One-Shot Imitation under Mismatched Execution

Kushal Kedia* **Prithwish Dan*** **Angela Chao** **Maximus A. Pace** **Sanjiban Choudhury**
Cornell University

Abstract: Human demonstrations as prompts are a powerful way to program robots to do long-horizon manipulation tasks. However, translating these demonstrations into robot-executable actions presents significant challenges due to execution mismatches in movement styles and physical capabilities. Existing methods either depend on human-robot paired data, which is infeasible to scale, or rely heavily on frame-level visual similarities that often break down in practice. To address these challenges, we propose `RHyME`, a novel framework that automatically aligns human and robot task executions using optimal transport costs. Given long-horizon robot demonstrations, `RHyME` synthesizes semantically equivalent human videos by retrieving and composing short-horizon human clips. This approach facilitates effective policy training without the need for paired data. `RHyME` successfully imitates a range of cross-embodiment demonstrators, both in simulation and with a real human hand, achieving over 50% increase in task success compared to previous methods. We release our datasets and graphics at this website.

Keywords: Imitation Learning, Manipulation, Representation Learning

1 Introduction

Human demonstrations offer an effective approach for programming robots to execute long-horizon manipulation tasks [1–4]. Unlike language instructions, demonstrations are grounded in the task environment, providing rich cues for what steps to follow, which objects to interact with, and how to interact with them [5, 6].

We view this as a translation problem where a human video must be translated into a series of robot actions [7–10]. However, training such policies typically requires paired human-robot demonstrations, which is impractical to collect at scale for long-horizon tasks. Although large-scale human videos (e.g., YouTube) and robot datasets exist [11, 12], they are unpaired, making them unsuitable for directly learning this translation.

Prior works leverage unpaired human and robot demonstrations to learn visual representations that map both human and robot images into a shared embedding space [4, 6, 13–15]. A policy is then trained to generate actions conditioned on robot video embeddings, and directly transferred at test time to work with embeddings from a human prompt video. However, a key assumption these works rely on is that the human and robot perform tasks with *matched execution*, i.e., the human executes tasks in a visually similar way to that of the robot (e.g. slowly moving one arm with a simple grasp). In reality, humans often act more swiftly, use both hands for manipulation, or even execute multiple tasks simultaneously, creating a mismatch in execution styles. This mismatch leads to misalignment between the human and robot embeddings, hindering direct policy transfer.

We tackle this problem of imitation under *mismatched execution*. ***Our key insight is that while a human and robot may perform the same task in visually and physically different ways, we can establish a high-level equivalence by reasoning over the entire sequence of image embeddings they generate.*** We show that while individual image embeddings may appear different between human and robot, we can construct sequence-level similarity functions where the two are closer. Notably, we can do this ***without fine-tuning representations*** on paired data.

* Denotes equal contribution.

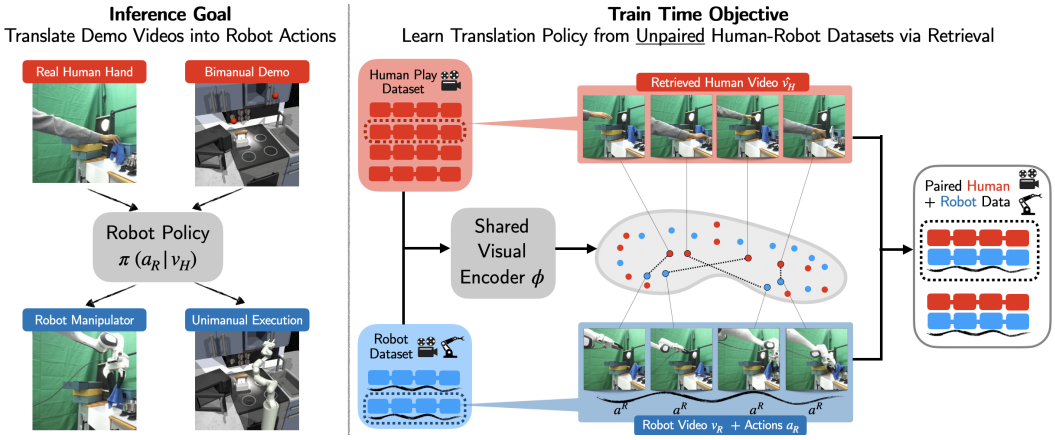

Figure 1: **Overview of RHyME.** We introduce RHyME, a hierarchical framework that trains a robot policy to mimic a long-horizon video from a demonstrator that exhibits mismatched task execution. **Inference Time (Left):** Our robot policy translates a demonstrator video into actions to complete the same long-horizon tasks specified by the input. **Train Time (Right):** Given unpaired robot-demonstrator datasets, RHyME "imagines" a paired dataset by employing sequence-level similarity metrics which can be used for training the policy.

We propose RHyME (Retrieval for Hybrid Imitation under Mismatched Execution), a framework which trains a robot policy to follow a long-horizon demonstration from a mismatched expert without access to paired human-robot videos (Fig. 1). First, RHyME defines a sequence-level similarity metric between human and robot embeddings, using optimal transport to measure alignment. Given a robot trajectory and a database of human play data, RHyME *imagines* a long-horizon video by retrieving and composing short-horizon snippets of human demonstrations similar to the robot video. This retrieval process is guided by an optimal transport similarity metric between human and robot sequences. The framework trains a policy using a hybrid approach, incorporating both real robot demonstrations and imagined human sequences. Our contributions can be summarized as:

1. We propose RHyME, a novel retrieval-based algorithm for one-shot imitation from videos of a human demonstrator with mismatched execution. Without access to paired datasets, RHyME aligns human and robot videos at a sequence-level using optimal transport costs.
2. We systematically study the performance of video retrieval methods both in simulation and the real world. In simulation, we release three novel cross-embodiment demonstrator datasets (totaling over 10 hours of videos) exhibiting increasing levels of execution mismatch. We further validate our algorithm by imitating real-world videos of a human hand.
3. We show that RHyME outperforms a range of baselines on all degrees of mismatches, yielding over 50% increase in success rates in the most challenging scenarios.

2 Related Work

Human demonstrations have been utilized to guide robot manipulation policies in several different ways. We place our work relative to each cluster of related research.

