HAND Me the Data: Fast Robot Adaptation via Hand Path Retrieval

Matthew Hong*, **Anthony Liang***, **Kevin Kim, Harshitha Rajaprakash,**
Jesse Thomason†, **Erdem Bıyık**†, **Jesse Zhang**†
Thomas Lord Department of Computer Science,
University of Southern California

Abstract: We hand the community HAND, a *simple* and *time-efficient* method for teaching robots new manipulation tasks through human hand demonstrations. Instead of relying on task-specific robot demonstrations collected via teleoperation, HAND uses easy-to-provide hand demonstrations to retrieve relevant behaviors from task-agnostic robot play data. Using a visual tracking pipeline, HAND extracts the motion of the human hand from the hand demonstration and retrieves robot sub-trajectories in two stages: first filtering by visual similarity, then retrieving trajectories with similar behaviors to the hand. Fine-tuning a policy on the retrieved data enables *real-time learning of tasks* in under four minutes, without requiring calibrated cameras or detailed hand pose estimation. Experiments also show that HAND outperforms retrieval baselines by over $2\times$ in average task success rates on real robots. Videos can be found at our project website: https://liralab.usc.edu/handretrieval/.

Keywords: Robot Adaptation, Retrieval, Play Data, Hand Demonstration

1 Introduction

Robots deployed in homes, warehouses, and other dynamic, human-centric settings will need to quickly learn many tasks specified by end-users. To support this goal, robot learning algorithms for these settings must (1) scale easily across many tasks and (2) enable fast adaptation for each new task. While imitation learning has shown promise in producing capable multi-task robot policies [1, 2, 3, 4, 5], it remains difficult to scale due to its reliance on large amounts of expert-collected, task-specific teleoperation data. In contrast, *task-agnostic play data*—collected through free-form robot teleoperation [6, 7, 8]—is much easier to gather, as it does not require constant environment resets or task-specific labeling. The key challenge is making this data usable for teaching robots new tasks quickly. In this work, we tackle this problem by retrieving relevant robot behaviors from the play dataset using only a single *human hand* demonstration.

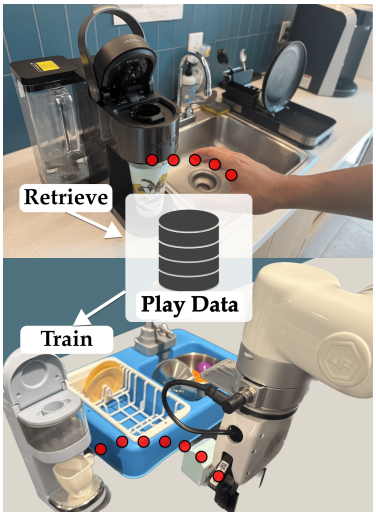

Figure 1: HAND learns a policy from as little as one (1) human hand demonstration.

We propose HAND, a *simple and time-efficient* approach that adapts pre-trained play policies to specific tasks using just one human hand demonstration (see Figure 1). Prior work on robot behavior retrieval relies on additional teleoperated target-task robot demonstrations [9, 10, 11, 12, 13], which are difficult for non-expert users to provide. Instead, our key insight is to extract *coarse guidance* from the hand demonstration—specifically, 2D relative hand motion paths—to retrieve diverse yet relevant behaviors from the play dataset.

*Equal Contribution, † Equal Advising

Calling back to the motivation, we aim for HAND to be *scalable* and *fast*. Towards scalability, HAND avoids the need for calibrated depth cameras [14, 15], specialized eye-in-hand setups [16], or detailed hand-pose estimation [16, 17]. Instead, it first labels a robot play dataset with 2D gripper positions relative to the RGB camera frame by tracking the gripper using a visual point-tracking model [18, 19]. When a human hand demonstration is provided, HAND tracks the hand trajectory with the same simple pipeline. The hand positions are then converted into 2D *relative* sub-trajectories, capturing motion independent of the starting point [20]. After an initial filtering step that removes unrelated behaviors using a visual foundation model [21], HAND retrieves matching sub-trajectories from the play dataset based on the 2D relative hand path. Finally, for fast learning, a policy pre-trained on the play dataset is LoRA-fine-tuned on the retrieved sub-trajectories, encouraging the policy to specialize in the demonstrated task. Because HAND retrieves primarily based on hand motion, it is more robust to irrelevant visual features such as background clutter and lighting changes compared to purely visual retrieval methods.

Our experiments, both in simulation in CALVIN [8] and in the real world on a WidowX robot, demonstrate that HAND enables quick adaptation to **8** diverse downstream tasks. Notably, HAND outperforms the best baseline by **2**$\times$ on the real robot. We also demonstrate that HAND works with hand demos collected from completely different scenes from the robot's. Finally, we perform a *real-time learning* experiment, where HAND learns a challenging long-horizon task in **under 4 minutes** of experiment time, from providing the hand demonstration to the trained policy, while being on average **5**$\times$ faster to collect data for than robot teleoperation demos on our WidowX arm.

2 Related Works

Robot Data Retrieval. Prior work has demonstrated *retrieval* as an effective mechanism for extracting relevant on-robot data for training robots [9, 10, 11, 12, 13, 22]. For example, SAILOR [9] and Behavior Retrieval [10] pre-train variational auto-encoders (VAEs) on prior robot images and actions to learn a latent embedding. This latent embedding is used to retrieve states and actions from an offline dataset similar to ones provided in expert demonstration trajectories. However, retrieving based on learned full image encodings or even raw pixel values [13] can be noisy; Flow-Retrieval [11] instead trains a VAE to encode *optical flows* indicating movement of objects and the robot arm in the scene. Similar to Flow-Retrieval, our method HAND also retrieves based on robot arm movement. However, rather than training a dataset-specific VAE model that may not be robust to large visual differences, we retrieve from our offline robot data by primarily matching motions of a human hand demonstration using *relative 2D paths* of the robot end-effector in the prior data. This hand path retrieval helps us robustly retrieve relevant robot arm *behaviors*.

STRAP [12] addresses visual retrieval robustness issues of prior work by using features from DINO-v2 [21], a large pre-trained image-input foundation model for retrieval. However, STRAP, along with all aforementioned retrieval work, assumes access to expert robot demonstrations for the target tasks. HAND on the other hand, only requires a *single*, easier-to-collect human hand demonstration that results in more *time-efficient* learning of demonstrated tasks compared to methods requiring robot teleoperation data for retrieval. Moreover, experiments demonstrate HAND actually retrieves more task-relevant trajectories and therefore attains higher success rates compared to these methods.

Learning From Human Hands. Similar to HAND, a separate line of work has proposed methods to use human hands to learn robot policies. One approach is to train models on human video datasets to predict future object flows [23, 24] or human affordances [25, 26]. These intermediate affordance and flow representations are then used to either train a policy conditioned on this representation [23] on robot data or control a heuristic policy [24, 25, 26]. Other works focus on learning directly from human hands [14, 15, 16, 27, 17]. These works generally use hand-pose detection models aided by multiple cameras or calibrated depth cameras to convert hand poses directly to robot gripper keypoints [14, 15, 17]. However, works that exclusively retrieve human data are restricted to constrained policy representations as they must match human hand poses to robot gripper poses. Kim et al. [16] instead use an eye-in-hand camera mounted on a human demonstrator's forearm to train an imitation learning policy conditioned on robot eye-in-hand camera observations. Unlike these prior

