [6] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.
- [7] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

- [8] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [9] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL <https://proceedings.mlr.press/v229/zitkovich23a.html>.
- [10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [11] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.
- [12] S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016. ISSN 1861-2776. doi:10.1007/s11370-015-0187-9.
- [13] X. Zhang and A. Boularias. One-shot imitation learning with invariance matching for robotic manipulation. *arXiv preprint arXiv:2405.13178*, 2024.
- [14] E. Valassakis, G. Papagiannis, N. D. Palo, and E. Johns. Demonstrate once, imitate immediately (DOME): Learning visual servoing for one-shot imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
- [15] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *Advances in Neural Information Processing Systems*, 30, 2017.
- [16] G. Papagiannis, N. D. Palo, P. Vitiello, and E. Johns. R+x: Retrieval and execution from everyday human videos. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [17] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. VIMA: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.
- [18] V. Jain, M. Attarian, N. J. Joshi, A. Wahid, D. Driess, Q. Vuong, P. R. Sanketi, P. Sermanet, S. Welker, C. Chan, I. Gilitschenski, Y. Bisk, and D. Dwibedi. Vid2Robot: End-to-end video-conditioned policy learning with cross-attention transformers. In *Proceedings of (RSS) Robotics Science and Systems*. Proceedings of Robotics: Science and Systems, May 2024.
- [19] Z. Mandi, F. Liu, K. Lee, and P. Abbeel. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2434–2444, 2022. doi:10.1109/ICRA46639.2022.9812450.

- [20] M. Xu, Y. Shen, S. Zhang, Y. Lu, D. Zhao, B. J. Tenenbaum, and C. Gan. Prompting decision transformer for few-shot policy generalization. In *Thirty-ninth International Conference on Machine Learning*, 2022.
- [21] R. Shah, S. Liu, Q. Wang, Z. Jiang, S. Kumar, M. Seo, R. Martín-Martín, and Y. Zhu. MimicDroid: In-context learning for humanoid manipulation from human play videos. *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [22] K. Dreczkowski, P. Vitiello, V. Vosylius, and E. Johns. Learning a thousand tasks in a day. *Science Robotics*, 10(108):eadv7594, 2025. doi:10.1126/scirobotics.adv7594. URL <https://www.science.org/doi/abs/10.1126/scirobotics.adv7594>.
- [23] L. Fu, H. Huang, G. Datta, L. Y. Chen, W. C.-H. Panitch, F. Liu, H. Li, and K. Goldberg. In-context imitation learning via next-token prediction. *International Conference on Robotics and Automation (ICRA)*, 2025. URL <https://arxiv.org/abs/2408.15980>.
- [24] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [25] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- [26] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, J. Fu, J. Gong, and X. Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.
- [27] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.
- [28] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [29] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pages 21875–21895, 2024.
- [30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.
- [31] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.
- [32] M. Torne, A. Tang, Y. Liu, and C. Finn. Learning long-context diffusion policies via past-token prediction. In *9th Annual Conference on Robot Learning*, 2025.
- [33] J. Fang, W. Chen, H. Xue, F. Zhou, T. Le, Y. Wang, Y. Zhang, J. Lv, C. Wen, and C. Lu. RoboPocket: Improve robot policies instantly with your phone. *arXiv preprint arXiv:2603.05504*, 2026. URL <https://arxiv.org/abs/2603.05504>.
- [34] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. M. 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. IEEE, 2025.

- [35] X. Xu, J. Park, H. Zhang, E. Cousineau, A. Bhat, J. Barreiros, D. Wang, J. Bohg, and S. Song. HoMMI: Learning whole-body mobile manipulation from human demonstrations. *arXiv preprint arXiv:2603.03243*, 2026.
- [36] Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, B. Burchfiel, and S. Song. Maniwav: Learning robot manipulation from in-the-wild audio-visual data. In *Conference on Robot Learning*, pages 947–962. PMLR, 2025.
- [37] Y. Gao, J. Liu, S. Li, and S. Song. Gated memory policy. *arXiv preprint arXiv:2604.18933*, 2026. URL <https://arxiv.org/abs/2604.18933>.

Appendix

A Case Study: Laundry Folding

Behavior prompts are a substantially more complex task representation than fixed-length language embeddings, and we want to understand whether this complexity poses challenges when we have *low* task diversity. To study this, we perform a case study with three sweater folding tasks (Fig. 1) collected with bimanual iPhUMI: *fold left arm* (requires only left robot arm), *fold right arm* (requires only right robot arm), and *fold bottom up* (requires both arms to individually grab the bottom of the sweater and then jointly fold it to the top). All tasks start with the sweater fully unfolded, so the policy must leverage the task descriptor to identify the correct task. We compare behavior prompting, where the policy must identify the task from the high amount of temporal and spatial prompt information, to CLIP [31] language conditioning, where the finetuned language encoding simply and directly identifies the task.

We present results in Tab. 1. We find that the Language completes all tasks with high success rate, while BPP occasionally completes a different task than instructed (Fig. 7). ***For this experiment, the spatial and temporal prompt information does not further clarify these tasks and instead introduces complexity and variation (i.e., prompts vary in duration and spatial configurations of objects), which we postulate causes BPP to have weaker task conditioning when there is low training task diversity.*** This indirectness might make BPP sensitive to overfitting to spurious task cues in the training data (e.g., background variations more often seen for a particular task).

Task	Result	Language	BPP (Ours)
<i>Fold left arm</i>	Success	24 (96%)	19 (76%)
	Failure (dropped left sleeve)	1 (4%)	4 (16%)
	Failure (did <i>fold right arm</i>)		1 (4%)
	Failure (did <i>fold bottom up</i>)		1 (4%)
<i>Fold right arm</i>	Success	25 (100%)	25 (100%)
	Success	25 (100%)	15 (60%)
<i>Fold bottom up</i>	Failure (did <i>fold right arm</i>)		6 (24%)
	Failure (right: grabs bottom ✓, left: <i>fold left arm</i> ✗)		4 (16%)

Table 1: **Laundry folding results.** We observe that BPP exhibits lower success rate compared to language conditioning as the policy occasionally completes the wrong task. Success rates are for one seed across 25 rollouts per task. The Language baseline is detailed in §F. Using iPhUMI, we collect ~150 full demonstrations per task and ~1000 total error correction demonstrations. We collect five additional behavior prompts per task so these evaluations are for *unseen* prompts for *seen* tasks. The *fold bottom up* task first grabs the bottom with the right arm and then with the left arm; we often observe failures where BPP correctly grasps the bottom with the right arm, but the left arm instead decides to fold the left sleeve instead of also grasping the bottom.

While BPP faces challenges in this low task diversity regime, we envision behavior prompting having substantial advantages with training data spanning *many* folding styles across *many* garments. In particular, a behavior prompt captures the *temporal* steps in the folding process as well as the *spatial* interaction points, both of which clarify and inform the desired execution in a high task diversity setting. This information could also help with visual adaptation to unseen garments and temporal adaptation to unseen folding orders. Practically, a user might also find it more natural to describe their desired folding preferences via a single demonstration, rather than through language.