Tracking Reference Motion. The imitation challenge is reduced to motion tracking when the robot receives the demonstrator's motion as input. Robots with human-like joint configurations can directly mimic human trajectories, a method applied to humanoid robots with similar morphologies [16–21] and robotic hands [22–29]. When the execution capabilities are mismatched, either human models are simplified [16, 18, 20] or robot trajectories [17, 19, 30, 31] are optimized to approximately match the reference trajectory. [21] shows a method to extract human poses from videos and use that as a reference for training reinforcement learning agents. Similarly, human demonstrations have been used to guide reinforcement learning for robotic hands [22–25]. More recently, large-scale video datasets of humans on the internet have been used to extract hand positions and adapted for robot manipulation [26–29, 32], or rely on other common abstractions such as optical flow as a common trajectory representation across embodiments [33], still requiring execution to be

matched. Distinct from these works, we focus on learning robot manipulation tasks directly from human RGB videos without explicit motion input.

Learning Reward Functions from Demonstrator Videos. These methods tackle the problem of matching the demonstrator behaviors when their motion cannot be simply mimicked. Still, their videos contain useful task information and can be used to learn reward functions for reinforcement learning on the robot. A general method in this line of research [2, 34–36] is to extract reward functions that encourage the robot to manipulate objects in the same way as the video. For example, [34] self-supervises the robot by making it learn to match the demonstrations of a demonstrator perturbing a rope. GraphIRL [2] extracts sequences of object pose movement from the demonstrator and enforces temporal cyclic consistency with the robot's execution. Other approaches frame the problem as task matching [37, 38], where the robot is rewarded when it is deemed to perform the same skill as the demonstrator. While these papers tackle the problem of mismatched execution directly, they require reinforcement learning to train robot policies, which is challenging for complex tasks.

Learning Aligned Human-Robot Representations. Another strategy for addressing this challenge is to train representations for both the robot and the human that are indistinguishable when performing the same task. This approach often frames the task as a video translation challenge, where the demonstrator's video is converted into a robot's perspective to simplify task mimicry [9, 10, 39]. Additionally, methods like WHIRL [3] align videos by effectively masking out both the robot and the human, creating a neutral visual field. Embeddings can be aligned using datasets that either directly pair human and robot actions or utilize human preference datasets to rank image frames, as shown in X-IRL [1] and RAPL [40]. Unlike these methods, our approach does not depend on labeled correspondences between robots and humans.

One-shot Visual Imitation from Demonstration Videos. We tackle this problem setting in our work where the robot imitates actions from human demonstration videos in a one-shot setting, i.e., the robot uses a prompt video as a guide, aiming to replicate the demonstrated actions after viewing them once [4, 6–8, 32, 41–43]. If a paired dataset of human and robot videos executing the same task exists, the robot can learn to translate a prompt video into actions directly [7, 8]. The closest to our work is the setting without paired data of human and robot skills. Prior works [4, 6] train policies conditioning on robot videos and zero-shot transfer to a prompt demonstration at test time using aligned visual embeddings. For example, XSkill [4] uses a self-supervised clustering algorithm based on visual similarity to align representations of human and robot videos. However, such an approach can falter when there are significant mismatches in execution. We address this issue by posing the visual imitation problem as a train-time retrieval problem. During training, we match robot videos to the closest human snippets from an unpaired play dataset to imagine synthetic demonstration videos. Training robot policies conditioned on these synthetic videos enable the robot to translate demonstration videos into robot actions at test time.

3 Problem Formulation

Inference Time: Translate Human Demonstration Video to Robot Actions. The robot's goal is to replicate a series of tasks demonstrated in a video using a policy $\pi(a_R|s_R, \mathbf{v_H})$ that translates the video into robot actions a_R at state s_R. The human demonstration video is a sequence of images $\mathbf{v_H} = \{v_H^0, v_H^1, \ldots, v_H^T\}$, where T is the length of the video.

Train Time: Learning from Unpaired Human and Robot Data. While training a policy with paired human and robot data is feasible, collecting such data at scale is impractical. Instead, we frame the problem as learning and leveraging aligned embeddings from unpaired data, enabling the transfer of policies trained on robot embeddings to human embeddings.

We assume access to two datasets —a *robot dataset* (D_{robot}) of long-horizon manipulation tasks and a *play dataset* (D_{play}) of short-horizon human video clips showing interactions with objects and the environment. The robot dataset, $D_{\text{robot}} = \{(\xi_{\mathbf{R}}, \mathbf{v_R})\}$, comprises pairs of state-action trajectories and robot videos. Each robot trajectory, $\xi_{\mathbf{R}} = \{(s_R^0, a_R^0), (s_R^1, a_R^1), \ldots, (s_R^T, a_R^T)\}$, represents the sequence of robot states and actions throughout an episode. Correspondingly, the video $\mathbf{v_R} = \{v_R^0, v_R^1, \ldots, v_R^T\}$ is a sequence of images of the robot executing the task. The play

dataset, $D_{\text{play}} = \{\mathbf{v_H}\}$, consists of human videos that do not have direct correspondences with the robot dataset. At test time, the demonstrator's video contains a set of tasks whose composition is unseen by the robot during training. However, we assume that the constituent tasks are individually covered both in D_{play} and D_{robot}. This assumption is consistent with prior work [4].

Our goal is to train two modules: a vision encoder and a robot policy. The vision encoder maps both human and robot videos into a shared embedding space to enable translation. We employ a video encoder $\phi(\mathbf{v})$ to extract a sequence of embeddings $\mathbf{z} = \{z_0, z_1, \ldots, z_T\}$ for each frame from all videos[1]. Then, given a human demonstration video $\mathbf{v_H}$, we generate a sequence of latent embeddings $\mathbf{z_H}$. We aim to train a policy that conditions on the sequence of embeddings to predict robot actions $\pi(a_R | s_R, \mathbf{z_H})$ without access to paired human and robot data. We discuss how to train both the encoder and the policy in Section 4.

