REDUNDANCY-AWARE VIDEO SEGMENT RETRIEVAL FOR FEW-SHOT ROBOT IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented imitation learning improves few-shot robot policy learning by augmenting limited target demonstrations with relevant segments from a large prior dataset. Existing methods, however, often rely on framewise image similarity, which can retrieve visually similar segments despite differences in the underlying manipulation behavior. Moreover, independently ranking candidates with a Top-K rule can fill the retrieval budget with redundant segments that exhibit highly similar spatiotemporal manipulation patterns, limiting the benefit of data augmentation. We propose RVSR, a redundancy-aware video segment retrieval framework that jointly considers target relevance and redundancy among selected segments. RVSR partitions robot demonstrations into temporally coherent segments using a frozen pretrained video representation and derives two role-specific features: a semantic feature for target relevance and a temporal-pattern feature for redundancy estimation. It then sequentially selects relevant segments while suppressing manipulation patterns already represented in the retrieved data. RVSR achieves 66.4% success on LIBERO-10, outperforming the strongest retrieval baseline at 63.5%, and obtains the highest average performance across three real-robot tasks. Ablations further show that temporally structured video representations and redundancy-aware selection are most effective when combined to improve few-shot policy learning. The source code is included in the supplementary materials.

1 INTRODUCTION

Imitation learning has become a powerful way to acquire manipulation policies directly from expert demonstrations (Chi et al., 2023; Zhao et al., 2023; Ravichandar et al., 2020), but collecting sufficient demonstrations for each new task requires substantial human effort (Memmel et al., 2025). Training a generalist policy on demonstrations pooled across many tasks can reduce this burden (Brohan et al., 2022; Open X-Embodiment Collaboration, 2023), but indiscriminately mixing unrelated data often causes negative transfer and leaves task-specific performance below that of specialist policies (Memmel et al., 2025; Kumar et al., 2025). Retrieval-augmented imitation learning offers a middle ground: given only a few target demonstrations, it retrieves related data from a large prior dataset and trains on both (Memmel et al., 2025; Du et al., 2023; Nasiriany et al., 2023). Because the retrieved data directly become training data, retrieval quality depends not only on how well each segment matches the target, but also on whether the selected segments collectively provide useful and non-redundant supervision.

Existing retrieval methods typically estimate the relevance of candidate segments to the target using framewise visual features, often extracted with a frozen image encoder such as DINOv2 (Oquab et al., 2024). They then rank candidates according to this *image-centric* similarity (Memmel et al., 2025). COLLAGE combines multiple modalities, but its visual matching component still relies on framewise image features (Kumar et al., 2025). As a result, retrieval can be dominated by static visual appearance rather than by the underlying manipulation behavior. For example, a target segment in which the robot grasps object A may retrieve segments involving object B—or even a different action—when the background, surrounding objects, and scene layout are visually similar. Conversely, behaviorally relevant segments may be overlooked when the same manipulation is performed with different object appearances, textures, or backgrounds. Figure 1 illustrates this failure mode on a real-robot task. Prior methods partially incorporate temporal structure through

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

subsequence dynamic time warping (S-DTW) (Memmel et al., 2025; Kumar et al., 2025), but their visual matching components still operate over sequences of framewise image features. This motivates using a pretrained video representation that encodes temporally contextualized, action-relevant information and can better match manipulation behaviors across visual variations.

A second limitation arises from independently ranking candidates and retaining the Top-K, as in STRAP (Memmel et al., 2025). Large robot datasets often contain repeated executions of similar successful behaviors, causing nearest-neighbor retrieval to concentrate on dense regions of the demonstration distribution (Xie et al., 2025). Consequently, a limited retrieval budget may be filled with segments exhibiting highly similar spatiotemporal manipulation patterns. Such repetition provides little additional coverage beyond the few target demonstrations and therefore limits the benefit of data augmentation. Simply increasing K does not directly address this redundancy and may eventually introduce less relevant candidates from lower ranks. Effective retrieval should therefore consider not only the relevance of each candidate to the target behavior, but also its redundancy with segments already selected.

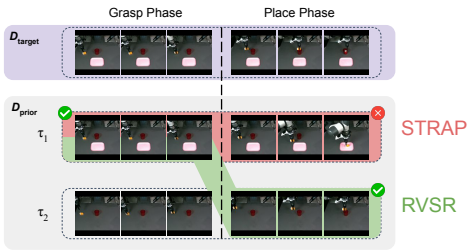


Figure 1: Retrieval comparison for the Put the Pencil Eraser in Red Cup task. Both STRAP and RVSR retrieve a valid grasp segment. During placement, STRAP (image-centric) retrieves a visually similar segment from a task with a different placement destination, whereas RVSR (video-based) retrieves a placement segment from the target task.

Our main contributions are as follows:

- **Temporally Structured Video Segment Representation.** We encode each segment with a frozen pretrained video encoder and derive two role-specific features: a semantic feature for measuring target relevance and a temporal-pattern feature that combines semantic information with low-frequency changes in the video features to estimate redundancy among candidate segments.
- **Redundancy-Aware Sequential Retrieval.** We move beyond independent Top-K ranking by sequentially selecting segments from a pre-filtered relevant set while penalizing spatiotemporal manipulation patterns already covered by the retrieved set.
- **Simulation and Real-Robot Validation.** RVSR achieves an average success rate of 66.4% on LIBERO-10, outperforming the retrieval baselines COLLAGE (63.5%) and STRAP (61.7%). It also achieves the highest average performance across three real-robot manipulation tasks using only three target demonstrations per task.

2 Related Work

2.1 Retrieval-Augmented Imitation Learning

Retrieval-augmented imitation learning methods differ in how they represent and select relevant prior data. Some learn retrieval-specific representations, including SAILOR's skill latents (Nasiriany et al., 2023), Behavior Retrieval's state–action VAE space (Du et al., 2023), and FlowRetrieval's optical-flow representation (Lin et al., 2025). Others use pretrained representations: STRAP aligns pretrained image features with S-DTW (Memmel et al., 2025), while COLLAGE adaptively weights subsets retrieved using appearance, shape, language, and optical flow (Kumar et al., 2025). IWR uses density-ratio importance weighting (Xie et al., 2025), and ROSER retrieves task-centric windows in a learned robot-state metric space (Rahman et al., 2026). Overall, prior methods emphasize relevance estimation, whereas we jointly consider temporally contextualized relevance and set-level redundancy.

2.2 Video Representations for Robot Learning

Pretrained image encoders such as R3M and DINOv2 are widely used in robot learning (Nair et al., 2023; Oquab et al., 2024). However, independently encoding individual frames does not directly capture the temporal evolution within a manipulation segment. STRAP addresses temporal correspondence by aligning sequences of frozen frame features with S-DTW (Memmel et al., 2025), but temporal structure is still incorporated through post hoc sequence matching rather than jointly encoded at the segment level. Meanwhile, video representation learning has advanced substantially. VideoMAE learns through masked pixel reconstruction (Tong et al., 2022), while latent-prediction methods for images and videos predict masked representations in feature space (Assran et al., 2023; Abdelfattah et al., 2025; Assran et al., 2025). V-JEPA 2 applies this paradigm to masked video and supports frozen downstream use (Assran et al., 2025). Prior analyses indicate that V-JEPA 2 captures task-relevant temporal dynamics (Miao et al., 2026), while temporal video pretraining induces action-relevant structure (Yeom et al., 2026). To leverage these properties, we use frozen V-JEPA 2 features for segment-level relevance and redundancy estimation.

2.3 Diversity-Aware Retrieval

Diversity-aware methods have been widely studied in information retrieval, summarization, and subset selection. Specifically, maximal marginal relevance (MMR) balances query relevance and novelty in document reranking and summarization (Carbonell & Goldstein, 1998), while related approaches include determinantal point processes for diverse subset selection (Kulesza & Taskar, 2012) and redundancy-aware code retrieval (Chen et al., 2025). In imitation learning, however, related data-selection concerns have mainly been addressed through offline dataset refinement, such as deduplication and suboptimal-sample removal (Zhang et al., 2025), or sample quality and influence scoring (Hejna et al., 2025; Dass et al., 2026). Diversity-aware retrieval remains less explored in retrieval-augmented imitation learning: STRAP retains Top-K matches, while COLLAGE samples from weighted relevance-based subsets (Memmel et al., 2025; Kumar et al., 2025). Such relevance-driven selection may fill a limited retrieval budget with non-overlapping segments that exhibit similar temporal patterns, thereby narrowing data coverage. To address this, we use MMR-based sequential selection to retrieve relevant yet non-redundant manipulation segments.