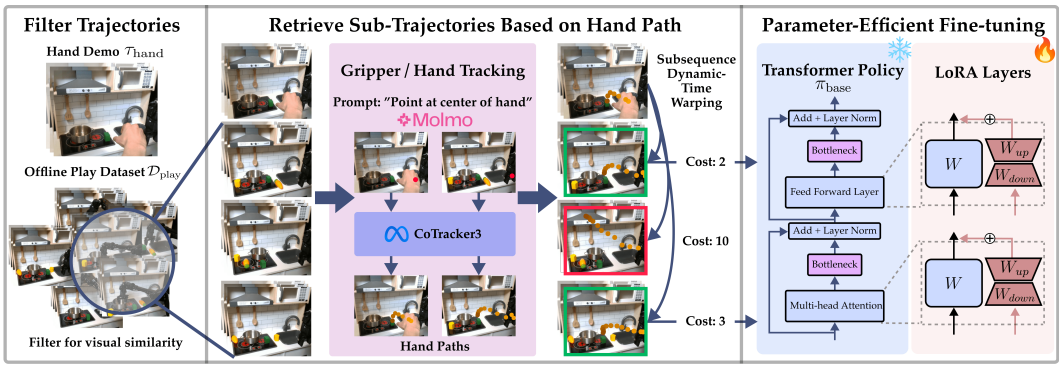

Figure 2: **HAND** enables fast-adaptation to a new target task by using an easy-to-provide hand demonstration of the target task (Left). We propose a two-step retrieval procedure where we first filter the trajectories in the offline play dataset, $\mathcal{D}_{\text{play}}$, for visually similar trajectories based on features from a pretrained vision model. We use off-the-shelf, pretrained hand detection and point tracking to construct 2D paths of the motion for both the human hand and robot end-effector. We use these paths as a distance metric to retrieve relevant trajectories from the play dataset (Middle) for quickly fine-tuning a pretrained transformer policy on the target task (Right).

works, HAND only requires a single RGB camera from which the robot gripper can be seen. Also, we focus on retrieving robot play data, allowing us to train arbitrarily expressive policies without constrained policy representations [14, 15, 17] or intermediate representations [23, 24, 25, 26].

3 HAND: Fast Robot Adaptation via Hand Path Retrieval

We assume access to a dataset of task-agnostic robot play data, $\mathcal{D}_{\text{play}}$, consisting of trajectories $\tau_i = \{(o_t, a_t)\}_{i=1}^{T}$, where each o_t is per-timestep observation that includes RGB images of the robot gripper and robot proprioceptive information, and a_t is the robot action. These trajectories may span many scenes or tasks and can vary in length, potentially covering long-horizon behavior. We do not assume task labels (e.g., language labels), as data collection is easier to scale without labeling each sub-trajectory in a long-horizon play trajectory.[1] We assume the RGB camera's angle relative to the robot base is fixed across trajectories, which is the case for tabletop robot manipulation setups.

In contrast to prior retrieval methods that rely on robot demonstrations for each target task [9, 10, 11, 12], we assume access to easy-to-provide human hand demonstrations.[2] For each task, a human records their hand movement without teleoperating the robot. On our real-world setup, these hand demonstrations, $\mathcal{D}_{\text{hand}}$, are on average $5\times$ faster to collect than robot teleoperation data. Moreover, producing high-quality hand demonstrations typically requires far less effort than robot teleoperation [28, 29]. Each video in $\mathcal{D}_{\text{hand}}$ consists of a sequence of RGB images $o_1, \ldots, o_H$, captured from a similar viewpoint relative to the human hand as the robot play data relative to the robot gripper.

Given $\mathcal{D}_{\text{play}}$ and $\mathcal{D}_{\text{hand}}$, we aim to train a policy $\pi_\theta(a \mid o)$ to perform the target task demonstrated by the human in $\mathcal{D}_{\text{hand}}$. Since we do not assume task labels in $\mathcal{D}_{\text{play}}$ and we are provided no expert robot teleoperation demonstrations, we must *retrieve* sub-trajectories indicating how to perform the behavior demonstrated in $\mathcal{D}_{\text{hand}}$ from $\mathcal{D}_{\text{play}}$ for training π. We denote this retrieved dataset, which we later use for imitation learning, as $\mathcal{D}_{\text{retrieved}}$. Moreover, following our motivation in Section 1, we aim for our method to be *fast*, so that non-expert end-users can easily train the robot for many downstream tasks. Thus, the key challenges we resolve in our method HAND are: (1) designing a representation that can unify the behaviors in robot sub-trajectories and human hand demonstrations (Section 3.1), (2) retrieving relevant sub-trajectories based on a suitable distance metric between these representations (Section 3.2), and (3) time-efficiently training a policy that can perform various unseen target tasks with a high success rate without expert demonstrations (Section 3.3). See Figure 2 for an overview and Algorithm 1 for full algorithm pseudocode.

[1] HAND can also easily incorporate task labels as an extra policy conditioning input.

[2] HAND also outperforms baselines even when they have access to robot demonstrations.

3.1 Path Distance as a Unifying Representation for Retrieval

Existing robot data retrieval methods assume access to expert demonstrations from which they extract proprioceptive information (e.g., joint states and actions) alongside visual features for retrieval [9, 10, 11, 12, 13]. However, since $\mathcal{D}_{\text{hand}}$ contains only visual data and no robot actions, retrieval based purely on appearance can be noisy—especially due to the visual domain gap between hand demonstrations in $\mathcal{D}_{\text{hand}}$ and robot demonstrations in $\mathcal{D}_{\text{play}}$ (c.f., Figure 2, left). To address these issues, we propose an embodiment-agnostic, behavior-centric retrieval metric that enables matching between $\mathcal{D}_{\text{hand}}$ and $\mathcal{D}_{\text{play}}$ based on demonstrated behaviors rather than appearance.

Using 2D Paths for Retrieval. The movement of the robot end-effector over time provides rich information about its behavior [4]. We represent behaviors in both datasets using the paths traced by the human hand or the gripper. Because we assume access only to an RGB camera from which the hand or the gripper is visible (i.e., no depth), we construct these paths in 2D relative to the camera viewpoint for both $\mathcal{D}_{\text{play}}$ and $\mathcal{D}_{\text{hand}}$.[3]

Obtaining Paths from Data. To extract paths, we use CoTracker3 [19], an off-the-shelf point tracker capable of tracking 2D points across video sequences, even under occlusion. CoTracker3 only requires a single point on the gripper or hand to generate a complete trajectory. We use Molmo-7B [30], an open-source 7B image-to-point foundation model, to automatically select this point by prompting it at the *midpoint* of each trajectory with either "Point at the center of the hand" or "Point to the robot gripper." Using the middle frame ensures a higher chance of visibility in case the gripper or hand is not yet in frame at the beginning or occluded at the end.[4]

Given the 2D point $(x, y)_{\text{hand}}$ or $(x, y)_{\text{play}}$ from the middle frame, we use CoTracker3 to perform bidirectional point tracking, resulting in a 2D path $p_{\text{hand}} = \{(x_t, y_t)_{\text{hand}}\}_{t=1}^{H}$ or $p_{\text{play}} = \{(x_t, y_t)_{\text{play}}\}_{t=1}^{T}$ for each trajectory. See the Gripper/Hand Tracking block of Figure 2 for a visualization of this pipeline. Next, we describe how we use 2D paths to retrieve sub-trajectories from $\mathcal{D}_{\text{play}}$.

3.2 Retrieving Relevant Sub-Trajectories using Path Distance

Background. For identifying relevant sub-trajectories in $\mathcal{D}_{\text{play}}$, we follow Memmel et al. [12] and use Subsequence Dynamic Time Warping (S-DTW) [31], an algorithm for aligning a shorter sequence to a portion of a longer reference sequence. Given a query sequence $Q = \{q_1, q_2, \ldots, q_H\}$ and a longer reference sequence $R = \{r_1, r_2, \ldots, r_T\}$, where $T > H$, the goal of S-DTW is to find a contiguous subsequence of R that minimizes the total cumulative distance between elements of both sequences. In HAND, the query sequences are the 2D hand demo paths $\{(x_t, y_t)_{\text{hand}}\}_{t=1}^{H}$ and the reference sequences are the 2D paths generated from long-horizon robot play data $\{(x_t, y_t)_{\text{play}}\}_{t=1}^{T}$.