4 Approach

We present RHyME, a one-shot imitation learning algorithm that translates human videos into robot actions, without paired data. Before policy training, we first train a video encoder using a dataset of unpaired human and robot videos (Section 4.1). Then, this trained video-encoder is frozen and utilized for retrieval during policy training (Section 4.2). At train time, given just a robot trajectory, RHyME imagines a corresponding demonstration by retrieving and composing short-horizon human snippets. It then trains a policy to predict robot actions, conditioned on the imagined demonstration. We discuss details of the retrieval, training process, and video embeddings below.

Algorithm 1: RHyME: Retrieval for Hybrid imitation under Mismatched Execution

Input: Robot Dataset D_{robot}, Human Play Dataset D_{play}, Video Encoder $\phi(\mathbf{z}|\mathbf{v})$
Output: Trained Robot Policy $\pi_\theta(a|s, \mathbf{z})$
Initialize Robot Policy π_θ
while not converged **do**
  Get robot video and actions
  $\xi_{\mathbf{R}}, \mathbf{v_R} \sim D_{robot}$
  Generate robot embeddings $\mathbf{z_R} = \phi(v_R)$
  // Retrieve human embeddings
  $\hat{\mathbf{z}}_{\mathbf{H}} \leftarrow$ Imagine-Demo$(\mathbf{z_R}, \mathbb{D}_{play})$
  // Hybrid Training
  for (s_t, a_t) in ξ_R **do**
    // Condition on imagined demo
    Update-Policy$(a_t, \pi_\theta(s_t, \hat{\mathbf{z}}_{\mathbf{H}}))$
    // Condition on robot video
    Update-Policy$(a_t, \pi_\theta(s_t, \mathbf{z_R}))$
Return Trained Robot Policy π

Algorithm 2: Imagine-Demo: Retrieving Matched Human Embeddings

Input: Robot Embeddings $\mathbf{z_R}$, Human Play Dataset D_{play}, Video Encoder $\phi(\mathbf{z}|\mathbf{v})$, Segment Length K, Distance Function d
Output: Imagined Demo $\hat{\mathbf{z}}_{\mathbf{H}}$
Initialize empty demo $\hat{\mathbf{z}}_{\mathbf{H}} \leftarrow \{\}$
// Divide long-horizon robot sequence into short-horizon clips
$Z_R = \{z_R^{1:K}, z_R^{K+1:2K}, \ldots, z_R^{T-K+1:T}\}$
for robot segment $z_R^{i:i+K}$ in Z_R **do**
  // Find closest short-horizon clip embedding in play dataset
  $\hat{\mathbf{z}}_{\mathbf{H}} \leftarrow \arg\min_{\mathbf{z_H} \in D_{\text{play}}} d(\mathbf{z_H}, z_R^{i:i+K})$
  // Extend imagined embedding sequence with retrieved demo
  $\hat{\mathbf{z}}_{\mathbf{H}}$.extend$(\hat{\mathbf{z}}_{\mathbf{play}})$
Return Imagined Demo $\hat{\mathbf{z}}_{\mathbf{H}}$

4.1 Training the Vision Encoder

We align the human and robot video embeddings in three ways: visually, temporally, and at the task level, all without requiring trajectory-level correspondences. We employ unsupervised losses $\mathcal{L}_{\text{vis}}(\phi)$ and $\mathcal{L}_{\text{temp}}(\phi)$ for visual and temporal alignment, following prior works [44, 45], and introduce an optional task alignment loss $\mathcal{L}_{\text{task}}(\phi)$.

Visual Alignment. To align human and robot embeddings ($\mathbf{z_R}$, $\mathbf{z_H}$), we use SwAV [44], a self-supervised method that clusters images based on shared visual features. SwAV learns a set of K prototype vectors, to which each image is assigned. The SwAV loss $\mathcal{L}_{\text{vis}}(\phi)$ updates both the encoder and prototypes, aligning human and robot videos by clustering similar visual features.

Temporal Alignment. To align temporally adjacent frames in human and robot videos, we use Time Contrastive Loss [45]. This loss encourages embeddings of frames close in time to be similar. For each frame z^t, we define a positive set $\mathbf{z}^+$ of frames within a temporal window w, and a negative set $\mathbf{z}^-$ for frames outside this window. Using the contrastive loss $\mathcal{L}_{\text{temp}}(\phi)$, we pull embeddings from

[1] We encode a 1-timestep sliding window of 8 neighboring images to generate each image embedding.

the positive set closer and push negative set embeddings further apart, capturing temporal continuity across videos.

Task Alignment. Task-level alignment $\mathcal{L}_{\text{task}}(\phi)$ is used when a small set of paired human and robot snippets is available. Unlike frame-level methods, this aligns video embeddings of the robot $\mathbf{z_R}$ and demonstrator $\mathbf{z_H}$. We compute the optimal transport distance $d(\mathbf{z_R}, \mathbf{z_H})$ to measure the similarity between two sequences of video embeddings. We then apply a contrastive learning objective (INFO-NCE [46]) to pull matched embeddings closer and push different-task embeddings apart. The final task alignment loss is: $\mathcal{L}_{\text{task}}(\phi) = -\sum_i \frac{\exp(-d(\mathbf{z_R}^i, \mathbf{z_H}^i))}{\exp(-d(\mathbf{z_R}^i, \mathbf{z_H}^i)) + \sum_{j \neq i} \exp(-d(\mathbf{z_R}^i, \mathbf{z_H}^j))}$

Our final loss function for training the visual encoder ϕ is:

$$\mathcal{L}(\phi) = \lambda_{\text{vis}}\mathcal{L}_{\text{vis}}(\phi) + \lambda_{\text{temp}}\mathcal{L}_{\text{temp}}(\phi) + \lambda_{\text{task}}\mathcal{L}_{\text{task}}(\phi) \tag{1}$$

where $\lambda_{\text{task}} = 0$ by default and non-zero only with access to short-horizon paired data.

4.2 Training the Robot Policy

Training Overview. Algorithm 1 details our approach to train robot policy π_θ using both robot trajectories and imagined human demonstration videos. The training process has two stages.