3 Preliminaries

3.1 Problem Definition

In few-shot manipulation, we are given a small dataset $\mathcal{D}_{\text{target}}$ containing n demonstrations of a target task specified by a language instruction ℓ. Our goal is to learn a policy $\pi_\theta(a \mid o, \ell)$, parameterized by θ, that predicts an action a given an observation o and the instruction ℓ. Because these demonstrations cover only a narrow range of observations and executions, a policy trained solely on $\mathcal{D}_{\text{target}}$ by behavioral cloning may generalize poorly to variations in object poses, scenes, and manipulation trajectories. Retrieval-augmented imitation learning addresses this limitation by retrieving relevant prior data $\mathcal{D}_{\text{retrieved}}$ from a large prior demonstration dataset $\mathcal{D}_{\text{prior}}$ and training the policy jointly on the target and retrieved data (Memmel et al., 2025; Du et al., 2023; Nasiriany et al., 2023). In our setting, $\mathcal{D}_{\text{prior}}$ contains previously collected demonstrations from multiple tasks. Following STRAP, retrieval operates at the sub-trajectory level: a *segment* is a contiguous span of frames within a trajectory (Memmel et al., 2025). Each trajectory is associated with a task instruction, but no annotation specifies which prior segments are relevant to a given target task. We train the policy by behavioral cloning, minimizing the negative log-likelihood of demonstrated actions:

$$\min_\theta \ \mathbb{E}_{(o,a,\ell)\sim\mathcal{D}_{\text{target}}\cup\mathcal{D}_{\text{retrieved}}}\big[-\log \pi_\theta(a \mid o, \ell)\big]. \tag{1}$$

The union denotes the combined training data, not a uniform sampling assumption; the sampling scheme is specified in Section 4.4.

3.2 Maximal Marginal Relevance

MMR, the redundancy-aware reranking method discussed in Section 2.3, is a greedy procedure that balances relevance and redundancy (Carbonell & Goldstein, 1998). Given a query q and a candidate

set $\mathcal{C}$, let S be a nonempty proper subset of $\mathcal{C}$ containing the previously selected items. MMR selects the next item from the remaining candidates $\mathcal{C} \setminus S$. For a candidate $c \in \mathcal{C} \setminus S$, it rewards query relevance $\mathrm{Sim}_1(c, q)$ and penalizes the maximum similarity $\mathrm{Sim}_2(c, c')$ to any previously selected item $c' \in S$:

$$\arg\max_{c \in \mathcal{C} \setminus S} \Big[\lambda \, \mathrm{Sim}_1(c, q) - (1 - \lambda) \max_{c' \in S} \mathrm{Sim}_2(c, c') \Big], \tag{2}$$

where $\lambda \in [0, 1]$ controls the trade-off between relevance and redundancy.

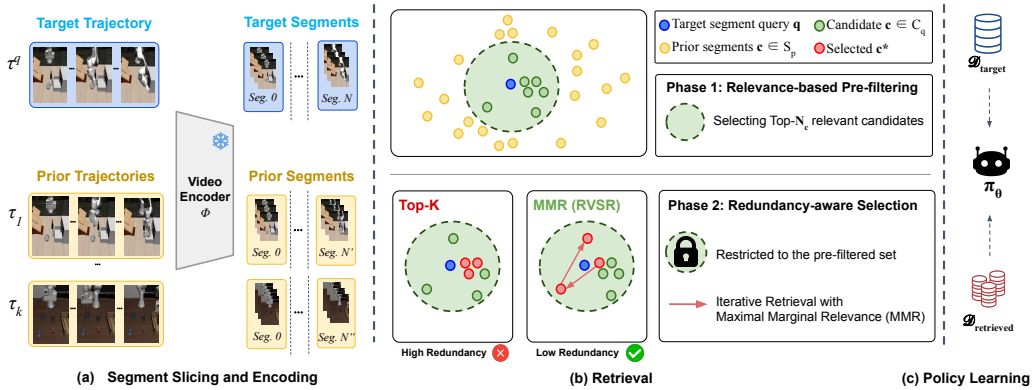


Figure 2: Overview of RVSR. (a) Target and prior trajectories are segmented and encoded using a frozen video encoder. (b) Relevance-based pre-filtering is followed by redundancy-aware selection; the schematic contrasts independent Top-K selection with MMR. (c) The policy is trained jointly on target and retrieved data.

4 Method

4.1 Overview

Given a few target demonstrations $\mathcal{D}_{\text{target}}$ and a large prior pool $\mathcal{D}_{\text{prior}}$, RVSR returns a collection of prior sub-trajectories to train on. Our key design is to use video segment features to assess both target relevance and redundancy with already-selected segments. Figure 2 provides an overview of RVSR. First, trajectories are segmented and encoded with a frozen video model, yielding role-specific representations for relevance and redundancy (Section 4.2). Second, candidates are scored and selected sequentially to preserve relevance while suppressing redundancy (Section 4.3). Finally, the policy is trained jointly on the target and retrieved data (Section 4.4). Algorithms 1 and 2 in Appendix A summarize segment encoding and retrieval, respectively.

4.2 Segment Encoding and Features

Segment slicing and encoding. Prior works (Memmel et al., 2025; Kumar et al., 2025) partition target demonstrations into query segments based on end-effector (EEF) pauses. Such segments can be behaviorally uninformative or near-static. Instead, we encode each trajectory $\tau \in \mathcal{D}_{\text{prior}} \cup \mathcal{D}_{\text{target}}$ with a frozen V-JEPA 2 encoder ϕ (Assran et al., 2025). The encoder produces spatiotemporal tokens. Averaging over spatial tokens at each temporal position yields $Z_\tau \in \mathbb{R}^{T \times d}$, where T is the number of temporal positions and d is the feature dimension. We apply kernel temporal segmentation (KTS) to Z_τ, minimizing within-segment representation variation while penalizing over-segmentation (Potapov et al., 2014; Afham et al., 2023). Compared with EEF-pause slicing, this segmentation empirically produces fewer near-static segments (Section 5).

Because ϕ uses bidirectional attention, features in Z_τ can incorporate context from outside a segment's boundaries. We therefore slice the raw frames at the KTS boundaries and re-encode each segment independently. Applying the same spatial averaging yields $\Phi_s \in \mathbb{R}^{T_s \times d}$ for each segment s, where T_s is the number of temporal positions within that segment. The Appendix compares these features with those sliced directly from Z_τ. We denote the resulting target and prior segment sets by $\mathcal{S}_{\text{target}}$ and $\mathcal{S}_{\text{prior}}$.

Semantic and temporal-pattern features. We derive a semantic feature for relevance estimation and a temporal-pattern feature for redundancy estimation. Let $\Phi_{s,t} \in \mathbb{R}^d$ denote the feature at temporal position t in segment s. The semantic feature is the L2-normalized temporal mean:

$$\mu_s = \frac{1}{T_s} \sum_{t=1}^{T_s} \Phi_{s,t}, \qquad z_{\text{sem}}(s) = \frac{\mu_s}{\|\mu_s\|_2}. \tag{3}$$

We use z_{sem} to measure target relevance in both retrieval phases.

To represent temporal patterns, we apply a discrete cosine transform (DCT) to each feature channel along time (Ahmed et al., 1974). The DCT expresses temporal variation as cosine components ordered from low to high frequency. The zero-frequency (DC) component is proportional to the temporal mean (Qin et al., 2021), which is already represented by the semantic feature. We therefore retain the first D non-DC components, $u_1(s), \ldots, u_D(s) \in \mathbb{R}^d$, to capture slower changes in the video features over the segment. We concatenate and L2-normalize these components to obtain $\bar{u}_{1:D}(s)$, then combine it with the semantic feature to form the temporal-pattern feature:

$$z_{\text{tp}}(s) = \left[\sqrt{1-\beta}\, z_{\text{sem}}(s); \sqrt{\beta}\, \bar{u}_{1:D}(s) \right], \tag{4}$$

where semicolons denote concatenation, and $\beta \in [0, 1]$ controls the contribution of the DCT block. The square-root weights give the semantic and DCT-block similarity weights $1 - \beta$ and β, respectively, in the resulting cosine similarity. We use $D = 3$; the Appendix provides sensitivity analyses and numerical normalization details.

4.3 Retrieval Metrics and Selection

The features defined above apply to both target and prior segments. During retrieval, we denote a target query segment by q and a prior candidate segment by c. For each query, we first retain N_c candidates by relevance, then select K of them using MMR (Section 3.2). We use the semantic feature for relevance (Sim_1) and the temporal-pattern feature for redundancy (Sim_2).

Phase 1: Relevance-based pre-filtering. For any two segments s and s', we measure relevance using cosine similarity between their semantic features:

$$\text{Sim}_1(s, s') = \cos\big(z_{\text{sem}}(s), z_{\text{sem}}(s')\big). \tag{5}$$

For each target query $q \in \mathcal{S}_{\text{target}}$, we evaluate $\text{Sim}_1(c, q)$ for every prior segment $c \in \mathcal{S}_{\text{prior}}$ and retain the N_c most relevant candidates as the pre-filtered set $\mathcal{C}_q$. This removes low-relevance candidates and restricts Phase 2 to the most relevant portion of the prior pool.