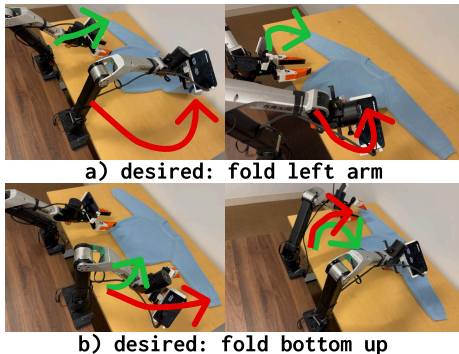


Figure 7: **BPP exhibits weak task conditioning under low task diversity.** Green is desired action, red is actual execution.

B DrawAnything-Real implementation

B.1 Scripted data generation

For our real drawing task we train on a mixture of data collected via iPhUMI and a scripted policy. Specifically, we collect 200 tasks with 5 demonstrations per task from a human using iPhUMI. We also collect 800 tasks with 6 demonstrations per task using a scripted policy. The scripted policy maps procedurally generated drawings used in DrawAnything-Sim to open-loop commands in the 6DOF robot end-effector space. Using iPhUMI data during training enables the policy to work well with iPhUMI prompts coming from humans at test time and the scripted policy helps reach the task diversity required to enable adaptation to new tasks at test time.

For each scripted demonstration, a human operator places the drawing canvas at a random position and orientation on the whiteboard. We localize the four red corner markers on the drawing canvas by running a segmentation model on an RGB image from the iPhone’s main rear camera, yielding 2D keypoints in the camera frame that we use to estimate the canvas pose. We then transform the resulting pose into the robot base frame using a fixed camera-to-robot extrinsic calibration. The robot first moves to a random pose in front of the whiteboard, starts recording, then moves to the estimated canvas pose and aligns its end-effector orientation with the canvas. It executes the scripted drawing trajectory for the task and finally retreats to a random pose. This is repeated across tasks and demonstrations.

B.2 Evaluation procedure

Qualitative results of BPP on some unseen drawings are shown in Fig. 8. Training and test tasks used for evaluation are shown in Fig. 9. For each evaluation trial, the drawing canvas is placed at a random position and orientation on the whiteboard. We sample a single demonstration to define a matched conditioning pair: the full demonstration is used as the behavior prompt for the prompting policy, and the corresponding goal image extracted from that same demonstration is used for the goal-image policy.

To control for initial conditions, we execute the prompting policy, reset the robot and environment to the same initial state, and then execute the goal-image policy with the matched goal image. We repeat this procedure after moving the canvas to a new random placement and sampling a new prompt/goal-image pair.

For evaluation on training tasks, conditioning pairs are sampled from the set of six demonstrations collected from four of the procedural training tasks. For evaluation on unseen test tasks, conditioning pairs are sampled from a set of three iPhUMI demonstrations collected for each of the six unseen evaluation tasks. Ten evaluation rollouts are done for each task.

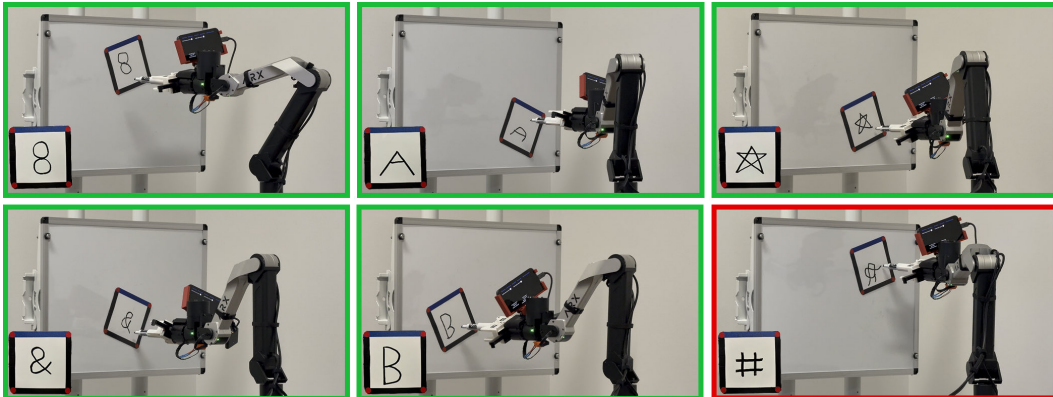


Figure 8: **BPP qualitative results on unseen tasks in DrawAnything-Real.** We find that BPP is able to reconstruct unseen drawings given a single iPhUMI demo. We show successful drawings (green) and a failure case (red).

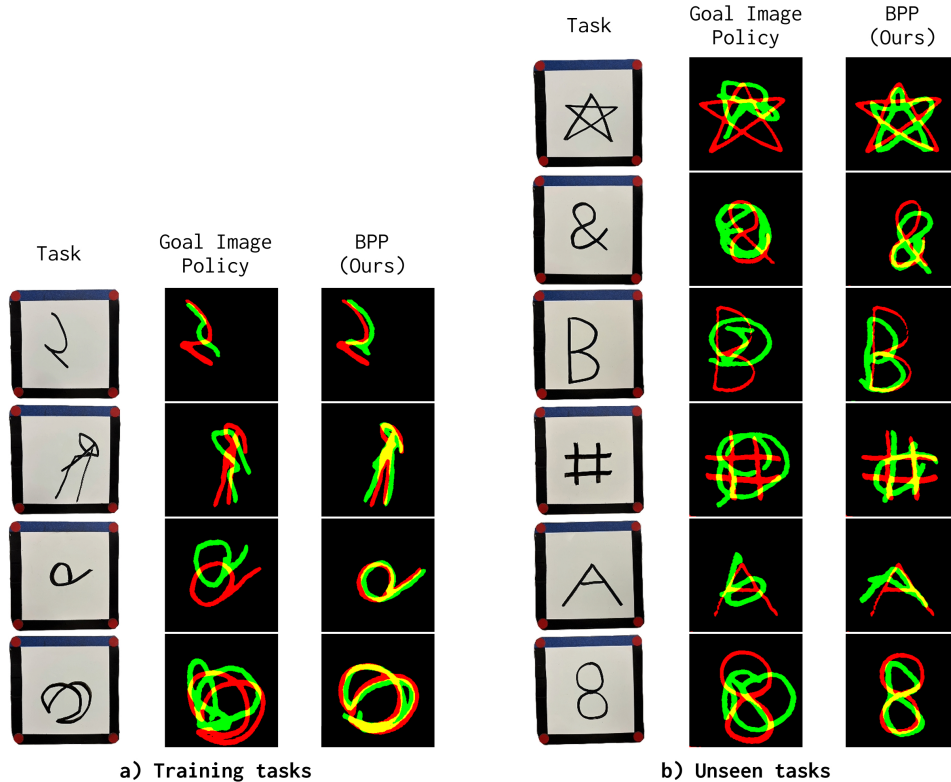


Figure 9: **DrawAnything-Real evaluation tasks with representative examples of robot executions.** Green: executed drawing by policy. Red: goal image input rendered as a reference overlay. We observe that Goal-Image can roughly match the structure of the training drawings, but fails to replicate unseen drawings. BPP is able to reconstruct both training and unseen drawings given a single demonstration.