Sub-Trajectory Preprocessing. To preprocess the datasets for S-DTW, we first segment the offline play dataset, $\mathcal{D}_{\text{play}}$, into variable-length sub-trajectories using a simple heuristic based on proprioception proposed in several prior works [32, 12]. In particular, we split the trajectories whenever the acceleration or velocity magnitude (depending on what proprioception data is available) drops below a predefined ϵ value, corresponding to when the teleoperator switches between tasks. We find that this simple heuristic can reasonably segment trajectories into atomic components resembling lower-level primitives. We also split the hand demonstrations evenly into smaller sub-trajectories based on how many subtasks the human operator determined they have completed. After sub-trajectory splitting, we have two sub-trajectory datasets, $\mathcal{T}_{\text{hand}} = \{t^i_{1:a}, t^i_{a:b}, \ldots, t^i_{H_i - |p^i_{\text{hand}}|:H_i} \forall\, \tau^i_{\text{hand}} \in \mathcal{D}_{\text{hand}}\}$ and $\mathcal{T}_{\text{play}} = \{t^j_{1:a}, t^j_{a:b}, \ldots, t^j_{T_j - |p^j_{\text{play}}|:T} \forall\, \tau^j_{\text{play}} \in \mathcal{D}_{\text{play}}\}$ where $|p^i_{\text{hand}}|$ and $|p^j_{\text{play}}|$ are the lengths of the last sub-trajectory paths of trajectories i, j from $\mathcal{D}_{\text{hand}}$ and $\mathcal{D}_{\text{play}}$, respectively. Inspired by prior work that proposes to cluster trajectories based on relative embedding differences [20], each sub-trajectory is represented in *relative 2D coordinates*, i.e., $p_t = [x_{t+1} - x_t, y_{t+1} - y_t]$. Relative coordinates ensure invariance based on the starting positions of the hand or gripper so that these starting positions do not influence how trajectories are retrieved.

[3] If both datasets have additional calibrated depth information, HAND can also operate on 3D paths.

[4] Points can also be obtained heuristically, e.g., if the robot starts from the same position in each $\mathcal{D}_{\text{play}}$ traj.

Visual Filtering. One issue with retrieving sub-trajectories based only on path distance is that different tasks can have similar movement patterns. For example, tasks like "pick up the mug" and "pick up the cube" can appear nearly identical in 2D path space. But, the retrieved trajectories for one task may not benefit learning of the other; since we don't assume task labels in $\mathcal{D}_{\text{play}}$, a policy directly trained on "pick up the cube" retrieved sub-trajectories may still fail to pick up a mug. Therefore, before retrieving sub-trajectories with paths, we first run a visual filtering step to ensure that the sub-trajectories we retrieve will be task-relevant. We use an object-centric visual foundation model, namely DINOv2 [21], to first filter out sub-trajectories performing unrelated tasks with different objects. Specifically, we use the DINOv2 first and final frame embedding differences, representing visual object movement from the first to last frame, between human hand demos and robot play data to filter $\mathcal{T}_{\text{play}}$. We find that using this simple method is sufficient to filter out most irrelevant sub-trajectories. For a given image sequence $o^{\text{hand}}_{1:H}$ from a hand sub-trajectory and image sequence $o^{\text{play}}_{1:T}$ from a robot play sub-trajectory, we define the cost as:

$$\text{C}_{\text{visual}}(o^{\text{hand}}_{1:H}, o^{\text{play}}_{1:T}) = \underbrace{||\text{DINO}(o^{\text{hand}}_1) - \text{DINO}(o^{\text{play}}_1)||^2_2}_{\text{first frame DINO embedding difference}} + \underbrace{||\text{DINO}(o^{\text{hand}}_H) - \text{DINO}(o^{\text{play}}_T)||^2_2}_{\text{last frame DINO embedding difference}}. \quad (1)$$

We sort these costs and take the M trajectories with lowest cost as possible retrieval trajectories for each human hand demo sub-trajectory in $\mathcal{T}_{\text{hand}}$. The rest are discarded for those hand demos.

Retrieving Sub-Trajectories. Finally, we then employ S-DTW to match the target sub-trajectories, $\mathcal{T}_{\text{hand}}$, to the set of visually filtered segments $\in \mathcal{T}_{\text{play}}$. Given two sub-trajectories, $t_i \in \mathcal{T}_{\text{play}}$ and $t_j \in \mathcal{T}_{\text{hand}}$, S-DTW returns the cost along with the start and end indices of the subsequence in t_j that minimizes the path cost (see Figure 2). We select the K matches from $\mathcal{D}_{\text{play}}$ with the lowest cost to construct our retrieval dataset, $\mathcal{D}_{\text{retrieved}}$.

3.3 Putting it All Together: Fast-Adaptation with Parameter-Efficient Policy Fine-tuning

We aim to enable fast, data-efficient learning of the task demonstrated in $\mathcal{D}_{\text{hand}}$. To this end, we first pretrain a task-agnostic base policy π_{base} on $\mathcal{D}_{\text{play}}$ with standard behavior cloning (BC) loss. While our approach is compatible with any policy architecture, we use action-chunked transformer policies [33] due to their suitability for low-parameter fine-tuning and strong performance in long-horizon imitation learning [34, 35, 36, 3].

Adapting to $\mathcal{D}_{\text{retrieved}}$. To rapidly adapt to a new task with minimal data, we leverage parameter-efficient fine-tuning using *task-specific adapters*—small trainable modules that modulate the behavior of the frozen base policy. Adapter-based methods have shown promise in few-shot imitation learning [37, 38], making them ideal for our limited retrieved dataset $\mathcal{D}_{\text{retrieved}}$. Following the findings of Liu et al. [38], we specifically insert LoRA layers [39] into the transformer blocks of π_{base}. These are low-rank trainable matrices (typically 0.1%–2% of the base policy's parameters) inserted between the attention and feedforward layers (see Figure 2, LoRA Layers). During fine-tuning, we keep π_{base} frozen and update only the parameters of these LoRA layers, θ, using $\mathcal{D}_{\text{retrieved}}$.

Loss Re-Weighting. While our retrieval mechanism identifies sub-trajectories relevant to the target task, not all will be equally useful. To prioritize the most behaviorally aligned examples, we reweight the BC loss with an exponential term $\in (0, \infty)$ following Advantage-Weighted Regression [40], where each sub-trajectory is weighted based on its similarity (from S-DTW) to the hand demonstration. Intuitively, this upweights the loss of the most relevant examples in $\mathcal{D}_{\text{retrieved}}$ and conversely downweights those that are less relevant. Finally, because trajectory cost scales vary depending on the task being retrieved and the features being used for S-DTW, we rescale the S-DTW costs $C_{i,\text{path}}$ to a fixed range. For each $\tau_i \in \mathcal{D}_{\text{retrieved}}$, its weight $e^{-C_{i,\text{path}}}$ is scaled to between $[0.01, 100]$, where the normalization term comes from the sum of costs of all trajectories in $\mathcal{D}_{\text{retrieved}}$. Our final training loss is:

$$\mathcal{L}_{\text{BC};\theta} = \frac{1}{|\mathcal{D}_{\text{retrieved}}|} \sum_{\tau_i \in \mathcal{D}_{\text{retrieved}}} \underbrace{\exp(-C_{i,\text{path}})}_{\text{Normalized Weight}} \times \underbrace{(-\log \pi_\theta(a \mid o))}_{\text{BC Loss}}. \quad (2)$$

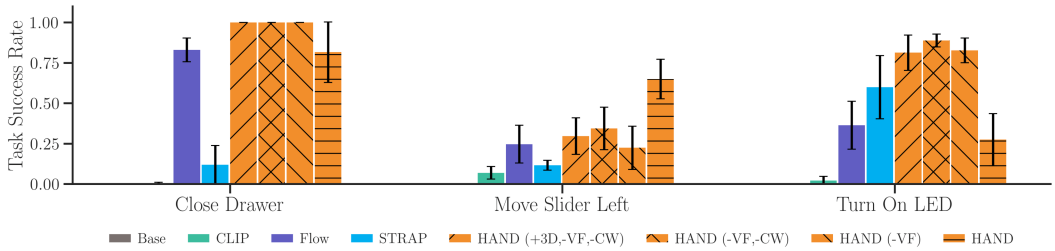

Figure 3: **CALVIN Results.** Task success rate of HAND and baseline methods on the CALVIN ABC-D task across three random seeds. Ablations of HAND are denoted by hatches. HAND and ablations outperform the next best baseline Flow on task success rate across all tasks.