Phase 2: Redundancy-aware selection. To assess redundancy between a candidate and previously selected segments, we use cosine similarity between their temporal-pattern features:

$$\text{Sim}_2(s, s') = \cos\big(z_{\text{tp}}(s), z_{\text{tp}}(s')\big). \tag{6}$$

Because z_{tp} combines semantic and DCT features (Eq. (4)), this score accounts for both segment-level semantics and temporal patterns.

To encourage task-level consistency, we also compare the instructions associated with the candidate and query. Let ℓ_c and ℓ_q be the task instructions of their source trajectories. Using a frozen CLIP text encoder ψ (Radford et al., 2021), we compute $\text{lc}(c, q) = \cos\big(\psi(\ell_c), \psi(\ell_q)\big)$.

For sequential selection, let $\mathcal{S}_q \subseteq \mathcal{C}_q$ denote the segments selected so far for query q. We initialize $\mathcal{S}_q$ with the candidate that maximizes $\text{Sim}_1(c, q)$. We then choose each of the remaining $K - 1$ segments from $\mathcal{C}_q \setminus \mathcal{S}_q$, comparing each candidate c with previously selected segments $c' \in \mathcal{S}_q$:

$$c^\star = \arg \max_{c \in \mathcal{C}_q \setminus \mathcal{S}_q} \Big[\lambda\, \text{Sim}_1(c, q) - (1 - \lambda) \max_{c' \in \mathcal{S}_q} \text{Sim}_2(c, c') + \nu\, \text{lc}(c, q) \Big]. \tag{7}$$

The three terms reward target relevance, penalize similarity to the most similar selected segment, and encourage language consistency, respectively. Here, $\lambda \in [0, 1]$ balances relevance and redundancy, and $\nu \geq 0$ weights language consistency. After each selection, we add $c^\star$ to $\mathcal{S}_q$ and repeat until $|\mathcal{S}_q| = K$. Redundancy suppression is therefore query-local: selections made for other target queries do not affect $\mathcal{S}_q$. We concatenate the query-wise selected sets to form $\mathcal{D}_{\text{retrieved}}$, retaining repeated selections when the same prior segment is retrieved for multiple queries.

Design choices in selection. Because task instructions are provided at the trajectory level rather than at the segment level, language similarity is coarse for fine-grained relevance filtering in Phase 1. We therefore use semantic similarity to construct the pre-filtered set. In Phase 2, however, redundancy-aware selection may favor less redundant candidates whose task context is less aligned with the target task, even though they remain visually similar within the pre-filtered set. The language score consequently acts as a task-level consistency regularizer, discouraging such drift while preserving diversity within the pre-filtered set.

4.4 Policy Learning

We train the policy on the target and retrieved data using the BC objective in Eq. (1). For RVSR, each training sample is drawn from $\mathcal{D}_{\text{target}}$ or $\mathcal{D}_{\text{retrieved}}$ with equal probability. We use the same language-conditioned transformer, GMM action head, and multi-step action prediction as STRAP and COLLAGE (Memmel et al., 2025; Kumar et al., 2025) for every method in Section 5.

5 Experiments

We evaluate RVSR on LIBERO-10 and three real-robot manipulation tasks. We compare against baselines with and without retrieval, analyze the retrieved segments, and use ablations to examine the contributions of the video representation, segmentation, and redundancy-aware selection.

Table 1: LIBERO-10 success rates (%). STRAP and COLLAGE use original (unmarked) or RVSR-matched (*) retrieval budgets and sampling. Bold and underlined values indicate the best and second-best results, including ties.

Method	Book	Bowl	Cheese-B	Moka	Mug-Mic	Mug-Mug	Mug-P	Stove	Soup-C	Soup-S	**Avg.**
BC	56.7	88.7	46.0	12.7	38.7	40.0	14.7	81.3	40.0	30.7	44.9 ± 2.2
MT	63.3	94.7	34.7	15.3	52.0	61.3	26.0	90.0	37.3	40.7	51.5 ± 7.8
STRAP	76.7	<u>98.7</u>	47.3	22.0	43.3	**67.3**	24.0	87.3	47.3	65.3	57.9 ± 6.6
STRAP*	74.7	**99.3**	52.7	<u>30.0</u>	52.7	63.3	<u>26.7</u>	88.7	60.7	<u>68.7</u>	61.7 ± 1.0
COLLAGE	76.7	<u>98.7</u>	44.0	**34.0**	46.7	59.3	**30.7**	88.7	<u>67.3</u>	63.3	60.9 ± 2.4
COLLAGE*	**91.3**	96.0	<u>59.3</u>	22.7	<u>53.3</u>	64.0	**30.7**	**96.7**	52.7	68.0	<u>63.5</u> ± 2.1
RVSR	<u>90.0</u>	98.0	**61.3**	21.3	**64.7**	<u>66.7</u>	<u>26.7</u>	<u>92.7</u>	**71.3**	**71.3**	$\mathbf{66.4} \pm 1.3$

5.1 Experimental Setup

Tasks and datasets. In simulation, we use the 10 long-horizon tasks of LIBERO-10 (Liu et al., 2023). Following STRAP (Memmel et al., 2025), we randomly select 5 demonstrations per task as $\mathcal{D}_{\text{target}}$ and use the 4,500 demonstrations in LIBERO-90 (90 tasks with 50 demonstrations each) as $\mathcal{D}_{\text{prior}}$. For the real-robot experiments, we use three pick-and-place tasks (BluePen, Adapter, and Eraser) on a Rainbow Robotics RB3 arm. Each target task has 3 demonstrations, and the prior dataset contains 100 demonstrations, with 10 collected for each of 10 tasks. Some prior tasks involve the same objects as the target tasks. Appendices B.1 and H detail the simulation and real-robot tasks and datasets, respectively.

Policy training. All methods use the same target demonstrations and policy architecture (Section 4.4). In simulation, we follow STRAP's policy-training setup (Memmel et al., 2025): a transformer with a 5-step observation history and a 5-mode GMM action head, ResNet-18 encoders trained from scratch for agent-view and eye-in-hand images, batch size 32, and 200 gradient steps per epoch for 300 epochs.

Evaluation. In simulation, we use three policy-training seeds and evaluate each checkpoint with 50 rollouts per task (Memmel et al., 2025; Kumar et al., 2025). Checkpoints are evaluated every 50 epochs, and the best-performing checkpoint is selected separately for each task and seed. We average

success rates over the 10 tasks within each seed, then report the mean and standard deviation across seeds. All methods use the same evaluation protocol and a maximum rollout horizon of 520 steps; the choice of horizon is discussed in the Appendix. The simulation environments randomize target object poses. For the real robot, we randomize object poses within a 20×20 cm grid, following STRAP (Memmel et al., 2025), and report task-completion success rates over 10 trials per task and their mean across tasks.

5.2 Baselines and Comparison Settings

We reproduce all baselines under the common protocol above. Retrieval comparisons focus on methods that select an explicit subset of prior data. *Without retrieval*: **BC** trains the policy on $\mathcal{D}_{\text{target}}$ only; **MT** trains a multitask policy on $\mathcal{D}_{\text{target}} \cup \mathcal{D}_{\text{prior}}$. *With retrieval*, each method selects $\mathcal{D}_{\text{retrieved}} \subset \mathcal{D}_{\text{prior}}$ and trains on it together with $\mathcal{D}_{\text{target}}$; **STRAP** retrieves sub-trajectories with frozen DINOv2 features and S-DTW (Memmel et al., 2025); **COLLAGE** combines subsets retrieved using visual, motion, shape, and language features through adaptive modality sampling (Kumar et al., 2025); **RVSR** is the proposed method described in Section 4.

Comparison settings. Table 1 reports STRAP and COLLAGE under both original and matched settings. The matched variants, denoted STRAP* and COLLAGE* in the tables, align retrieval budgets and sampling with RVSR for a more controlled comparison. The matched budget is $K = 5$ segments per target query, applied separately to COLLAGE's visual, motion, and shape retrievers; its language subset approximately matches the visual subset in frame count. Matched sampling assigns equal probability to target and prior data: the full prior dataset for MT and retrieved data for retrieval methods. COLLAGE retains adaptive modality weighting within the retrieved group. Appendix B.5 details the settings and COLLAGE modality-weight estimation. RVSR uses a fixed retrieval configuration across tasks, seeds, and environments, with full settings and sensitivity analyses in the Appendix.

5.3 Simulation Results

RVSR achieves the highest mean success rate of 66.4% on LIBERO-10, outperforming both the original and matched baseline variants (Table 1). Both retrieval baselines improve under matched settings, but RVSR maintains its advantage.