B.3 Qualitative comparison: goal image vs. behavior prompting

Fig. 9 shows representative examples comparing BPP to Goal-Image on training and unseen drawing tasks. In DrawAnything-Sim, we observed that Goal-Image performs well on training tasks (Fig. 4a). In DrawAnything-Real, we find however that Goal-Image underperforms quantitatively on training tasks (Fig. 4b). However, qualitative inspection of Fig. 9a suggests a consistent failure mode: the policy often produces a recognizable rendition (or partial rendition) of the intended shape, but with spatial misalignment or local geometric errors that cause large Chamfer distances.

For unseen tasks, Goal-Image frequently degenerates into near-random strokes that do not resemble the target shape, whereas BPP produces coherent drawings aligned with the demonstrated behavior. Interestingly, we find that Goal-Image has similar quantitative error rates between training and unseen tasks (Fig. 4b). However, we see that unlike on training tasks, Goal-Image has poor qualitative fidelity on these test tasks (Fig. 9b). We believe this is because the unintelligible drawings produced by Goal-Image on test tasks can achieve moderate levels of Chamfer error by roughly covering the target drawing region without reproducing the intended structure.

C DrawAnything-Sim Tasks

We visualize the training (Fig. 10) and evaluation dataset (Fig. 11) for DrawAnything-Sim.

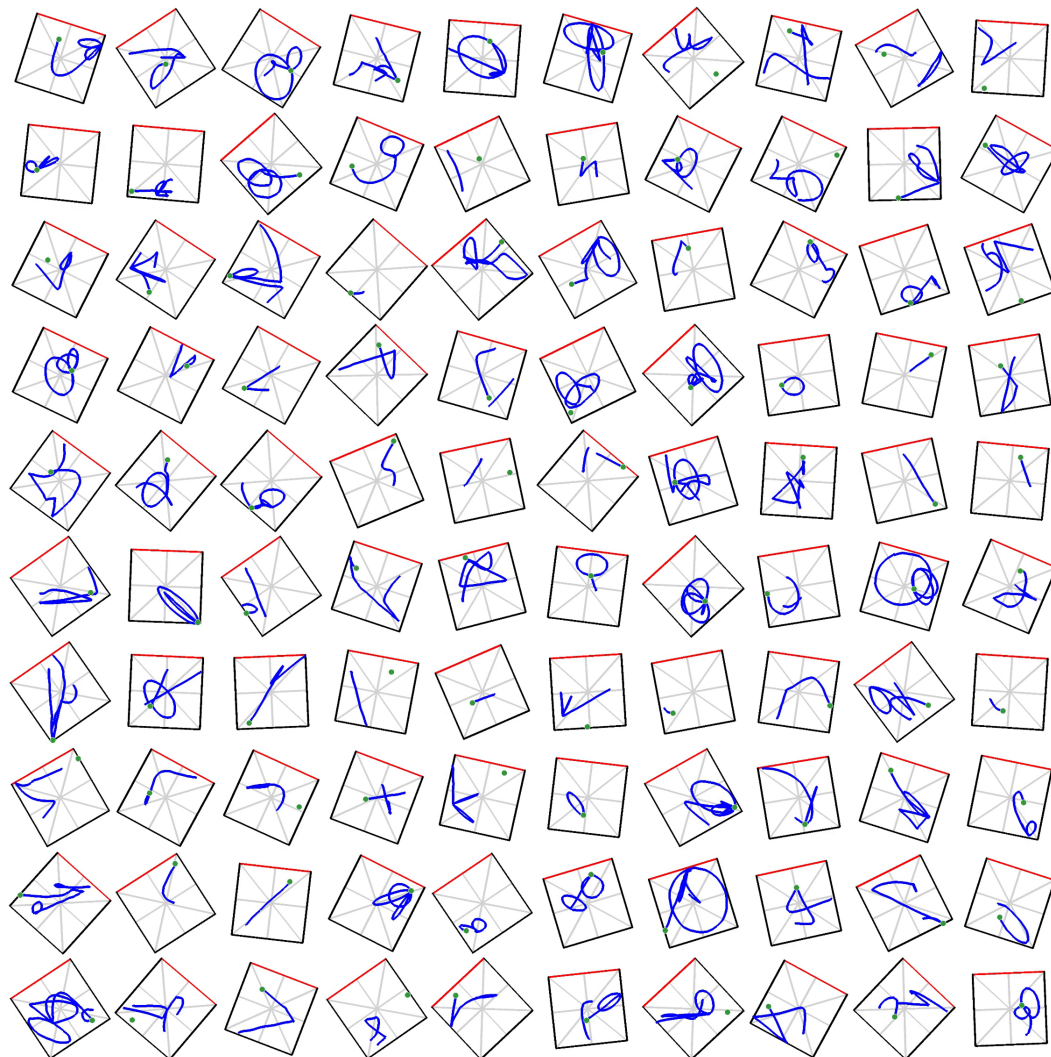


Figure 10: **DrawAnything-Sim training tasks.** We show a subset of 100 of the 2000 procedurally generated tasks (each having 5 demonstrations per task) using a combination of 1 to 6 parts. Parts include lines, Bézier curves, partial/full ovals, and free space (pen up) movement. The parameters for each part (such as start/end position, Bézier control points, proportion of oval, clockwise/counterclockwise direction, etc.) are randomly sampled. With a specified probability, parts will connect back to the start/end position of a previous part. Each demonstration includes varied whiteboard rotation (from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$), has varied speed, and includes noise inserted into the trajectories.

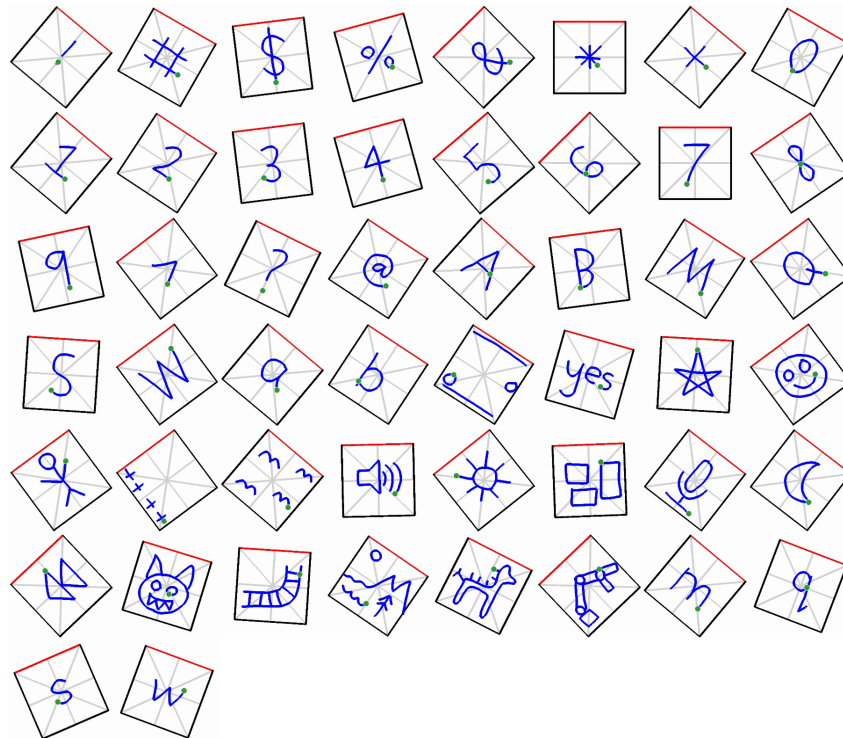


Figure 11: **DrawAnything-Sim evaluation tasks.** We hand-collect 50 evaluation tasks that were not seen during training with 5 demonstrations per task at varying board rotations. Tasks have varied complexity and duration.