Stage 1: Create a Paired Dataset. For each robot trajectory $\xi_{\mathbf{R}}$ and video $\mathbf{v_R}$ in D_{robot}, we encode the robot video into embeddings $\mathbf{z_R} = \phi(\mathbf{v_R})$ using the learned video encoder ϕ. We then retrieve imagined human embeddings $\hat{\mathbf{z}}_\mathbf{H}$ by aligning $\mathbf{z_R}$ with demonstration snippets from the play dataset D_{play}, through the function `Imagine-Demo`. This produces a paired dataset D_{paired} containing $(\hat{\mathbf{z}}_\mathbf{H}, \mathbf{z_R}, \xi_\mathbf{R})$.

Stage 2: Train Policy on Paired Dataset. The policy π_θ is trained on the paired dataset D_{paired} in a hybrid fashion. For each element in D_{paired}, we update the policy in two modes —*Mode 1:* The policy is conditioned on the robot video embeddings $\mathbf{z_R}$ to predict actions $\pi_\theta(a_t|s_t, \mathbf{z_R})$, *Mode 2:* The policy is conditioned on the imagined human demonstration embeddings $\hat{\mathbf{z}}_\mathbf{H}$ to predict actions $\pi_\theta(a_t|s_t, \hat{\mathbf{z}}_\mathbf{H})$. By alternating between these two modes, the policy learns to generalize from both robot and imagined human videos, enabling it to handle execution mismatches.

Imagining Human Demonstration Videos. Algorithm 2 details the retrieval process for imagining a sequence of human embeddings. We break the robot's video into short-horizon windows and compare the embeddings with those from the play dataset, retrieving snippets with the lowest sequence-level distance. These retrieved snippets are concatenated to form an imagined long-horizon human demonstration video. The key challenge is defining a distance function $d(\mathbf{z_R}, \mathbf{z_H})$ that can handle video sequences of varying lengths. We propose two methods to compute this distance: Optimal Transport Distance and Temporal Cyclic Consistency (TCC) Distance.

Method 1: Optimal Transport Distance. We calculate the Wasserstein distance (Optimal Transport) [47] between the human and robot video embeddings, i.e., the cost of the optimal transport plan that transfers one sequence of video embeddings into another. The robot's embedding distribution is defined as $\rho_R = \{1/T, 1/T, \ldots, 1/T\}$, and the human's embedding distribution is defined as $\rho_H = \{1/T', 1/T', \ldots, 1/T'\}$, where T and T' are the lengths of the video sequences respectively. The cost function for the transport is $C \in \mathbb{R}^{T \times T'}$ where C^{ij} is the cosine distance between the robot embedding z_R^i and the human embedding z_H^j. Our goal is to find the optimal assignment $M \in \mathbb{R}^{T \times T'}$ that transports the distribution from ρ_R to ρ_H while minimizing the cost of the plan. Formally, we need to find $M^* = \arg\min_M \sum_i \sum_j C^{ij} M^{ij}$. After solving the optimal transport assignment, the distance function is the cost of the plan, i.e., $d(\mathbf{z_R}, \mathbf{z_H}) = \sum_i \sum_j C^{ij} M^{*ij}$. In practice, we optimize an entropy-regularized version of this problem to find an approximate solution efficiently using the Sinkhorn-Knopp algorithm [47].

Method 2: Temporal-Cyclic Consistency (TCC) Distance. We calculate the TCC loss between human and robot videos following [48] which computes cycle consistency between robot video embeddings $\mathbf{z_R} = \{z_R^1, z_R^2, \ldots, z_R^T\}$ and human video embeddings $\mathbf{z_H} = \{z_H^1, z_H^2, \ldots, z_H^{T'}\}$. For each robot frame z_R^t, we first compute a similarity distribution α of z_t^R with respect to the human's embeddings, to find a soft-nearest neighbor $\tilde{z}_H = \sum_{t'=1}^{T'} \alpha_t z_H^{t'}$. Then, $\tilde{z}_H$ cycles back to the robot video by again computing its similarity distribution β with respect to robot video embeddings to get

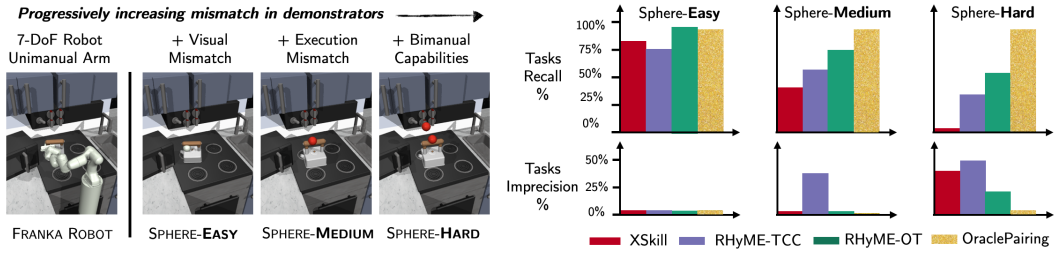

Figure 2: **Performance on Mismatched Execution Datasets.** We present results on three datasets (left). As the demonstrator's actions visually and physically deviate further from those of the robot, policies trained with our framework `RHyME` consistently outperforms `XSkill` measured by task recall and imprecision rates.

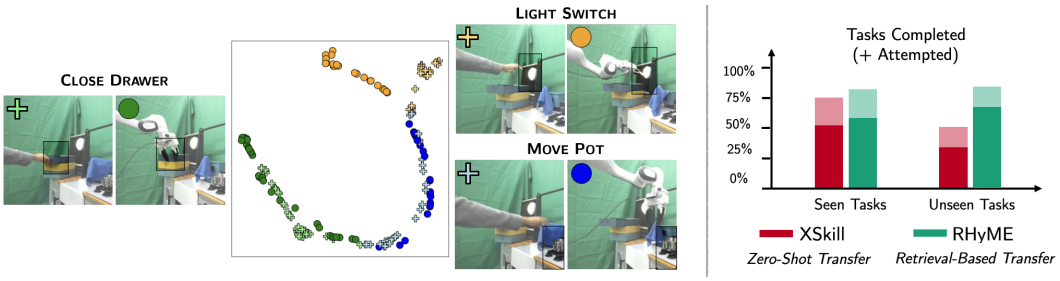

Figure 3: **Realworld Results**. (Left) **Task Embeddings:** We use t-SNE to visualize cross-embodiment latent embeddings from the human and robot completing three tasks. (Right) **Task Completion:** We compare the performance of `RHyME` with `XSkill` on seen and unseen long-horizon tasks specified by human prompt videos. Opaque segments indicate *Task Completion* rate, and augmented transparent bars indicate *Task Attempt* rate.

its soft-nearest neighbor $\tilde{z}_R^t = \sum_{t=1}^{T} \beta^t z_R^t$. The TCC distance for a robot frame z_R^t is the mean square error with its cycled-back frame $\tilde{z}_R^t$ as $\ell_{\text{tcc}} = ||z_R^t - \tilde{z}_R^t||_2$. We define the video-level TCC distance function by summing over the frame-level losses $d(\mathbf{z_R}, \mathbf{z_H}) = \sum_{t=1}^{T} \ell_{\text{tcc}}(z_R^t)$.