Per-task analysis. RVSR achieves the highest success rate on four of the ten tasks. On Cheese-B and Mug-Mic, it exceeds BC by 15.3 and 26.0 percentage points, respectively. Its advantage over the matched retrieval baselines is particularly pronounced on Mug-Mic, where RVSR achieves 64.7% compared with 53.3% for COLLAGE and 52.7% for STRAP. Some parts of the target behavior lack exact matches in the prior dataset, motivating video-based retrieval of behaviorally similar segments from other tasks.

Retrieval-distribution analysis. Both COLLAGE and RVSR retrieve cross-scene data, whereas STRAP concentrates on the same-scene *close microwave* task (Figure 3). Compared with COLLAGE, RVSR distributes retrieval across sources more evenly in the earlier target-progress bins, while emphasizing the *close microwave* task in the final bin. This pattern is consistent with combining cross-scene retrieval with redundancy-aware selection.

5.4 Real-Robot Results

Table 2: Real-robot success rates (%).

Method	BluePen	Adapter	Eraser	**Avg.**
BC	20	30	10	20.0
MT	<u>80</u>	30	50	53.3
STRAP*	<u>80</u>	<u>40</u>	<u>60</u>	<u>60.0</u>
RVSR	**100**	**80**	**70**	**83.3**

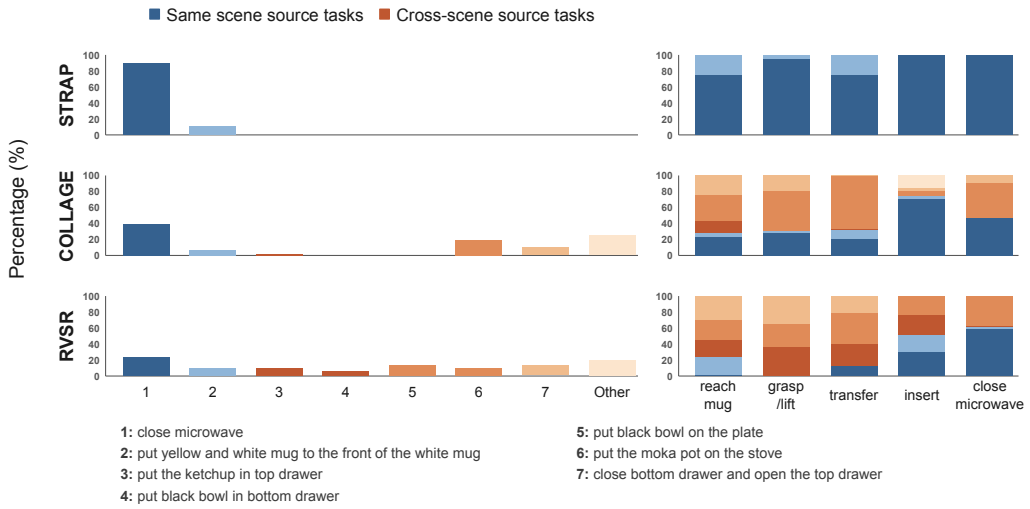


Figure 3: Retrieval distributions for Mug-Mic under matched settings. Left: overall source-task composition. Right: source composition across target progress. Aggregation details are provided in Appendix G.

We compare RVSR with BC, MT, and STRAP under matched settings on three real-robot tasks. COLLAGE is limited to simulation because our reproduction relies on released LIBERO embeddings and lacks a real-robot feature-extraction pipeline. RVSR achieves the highest success rate on all three tasks, averaging 83.3% compared with 60.0% for STRAP, the strongest baseline (Table 2).

Table 3: Source-task shares (%) among retrieved segments. Pick/Place identify source objects/destinations; bold marks objects or destination categories shared with the target.

Method	Pick	Place	Ratio
Target: Blue pen → white basket			
STRAP*	**Blue pen**	**white basket**	100%
RVSR	**Blue pen**	**white basket**	100%
Target: USB adapter → green plate			
STRAP*	Red pen	**green plate**	90.0%
	USB adapter	pink **plate**	10.0%
RVSR	**USB adapter**	pink **plate**	83.3%
	Red pen	**green plate**	16.7%
Target: Pencil eraser → red cup			
STRAP*	**Pencil eraser**	pink plate	100%
RVSR	**Pencil eraser**	**red cup**	46.7%
	Pencil eraser	blue **cup**	31.7%
	Pencil eraser	pink plate	21.6%

Retrieval analysis. The largest success-rate gain over STRAP occurs on Adapter (80% vs. 40%), whose target object–destination pair is absent from the prior dataset. Table 3 shows a corresponding difference in the retrieved data. STRAP retrieves mostly red-pen segments with a green plate, matching the target destination but involving a different object. In contrast, 83.3% of the segments retrieved by RVSR come from the USB-adapter–pink-plate task. These match both the target object and the destination type (a plate), with a different plate color.

For Eraser, RVSR retrieves segments with cup destinations, including the target red cup, whereas STRAP retrieves only segments with a plate destination. For BluePen, both methods retrieve from the same source task despite different success rates.

5.5 Ablation Studies and Analysis

Table 4: LIBERO-10 ablations. Avg. is the mean success (%); Δ is the change from full RVSR in percentage points. DINO/JEPA denote DINOv2/V-JEPA 2. TP and Lang. indicate the DCT block and language-consistency term; dashes mean disabled or unused. KTS^D uses DINOv2 features for segmentation; EEF denotes EEF-pause segmentation.

Encoder	Seg.	TP	Lang.	Select.	Avg. (Δ)
DINO	KTS^D	✓	✓	MMR	57.0 (-9.4)
JEPA	KTS	-	-	Top-K	56.5 (-9.9)
JEPA	KTS	✓	-	MMR	61.7 (-4.7)
JEPA	KTS	-	✓	MMR	63.0 (-3.4)
JEPA	EEF	✓	✓	MMR	60.5 (-5.9)
JEPA	KTS	✓	✓	MMR	**66.4**

Component ablations. The largest performance drop in Table 4 occurs when replacing redundancy-aware Phase 2 with relevance-only Top-K selection, supporting the benefit of the complete second-phase selection rule. Replacing the V-JEPA 2 pipeline with DINOv2 also substantially reduces performance, supporting the contribution of the video representation and its corresponding segment boundaries. KTS segmentation outperforms EEF-pause segmentation under the same remaining configuration. Removing the language-consistency term or the DCT block reduces success from 66.4% to 61.7% and 63.0%, respectively, supporting the contribution of both components to policy performance.

Diagnostic analysis. These operational diagnostics assess whether the observed trends are consistent with our design goals rather than serving as ground-truth measures. KTS produces fewer **low-activity segments** (3.2%) than EEF-pause segmentation (15.5%) under the near-static criterion, although this does not evaluate sub-task boundary accuracy.

Using the same candidate pool and retrieval budget, the full RVSR selection rule yields a lower within-query Action-DTW-defined near-duplicate rate than relevance-only Top-K selection (23.9% vs. 43.1%), consistent with lower action-sequence redundancy under this proxy. Because the selector also includes language consistency, this comparison evaluates the complete selection rule rather than isolating the temporal-pattern term. Full definitions and additional analyses are provided in the Appendix.

6 Conclusion

We presented RVSR, a redundancy-aware video segment retrieval framework for few-shot imitation learning. Using a frozen pretrained video encoder, RVSR derives a semantic feature for target relevance. For redundancy estimation, we construct a temporal-pattern feature that combines semantic information with low-frequency changes in the video features. These two role-specific features guide sequential selection of relevant segments while suppressing redundancy. RVSR achieves 66.4% on LIBERO-10 and the highest average performance among the evaluated methods across three real-robot tasks. Ablations show that combining the video representation with redundancy-aware selection yields the best performance among the evaluated configurations.

Limitations and future work. Our real-robot evaluation is limited to a single platform and a relatively small prior dataset. Future work should evaluate RVSR with larger datasets, more diverse tasks, and cross-embodiment retrieval. As with any retrieval method, RVSR is also limited by the behavioral coverage of the prior demonstration dataset.

Additionally, although RVSR uses a single fixed retrieval configuration across all tasks and environments in our experiments, the optimal relevance–redundancy balance and temporal-pattern weight may vary with the characteristics of the prior dataset and pre-filtered set. Future work could investigate data-driven strategies for adapting these parameters accordingly.

AI USE STATEMENT

We used Claude and Codex to assist with reviewing baseline implementations and writing code for RVSR, experiment orchestration, and auxiliary retrieval diagnostics. These tools also assisted with literature search and review, manuscript drafting and editing, and feedback on methodology, experimental design, and result interpretation. We additionally used them for table preparation, data extraction for Figure 3, and preparation of Figures 4–6. The authors reviewed the AI-assisted text, code, and analyses, reviewed and finalized the figures and tables, and take responsibility for the final manuscript, code, and reported results.

REPRODUCIBILITY STATEMENT

Section 4 describes RVSR, and Section 5 specifies the experimental protocol. Appendix A provides pseudocode for segment encoding and retrieval, while Appendices B and C document dataset construction, preprocessing, hyperparameters, and training and evaluation settings. Appendices H and I describe the real-robot setup and computational infrastructure. Source code for the simulation experiments is provided in the supplementary materials. The simulation experiments use publicly available LIBERO demonstrations (Liu et al., 2023), with data sources and baseline repositories documented in Appendix B.