4 Experiments

Our aim in the experiments is to study the efficacy of HAND as a robot data retrieval pipeline and evaluate its ability to quickly learn to solve new downstream tasks. To this end, we organize our experiments to answer the following questions, in order:

- **(Q1)** How effective is HAND, using 2D relative paths, in retrieving *task-relevant* behaviors?
- **(Q2)** Does HAND work with hand demonstrations from *unseen scenes*?
- **(Q3)** Does HAND enable learning tasks in *new scenes* in simulation?
- **(Q4)** Can HAND enable *real-time, fast* adaptation on a real robot?

4.1 Experimental Setup

We evaluate HAND both in simulation using the CALVIN benchmark [8] and on real-world manipulation tasks with the WidowX-250 robot arm.

CALVIN contains unstructured, teleoperated play data in four tabletop manipulation environments {A,B,C,D}, that share the same set of objects, but have different visual textures and static object locations (e.g., slider, button, switch), shown in Figure 6 (Left). Because it is infeasible to provide explicit human hand demonstrations in CALVIN, we instead perform end-effector point-tracking on expert task demonstrations to mimic the effect of hand-based tracking. We uniformly sample $N = 6$ task-specific expert trajectories from environment D as $\mathcal{D}_{\text{hand}}$, and utilize about 17k trajectories from environments {A,B,C} as $\mathcal{D}_{\text{play}}$. We evaluate our fine-tuned policy in environment D across 3 tasks.

Real World. We demonstrate that HAND can also scale to real-world scenarios by evaluating on several manipulation tasks in a kitchen setup shown in Figure 7. We collect a task-agnostic play dataset of about 50k transitions. Human teleoperators were instructed to freely interact with the available objects in the scene without being bound to specific task goals. Object positions are randomized within the workspace during data collection and evaluation. We test three tasks: `Reach Green Block`, `Press Button`, and `Close Microwave`. We also introduce two additional difficult, long-horizon tasks, `Put K-Cup in Coffee Machine` and `Blend Carrot`, which require great precision and more than 150 real-world timesteps at a 5hz control frequency to execute, highlighting the capabilities of HAND to learn complex behaviors in real-time. Partial success is provided for tasks composed of multiple subtasks. Refer to Appendix A.2 for description of each task.

Baselines: We compare HAND to several retrieval baselines. All methods use the same transformer policy where applicable. We refer the reader to Appendix A for implementation details and Appendix E for extensive ablation results. We consider the following baseline methods:

- $\pi_{\textbf{base}}$ is the base policy pre-trained only on *task-agnostic* play data;
- CLIP retrieves based on cosine similarity between target task language description's CLIP embeddings (instead of hand demonstrations) and play data's CLIP frame embeddings;
- Flow [11] trains a dataset-specific VAE on pre-computed optical flows for $\mathcal{D}_{\text{play}}$ from GMFlow [41] and retrieves individual states-action pairs based on latent motion similarity; and
- STRAP [12] also uses S-DTW for sub-trajectory retrieval but computes S-DTW distance based solely on Euclidean distance between pre-trained DINO-v2 image embeddings.

Figure 4: **Qualitative retrieval results on out-of-distribution scene.** We visualize the top sub-trajectory match of Flow , STRAP , HAND without visual filtering (HAND(-VF)), and HAND on two out-of-domain demonstrations recorded from an iPhone camera, showing approaching a K-Cup and putting it into the machine. Only HAND's top match is relevant for both hand demonstrations.

STRAP and Flow assume access to expert *robot* demonstrations for both retrieval and fine-tuning. Unless otherwise stated, we adopt them for our setting without expert robot demonstration fine-tuning because we do not assume access to them. STRAP and Flow propose training policies from scratch, but we LoRA fine-tune both of them, like with HAND, because we found it to perform better.

4.2 Experimental Evaluation

(Q1): HAND retrieves more task-relevant data. We analyze the quality of retrieved sub-trajectories between Flow, STRAP and HAND. STRAP and HAND both use S-DTW-based trajectory retrieval, but STRAP relies purely on visual DINO-v2 embeddings for retrieval. We provide a single hand demonstration of three real robot tasks and retrieve the top $K = 25$ matches from $\mathcal{D}_{\text{play}}$. Compared to STRAP and Flow, we observe in Table 1 that HAND retrieves more trajectories in which the robot performs the demonstrated task. As both STRAP and Flow retrieve solely based on visual similarity, they perform poorly when there is a visual gap between the target demonstrations, e.g., human hand videos, and the offline robot play dataset. In particular, for the Push Button task, STRAP cannot retrieve any button pushing trajectories in its top 25 matches. Moreover, we ablate HAND's visual filtering step and show that it helps retrieve $+30\%$ more relevant trajectories across all tasks. We provide qualitative comparisons of retrieved trajectories by in Appendix D.

	Block	Button	Microwave
Flow	7/25	0/25	0/25
STRAP	5/25	0/25	2/25
HAND(-VF)	9/25	13/25	9/25
HAND	**15/25**	**18/25**	**11/25**

Table 1: **Number of retrieved sub-trajectories performing demonstrated task.** HAND retrieves more sub-trajectories performing the task compared to Flow and STRAP.

(Q2): HAND works with hand demos from unseen environments. Because HAND retrieves based on *relative hand motions*, it can work with target hand demos from out-of-distribution scenes, provided the camera angle remains relatively close to that in the play dataset. To demonstrate this scene robustness, we collect hand demos from a different scene with a handheld iPhone camera and a real coffee machine. We retrieve from robot play data containing a completely different scene and a toy coffee machine. In Figure 4, we show the lowest cost retrieved sub-trajectory of STRAP and Flow compared to HAND and a HAND ablation without the visual filtering step, HAND(-VF). Both of the retrieved trajectories for STRAP and Flow, along with the top trajectory for HAND(-VF) are irrelevant to the demonstrated task. Only HAND retrieves relevant robot trajectories for both hand demos because it focuses on the *motion* demonstrated by the human hand after *visual filtering*.

(Q3): HAND enables policy learning in simulation and real world. In Figure 3, we demonstrate that HAND and ablations outperform baselines in CALVIN, with a **16%** average improvement over Flow and **123%** over STRAP. In Figure 3, we also ablate the use of S-DTW-based loss weighting from Equation (2) with HAND(-CW), visual filtering from Equation (1) with HAND(-VF), and ground

truth 3D pose information with HAND(+3D,-VF,-CW). HAND outperforms all of these ablations in Move Slider Left. Surprisingly, in this task, HAND(+3D,-VF,-CW) with priviledged 3D information, even underperforms HAND(-CW). We believe this is because, as HAND(+3D,-VF,-CW) retrieves trajectories based on an exact match in 3D end-effector pose, the retrieved trajectories have little variability and thus fail to generalize to changes in object placement in the scene. In some tasks, we notice that adding visual filtering can negatively impact performance, likely for a similar reason that filtering constrains the diversity of the resulting data subset. However, we demonstrated in the above two paragraphs that visual filtering helps in the real world to retrieve task-relevant trajectories.

Real-world experiments in Figure 5 demonstrate that fine-tuning with HAND improves success rates by **+45%** over the next best baseline, STRAP. In contrast, Flow fails to learn a policy that achieves reasonable success rates in any of the tasks, despite it being the best-performing baseline in CALVIN. Despite visual filtering not always helping in simulation in CALVIN, we observe that **visual filtering is necessary in the real world** to retrieve trajectories where the target object is interacted with, as demonstrated with HAND(-VF)'s worse retrieval performance in Table 1. We ablate different K values for real robot tasks in Appendix F. We also report the performance of π_{base}, trained on all of $\mathcal{D}_{\text{play}}$.