We hypothesize that video retrieval using TCC distance can be inaccurate in two cases: (1) When human and robot embeddings differ due to variations in execution speed or style, leading to poor frame alignment. (2) When multiple robot embeddings correspond to a single human frame, as in sequential robot tasks versus parallel human actions, causing ambiguity.

5 Experiments

Setup. `Simulation:` We evaluate our approach using the Franka Kitchen simulator [49], where a 7-DOF Franka arm performs 7 different tasks. We generate 3 cross-embodiment video datasets each progressively increasing the embodiment and execution mismatch, which contain 580 long-horizon ($\sim$25 seconds) robot trajectories completing a sequence of 4 tasks and a bank of cross-embodiment demonstrator play data ($>$ 3 hours) for training our models. First, in **Sphere-Easy**, we replace the robot's visual rendering with a sphere following the gripper's position, creating a visual gap between robot and demonstrator. Second, in **Sphere-Medium**, we introduce manipulation style mismatches by applying randomized motion primitives to the demonstrator, such as the robot dragging an object while the demonstrator lifts and carries it. Finally, in **Sphere-Hard**, we create a further divergence where the demonstrator performs two tasks simultaneously, similar to how humans use two hands. Fig. 2 (left) illustrates the cross-embodiment datasets. `Realworld:` We use a 7-DOF Franka arm to perform 4 different tasks. We train our models on 40 long-horizon ($\sim$25 seconds) robot trajectories completing a sequence of 3 tasks and $\sim$15 minutes of human play data with natural execution mismatch, holding out unseen compositions of 3 tasks for testing.

Baselines. `XSkill` [4] simply conditions on robot videos during train-time, and uses its shared representation space to zero-shot generalize to inputs of human videos at test-time. `OraclePairing` [7, 8] is the gold-standard approach, assuming an oracle pairs human demonstrations with robot trajectories, enabling conditioning on the human at train time. Our approach, `RHyME`, finds a middle ground. Without pairing, it *imagines* human videos that perform the same tasks as a robot trajectory (Section 4.2) by exploiting sequence-level correspondences. We compare two variants of our algo-

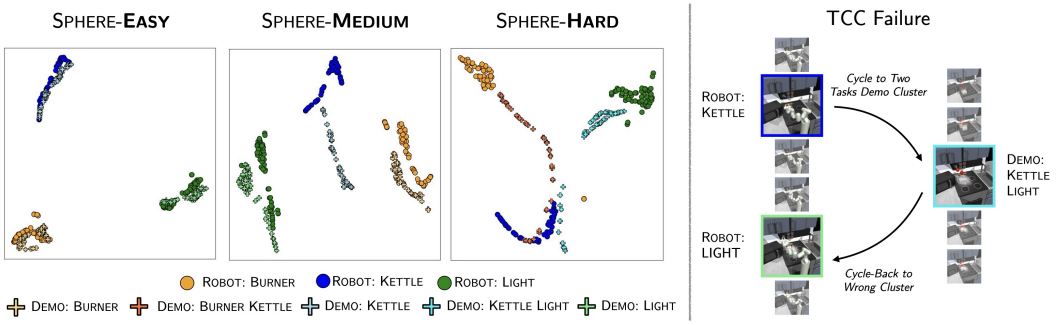

Figure 4: **Cross-Embodiment Vision Embeddings.** (Left) **Visualizing task embeddings.** We use t-SNE to visualize cross-embodiment latent embeddings generated by robot and demonstrator when executing different tasks on all three datasets. (Right) **TCC Failure Example:** The robot and video clip are equivalent, but specific frames have high TCC losses. For example, a frame showing the robot performing the 'kettle' action has a high loss due to its nearest neighbor in the video performing both 'kettle' and 'light' actions. This frame cycles back to the robot performing 'light', which is mismatched.

rithm, `RHyME-TCC` and `RHyME-OT`, which differ in their distance functions used for retrieval. In the realworld, we use `XSkill` [4] as a baseline for `RHyME-OT`. We also show how vision representations can be improved using short-horizon robot-demonstrator task pairs.

Evaluation and Success Metrics. At test-time, we provide the robot policy with long-horizon human videos as prompts. ***Simulation:*** We evaluate one-shot imitation performance across 20 different demonstrator videos, rolling out the robot policy from the last 5 model checkpoints, yielding 100 total trials per dataset. We measure *Task Recall*, which assesses recall by counting the successfully completed tasks shown in the demonstrator's video, and *Task Imprecision*, which measures imprecision and reports the percentage of tasks the robot attempts incorrectly—those not specified in the demonstrator's video. ***Realworld:*** We evaluate performance across 30 different human videos (20 seen, 10 unseen). We break down *Task Recall* into two metrics: *Task Attempts* and *Task Completions*, which measure (a) the robot's ability to attempt tasks specified by the human video and (b) the low-level control policy's ability to fully complete the tasks.

Q1. **How does performance vary across different levels of execution mismatch?**

As the cross-embodiment demonstrator's execution deviates further from those of the robot, policies trained with our framework `RHyME` consistently outperform `XSkill` (Fig. 2), with the largest gap in the bimanual demonstrator setting **SPHERE-HARD** (53% vs 1%). The `OraclePairing` baseline serves as an upper bound on performance.