REFERENCES

Mohamed Abdelfattah, Kaouther Messaoud, and Alexandre Alahi. OSKAR: Omnimodal self-supervised knowledge abstraction and representation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Mohamed Afham, Satya Narayan Shukla, Omid Poursaeed, Pengchuan Zhang, Ashish Shah, and Sernam Lim. Revisiting kernel temporal segmentation as an adaptive tokenizer for long-form video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 1189–1194, 2023.

Nasir Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. arXiv preprint arXiv:2506.09985, 2025.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale. arXiv preprint arXiv:2212.06817, 2022.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336, 1998.

Xiaohan Chen, Zhongying Pan, Quan Feng, Yu Tian, Shuqun Yang, Mengru Wang, Lina Gong, Yuxia Geng, Piji Li, and Xiang Chen. SaraCoder: Orchestrating semantic and structural cues for resource-optimized repository-level code completion. arXiv preprint arXiv:2508.10068, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Jeremy A. Collins, Loránd Cheng, Kunal Aneja, Albert Wilcox, Benjamin Joffe, and Animesh Garg. AMPLIFY: Actionless motion priors for robot learning from videos. arXiv preprint arXiv:2506.14198, 2025.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. DataMIL: Selecting data for robot imitation learning with datamodels. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/033d9e8dbbbc0ad90e59222cf1db0fc2-Abstract-Conference.html.

Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. In *Robotics: Science and Systems*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. In *Robotics: Science and Systems*, 2025.

Moo Jin Kim, Chelsea Finn, and Percy Liang. OpenVLA-OFT LIBERO evaluation script. https://github.com/moojink/openvla-oft/blob/main/experiments/robot/libero/run_libero_eval.py, 2025. Accessed September 21, 2026.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

Sateesh Kumar, Shivin Dass, Georgios Pavlakos, and Roberto Martín-Martín. COLLAGE: Adaptive fusion-based retrieval for augmented policy learning. In *Proceedings of the Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4607–4624, 2025. URL https://proceedings.mlr.press/v305/kumar25a.html.

Li-Heng Lin, Yuchen Cui, Amber Xie, Tianyu Hua, and Dorsa Sadigh. FlowRetrieval: Flow-guided data retrieval for few-shot imitation learning. In *Proceedings of the Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4084–4099, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://github.com/Lifelong-Robot-Learning/LIBERO.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690, 2022.

Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In *International Conference on Learning Representations*, 2025.

Shangchen Miao, Ningya Feng, Jialong Wu, Ye Lin, Xu He, Dong Li, and Mingsheng Long. JEPA-VLA: Video predictive embedding is needed for VLA models. arXiv preprint arXiv:2602.11832, 2026.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Proceedings of the Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909, 2023.

Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Proceedings of the Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 2181–2204, 2023.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. arXiv preprint arXiv:2310.08864, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Physical Intelligence. `openpi` LIBERO evaluation script. `https://github.com/Physical-Intelligence/openpi/blob/main/examples/libero/main.py`, 2025. Accessed September 21, 2026.

Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *European Conference on Computer Vision*, pp. 540–555, 2014.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Zequn Qin, Pengyi Zhang, Fei Wu, and Xi Li. FcaNet: Frequency channel attention networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 783–792, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, 2021.

Zillur Rahman, Eddison Pham, Alejandro Daniel Noel, and Cristian Meo. ROSER: Few-shot robotic sequence retrieval for scalable robot learning. arXiv preprint arXiv:2603.01474, 2026.

Rainbow Robotics. RB3-730 collaborative robot specifications, n.d. URL `https://rainbow-robotics.com/products/rb3-730/`. Accessed September 21, 2026.

Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:297–330, 2020. doi: 10.1146/annurev-control-100819-063206.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Amber Xie, Rahul Chand, Dorsa Sadigh, and Joey Hejna. Data retrieval with importance weights for few-shot imitation learning. In *Proceedings of the Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 1–16, 2025. URL `https://proceedings.mlr.press/v305/xie25a.html`.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. GMFlow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8121–8130, 2022. doi: 10.1109/CVPR52688.2022.00795.

Jewon Yeom, Hanseul Kim, Jeongjae Park, Sungmok Jung, Jaejin Lee, and Taesup Kim. What makes video world model latents action-relevant: Prediction over reconstruction. arXiv preprint arXiv:2606.07687, 2026.

Yu Zhang, Yuqi Xie, Huihan Liu, Rutav Shah, Michael Wan, Linxi Fan, and Yuke Zhu. SCIZOR: A self-supervised approach to data curation for large-scale imitation learning. arXiv preprint arXiv:2505.22626, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

A ALGORITHMS

Algorithm 1 constructs the segment features (Section 4.2); Algorithm 2 uses them to select prior segments (Section 4.3).

Algorithm 1 Segment Slicing and Encoding

Input: target and prior demonstrations $\mathcal{D}_{\text{target}}, \mathcal{D}_{\text{prior}}$; frozen video encoder ϕ
Parameters: number of non-DC DCT components D; temporal-pattern weight β
Output: segment sets $\mathcal{S}_{\text{target}}, \mathcal{S}_{\text{prior}}$ and their semantic and temporal-pattern features

for each trajectory $\tau \in \mathcal{D}_{\text{target}} \cup \mathcal{D}_{\text{prior}}$ **do**
  Extract temporal video features with ϕ and apply KTS to obtain segment boundaries.
  for each resulting segment s **do**
    Slice its raw frames and independently re-encode them with ϕ.
    Compute $z_{\text{sem}}(s)$ and $z_{\text{tp}}(s)$ using Eqs. (3) and (4).
  end for
end for
Collect target and prior segments into $\mathcal{S}_{\text{target}}$ and $\mathcal{S}_{\text{prior}}$, respectively.

Algorithm 2 Redundancy-Aware Retrieval

Input: segments and features from Algorithm 1; task instructions; frozen text encoder ψ
Parameters: per-query retrieval budget K; pre-filtered set size N_c; MMR relevance weight λ; language-consistency weight ν
Output: retrieved data $\mathcal{D}_{\text{retrieved}}$

Encode task instructions with ψ for language consistency.
$\mathcal{D}_{\text{retrieved}} \leftarrow \emptyset$.
for each target query $q \in \mathcal{S}_{\text{target}}$ **do**
  Phase 1: Relevance-based pre-filtering
  Rank prior segments $c \in \mathcal{S}_{\text{prior}}$ by semantic cosine similarity $\text{Sim}_1(c, q)$.
  $\mathcal{C}_q \leftarrow$ the top N_c prior segments.
  Phase 2: Redundancy-aware selection
  $c^\star \leftarrow \arg\max_{c \in \mathcal{C}_q} \text{Sim}_1(c, q)$
  $\mathcal{S}_q \leftarrow \{c^\star\}$
  while $|\mathcal{S}_q| < K$ **do**
    Select $c^\star$ from $\mathcal{C}_q \setminus \mathcal{S}_q$ by balancing relevance, redundancy, and language consistency (Eq. (7)).
    $\mathcal{S}_q \leftarrow \mathcal{S}_q \cup \{c^\star\}$.
  end while
  Append the segments in $\mathcal{S}_q$ to $\mathcal{D}_{\text{retrieved}}$.
end for

After retrieval, we train π_θ on $\mathcal{D}_{\text{target}}$ and $\mathcal{D}_{\text{retrieved}}$ using the BC objective and equal sampling probabilities (Section 4.4).

Table 5: Full LIBERO-10 task names for the abbreviations in Table 1.

Abbr.	Task Name in LIBERO-10
Book	STUDY SCENE1 pick up the book and place it in the back compartment of the caddy
Bowl	KITCHEN SCENE4 put the black bowl in the bottom drawer of the cabinet and close it
Cheese-B	LIVING ROOM SCENE2 put both the cream cheese box and the butter in the basket
Moka	KITCHEN SCENE8 put both moka pots on the stove
Mug-Mic	KITCHEN SCENE6 put the yellow and white mug in the microwave and close it
Mug-Mug	LIVING ROOM SCENE5 put the white mug on the left plate and put the yellow and white mug on the right plate
Mug-P	LIVING ROOM SCENE6 put the white mug on the plate and put the chocolate pudding to the right of the plate
Stove	KITCHEN SCENE3 turn on the stove and put the moka pot on it
Soup-C	LIVING ROOM SCENE1 put both the alphabet soup and the cream cheese box in the basket
Soup-S	LIVING ROOM SCENE2 put both the alphabet soup and the tomato sauce in the basket

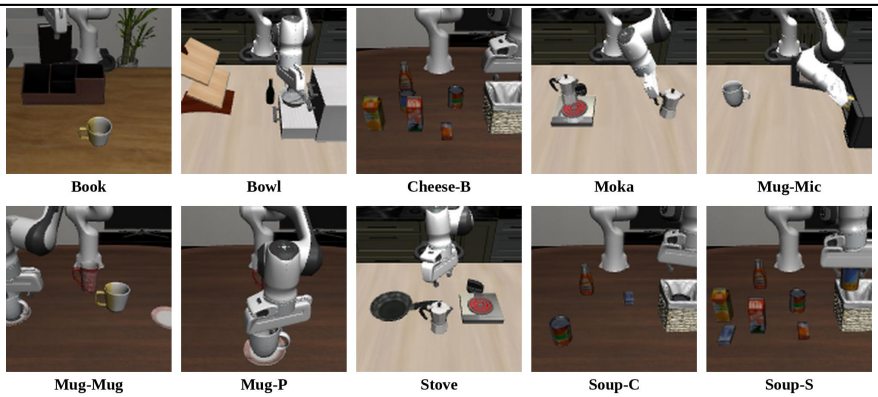


Figure 4: Representative observations from the ten LIBERO-10 target tasks. Labels follow the abbreviations in Table 5.