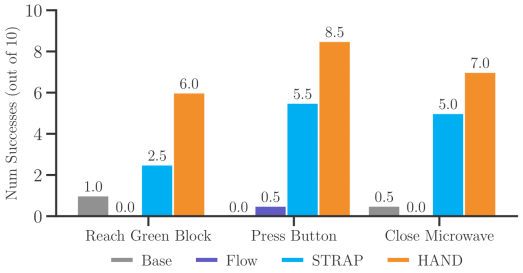

Figure 5: **Real-Robot Results.** Task completion out of 10 of π_{base}, STRAP, Flow, and HAND.

(Q4): HAND enables real-time, data-efficient policy learning of long-horizon tasks. We performed two small-scale user studies with IRB approval from our institution to demonstrate real-time learning. In the first study, a participant familiar with HAND iteratively demonstrated each part of a long-horizon `Blend Carrot` task (shown in Figure 7) and trained a HAND policy ***with over 70% success rate in under four (4) minutes*** from providing a single hand demonstration to deploying the fine-tuned policy. A video of a similar experiment can be found on our project website.

In the second study, two external users with prior teleoperation experience—but not affiliated with this research—each collected 10 demonstrations, using both their hand and robot teleoperation, to train the robot for the `Put Keurig Cup in Coffee Machine` task (see Figure 7). We employ HAND retrieval for hand-collected demonstrations and STRAP retrieval for robot teleoperation demonstrations. For a direct comparison, we additionally fine-tune STRAP with the human-collected teleoperated demonstrations as their paper suggests.

Method	User 1 (Minutes)	User 2 (Minutes)
Hand Demos (Min) ↓	**3**	**2**
Robot Demos (Min) ↓	10	14
Hand Demos (SR) ↑	**5/10**	**4/10**
Robot Demos (SR) ↑	3/10	2.5/10

Table 2: **Hand vs. Robot Teleoperation**. Comparison of time taken and success rates between hand and teleoperated demonstrations.

As reported in Table 2, teleoperated demonstrations required over 3× more time to collect than hand demonstrations. Notably, using a single hand demonstration per user, we fine-tuned a policy exceeding 40% task completion compared to STRAP which only achieves 25% using a single *robot teleoperation* demonstration. We observed that increasing the number of expert demonstrations used for STRAP actually hurt downstream performance as it reduced the quality of retrieved trajectories. Thus, our results demonstrate that HAND is effective in enabling *fast* adaptation to downstream tasks.

5 Conclusion

We presented HAND a simple and time-efficient framework for adapting robots to new tasks using easy-to-provide human hand demonstrations. We demonstrated that HAND enables *real-time* task adaptation with a *single* hand demonstration in just several minutes of policy fine-tuning. Our results highlight the scalability of HAND to train performant real-world, task-specific policies.

6 Limitations

Relative Camera Viewpoint. One limitation of HAND is that we assume the relative camera viewpoint between the hand demonstration and play trajectories are similar. However, this is a reasonable assumption given that many tabletop manipulation works assume a fixed external camera view. Many open-sourced large-scale offline robot datasets similarly assume standardized camera viewpoints [42, 43, 44, 45]. Moreover, we demonstrated the flexibility of HAND as it is robust to out-of-distribution scenes that are completely different from the ones in the play dataset. In particular, we show that our 2D path retrieval metric is able to retrieve relevant task trajectories even when using a hand demonstration from a regular iPhone camera.

Extending to 3D paths for retrieval. While HAND uses 2D paths for retrieval, one future direction could extend HAND to estimate the hand trajectory in 3D using foundation depth prediction models. Incorporating depth information could provide more fine-grained information about the hand path. Furthermore, 2D hand paths do not provide any explicit information about the gripper for retrieval, which could be useful for more dexterous manipulation tasks. Another direction future work could consider is a mixture of features for improving retrieval for tasks that require more dexterous control, i.e., cloth folding or deformable object manipulation.

Acknowledgments

We thank Yutai Zhou and Yigit Korkmaz for participating in our user studies. We also thank Sid Kaushik for feedback on the final draft of the paper.

References

[1] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, C. R. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, and A. Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, 2025.

[5] G. R. Team et al. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

[6] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, 2020.

[7] S. Young, J. Pari, P. Abbeel, and L. Pinto. Playful interactions for representation learning. In *International Conference on Intelligent Robots and Systems*. IEEE, 2022.

[8] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *Robotics and Automation Letters (RA-L)*, 2022.

[9] S. Nasiriany, T. Gao, A. Mandlekar, and Y. Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Conference on Robot Learning*, 2022.

[10] M. Du, S. Nair, D. Sadigh, and C. Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. In *Robotics: Science and Systems*, 2023.

[11] L.-H. Lin, Y. Cui, A. Xie, T. Hua, and D. Sadigh. Flowretrieval: Flow-guided data retrieval for few-shot imitation learning. In *Conference on Robot Learning*, 2024.

[12] M. Memmel, J. Berg, B. Chen, A. Gupta, and J. Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In *International Conference on Learning Representations*, 2025.

[13] K. Sridhar, S. Dutta, D. Jayaraman, and I. Lee. REGENT: A retrieval-augmented generalist agent that can act in-context in new environments. In *International Conference on Learning Representations*, 2025.

[14] G. Papagiannis, N. D. Palo, P. Vitiello, and E. Johns. R+x: Retrieval and execution from everyday human videos, 2024.

[15] S. Haldar and L. Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. *arXiv preprint arXiv:2502.20391*, 2025.

[16] M. J. Kim, J. Wu, and C. Finn. Giving robots a hand: Learning generalizable manipulation with eye-in-hand human video demonstrations. *CoRR*, 2023.

[17] M. Lepert, J. Fang, and J. Bohg. Phantom: Training robots without robots using only human videos, 2025.

[18] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. In *Proc. ECCV*, 2024.

[19] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. In *European Conference on Computer Vision*, 2025.

[20] J. Zhang, M. Heo, Z. Liu, E. Biyik, J. J. Lim, Y. Liu, and R. Fakoor. EXTRACT: Efficient policy learning by extracting transferrable robot skills from offline data. In *Conference on Robot Learning*, 2024.

[21] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[22] K. Kedia, P. Dan, A. Chao, M. A. Pace, and S. Choudhury. One-shot imitation under mismatched execution. In *International Conference on Robotics and Automation (ICRA)*, 2025.

[23] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song. Flow as the cross-domain manipulation interface. In *Conference on Robot Learning*, 2024.

[24] C. Yuan, C. Wen, T. Zhang, and Y. Gao. General flow as foundation affordance for scalable robot learning. *Conference on Robot Learning*, 2024.

[25] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics. 2023.

[26] Y. Kuang, J. Ye, H. Geng, J. Mao, C. Deng, L. Guibas, H. Wang, and Y. Wang. Ram: Retrieval-based affordance transfer for generalizable zero-shot robotic manipulation. *Conference on Robot Learning*, 2024.

[27] S. Kareer, D. Patel, R. Punamiya, P. Mathur, S. Cheng, C. Wang, J. Hoffman, and D. Xu. Egomimic: Scaling imitation learning via egocentric video, 2024.

[28] J. Xie, Z. Xu, J. Zeng, Y. Gao, and K. Hashimoto. Human–robot interaction using dynamic hand gesture for teleoperation of quadruped robots with a robotic arm. *Electronics*, 2025.

[29] H. Li, Y. Cui, and D. Sadigh. How to train your robots? the impact of demonstration modality on imitation learning, 2025.

[30] M. Deitke et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024.

[31] M. Müller. *Fundamentals of music processing: Using Python and Jupyter notebooks*, volume 2. Springer, 2021.