Fig. 4 (left) investigates this trend by probing the visual representations of the video encoder ϕ, common across policies. We plot the image embeddings of the robot and demonstrator across three different tasks using a t-SNE plot. As execution mismatch increases, the robot and demonstrator embeddings become less clustered by task, supporting `XSkill`'s inability to zero-shot transfer to demonstrator embeddings at test-time in **SPHERE-HARD**. `RHyME` algorithms overcome this problem and successfully retrieve the correct demonstration videos at train-time by reasoning over sequences of embeddings. However, `RHyME-OT` outperforms `RHyME-TCC` across all three datasets in both metrics, suggesting inaccurate train-time retrievals with TCC.

Q2. **How does `RHyME` perform on real kitchen tasks when prompted with human videos?**

With natural visual and execution mismatches between human and robot videos, `RHyME` consistently outperforms `XSkill` when prompted with both seen and unseen human prompt videos (Fig. 3 Right). We observe marginal benefits in *Task Attempts* and *Task Completions* when faced with seen task compositions, but record significant improvements in both metrics (83% vs. 50% and 67% vs. 33%, respectively) in the unseen setting which our framework aims to generalize to. These results encourage greater investigation into the performance of both methods.

Q3. **How does video retrieval using Optimal Transport and TCC impact policies at test-time?**

As task embedding clusters deviate due to execution mismatches, we observe inaccuracies in TCC retrievals: in **SPHERE-HARD** (Fig. 4 (right)) when both clips complete the same two tasks, a bimanual task embedding lies in between two robot task clusters which results in cycling-back to the incorrect robot frame, leading to high task imprecision. `RHyME-OT` performs strictly better across datasets (Fig. 2).

The key reason for this performance difference is that optimal transport computes distances by matching videos across a sequence of embeddings. Fig. 5 visualizes the cost of the optimal transport plan between prompt robot clips and demonstration videos in the hard **SPHERE-HARD** dataset. Comparing a robot clip doing two tasks (e.g. kettle and light), the transport cost across assignments is minimum only when compared to the demonstrator performing those same two tasks. TCC, on the other hand, attempts to establish one-to-one correspondences between the robot and demonstration frames, which are lacking in this dataset.

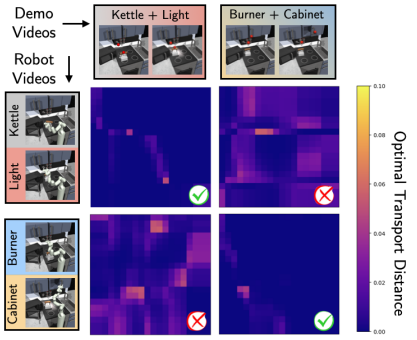

Figure 5: ***Optimal Transport Distances.*** We measure the similarity between robot and demonstrator videos on the **SPHERE-HARD** dataset by computing the cost of the Optimal Transport (OT) plans. The sum over the entire transport cost matrix costs yields the distance between videos. OT costs are lowest when tasks are the same between videos (highlighted by a tick mark).

***Q4.* Where does `RHyME` succeed, and what are common failure modes of other methods?**

We visualize the vision embeddings using t-SNE (Fig. 3 Left). We find that task embeddings in the realworld are generally clustered by task, but tend to deviate between human robot embeddings when completing the *Light Switch* task. This is directly reflected in the performance of both methods, as we observed that `XSkill` never attempts the task when prompted with human embeddings.

On the other hand, in unseen settings, `RHyME` always attempts the *Light Switch* task and completed it 9 out of 10 times. The Optimal Transport retrieval (Sec. 4) used to imagine the paired dataset recognizes can correctly match human and robot clips completing the same task by reasoning over the distribution of embeddings rather than relying on perfect embedding alignment, so `RHyME` is able to accurately pair action labels with imagined human videos at train time and obtain better performance (Fig. 3 Right).

***Q5.* Does fine-tuning visual representations with task-equivalent pairs improve one-shot imitation?**

Following Section 4.1, we assume access to short-horizon task pairings across embodiments and apply $L_{task}(\phi)$ on vision representations in the **SPHERE-HARD** setting. We find that encouraging induced distributions over embeddings to be similar lifts the performance of both `XSkill` and `RHyME-OT` (Fig. 6), and scales up with more paired clips. Ultimately, comparing induced distributions over embeddings with optimal transport is a beneficial design choice for matching clips at a task-level in the face of execution mismatches, as `RHyME-OT (0% fine-tuned)` still significantly outperforms `XSkill (fine-tuned)`.

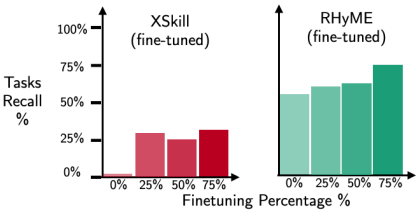

Figure 6: **Performance Improves by Pairing Skills on SPHERE-HARD.** For both non-retrieval and retrieval-based methods, performance improves when fine-tuned when the visual encoder is finetuned short-horizon robot-demonstrator snippet pairs using a contrastive optimal transport loss.

6 Discussion and Limitations

This work addresses the challenge of one-shot imitation in the presence of *mismatched execution* by the demonstrator. We propose `RHyME`, a novel framework that leverages task-level correspondences to bridge frame-level visual disparities between the robot and the demonstrator, enabling the learning of a video-conditioned policy without paired data.

Limitations. While the exact test-time task compositions are unseen during training, our method relies on transitions between task pairs in the robot dataset to learn transition actions. This limits the

ability to learn entirely new task sequences. We note that our method still generalizes well to new compositions when such transitions are present.

7 Acknowledgements

This work was partially funded by NSF RI (#2312956). Sanjiban Choudhury is supported in part by Google Google Faculty Research Award and OpenAI SuperAlignment Grant.

References

[1] K. Zakka, A. Zeng, P. R. Florence, J. Tompson, J. Bohg, and D. Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, 2021.

[2] S. Kumar, J. Zamora, N. Hansen, R. Jangir, and X. Wang. Graph inverse reinforcement learning from diverse videos. 2022.

[3] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. 2022.

[4] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song. XSkill: Cross embodiment skill discovery. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=8L6pHd9aS6w.

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

[6] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. 2023.

[7] 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. volume abs/2202.02005, 2022.