B Experimental Setup

B.1 Simulation Benchmark and Data

LIBERO is a language-conditioned robot-manipulation benchmark designed to study transfer across tasks (Liu et al., 2023). Its public LIBERO-100 demonstrations are partitioned into LIBERO-90 and LIBERO-10. We use all 4,500 LIBERO-90 trajectories (90 tasks, 50 demonstrations each) as $\mathcal{D}_{\text{prior}}$ and five demonstrations from each of the ten LIBERO-10 tasks as $\mathcal{D}_{\text{target}}$. This provides diverse prior and held-out target tasks while keeping the robot embodiment, observation space, and action space fixed. The data and download script are available in the official release (Liu et al., 2023). Table 5 expands the task abbreviations used in the main paper, and Figure 4 shows one representative observation from each target task.

The five target demonstrations for each task were sampled once using retrieval seed 42 (IDs 1, 7, 15, 24, and 45) and fixed across all methods and policy seeds.

B.2 BASELINES

The original baseline repositories are available at `https://github.com/WEIRDLabUW/STRAP` for STRAP and `https://github.com/SateeshKumar21/collage-retrieval-code` for COLLAGE. Our original and matched comparison settings are described in Appendix B.5.

Baselines without retrieval. BC trains only on the target demonstrations. MT trains on the target demonstrations and the complete LIBERO-90 prior dataset.

STRAP. We reproduce STRAP using its released retrieval implementation and frozen DINOv2-base feature artifacts for every observation in the target and prior datasets (Memmel et al., 2025; Oquab et al., 2024). Only the target trajectories are partitioned at end-effector (EEF) pauses to form query sub-trajectories. For each query, subsequence dynamic time warping (S-DTW) aligns its framewise DINOv2 sequence to the best contiguous match within each unsegmented prior trajectory; the lowest-cost matched ranges form the retrieved dataset.

COLLAGE. We use the released COLLAGE implementation and author-provided visual, motion, and shape representations (Kumar et al., 2025): DINOv2 visual features (Oquab et al., 2024), optical-flow embeddings from a reconstruction encoder trained on GMFlow outputs (Xu et al., 2022), and PointNet++ shape features (Qi et al., 2017). For each of these three modalities, COLLAGE uses the same target EEF-pause queries and S-DTW retrieval procedure as STRAP, producing a separate retrieved subset. Its fourth subset is constructed at the trajectory level: task instructions are embedded using OpenAI `text-embedding-ada-002`, source tasks are selected by instruction cosine similarity, and complete demonstrations are sampled to approximately match the number of frames in the visual subset.

COLLAGE combines the retrieved subsets using task-specific modality weights. Appendix B.5 distinguishes the released weights used in the original setting from the recomputed weights used in the matched setting. Appendix C.2 provides the weight-estimation procedure.

For STRAP and COLLAGE, we use the released precomputed features without re-encoding the RGB observations or changing their preprocessing. Appendix C.1 specifies their EEF-pause segmentation and S-DTW retrieval.

B.3 BASELINE COMPARISON PROTOCOL

All methods use the same target demonstrations, policy architecture, training schedule, policy seeds, and evaluation protocol. In the matched protocol, MT and the retrieval methods sample target and prior data with equal probability. The prior group is the full LIBERO-90 dataset for MT and the retrieved data for retrieval methods; BC uses only target demonstrations. COLLAGE retains adaptive modality weighting within the retrieved group.

Matched retrieval methods use a budget of $K = 5$ segments per target query while retaining their method-specific retrieval procedures. The following section describes how this budget is applied and how the resulting data volumes differ.

B.4 TASK-LEVEL AND PER-QUERY RETRIEVAL BUDGETS

A query is a segment of a target demonstration; each retrieval selects a segment from the prior data. The following tables distinguish the number of target queries from the number of prior segments retrieved for them. All tasks use five target demonstrations.

RVSR. RVSR uses KTS to form target queries and retrieves $K = 5$ prior segments for each query. Table 6 shows the number of queries and total retrieved segments.

STRAP and COLLAGE. STRAP forms target queries by splitting demonstrations at pauses in end-effector (EEF) motion. Section C.1 describes the segmentation procedure in detail. COLLAGE retrieves four subsets using visual, motion, shape, and language information. Its visual, motion, and

Table 6: RVSR retrieval per task with $K = 5$ prior segments per query. Queries are KTS segments from the five target demonstrations.

Task	Queries	**Retrieved/query**	Total retrieved
Book	21	5	105
Bowl	31	5	155
Cheese-B	38	5	190
Moka	46	5	230
Mug-Mic	37	5	185
Mug-Mug	27	5	135
Mug-P	32	5	160
Stove	31	5	155
Soup-C	39	5	195
Soup-S	34	5	170
Total	336	—	1,680

shape retrievers each use the same EEF-pause queries as STRAP, with a separate retrieval budget for each modality. The language retriever instead selects complete demonstrations by task instruction, approximately matching the visual subset in frame count (Kumar et al., 2025).

In the original setting, the nominal task-level retrieval budget $K_{\text{task}} = 100$ is divided among the Q_{EEF} queries. Each query retrieves $\max(1, \lfloor K_{\text{task}}/Q_{\text{EEF}} \rfloor)$ prior segments; the unused remainder is not reassigned (Memmel et al., 2025). In the matched setting, each query retrieves $K = 5$ segments. Table 7 shows both settings for STRAP and for each of COLLAGE's three segment-based retrievers. The language subset is constructed separately as described above.

Table 7: Retrieval settings for STRAP and COLLAGE. Original divides a task-level budget of 100 prior segments among the target queries, whereas matched retrieves five segments per query. Both settings use EEF-pause queries from five target demonstrations. For COLLAGE, the table applies to each of its visual, motion, and shape retrievers; language retrieval is handled separately.

	Original ($K_{\text{task}} = 100$)			Matched ($K = 5$)		
Task	Queries	Retrieved/ query	**Total retrieved**	Queries	**Retrieved/ query**	Total retrieved
Book	23	4	92	23	5	115
Bowl	20	5	100	20	5	100
Cheese-B	22	4	88	22	5	110
Moka	38	2	76	38	5	190
Mug-Mic	36	2	72	36	5	180
Mug-Mug	26	3	78	26	5	130
Mug-P	32	3	96	32	5	160
Stove	31	3	93	31	5	155
Soup-C	30	3	90	30	5	150
Soup-S	30	3	90	30	5	150
Total	288	—	875	288	—	1,440

Retrieved data volume. Table 8 compares the total retrieved segments and frames for STRAP and RVSR under the matched $K = 5$ setting. Total retrieved segments include repeated selections across queries. Frame totals exclude added context and padding.

RVSR retrieves more segments but fewer frames than STRAP because its segments are shorter on average. Thus, matching the per-query budget does not equalize the total retrieved data. COLLAGE

Table 8: Retrieved segments and frames across all ten LIBERO-10 tasks for STRAP and RVSR, both using $K = 5$ prior segments per query.

Method	Total retrieved	Total frames	Segment length (frames) mean / median
STRAP*	1,440	79,531	55.2 / 55
RVSR	1,680	64,804	38.6 / 37

is omitted from this comparison because it combines modality-specific segment subsets with full demonstrations.

B.5 Original and Matched Baseline Comparison

Table 9 summarizes the original and matched settings, and Table 10 repeats the main results using the same method labels. Original retains each baseline's retrieval budget and sampling scheme; matched uses the per-query retrieval budget above and the sampling protocol in Appendix B.3. In the settings table, Union denotes uniform sampling from the combined target and retrieved data, Equal denotes equal target/prior sampling probabilities, and Adaptive denotes COLLAGE's original adaptive mixture. MT is reported only under the matched protocol.