[32] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2023.

[33] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[34] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems*, 2023.

[35] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity, 2024.

[36] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *Neural Information Processing Systems*, 2024.

[37] A. Liang, I. Singh, K. Pertsch, and J. Thomason. Transformer adapters for robot learning. In *CoRL 2022 Workshop on Pre-training Robot Learning*, 2022.

[38] Z. Liu, J. Zhang, K. Asadi, Y. Liu, D. Zhao, S. Sabach, and R. Fakoor. TAIL: Task-specific adapters for imitation learning with large pretrained models. In *International Conference on Learning Representations*, 2024.

[39] 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. *International Conference on Learning Representations*, 2022.

[40] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[41] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao. Gmflow: Learning optical flow via global matching. In *Conference on Computer Vision and Pattern Recognition*, 2022.

[42] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, 2023.

[43] O. X.-E. Collaboration et al. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[44] A. Khazatsky et al. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[45] H. Xiong, H. Fu, J. Zhang, C. Bao, Q. Zhang, Y. Huang, W. Xu, A. Garg, and C. Lu. Robotube: Learning household manipulation from human videos with simulated twin environments. In *Conference on Robot Learning (CoRL)*. PMLR, 2023.

[46] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

[47] L. Wang, X. Chen, J. Zhao, and K. He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Neural Information Processing Systems*, 37:124420–124450, 2024.

A Environment Details and Hyperparameters

(a) CALVIN [8] (b) Real-World WidowX-250

Figure 6: **Environments.** We retrieve data from a prior dataset to train on *new scenes* in CALVIN. On our real-world WidowX-250 robot, we demonstrate real-world learning from HAND-retrieved trajectories along with real-time adaptation to long-horizon tasks.

A.1 CALVIN.

The CALVIN benchmark is built on top of the PyBullet [46] simulator and involves a 7-DOF Franka Emika Panda Robot arm that manipulates the scene. CALVIN consists of 34 tasks and 4 different environments (`ABCD`). All environments are equipped with a desk, a sliding door, a drawer, a button that turns on/off an LED, a switch that controls a lightbulb and three different colored blocks (red, blue and pink). These environments differ from each other in the texture of the desk and positions of the objects. CALVIN provides 24 hours of tele-operated unstructured play data, 35% of which are annotated with language descriptions. We utilize this 35% as a natural way to obtain a smaller subset of the data as the full dataset is very large, but we do not use the task-oriented language instructions. In total, $\mathcal{D}_{\text{play}}$ corresponds to $\sim$ 17k trajectories for our experiments.

We evaluate on the following tasks:

- **Close Drawer.** For this task, the arm is required to push an opened drawer and close it. The drawer's degree of openness is randomized.
- **Move Slider Left.** This task requires the robot arm to move a slider located on the desk from the right to the left. The slider position is randomized.
- **Turn On Led.** In this task, the robot arm needs to navigate its way to a button and press down on it such that an LED turns on.
- **Lift Blue Block Table.** For this task, the robot arm needs to pick up a blue block from the table. The location of the blue block on the table is randomized.

A.2 Real Robot Experimental Setup

Figure 7: **Real Robot Tasks.** We evaluate HAND on 5 different real robot tasks. The last two are long-horizon tasks, requiring more than 100 timesteps of execution.

Hardware Setup. We evaluate HAND on a real-world multi-task kitchen environment using the WidowX robot arm. The WidowX is a 7-DoF robot arm with a two-fingered parallel jaw gripper. Our robot environment setup is shown in Figure 6. We use an Intel Realsense D435 RGBD camera as a static external camera and a Logitech webcam as an over-the-shoulder camera view. We use a Meta Quest 2 VR headset for teleoperating the robot.

Task-agnostic play dataset. Our play dataset contains a total of 50k transitions, each trajectory having an average of 230 timesteps and covering multiple tasks, collected at 5hz. To encourage diverse behaviors and motions, human teleoperators were instructed to freely interact with the available objects in the scene without being bound to specific task goals.

Evaluation protocol. We introduce distractor objects in the scene that are not part of the task so that the policy does not just memorize the expert demonstrations. Moreover, movable task object positions are randomized in a fixed region if applicable. We evaluate on four manipulation tasks described below:

- **Reach Block.** In this task, the robot arm must reach and hover directly above a green block placed on the table. Success is achieved when the gripper remains positioned clearly above the block. Partial success is awarded if the gripper end-effector touches the block without hovering steadily above it.
- **Push Button.** This task requires the robot arm to press the right-side button on a stovetop. Success is achieved upon pressing the button. Partial success is awarded if the robot arm approaches sufficiently close to the button without making contact.
- **Close Microwave.** This task requires the robot to close a microwave door from various starting angles. Partial successe is awarded if the robot pushes the door without completely closing it. A successful closure is confirmed by an audible click sound.
- **Put K-Cup in Coffee Machine.**[5] In this task, the robot needs to first pick up the Keurig cup and then transport it to the coffee machine and insert the cup into the cup holder. This task requires precision low-level control as the Keurig cup is small, making it difficult to grasp reliably. Additionally, the cup holder on the coffee machine is just large enough to fit the Keurig cup, leaving small margin of error during the insertion. The coffee machine is fixed to the kitchen stovetop, while the initial location of the Keurig cup is randomized. Given the difficulty of the task, we provide partial success for successfully grasping the Keurig cup.
- **Blend Carrot.** The robot first picks up a toy carrot and then drops it into the blender. Once the carrot is inside the blender, it will press a button at the blender base to activate the blender and hold the button for 2 seconds. The location of the blender is static, but the carrot is randomized. Partial success is provided for picking up the carrot and also successfully dropping it into the blender. Due to the long-horizon nature of the task, we allocate a budget of 300 timesteps to account for its extended duration.

Robot Policy. For our policy, we are inspired by the architectural components introduced in Wang et al. [47] and Zhao et al. [33]. A diagram of our policy architecture is shown in Figure 8. For both external and over-the-shoulder RGB images, we use a pretrained ResNet to first extract 7×7 feature

[5] https://www.samsclub.com/p/members-mark-gourmet-kitchen-appliances-playset/P990340349?xid=plp_product_2

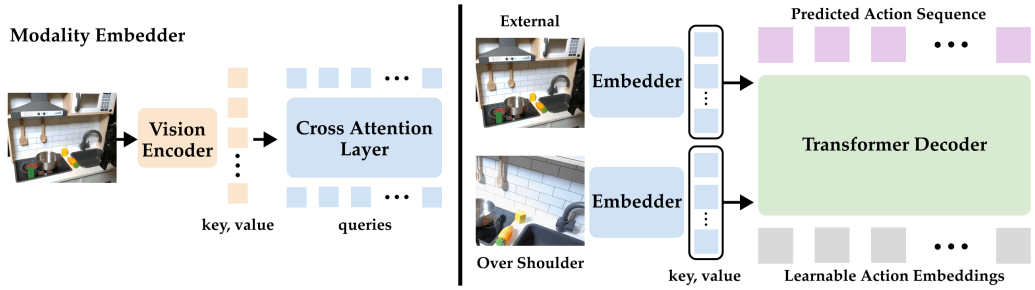

Figure 8: **Real Robot Policy Architecture.** (Left) Learnable image embeddings following [47]. (Right) The learned image embeddings for each modality are concatenated and provided to a transformer decoder similar to [33]. We also perform action chunking with a chunk size of 5 timesteps for 1 second of execution.

maps and flatten these features across the spatial dimension to create a sequence of d_v dimension tokens where d_v is the output dimension of ResNet. In particular, we use ResNet18 where $d_v = 512$. We feed these visual tokens as learnable input tokens to a causal transformer decoder with dimension d. We use the flattened image feature map as the keys and values and apply a cross-attention between the image features and learnable tokens. We concatenate all modality tokens and add additional modality-specific embeddings and sinusoidal positional embeddings.