[8] V. Jain, M. Attarian, N. 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. volume abs/2403.12943, 2024.

[9] L. M. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. AVID: learning multi-stage tasks via pixel-level translation of human videos. In M. Toussaint, A. Bicchi, and T. Hermans, editors, *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020. doi:10.15607/RSS.2020.XVI.024. URL https://doi.org/10.15607/RSS.2020.XVI.024.

[10] E. Chane-Sane, C. Schmid, and I. Laptev. Learning video-conditioned policies for unseen manipulation tasks. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 909–916, 2023.

[11] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[12] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[13] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, 2022.

[14] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[15] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[16] N. S. Pollard, J. K. Hodgins, M. Riley, and C. G. Atkeson. Adapting human motion for the control of a humanoid robot. volume 2, pages 1390–1397 vol.2, 2002.

[17] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. volume 3, pages 3905–3910 vol.3, 2003.

[18] S. Kim, C. Kim, B.-J. You, and S.-R. Oh. Stable whole-body motion generation for humanoid robots to imitate human motions. pages 2518–2524, 2009.

[19] W. Suleiman, E. Yoshida, F. Kanehiro, J.-P. Laumond, and A. Monin. On human motion imitation by humanoid robot. pages 2697–2704, 2008.

[20] J. Koenemann, F. Burget, and M. Bennewitz. Real-time imitation of human whole-body motions by humanoids. pages 2806–2812, 2014.

[21] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine. Sfv: Reinforcement learning of physical skills from videos. volume 37, page 178, 2018.

[22] A. Handa, K. V. Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. D. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. pages 9164–9170, 2019.

[23] G. Garcia-Hernando, E. Johns, and T.-K. Kim. Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. pages 9561–9568, 2020.

[24] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, 2021.

[25] P. Mandikal and K. Grauman. DexVIP: Learning dexterous grasping with human hand pose priors from video. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=kSnfpHJBJOt.

[26] J. Ye, J. Wang, B. Huang, Y. Qin, and X. Wang. Learning continuous grasping function with a dexterous hand from human demonstrations. volume 8, pages 2882–2889, 2022.

[27] K. Shaw, S. Bahl, and D. Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, 2022.

[28] A. Sivakumar, K. Shaw, and D. Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. 2022.

[29] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. 2022.

[30] 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. *arXiv preprint arXiv:2407.04689*, 2024.

[31] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani. Track2act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation. *arXiv preprint arXiv:2405.01527*, 2024.

[32] Y. Zhu, A. Lim, P. Stone, and Y. Zhu. Vision-based manipulation from single human video with open-world object graphs. *arXiv preprint arXiv:2405.20321*, 2024.

[33] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song. Flow as the cross-domain manipulation interface. *arXiv preprint arXiv:2407.15208*, 2024.

[34] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. pages 2146–2153, 2017.

[35] M. Sieb, X. Zhou, A. Huang, O. Kroemer, and K. Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning*, 2019.

[36] K. Schmeckpeper, A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine, and C. Finn. Learning predictive models from observation and interaction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX*, pages 708–725. Springer, 2020.

[37] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. volume 40, pages 1419 – 1434, 2020.

[38] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. volume abs/2103.16817, 2021.

[39] H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg. Learning by watching: Physical imitation of manipulation skills from human videos. pages 7827–7834, 2021.

[40] R. Tian, C. Xu, M. Tomizuka, J. Malik, and A. V. Bajcsy. What matters to you? towards visual representation alignment for robot learning. volume abs/2310.07932, 2023.

[41] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *Neural Information Processing Systems*, 2017.

[42] S. Dasari and A. K. Gupta. Transformers for one-shot visual imitation. *ArXiv*, abs/2011.05970, 2020.

[43] Z. Mandi, F. Liu, K. Lee, and P. Abbeel. Towards more generalizable one-shot visual imitation learning. *2022 International Conference on Robotics and Automation (ICRA)*, pages 2434–2444, 2021.

[44] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[45] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine. Time-contrastive networks: Self-supervised learning from video. pages 1134–1141, 2017.

[46] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[47] G. Peyré and M. Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11: 355–607, 2018.

[48] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1801–1810, 2019.

[49] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *ArXiv*, abs/1910.11956, 2019.

[50] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

A Appendix

We report metrics associated with Fig 2 in Table 2 and run additional ablations to investigate the importance of segment lengths when performing retrievals. We additionally provide more details about our model architecture and implementation.

	Model →	BASELINE	OURS		GOLD STANDARD
	Metric ↓	XSKILL	RHYME-TCC	RHYME-OT	ORACLEPAIRING
EASY	Task Recall	82% (± 3.5)	75% (± 1.7)	96% (± 1.3)	92% (± 2.7)
	Task Imprecision	5% (± 1.7)	5% (± 1.7)	5% (± 1.0)	7% (± 2.3)
MED.	Task Recall	40% (± 2.7)	57% (± 4.2)	72% (± 5.8)	92% (± 2.9)
	Task Imprecision	2% (± 1.1)	51% (± 3.4)	4% (± 1.9)	0% (± .00)
HARD	Task Recall	1% (± 1.2)	34% (± 4.2)	53% (± 5.5)	92% (± 2.2)
	Task Imprecision	53% (± 5.2)	65% (± 4.5)	28% (± 5.1)	1% (± 1.1)

Table 1: We report exact metrics and standard errors for the bar plots in Fig 2.

We also report metrics for the bar graphs in Fig. 3 for real world experiments.

	Model → Metric ↓	XSKILL	RHYME
SEEN	Tasks Completed	52% (± 4.2)	58% (± 2.0)
	Tasks Attempted	75% (± 2.9)	83% (± 2.3)
UNSEEN	Tasks Completed	33% (± 1.5)	67% (± 0.0)
	Task Attempted	50% (± 1.5)	83% (± 0.0)

Table 2: We report exact metrics and standard errors for the bar plots in Fig 3.