For COLLAGE in the original setting, we use the modality weights released by its authors. In the matched setting, the per-modality retrieved subsets change with the retrieval budget, so we recompute the weights using COLLAGE's reference-policy procedure. We repeat weight estimation with three seeds and average the resulting weight vectors to reduce seed-dependent variation (Appendix C.2). Original and matched therefore differ in retrieval budget and sampling, and additionally in the estimated modality weights for COLLAGE; their differences do not isolate the effect of sampling alone.

Table 9: Retrieval budget and sampling settings for Table 1. Original uses a task-level budget ($K_{\text{task}} = 100$); matched uses a per-query budget ($K = 5$). * marks matched baselines. For COLLAGE (†), budgets apply separately to visual, motion, and shape retrieval; Equal retains adaptive modality weighting within the retrieved group.

Method	Setting	Retrieval budget	Sampling
BC	–	–	Target only
MT	Matched	–	Equal
STRAP	Original	$K_{\text{task}} = 100$	Union
STRAP*	Matched	$K = 5$	Equal
COLLAGE	Original	$K_{\text{task}} = 100^\dagger$	Adaptive
COLLAGE*	Matched	$K = 5^\dagger$	Equal†
RVSR	–	$K = 5$	Equal

B.6 Policy Training and Evaluation

The main simulation comparison uses the language-conditioned BC Transformer released with STRAP, based on robomimic (Memmel et al., 2025; Mandlekar et al., 2022); COLLAGE also adopts this policy family (Kumar et al., 2025). Table 11 summarizes the shared architecture and optimization settings. Training-data sampling is specified in Appendix B.3 and Appendix B.5.

B.6.1 Policy Evaluation Protocol

For each policy seed, we evaluate checkpoints from epochs $50, 100, 150, 200, 250$, and 300. Each checkpoint is evaluated using 50 rollouts per task with a maximum horizon of 520 environment steps and early termination upon success. We select the best evaluated checkpoint separately for each task–seed policy and apply this same rule to every method. We first average the ten task

Table 10: LIBERO-10 success rates (%), repeated from Table 1. * marks matched baselines (Table 9). Avg. reports mean±standard deviation across seeds. Bold and underlined values indicate the best and second-best results, including ties.

Method	Book	Bowl	Cheese-B	Moka	Mug-Mic	Mug-Mug	Mug-P	Stove	Soup-C	Soup-S	**Avg.**
BC	56.7	88.7	46.0	12.7	38.7	40.0	14.7	81.3	40.0	30.7	44.9 ± 2.2
MT	63.3	94.7	34.7	15.3	52.0	61.3	26.0	90.0	37.3	40.7	51.5 ± 7.8
STRAP	76.7	<u>98.7</u>	47.3	22.0	43.3	**67.3**	24.0	87.3	47.3	65.3	57.9 ± 6.6
STRAP*	74.7	**99.3**	52.7	<u>30.0</u>	52.7	63.3	<u>26.7</u>	88.7	60.7	<u>68.7</u>	61.7 ± 1.0
COLLAGE	76.7	<u>98.7</u>	44.0	**34.0**	46.7	59.3	**30.7**	88.7	<u>67.3</u>	63.3	60.9 ± 2.4
COLLAGE*	**91.3**	96.0	<u>59.3</u>	22.7	<u>53.3</u>	64.0	**30.7**	**96.7**	52.7	68.0	$\underline{63.5} \pm 2.1$
RVSR	<u>90.0</u>	98.0	**61.3**	21.3	**64.7**	<u>66.7</u>	<u>26.7</u>	<u>92.7</u>	**71.3**	**71.3**	$\mathbf{66.4} \pm 1.3$

Table 11: Shared policy architecture and training settings for the simulation experiments.

Setting	Value
Inputs and encoders	
Camera views	Agent-view and eye-in-hand RGB
Observation history	5 steps
Visual encoder	ResNet-18 per view, trained from scratch (He et al., 2016)
Language encoder	Frozen CLIP ViT-L/14 (Radford et al., 2021)
Policy architecture	
Backbone	BC Transformer
Transformer layers	8
Embedding dimension	256
Attention heads	4
Dropout (embedding / attention / output)	0.1 / 0.1 / 0.1
Action head	Gaussian mixture model (GMM)
GMM components	5
Action prediction horizon	5 steps
Optimization	
Optimizer	AdamW
Learning rate	10^{-4}
L2 regularization	10^{-5}
Batch size	32
Warmup	10,000 gradient steps
Training epochs	300
Gradient steps per epoch	200

success rates within a seed and then report the mean and standard deviation of those three seed-level averages (seeds 1234, 42, and 4325).

Evaluation horizon. We use a horizon of 520 steps for all methods. Table 12 reports the lengths of the official LIBERO-10 demonstrations in our dataset copy. Across the 500 demonstrations, lengths range from 150 to 517 steps. Moka has a mean length of 415.9 steps, a maximum of 517, and a mean of 386.4 for the five selected target demonstrations. A 300-step evaluation horizon is therefore shorter than the recorded duration of many successful demonstrations, whereas 520 accommodates the longest demonstration in this dataset copy.

Table 12: Lengths, in environment steps, of the official LIBERO-10 offline demonstrations. "Mean (50)" and "Max." use all 50 demonstrations for a task; the last column uses the five target demonstrations selected for our experiments.

Task	Mean (50)	Max.	Mean (selected 5)
Book	189.4	259	197.8
Bowl	248.7	317	230.6
Cheese-B	260.4	322	250.4
Moka	415.9	517	386.4
Mug-Mic	304.6	449	302.6
Mug-Mug	258.2	331	257.2
Mug-P	255.1	342	259.8
Stove	266.0	340	260.4
Soup-C	269.5	334	281.4
Soup-S	294.0	388	304.4

Public OpenVLA-OFT and `openpi` evaluation implementations also use 520 steps for LIBERO-10 (Kim et al., 2025; Physical Intelligence, 2025); AMPLIFY uses a 500-step limit (Collins et al., 2025). We fixed the 520-step horizon before the reported comparison and applied it to every method, including the retrained STRAP and COLLAGE baselines. Their original evaluations used a 300-step horizon; all baseline results reported here are reevaluated under our common 520-step protocol. The horizon was not tuned separately for individual methods, and we did not conduct a horizon sweep.

B.6.2 Randomness and Determinism

Training and evaluation set the NumPy and PyTorch seeds from the policy seed. Environment seeds are derived from the policy seed and environment index. Data-loader generators use the training seed, and worker NumPy seeds are derived from PyTorch worker seeds. cuDNN benchmarking is disabled and deterministic mode is enabled.

C Implementation Details

C.1 STRAP and COLLAGE Retrieval Details

EEF-pause Query Segmentation. STRAP and the three sub-trajectory branches of COLLAGE share the same query construction. They use EEF positions only to determine temporal ranges in the target trajectories; similarity is subsequently computed from the corresponding modality features. Prior trajectories remain unsegmented for S-DTW search. Let $p_t \in \mathbb{R}^3$ be the Cartesian EEF position at frame t. The released implementation places a provisional boundary after frame t whenever

$$\|p_{t+1} - p_t\|_1 < 5 \times 10^{-3}.$$

It then scans the provisional segments in temporal order and merges a segment shorter than 20 frames forward into the next segment; if the final segment is still shorter than 20 frames, it is merged into the preceding segment. The resulting contiguous ranges cover the complete target trajectory and serve as the S-DTW queries. Thus, the threshold above defines provisional boundaries, whereas all reported segment counts and lengths are measured after the 20-frame merging step. STRAP represents each resulting query with its framewise DINOv2 features. COLLAGE applies the same query ranges separately to its DINOv2 visual, GMFlow motion, and PointNet++ shape features; its language branch does not use EEF segmentation and instead retrieves complete demonstrations by instruction similarity. Across all 500 LIBERO-10 demonstrations, this procedure produces 3,001 final segments, or 6.00 per demonstration, as reported in Table 13. On the 50 demonstrations actually used as retrieval targets (five per task), it produces 288 queries, or 5.76 per demonstration.

DTW and S-DTW. For two sequences $X = (x_1, \ldots, x_L)$ and $Y = (y_1, \ldots, y_M)$, dynamic time warping (DTW) accumulates a local feature distance $\delta(x_i, y_j)$ using

$$D(i,j) = \delta(x_i, y_j) + \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\}.$$

For these baselines, $\delta(x_i, y_j) = \|x_i - y_j\|_2$ in the corresponding modality feature space. The released retrieval code L2-normalizes the frame features before computing these local distances and ranks matches by the accumulated S-DTW cost without an additional path- or sequence-length normalization. For ordinary DTW, $D(0,0) = 0$ and the remaining zeroth row and column are infinite; it therefore aligns both complete sequences and returns $D(L, M)$. STRAP uses subsequence DTW (S-DTW) for segment-level retrieval. COLLAGE applies the same sub-trajectory retrieval procedure separately to its visual, motion, and shape features; its language subset is instead retrieved at the trajectory level using instruction similarity. Given a short target sub-trajectory X and a longer prior trajectory Y, S-DTW allows the matched region in Y to have free start and end points:

$$d_{\text{S-DTW}}(X, Y) = \min_{1 \le s \le e \le M} \text{DTW}(X, Y_{s:e}).$$

Thus, the entire target sub-trajectory is aligned to its best contiguous match in the prior trajectory (Memmel et al., 2025; Kumar et al., 2025).