The policy base is a transformer decoder similar to the one used in ACT [33]. The input sequence to the transformer is a fixed position embedding, with dimensions $k \times 512$ where k is the chunk size and the keys and values are the combined image tokens from the stem. Given the current observation, we predict a chunk of $k = 5$ actions, which corresponds to 1 second of execution. During inference time, we also apply temporal ensembling similar to [33] with exponential averaging parameter $m = 0.5$, which controls the weight of previous actions.

We train the policy for 20k update steps with batch size of 256 and a learning rate of $3e^{-4}$ (around 2 hours of wall time). The action dimension is 7 comprising of continuous Cartesian end-effector motion (6), corresponding to relative changes in pose and the gripper state (1).

For our experiments, we employ two transformer architectures: small and large. The small transformer is used by default; however, for more challenging, long-horizon tasks that require greater precision such as the `Put K-Cup in Coffee Machine` and `Blend Carrot` tasks, we utilize the large architecture.

Hyperparameter	Small	Large
Number of attention heads	4	8
Number of transformer layers	3	6
Embedding dimension (d)	256	512

Table 3: **Transformer Hyperparameters.** Small and large transformer architectures depending on task complexity.

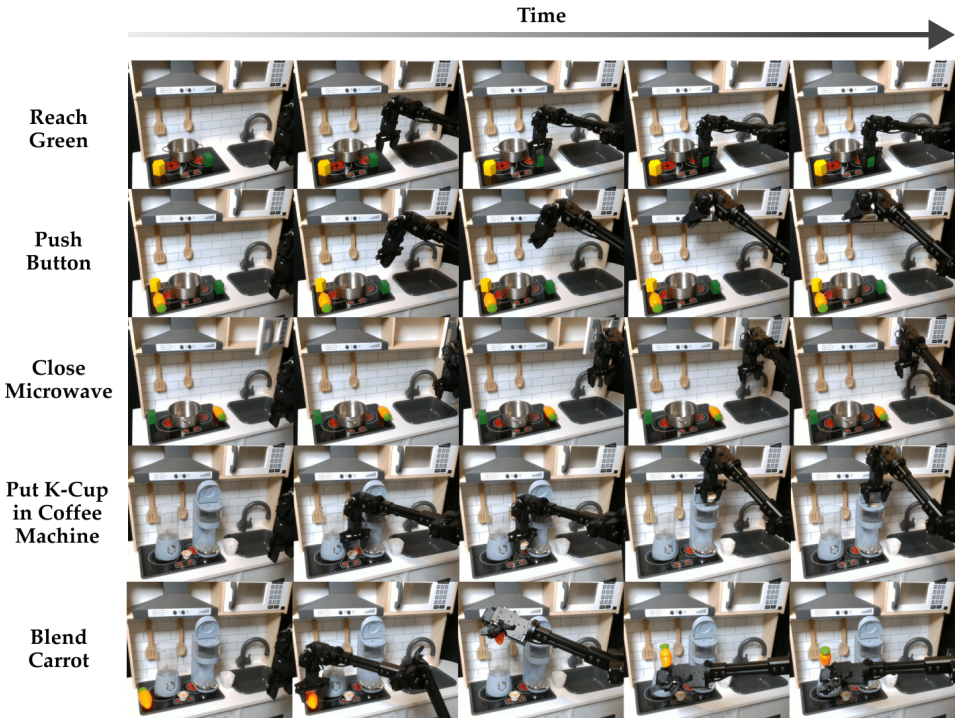

Figure 9: Task Rollouts

B HAND Algorithm

Algorithm 1 HAND FULL ALGORITHM

Require: Hand demonstrations $\mathcal{D}_{\text{hand}}$, offline play datsaet $\mathcal{D}_{\text{play}}$, CoTracker3, Molmo-7B, # retrieved sub-trajectories K, threshold ϵ, DINO, # visual filtered sub-trajectoies M

/ Policy Pretraining */*

Train π_{base} on $\mathcal{D}_{\text{play}}$ using regular behavior cloning loss $\mathcal{L}_{BC}$

/ Sub-Trajectory Pre-processing */*

$\mathcal{T}_{\text{hand}} \leftarrow \texttt{SubTrajSegmentation}(\mathcal{D}_{\text{hand}}, \epsilon)$ ▷ Heuristic demo segmentation

$\mathcal{T}_{\text{play}} \leftarrow \texttt{SubTrajSegmentation}(\mathcal{D}_{\text{play}}, \epsilon)$ ▷ Heuristic demo segmentation

/ Retrieval using S-DTW and 2D Hand Paths Section 3.2 */*

$\mathcal{D}_{\text{retrieved}} \leftarrow \{\}$

for $\tau_{\text{hand}} \in \mathcal{T}_{\text{play}}$ **do**

  $o^{\text{hand}}_{1:H} \leftarrow$ image obs sequence of τ_{hand}

  for $\tau_{\text{play}} \in \mathcal{T}_{\text{play}}$ **do**

    $o^{\text{play}}_{1:T} \leftarrow$ image obs sequence of τ_{play}

    / Visual Filtering */*

    Compute $\text{C}_{\text{visual}}(o^{\text{hand}}_{1:H}, o^{\text{play}}_{1:T})$ with DINO ▷ Equation (1)

  end for

  $\mathcal{T}^{M}_{\text{play}} \leftarrow M$ sub-trajectories with lowest C_{visual}

  for $\tau_{\text{play}} \in \mathcal{T}^{M}_{\text{play}}$ **do**

    $o^{\text{play}}_{1:T} \leftarrow$ image obs sequence of τ_{play}

    $(x, y)_{\text{hand}} = \text{Molmo}(o_{H/2}), (x, y)_{\text{play}} = \text{Molmo}(o_{T/2})$ ▷ Get middle frame query point

    $p_{\text{hand}} = \{(x_t, y_t)_{\text{hand}}\}^{H}_{1} = \text{CoTracker3}((x, y)_{\text{hand}})$ ▷ Track hand point

    $p_{\text{play}} = \{(x_t, y_t)_{play}\}^{T}_{1} = \text{CoTracker3}((x, y)_{\text{play}})$ ▷ Track robot gripper point

    $p_{\text{hand}} = p_{\text{hand}}[:-1] - p_{\text{hand}}[1:]$ ▷ Convert p_{hand} and p_{play} to *relative 2D paths*

    $p_{\text{play}} = p_{\text{play}}[:-1] - p_{\text{play}}[1:]$

    $(\text{C}_{\text{path}}, \tau^{\text{play}}_{i:j}) \leftarrow \text{S-DTW}(p_{\text{hand}}, p_{\text{play}})$ ▷ Path cost and corresponding retrieved sequence

  end for

  Add K lowest C_{path} $\tau^{\text{play}}_{i:j}$ sub-trajectories to $\mathcal{D}_{\text{retrieved}}$

end for

/ Parameter-Efficient Policy Fine-tuning */*

Insert *task-specific* adapter LoRA layers θ in π_{base}

Update π_{base} on $\mathcal{D}_{\text{retrieved}}$ with loss $\mathcal{L}_{BC;\theta}$ ▷ Equation (2)

return π_{θ}

C User Studies

C.1 Efficiency of Hand Demonstrations

Teleoperation **Hand Demo**

Figure 10: **Efficient Demonstrations.** Two users, unfamiliar with HAND are asked to collect trajectories either via teleoperation (Left) or using their hands (Right). HAND retrieval achieves a 50% success rate with the same amount of demonstrations using $3\times$ less time. STRAP retrieval is unable to reach 50% even when provided with more expert demonstrations.

In our first study, two users collect 10 demonstrations each either by manually teleoperating using a VR controller or by providing a hand demonstration. For manual teleoperation, we explain to the users how to operate the robot using the VR controller and allow them a couple trials to get accustomed to the interface. For hand demonstrations, we ask the users to mimic the trajectory of the robot end effector using their hands. Figure 10 shows an example of a user performing both forms of demonstrations. We observe that providing hand demonstrations is significantly more time efficient (over $3\times$) compared to manual teleoperation. Furthermore, with just a single hand demonstration, we are able to learn a performant policy with 45% average success rate, while STRAP struggles even when provided 5 expert demonstrations.