D Results on original LIBERO dataset

We report results on the original LIBERO [2] dataset. The original LIBERO tasks are sufficiently described by language and the original benchmark splits do not evaluate test-time adaptation to unseen tasks. As expected, we find that using behavior prompting achieves similar performance to the language-conditioned diffusion policy. We empirically found that finetuning a CLIP language encoder gives substantially better performance compared to the frozen language embeddings used in some prior diffusion policy LIBERO evaluations [8]. Modern policies have also started to saturate performance of the original LIBERO benchmark [10, 7], emphasizing the importance of LIBERO-Gen which extends the dataset to evaluate test-time adaptation to unseen tasks.

Split	Language Diffusion Policy	Behavior Prompting Policy (Ours)	$\pi_{0.5}$
LIBERO Spatial	97.80 \pm 0.71	97.40 \pm 1.02	97.8
LIBERO Object	97.80 \pm 0.49	98.93 \pm 0.52	97.4
LIBERO Goal	95.60 \pm 1.07	98.33 \pm 0.38	97.8
LIBERO 10	93.73 \pm 2.07	95.27 \pm 0.66	95.6
Average	96.23 \pm 0.55	97.48 \pm 0.33	97.15

Table 2: **Original LIBERO dataset results.** Model success rate (%) on the original LIBERO [2] dataset by suite across 3 seeds (mean \pm stdev). Language is detailed in §F. We finetuned $\pi_{0.5}$ [5] for 30K LoRA [28] steps using openpi and report results for one seed. The $\pi_{0.5}$ training does not include LIBERO-90 in the finetuning mix, while the other models presented include LIBERO-90 split in the training data. Due to this difference, we put $\pi_{0.5}$ results in a separated column. For each of the three policy architectures listed we train one checkpoint per seed across all of the listed LIBERO splits rather than training and evaluating separate checkpoints per split.

E LIBERO-Gen implementation details

LIBERO-Gen procedurally generates new task definitions consisting of an environment definition and execution steps (§E.1). It also generates corresponding demonstrations for those tasks (§E.2). We then discuss details for LIBERO-Gen Combination (§E.3) and LIBERO-Gen Chain (§E.4), two benchmarks created using this tool.

E.1 Procedural task generation

The 130 original tasks in the LIBERO dataset are defined using BDDL, which defines both the environment and the intended robot task. The environment is defined by a base scene, objects present in that scene, and the initial locations of those objects. The task is defined through a set of predicates that specify the desired final states of the objects in the scene. LIBERO-Gen takes these BDDL files and enumerates the possible initial object locations and the desired final object states to generate new task definitions. The user specifies the set of procedural operations they would like to occur to generate a desired set of task variation, such as: 1) *generate environments where the goal is the same, but the plate is initialized at all of the possible starting locations* or 2) *generate environments where the initial object configuration stays the same, but the goal states are all the places the cream cheese could be placed*. LIBERO-Gen also supports the generation of chained tasks where the goal states consist of multiple action primitives applied sequentially (e.g., *turn on the stove and then put the bowl on the stove*).

In practice, we manually define a set of valid operators for each object (e.g., bowls can be grasped or you can put an object in them). We also start with a base set of environments from existing LIBERO tasks for which we apply the variations on top of. Doing so typically creates a large combinatorial space of tasks which either 1) may extend beyond the desired set of tasks for a targeted scientific experiment or 2) may want to be partitioned into a training and test set. We address this through the use of *views* which include filters that select desired portions of the generated tasks to put into training or testing splits.

E.2 Procedural demonstration generation

Given the procedurally generated task definitions, we need to procedurally generate demonstration data for these tasks. We do so through a scripted policy that leverages ground simulator object states and the existing demonstration data for the 130 original LIBERO tasks. From the existing teleoperation data, we extract object-relative grasp poses for each of the target objects. Many tasks also involve replaying a portion of teleoperation data and then switching to a scripted policy to complete the new part of the task. We introduce variations at multiple points to simulate differences present within teleoperated demonstrations: grasp location, placement position, intermediate poses on the way to or from each action primitive, and initial robot and object states. We validate the success of each demonstration using the predicate checking mechanism present in LIBERO. For chained tasks, the BDDL does not encode the ordering of each action primitive, so we encode the desired sequential order in a separate configuration file.

E.3 LIBERO-Gen Combination test tasks

Here we extend the discussion of LIBERO-Gen Combination from §4. We modify the LIBERO Spatial environments to have additional starting bowl locations as pictured in Fig. 12.

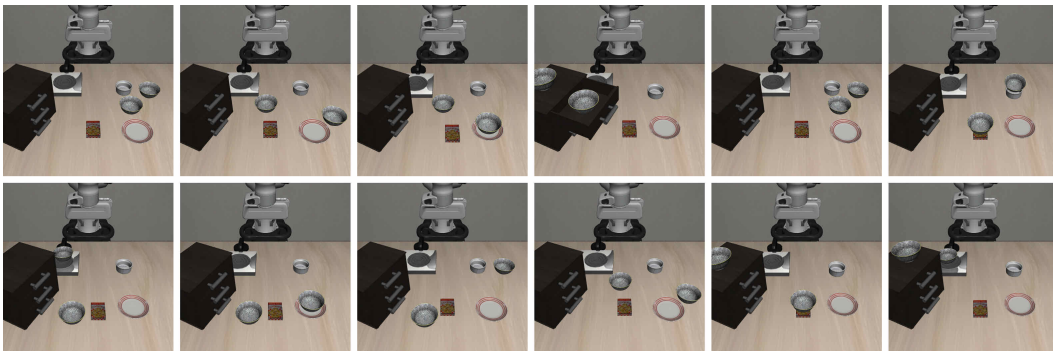


Figure 12: **Initial environments for LIBERO-Gen Combination.** There are two identical bowls in each environment. The task involves moving a specified bowl to a specified target location.

From each starting initialization, we generate the combinatorial space of all possible bowl pick locations and place locations to generate a task distribution. We withhold ten tasks (combinations of pick-place locations) to use as evaluation tasks. We document each of these unseen tasks below including the count of training tasks that have the same pick location and training tasks that have the same place location; this indicates that the pick and place locations have individually been seen during training, but never jointly. This experiment design evaluates the capability of the policy to adapt to unseen instructions at test time without evaluating unseen action primitives.