A.1 Segment Length for Retrievals

In this section, we vary the segment length K from Algorithm 2 for RHyME-OT on the Sphere-Hard dataset and report Task Recall and Task Imprecision results in Table 3. Instead of a constant K for all videos, we deploy K as a function of the video length: that is, for a sequence of T images, $K = \frac{T}{K'}$ where K' is the total number of short horizon segments we split the video into. Increasing K' yields more demonstrator clip retrievals. We qualitatively find that the synthesized videos with lower K' often only consist of a strict subset of tasks from the original robot demonstration (thus yielding the highest task imprecision when $K' = 1$), while those from higher K' are more likely to accurately capture all the tasks in the robot demonstration. For our results throughout the paper, we simply select $K' = 2$, but we show that as long as the segment lengths are not too long (i.e. K' is not too small), Optimal Transport retrievals provide similar results regardless of segment length. While higher values of K' may lead to the construction of demonstrations with redundant tasks, the Transformer-based Skill Alignment Transformer described in Section A.3 is able to learn relations between the robot's current state and the tasks in the entire demonstration video to extract the most relevant task embedding for the policy.

	Video Splits K' →	RHYME-OT			
	Metric ↓	$K' = 1$	$K' = 2$	$K' = 3$	$K' = 4$
HARD	Task Recall	49% (± 2.9)	53% (± 3.6)	58% (± 4.5)	53% (± 2.6)
	Task Imprecision	28% (± 4.2)	21% (± 3.7)	12% (± 3.8)	14% (± 3.7)

Table 3: We report Task Recall and Task Imprecision rates when varying the number of short horizon segments K' we divide each long-horizon robot demonstration into to perform retrievals with RHyME-OT on the Sphere-Hard dataset

.

A.2 Representation Alignment

We utilize common representation alignment methods [4, 13, 44–46] to train our vision encoder ϕ that is used to produce image embeddings from videos, and describe them briefly.

Temporal Alignment. This alignment method leverages that image frames temporally close in a video are likely to be similar. We utilize the Time Contrastive Loss, used extensively in learning representations for robotics [13, 45]. For an image embedding at timestep t, z^t, we define a positive set $\mathbf{z}^+ = \{z^{t'}, |t' - t| \leq w\}$ and negative set $\mathbf{z}^- = \{z^{t'}, |t' - t| > w\}$ (where w is a hyperparameter specifying the positive window size). Intuitively, z^t should be closer to the positive set embeddings and further from the negative set embeddings measured by a similarity function $s(z^t, z^{t'})$. Using the contrastive INFO-NCE [46] learning objective, we can define $\mathcal{L}_{temp}(\phi) = -\sum_{z^+ \in \mathbf{z}^+} \frac{\exp(s(z^t, z^+)/\tau)}{\exp(s(z^t, z^+)/\tau) + \sum_{z^- \in \mathbf{z}^-} \exp(s(z^t, z^-)/\tau)}$, with temperature parameter τ.

Visual Alignment. We utilize SwAV [44], a self-supervised learning algorithm to cluster images based on visual features. The algorithm learns a set of K learnable *prototype* vectors, $\mathbf{c} = \{c^1, c^2, \ldots, c^K\}$ that are matched with individual images. For training these representations, an image is first augmented in two different ways producing different embeddings z^1 and z^2. Then, each embedding's soft assignment to the K prototypes is computed using the Sinkhorn-Knopp algorithm to produce *codes*, q^1 and q^2, K dimensional assignment probabilities to each prototype. The SwAV loss function leverages that both embeddings, only differentiated by augmentations, should map to the same codes. The loss function $\mathcal{L}_{vis}(\phi, \mathbf{c}) = l_{swav}(z^1, q^2) + l_{swav}(z^2, q^1)$, updates both the video encoder as well as the prototype set. We refer the reader to the original paper for more details. As used by XSkill [4], we map demonstrator and robot images to the same set of prototypes using this loss function, where batches only consist of images from one embodiment. In our experiments, we show that this way of learning representations maps robot and demonstrator tasks to the same embedding space, *but only when their object movements are similar*.

A.3 Model Architecture Details

Video Encoding

The video encoder ϕ is modeled by a CNN-based vision backbone and transformer encoder. ϕ individually extracts embeddings for each frame in a demonstration video, where in practice each frame is represented by a 1-timestep sliding window of 8 neighboring images passed into the network to produce a 256-dimensional flattened vision feature vector. At train time, we perform random image augmentations to compute a self-supervised loss [44] and use $K = 128$ learnable prototype vectors implemented as a linear layer with no bias as described in Section A.2.

Policy Structure

The policy π consists of two components: a Skill Alignment Transformer (SAT) (introduced by [4]) to model $p(z_{t+1}|s_t, \mathbf{z_R})$, which allows the policy to extract z_{t+1}, the next task embedding induced by progressing in the tasks, based on the robot state s_t and prompt video $\mathbf{z_R}$. The second component is a task-conditioned policy $\pi(a_t|s_t, z_{t+1})$, which essentially serves as an inverse dynamics model to decode the robot's current state s_t and the next task embedding z_{t+1} (predicted by SAT) into the correct action. The policy is modeled by Diffusion Policy [50].

Hyperparameters

We borrow hyperparameters from prior works [4, 44] for Temporal Alignment and Visual Alignment (Section A.2), and present hyperparameters for our retrieval algorithm IMAGINE-DEMO (Alg. 2), as well as the optional hyperparameters for fine-tuning the visual representation space (Section 4.2).

IMAGINE-DEMO Hyperparam. ↓	Value
OT similarity temperature	0.05
TCC similarity temperature	0.1
Policy π Hyperparam. [50]↓	**Value**
Observation Horizon	2
Action Horizon	2
Action Pred. Horizon	16
State - Vision Feature Dim.	64
State - Proprio. Feature Dim.	9
Action Dim.	9
Batch Size	128
Training iteration	200
Learning rate	1e-4
Weight decay	1e-6
Optimizer	ADAM

Video Encoder ϕ Hyperparam. [4] ↓	Value
Video Clip length l	8
Sampling Frames T	100
Sinkhorn iterations	3
Sinkhorn epsilon	0.03
Prototype loss coef	0.5
Prototype loss temperature	0.1
TCN loss coef	1
TCN positive window w_p	4
TCN negative window w_n	12
TCN negative samples	16
TCN temperature τ_{tcn}	0.1
Batch Size	28
Training iteration	100
Learning rate	1e-4
Optimizer	ADAM