C.2 Fast Adaptation to Long-Horizon Tasks

Collect One Hand Demo (~15 seconds) **Retrieval + Fine-Tune (~2 minutes)** **Evaluate (~ 15 seconds)**

Figure 11: **Fast Adaptation.** We conduct a small-scale user study to demonstrate HAND's ability to learn robot policies in real-time. From providing the hand demonstration (Left), to retrieval and fine-tuning a base policy (Middle), to evaluating the policy (Right), we show that HAND can learn to solve the `Put Carrot in Blender` task with 7.5/10 task completion in less than 3 minutes.

We conduct a small study demonstrating that HAND enables real-time fast, adaptation to unseen downstream tasks. Snapshots at various stages of this experiment is shown in Figure 11. In our study, we measure the total time required for a user to provide a hand demonstration of a new target task to evaluating the performance of a fine-tuned policy. The hand demonstration is simple to provide and typically takes between $10 - 15$ seconds to collect. Data preprocessing, which involves computing the 2D path features of the hand demonstration and performing retrieval, takes around $30 - 40$ seconds. We assume that the offline play dataset is already preprocessed prior to the study and we do not include this time in our estimate. We also assume a base policy has already been trained on this data; however, it performs poorly on the target task. We fine-tune the base policy with 4 LoRA adapter layers for 1000 batch updates, which takes $\sim$ 2 minutes on a NVIDIA 4070 GPU. The resulting policy, which took less than 3 minutes to train and achieves 7.5/10 task completion,

highlighting the efficacy of HAND for real-time policy learning. A video of a similar experiment can be found on our project website at https://handretrieval.github.io/.

D Qualitative Retrieval Analysis

In Figure 12, we provide more qualitative results comparing STRAP retrieval results to HAND on each of our real robot tasks. Across all tasks, HAND retrieves more relevant trajectories that perform the task demonstrated by the human hand.

Figure 12: **Qualitative Retrieval Examples.** We show the top 5 matches from $\mathcal{D}_{\text{play}}$ for STRAP (top) and HAND (bottom) provided the hand demonstration for each of our evaluation tasks.

E CALVIN Results

Method	K=25	K=50	K=100	K=250
With Expert				
FT	0.425 ± 0.059	-	-	-
Flow	0.694 ± 0.089	0.797 ± 0.045	0.633 ± 0.127	0.747 ± 0.039
STRAP	0.481 ± 0.119	0.286 ± 0.073	0.703 ± 0.075	0.600 ± 0.085
Without Expert				
π_{base}	0.233 ± 0.024	-	-	-
CLIP	0.003 ± 0.004	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
Flow	0.808 ± 0.080	0.831 ± 0.058	0.533 ± 0.106	0.653 ± 0.055
STRAP	0.000 ± 0.000	0.011 ± 0.010	0.006 ± 0.008	0.031 ± 0.004
HAND(+3D,-VF,-CW)	0.994 ± 0.004	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
HAND(-VF,-CW)	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
HAND(-VF)	1.000 ± 0.000	1.000 ± 0.000	0.997 ± 0.004	1.000 ± 0.000
HAND	0.828 ± 0.169	0.464 ± 0.061	0.536 ± 0.082	0.436 ± 0.136

Table 4: **CALVIN Close Drawer**: Performance with and without expert demonstrations

Method	K=25	K=50	K=100	K=250
With Expert				
FT	0.564 ± 0.309	-	-	-
Flow	0.092 ± 0.038	0.086 ± 0.017	0.156 ± 0.046	0.039 ± 0.039
STRAP	0.053 ± 0.034	0.075 ± 0.012	0.111 ± 0.014	0.094 ± 0.037
Without Expert				
π_{base}	0.011 ± 0.010	-	-	-
CLIP	0.017 ± 0.024	0.033 ± 0.047	0.006 ± 0.004	0.031 ± 0.024
Flow	0.000 ± 0.000	0.247 ± 0.116	0.094 ± 0.046	0.053 ± 0.014
STRAP	0.058 ± 0.018	0.122 ± 0.022	0.075 ± 0.025	0.028 ± 0.024
HAND(+3D,-VF,-CW)	0.028 ± 0.008	0.047 ± 0.010	0.192 ± 0.049	0.139 ± 0.040
HAND(-VF,-CW)	0.186 ± 0.088	0.081 ± 0.017	0.364 ± 0.149	0.619 ± 0.092
HAND(-VF)	0.069 ± 0.042	0.167 ± 0.056	0.200 ± 0.123	0.325 ± 0.014
HAND	0.647 ± 0.229	0.483 ± 0.041	0.636 ± 0.103	0.431 ± 0.107

Table 5: **CALVIN Move Slider Left**: Performance with and without expert demonstrations

Method	K=25	K=50	K=100	K=250
With Expert				
FT	0.000 ± 0.000	-	-	-
Flow	0.131 ± 0.085	0.344 ± 0.092	0.697 ± 0.082	0.581 ± 0.134
STRAP	0.200 ± 0.147	0.125 ± 0.042	0.056 ± 0.017	0.372 ± 0.220
Without Expert				
π_{base}	0.036 ± 0.014	-	-	-
CLIP	0.025 ± 0.035	0.006 ± 0.008	0.019 ± 0.016	0.000 ± 0.000
Flow	0.017 ± 0.024	0.011 ± 0.008	0.364 ± 0.147	0.436 ± 0.031
STRAP	0.500 ± 0.131	0.600 ± 0.184	0.525 ± 0.150	0.633 ± 0.112
HAND(+3D,-VF,-CW)	0.333 ± 0.111	0.661 ± 0.093	0.814 ± 0.059	0.489 ± 0.136
HAND(-VF,-CW)	0.675 ± 0.065	0.719 ± 0.155	0.886 ± 0.032	0.431 ± 0.103
HAND(-VF)	0.428 ± 0.016	0.467 ± 0.138	0.828 ± 0.058	0.881 ± 0.034
HAND	0.136 ± 0.102	0.278 ± 0.073	0.186 ± 0.051	0.094 ± 0.017

Table 6: **CALVIN Turn On LED**: Performance with and without expert demonstrations

F Real Robot Results

Task \ Method	π_{base}	Flow	STRAP			HAND(-VF)	HAND		
		K=25	K=10	K=25	K=50	K=25	K=10	K=25	K=50
Reach Green Block	0/1.0	0/1.5	0/2.5	1/2.0	1/2.5	2/2.5	3/6.0	6/7.5	3/5.0
Press Button	0/0.0	0/0.0	2/5.5	1/5.0	0/2.5	3/4.0	7/8.5	2/5.0	0/4.0
Close Microwave	0.5/0	0.5/0	1/5.0	1/2.5	1/4.0	2/3.0	5/7.0	6/8.0	3/4.5

Table 7: **Real-world expert demonstrations** ($N = 3$). Success rates/task completions out of 10 trials per task.

Task \ Method	π_{base}	Flow	STRAP			HAND(-VF)	HAND		
		K=25	K=10	K=25	K=50	K=25	K=10	K=25	K=50
Reach Green Block	0/1.0	0/0.0	2/3.0	0/1.0	0/1.0	0/1.0	5/6.5	5/7.0	4/6.0
Press Button	0/0.0	0/0.5	1/1.5	0/0.0	0/0.5	0/0.0	4/4.5	5/6.0	2/3.5
Close Microwave	0/0.5	0/0.0	0/0.0	0/0.0	0/0.0	2/5.0	7/8.0	2/4.0	0/1.0

Table 8: **Real-world hand demonstration** ($N = 1$). Success rates/task completions out of 10 trials per task.

Metric	Value
Task Completion	7.5/10
Success Rate	6/10

Table 9: **Real-world Blend Carrot**. Task completion and success rate.