Unseen training tasks for LIBERO-Gen Combination and relevant training tasks:

```
Unseen: Pick the bowl from table center and place it on the cookies box
Training: Same pick (bowl table center), different place [9 tasks]
Training: Different pick, same place (bowl on the cookies box) [8 tasks]

Unseen: Pick the bowl in the top layer of the wooden cabinet and place it on the plate
Training: Same pick (bowl wooden cabinet top), different place [7 tasks]
Training: Different pick, same place (bowl on the plate) [9 tasks]

Unseen: Pick the bowl next to the cookies box and place it on the bowl next to the ramekin
Training: Same pick (bowl next to box), different place [9 tasks]
Training: Different pick, same place (bowl on the bowl next to the ramekin) [4 tasks]

Unseen: Pick the bowl next to the plate and place it on the stove
Training: Same pick (bowl next to plate), different place [9 tasks]
Training: Different pick, same place (bowl on the stove) [9 tasks]
```

Unseen: Pick the bowl on the cookies box and place it on the bowl on the wooden cabinet
 Training: Same pick (bowl cookies), different place [9 tasks]
 Training: Different pick, same place (bowl on the bowl on the wooden cabinet) [8 tasks]

Unseen: Pick the bowl on the cookies box and place it on the wooden cabinet
 Training: Same pick (bowl cookies), different place [9 tasks]
 Training: Different pick, same place (bowl on the wooden cabinet) [8 tasks]

Unseen: Pick the bowl on the plate and place it on the bowl on the table center
 Training: Same pick (bowl plate), different place [9 tasks]
 Training: Different pick, same place (bowl on the bowl on the table center) [4 tasks]

Unseen: Pick the bowl on the plate and place it on the table center
 Training: Same pick (bowl plate), different place [9 tasks]
 Training: Different pick, same place (bowl on the table center) [7 tasks]

Unseen: Pick the bowl on the stove and place it on the ramekin
 Training: Same pick (bowl stove), different place [9 tasks]
 Training: Different pick, same place (bowl on the ramekin) [9 tasks]

Unseen: Pick the bowl on the wooden cabinet and place it between the plate and the ramekin
 Training: Same pick (bowl wooden cabinet top side), different place [8 tasks]
 Training: Different pick, same place (bowl between the plate and the ramekin) [9 tasks]

E.4 LIBERO-Gen Chain test tasks

Here we extend the discussion of LIBERO-Gen Chain from §4. In LIBERO-Gen Chain we have one environment initialization and enumerate the set of two-step sequences where we chain together two action primitives sequentially (see Fig. 13).

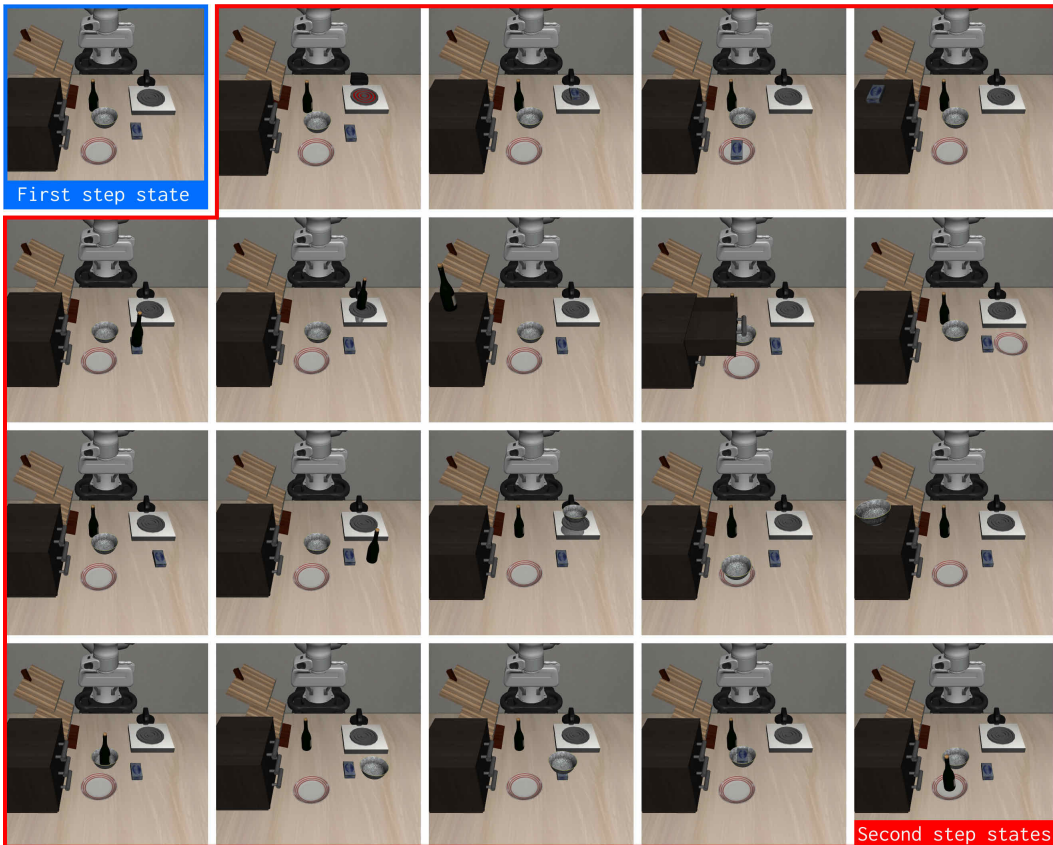


Figure 13: **Initial environment for LIBERO-Gen Chain.** All two-step chained tasks in this experiment start from the single first step state (blue). We also include one step actions starting from the first step state (blue) as well as second step actions which start assuming one action primitive has already been completed (red). The ablation for no second step excludes the single-step tasks that start from the second step initializations (red).

At test time, we evaluate adaptation to *unseen* sequential chains of individually *known* primitives. The purpose is not to adapt to an entirely new action primitive at test time that has not been seen during training, but rather to adapt to a new sequence of these primitives. This setup evaluates the capability of a model to follow previously unseen instructions. Here we enumerate all ten unseen evaluation tasks and, for each one, include the counts of the similar tasks that the model has seen during training. In short, this list demonstrates that the primitives for each unseen task have been seen individually during training, but never sequentially chained. Our ablation where we remove the second step tasks from the training mixture removes the guarantee that the second step task in the chain has been seen with the exact same environment initialization (see Fig. 13).

Unseen training tasks for LIBERO-Gen Chain and relevant training tasks:

Unseen: [1st] Open the middle layer of the drawer and then [2nd] put the cream cheese on the stove
 Training: similar chain (same 1st, different 2nd) [3 tasks]:
 - Open the middle layer of the drawer and put the cream cheese inside
 - Open the middle layer of the drawer and then put the cream cheese in front of the stove
 - Open the middle layer of the drawer and then put the cream cheese on the top of the drawer
 Training: similar single-step (2nd step match with wooden cabinet middle open): not present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Open the top layer of the drawer and then [2nd] put the wine bottle on the top of the drawer
 Training: similar chain (same 1st, different 2nd) [5 tasks]:
 - Open the top layer of the drawer and put the wine bottle inside
 - Open the top layer of the drawer and then put the wine bottle in front of the stove
 - Open the top layer of the drawer and then put the wine bottle on the bowl
 - Open the top layer of the drawer and then put the wine bottle on the cream cheese
 - Open the top layer of the drawer and then put the wine bottle on the stove
 Training: similar single-step (2nd step match with wooden cabinet top open): present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Push the plate to the front of the stove and then [2nd] put the bowl on the plate
 Training: similar chain (same 1st, different 2nd) [3 tasks]:
 - Push the plate to the front of the stove and then put the bowl on the cream cheese
 - Push the plate to the front of the stove and then put the bowl on the stove
 - Push the plate to the front of the stove and then put the bowl on the top of the drawer
 Training: similar single-step (2nd step match with plate on stove front): present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the bowl on the plate and then [2nd] put the cream cheese on the bowl
 Training: similar chain (same 1st, different 2nd) [4 tasks]:
 - Put the bowl on the plate and then put the cream cheese in front of the stove
 - Put the bowl on the plate and then put the cream cheese on the bowl region
 - Put the bowl on the plate and then put the cream cheese on the stove
 - Put the bowl on the plate and then put the cream cheese on the top of the drawer
 Training: similar single-step (2nd step match with bowl on plate): present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the bowl on the top of the drawer and then [2nd] put the cream cheese on the plate
 Training: similar chain (same 1st, different 2nd) [4 tasks]:
 - Put the bowl on the top of the drawer and then put the cream cheese in front of the stove
 - Put the bowl on the top of the drawer and then put the cream cheese on the bowl
 - Put the bowl on the top of the drawer and then put the cream cheese on the bowl region
 - Put the bowl on the top of the drawer and then put the cream cheese on the stove
 Training: similar single-step (2nd step match with bowl on wooden cabinet top side): present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the cream cheese on the plate and then [2nd] put the bowl on the cream cheese
 Training: similar chain (same 1st, different 2nd) [4 tasks]:
 - Put the cream cheese on the plate and then put the bowl in front of the stove
 - Put the cream cheese on the plate and then put the bowl on the cream cheese region
 - Put the cream cheese on the plate and then put the bowl on the stove
 - Put the cream cheese on the plate and then put the bowl on the top of the drawer
 Training: similar single-step (2nd step match with cream cheese on plate): present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the cream cheese on the top of the drawer and then [2nd] put the wine bottle on the bowl
 Training: similar chain (same 1st, different 2nd) [5 tasks]:
 - Put the cream cheese on the top of the drawer and then put the wine bottle in front of the stove
 - Put the cream cheese on the top of the drawer and then put the wine bottle on the cream cheese
 - Put the cream cheese on the top of the drawer and then put the wine bottle on the cream cheese region
 - Put the cream cheese on the top of the drawer and then put the wine bottle on the plate
 - Put the cream cheese on the top of the drawer and then put the wine bottle on the stove
 Training: similar single-step (2nd step match with cream cheese on wooden cabinet top side): present
 Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the wine bottle in front of the stove and then [2nd] put the bowl on the stove

Training: similar chain (same 1st, different 2nd) [3 tasks]:

- Put the wine bottle in front of the stove and then put the bowl on the plate
- Put the wine bottle in front of the stove and then put the bowl on the wine bottle region
- Put the wine bottle in front of the stove and then put the bowl on the top of the drawer

Training: similar single-step (2nd step match with wine bottle on stove front): present

Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the wine bottle on the rack and then [2nd] put the bowl on the top of the drawer

Training: similar chain (same 1st, different 2nd) [5 tasks]:

- Put the wine bottle on the rack and then put the bowl in front of the stove
- Put the wine bottle on the rack and then put the bowl on the cream cheese
- Put the wine bottle on the rack and then put the bowl on the plate
- Put the wine bottle on the rack and then put the bowl on the stove
- Put the wine bottle on the rack and then put the bowl on the wine bottle region

Training: similar single-step (2nd step match with wine bottle on wine rack top): not present

Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Turn on the stove and then [2nd] put the bowl on the stove

Training: similar chain (same 1st, different 2nd) [4 tasks]:

- Turn on the stove and then put the bowl in front of the stove
- Turn on the stove and then put the bowl on the cream cheese
- Turn on the stove and then put the bowl on the plate
- Turn on the stove and then put the bowl on the top of the drawer

Training: similar single-step (2nd step match with flat stove turned on): present

Training: similar single-step (2nd step match with diff background object config): present

F Baseline details

Our Language and Goal-Image baseline models follow the same diffusion CNN U-Net architecture from Chi et al. [25] which also matches the architecture of the BPP action decoder. The difference between BPP and the baselines is the observation encoding; in place of a prompt encoder, Language uses a finetuned CLIP [31] language encoder and Goal-Image shares a finetuned CLIP vision encoder with the corresponding image input.

LIBERO:

- Language uses a fine-tuned CLIP encoding which we found performed substantially better than frozen language embeddings. For specific LIBERO splits where it’s not possible to use the initial environment state to infer the task (such as LIBERO Goal), we observed that freezing the language encoding would often cause the policy to complete the wrong task.
- Goal-Image shares the same fine-tuned vision encoder used by the 3rd person camera. The goal image is the final frame of a demonstration from the 3rd person camera. Using goal images has ambiguity for some tasks within the original LIBERO dataset, so we only provide results for this baseline for LIBERO-Gen Combination and LIBERO-Gen Chain which do not contain tasks for which the goal image is ambiguous about the final state. For LIBERO-Gen Chain, goal images have ambiguity as they do not indicate the ordering in which the two chained operations should occur.
- $\pi_{0.5}$ is finetuned using LoRA [28]. For training $\pi_{0.5}$ we follow the LIBERO preprocessing steps from [8] to regenerate our LIBERO-Gen datasets at a higher resolution, but do not filter out no-op operations. We finetune for 100K steps for LIBERO-Gen experiments (Fig. 4) and finetune for 30K steps for the results on the original LIBERO dataset (Tab. 2).

DrawAnything-Sim:

- Goal-Image shares the fine-tuned vision encoder with the main image encoder.
- ICRT extends the provided implementation by Fu et al. [23] to support action chunking at the token input level; this means that 1) each input action token contains N sequential action steps rather than a single step and 2) we no longer encode observation tokens after every step, only after every N action steps. Additionally, at test time we only provide a single, full demonstration in the context at test-time. Both changes enable a more direct comparison to BPP, which chunks actions together in the prompt and uses a single behavior prompt during deployment.

DrawAnything-Real:

- Goal-Image shares the vision encoder with the iPhone main RGB camera in real. The drawing is not fully visible in all frames of the demonstrations due to the moving, wrist-mount camera and gripper occlusion. Thus, to ensure fair comparison, we select the goal image as the first frame searching backwards from the end of demonstration that fully shows the drawing region.

Laundry Folding:

- Language uses a finetuned CLIP language encoding.

G BPP architectural comparison to ICRT

Model architecture. ICRT [23] is a prior behavior prompting architecture that we baseline against. ICRT is a causal transformer decoder, while BPP has separate modules for prompt understanding and action generation (detailed in §3.3). BPP leverages cross-attention to reason over the prompt contents while ICRT leverages causal self-attention. Furthermore, BPP leverages action diffusion [25], while ICRT uses L1 action loss.

Training. Each step of training, ICRT samples a sequence of prompts and entire rollouts concatenated along the time dimension into a single sequence to fill the transformer context length. The loss is computed by predicting action loss for each step of the rollout sequence using teacher forcing. In contrast, for each training step of BPP, we sample a single behavior prompt along with a batch of observation-action pairs randomly sampled across many demonstrations. Action loss is computed for each of these randomly sampled observation-action pairs. Both models use prompts and rollouts from a single task per batch. In practice, due to DDP training across multiple GPUs or the use of gradient accumulation, each gradient update will take into account losses across multiple tasks.

Inference. BPP has a fixed length history, common in other visuomotor policies [25]. In contrast, ICRT retains the full rollout history in the model context which can be susceptible to OOD issues during deployment due to spurious correlations during training [32]. This also means that ICRT can only be rolled out for a fixed duration before reaching the transformer context limit.

The separate modules for prompt understanding and action generation in BPP enable us to preprocess the prompt once per rollout, extract relevant prompt information using the prompt encoder once per inference call, and perform many steps of action denoising without needing to reference the entire prompt each time. On the other hand, each forward pass of ICRT references the entire prompt and the entire observation-action history since the start of the rollout, but leverages KV caching to reduce the causal attention computation.

H BPP Training details

Benchmark	Duration (hrs)	GPUs
LIBERO-Gen Combination	28	4× NVIDIA RTX A6000
LIBERO-Gen Chain	34	4× NVIDIA L40S
DrawAnything-Sim	27	4× NVIDIA L40S
DrawAnything-Real	42	4× NVIDIA L40S
Laundry Folding	24	4× NVIDIA L40S

Table 3: **Compute requirements for training BPP for each training benchmark.** Simulation experiments include time for intermediary and final rollout evaluations. We train using DDP.

Layers	Hidden size D	MLP size	Heads	Params
6	768	3072	8	57M

Table 4: **BPP prompt encoder architecture (transformer decoder).**

Structure	Step emb dim	Down dims	Kernel	Params
U-Net	128	[256, 512, 1024]	5	151M

Table 5: **Action decoder model architecture details.** We use the CNN U-Net action diffusion architecture from Chi et al. [25]. BPP, Language, and Goal-Image models use this for action diffusion.

I iPhUMI

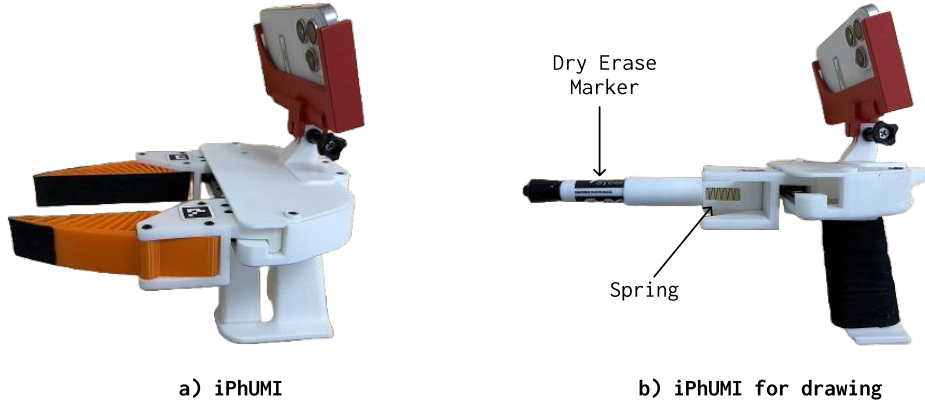


Figure 14: **The iPhUMI handheld data collection gripper.** iPhUMI enables real-time localization in new environments, which dramatically reduces the setup time required for collecting demonstration data compared to the original UMI [1]. It features a custom-built application that facilitates data collection and policy deployment. With iPhUMI, a user can also specify behavior prompts at test-time to immediately condition behavior prompting policies. We present two different instantiations of iPhUMI: a) with fingers for standard manipulation tasks and b) with a marker attachment and spring for compliant drawing.

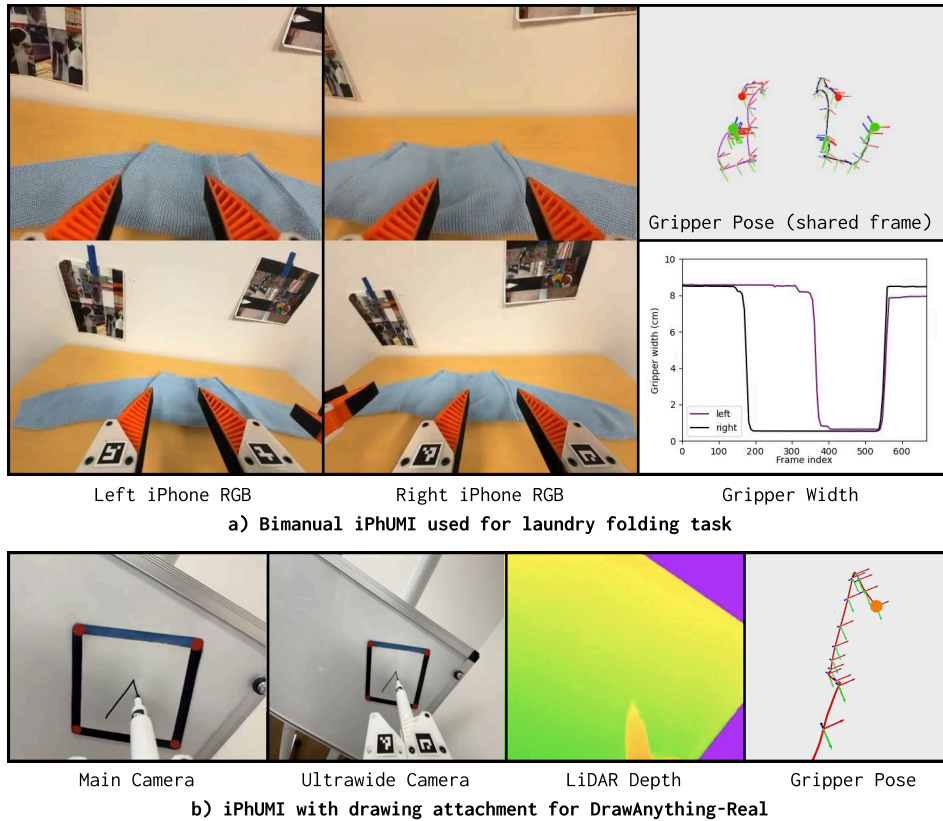


Figure 15: **iPhUMI collected modalities.** Data collected with bimanual iPhUMI for a) laundry folding and b) drawing visualized part-way through a user drawing the letter A. For DrawAnything-Real we do not use the ultrawide or depth camera as policy inputs, but include them for reference.

iPhUMI is a handheld data collection interface designed to enable rapid data collection across new tasks and environments. We modify the original UMI gripper design [1] by replacing the GoPro with a custom 3D-printed mount to hold an iPhone (Fig. 14a). For drawing tasks, the UMI gripper is further modified to attach a marker with a spring to allow compliance for drawing without the need for a force-torque sensor (Fig. 14b). We open source a custom iOS application for iPhUMI that provides an intuitive application interface for recording demonstration data and for using the iPhone as a camera on the robot during policy deployment.

I.1 iPhUMI data collection

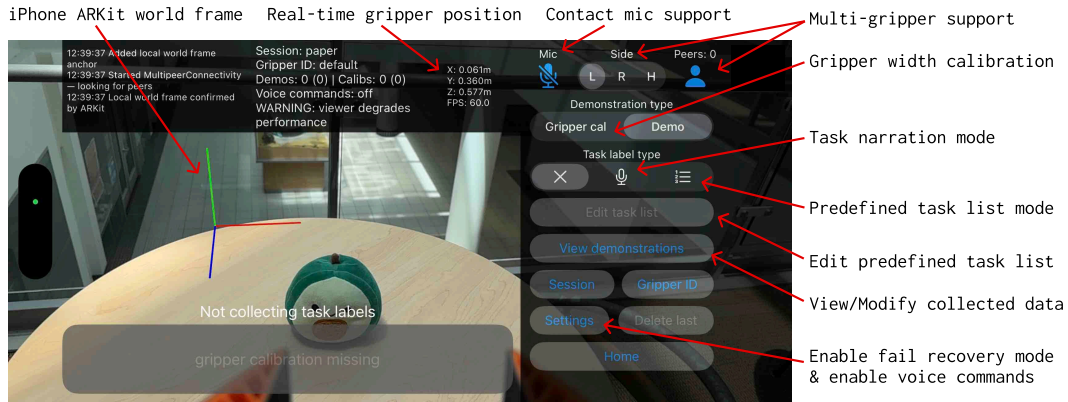


Figure 16: **iPhUMI data collection interface.** We provide an interface to perform gripper calibration, collect demonstrations, and set appropriate settings for data collection.

iPhUMI is capable of collecting five types of data from the iPhone simultaneously: main camera (1920x1440 at 60Hz), ultrawide camera (640x480 at 10 Hz), LiDAR depth (256x192 at 60Hz), gripper pose (60Hz), and gripper width (10Hz) (Fig. 15). Note that depth data is not used in any models or experiments presented in this paper. Unlike previous iPhone-based grippers that either rely on an external fisheye camera [33] or do not capture the ultrawide camera [34], our iPhUMI application is capable of capturing the ultrawide camera input. This provides increased visual context to the policy, which has been found to improve performance [1].

The primary data collection interface is shown in Fig. 16. Upon launching the app, ARKit initializes a world coordinate frame and then continually estimates the iPhone pose using SLAM. The resulting 6-DoF pose of the iPhone in this world frame is logged in real time throughout each demonstration. The app also supports simultaneous recording from multiple grippers. Up to three devices (bimanual iPhUMI and head-mounted iPhone) can be connected into a shared ARKit session to share a single world coordinate frame among them. The triple iPhone setup was co-developed with Xu et al. [35]. The iPhones will connect into a shared session once common world features have been detected between them.

Before collecting demonstrations, the user performs a one-time gripper width calibration by recording a ~ 10 s clip while opening and closing the gripper multiple times in calibration mode. The iPhUMI uses the ultrawide camera to detect ArUco tags on the fingers to determine the minimum and maximum gripper width for the specific hardware.

When collecting data, the user can specify the task name(s) associated with the demonstration and switch labels online as the task changes. Alternatively, the user may narrate the task while demonstrating the task, and iPhUMI uses speech-to-text to label segments with the active task. We also have support for connecting the contact microphone used by [36] through the USB-C connector on the iPhone to capture synchronized audio data.

Recorded demonstrations can be reviewed using the interface shown in Fig 17. Selecting a demonstration displays the main, ultrawide, and depth videos, while a left swipe allows the user to delete a demonstration. Tapping the Export button saves the demonstration data to an external location such as an SD card connected via a USB C adapter.

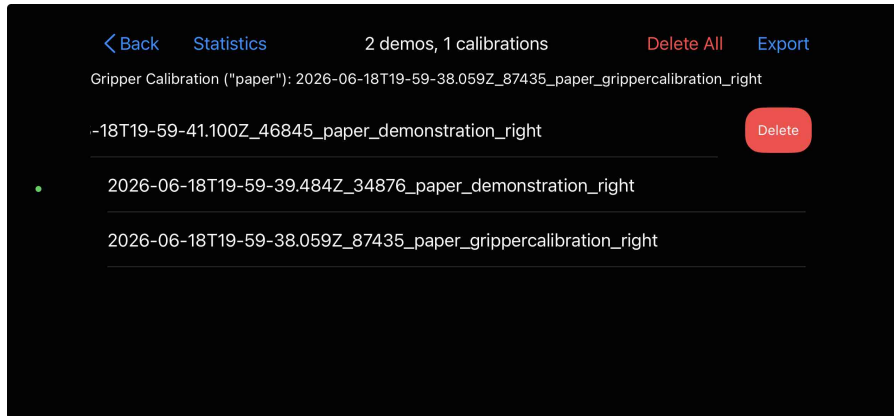


Figure 17: **iPhUMI Demonstration management interface.** This interface lets the user view and manage collected demonstration data. The data can also be exported to an external SD card connected with a USB C adapter.

I.2 iPhUMI policy deployment

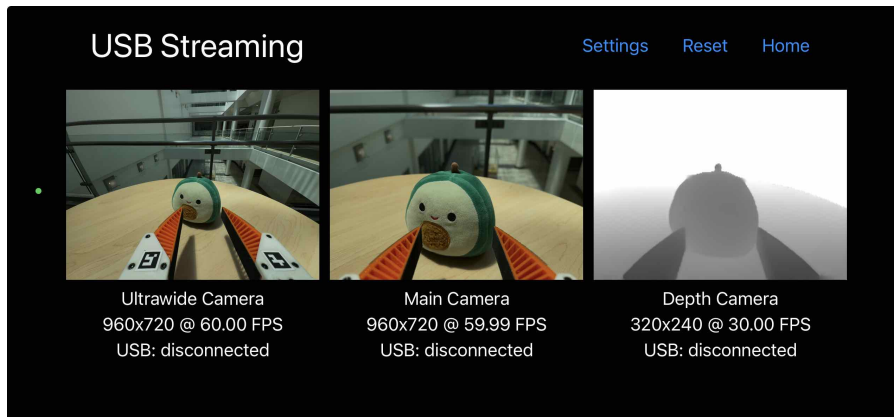


Figure 18: **iPhUMI deployment interface.** Both USB and Ethernet streaming are supported to stream main camera, ultrawide camera, and LiDAR depth for use during robot deployment.

The deployment interface (Fig. 18) can be used to deploy a policy on the robot using an iPhone as the camera. The iPhone streams its camera feeds and depth through a USB or Ethernet connection. The main camera and ultrawide camera feeds are each streamed 960x720 at 60Hz. Depth is streamed at 320x240 at 30Hz. Our robot deployment infrastructure is built off the code from [37].

I.3 iPhUMI for test-time behavior prompting

The iPhUMI app supports wireless transfer of behavior prompts from the iPhone to a deployment desktop connected to a robot. This enables rapid behavior prompt collection and conditioning of robot execution with BPP. When in behavior prompting mode, a user collects a single demonstration using iPhUMI that is then wirelessly transmitted to a desktop connected to a robot. The desktop processes the raw iPhone data into a behavior prompt that is used to condition BPP. This enables a rapid process for practically leveraging behavior prompting at test-time to specify new tasks.