COLLAGE language retrieval. The language branch selects source tasks whose instruction cosine similarity exceeds 0.90 using `text-embedding-ada-002` embeddings. It samples complete demonstrations to approximately match the number of frames in the visual subset.

C.2 COLLAGE Modality-Weight Estimation

COLLAGE estimates modality weights by briefly training a reference behavioral cloning policy on each modality-specific retrieved subset and evaluating the demonstrated actions' log-likelihood on the target data (Kumar et al., 2025). We use the author-released weights for the original setting. For the matched setting, we rerun this procedure on the newly retrieved subsets.

Each reference policy uses COLLAGE's released Transformer implementation, a batch size of 32, and 100 training epochs. Following its weight-estimation procedure, we average the target-data log-likelihood scores over checkpoints at epochs 60, 70, 80, 90, and 100, then apply a softmax with temperature 2 to obtain the modality weights. For each target task, we repeat the procedure with seeds 1234, 42, and 4325 and average the three resulting weight vectors. The averaged weights are then used for modality sampling within the retrieved-data group in the matched protocol.

C.3 Video Encoding and Preprocessing

For retrieval, we encode only the agent-view RGB sequence using the frozen V-JEPA 2 ViT-g/16 model at 384×384 input resolution (Assran et al., 2025). For each sequence, we resize the shorter side to 438 pixels using bilinear interpolation, apply a 384×384 center crop and ImageNet normalization, and vertically flip each stored LIBERO frame to correct its orientation. From the final encoder output, we average over spatial patch locations to obtain one 1408-dimensional vector per two-frame tubelet.

We encode each trajectory in one bidirectional forward pass over its complete even-length prefix, without temporal subsampling or clip stitching. We refer to this procedure as full-trajectory encoding.

Trajectories and segments are encoded at their original temporal lengths. Because the tubelet size is two, full-trajectory encoding removes an unmatched final frame before KTS. KTS therefore sees only complete tubelets. When token boundaries are converted to half-open raw-frame ranges, only the end of the final range is extended to the original trajectory length. The subsequent segment encoder reads that raw range; if its length is odd, it repeats the final frame once. Consequently, the odd frame is excluded from boundary detection but is retained in the retrieved data and in the independently re-encoded segment representation.

Table 13: Final segmentation statistics over the same 500 LIBERO-10 demonstrations. EEF-pause segments are measured after applying the 20-frame merging rule in Section C.1; KTS segments are the output of Section C.4. Means are reported with sample standard deviations. Ranges are inclusive minima and maxima.

		Segments per demonstration		Frames per segment	
Segmentation	Total segments	Mean ± SD	Median [range]	Mean ± SD	Median [range]
EEF pause	3,001	6.00 ± 2.09	6 [2, 15]	46.01 ± 23.58	46 [20, 167]
V-JEPA 2 KTS	3,417	6.83 ± 1.51	7 [4, 11]	40.41 ± 18.46	38 [20, 142]

C.4 Kernel Temporal Segmentation

KTS is applied independently to the full-trajectory V-JEPA 2 temporal tokens of every target and prior trajectory. We construct its affinity matrix using a Gaussian radial basis function (RBF) kernel. For token features x_i, $A_{ij} = \exp(-\|x_i - x_j\|_2^2/(2\sigma^2))$, where σ is the square root of the median nonzero pairwise squared distance within that trajectory. Thus, tokens that are close in the V-JEPA 2 feature space have high affinity. For a candidate interval $[a, b)$, the implementation assigns the coherence score $\sum_{i,j=a}^{b-1} A_{ij}/(b-a)$. Dynamic programming then chooses the partition that maximizes the sum of these within-segment scores minus a fixed 1.2 penalty for every segment. The coherence term favors internally consistent segments, while the penalty prevents the trajectory from being split into too many short pieces. Candidate segments contain 10–200 video tokens. A trajectory with at most 10 tokens is retained as a single segment. These values are fixed for all reported policy experiments. The KTS parameters were not selected through a policy-success sweep. The dynamic program follows the video segmentation formulation of Potapov et al. (2014); the RBF bandwidth, linear penalty, and segment-length constraints above are our documented implementation choices.

C.5 Segmentation Statistics and Low-Activity Diagnostic

We compare the KTS partitions with the EEF-pause partitions defined in Section C.1. Both are computed on the same 500 LIBERO-10 demonstrations and enforce a minimum final segment length of 20 input frames. Table 13 shows that the two methods operate at similar granularity: EEF produces 6.0 segments per demonstration on average, and KTS produces 6.8. KTS produces more segments of shorter average length, with less variation in segment count across demonstrations.

Query counts and retrieved segment totals for the five target demonstrations per task are reported in Tables 6 and 7.

Segment count alone does not measure behavioral usefulness, so we additionally measure the prevalence of low-activity fragments. For a segment s with F_s frames, end-effector positions $p_t \in \mathbb{R}^3$, and scalar gripper states g_t, define

$$m(s) = \frac{1}{F_s - 1} \sum_{t=1}^{F_s - 1} \|p_{t+1} - p_t\|_1, \tag{8}$$

$$r_g(s) = \max_t g_t - \min_t g_t. \tag{9}$$

Eq. (8) is the segment-level mean frame-to-frame end-effector displacement, with motion along the three Cartesian axes summed by the L1 norm. Eq. (9) is the range of the gripper signal within the segment. We use 5×10^{-3} as the motion threshold because it is the per-timestep L1 displacement threshold used to construct the EEF-pause boundaries in Section C.1, rather than a value tuned from this diagnostic. The operations are nevertheless distinct: EEF segmentation tests each consecutive frame pair, whereas this diagnostic tests the mean displacement over a complete final segment.

We label s low activity only when both $m(s) < 5 \times 10^{-3}$ and $r_g(s) < 0.01$: the arm moves very little and the gripper does not meaningfully open or close. The gripper threshold is used only for this diagnostic, not by KTS; it excludes segments with gripper-state changes above the threshold, even when arm motion is small. For reference, the median demonstration-level gripper range is

Table 14: RVSR retrieval configuration and values used in additional ablations or sensitivity analyses.

Parameter	RVSR value	Tested alternatives
Relevance-based pre-filtering		
Pre-filtered set size N_c	30	50, 100
Sequential selection		
Per-query retrieval budget K	5	3, 10, 20
MMR relevance weight λ	0.5	0.3, 0.7, 1.0
Language-consistency weight ν	0.3	0
Temporal-pattern feature		
Temporal-pattern weight β	0.2	0, 0.4, 0.5, 0.6, 0.8, 1.0
Non-DC DCT components D	3	1, 9

approximately 0.034. The denominator is all segments produced by the corresponding method: 15.5% of the 3,001 EEF segments and 3.2% of the 3,417 KTS segments satisfy the primary criterion. The directional result is unchanged across the tested thresholds: EEF/KTS low-activity rates are 12.0/2.5% and 21.4/5.0% when the gripper threshold is changed to 0.005 and 0.020, respectively, and 4.8/0.5% when the motion threshold is halved to 2.5×10^{-3}.

These measurements indicate that a smaller fraction of KTS segments are nearly static under the tested thresholds. This diagnostic measures low activity, rather than agreement with ground-truth sub-task boundaries.

C.6 Segment Encoding and Features

After detecting KTS boundaries, we slice the raw frames and independently re-encode each segment; we do not slice tokens from the full-trajectory forward pass. Let $\Phi_s \in \mathbb{R}^{T_s \times 1408}$ denote the resulting temporal representation, where T_s is the number of temporal positions in segment s. Variable-length token sequences are zero-padded only for HDF5 storage and accompanied by their valid lengths; all readouts use only the valid prefix. The semantic feature z_{sem} is the L2-normalized temporal mean, whose normalized direction equals that of the zero-frequency (DC) component. Let $\bar{u}_{1:D}$ denote the L2-normalized concatenation of components $u_1, \ldots, u_D$ from an orthonormal type-II discrete cosine transform (DCT-II) along time, excluding the DC component u_0. Following the main-paper definition, z_{tp} combines z_{sem} with this DCT block using β. The implementation calls `scipy.fft.dct` with `type=2`, `norm=ortho`, and the temporal axis. Both the temporal mean and the concatenated non-DC block are divided by their L2 norm plus 10^{-9}; hence a zero non-DC block remains zero rather than producing an undefined value. The two normalized blocks are combined without further normalization; redundancy is computed as their weighted inner-product sum. For nonzero blocks with exact L2 normalization, this equals cosine similarity between the combined features in Eq. (6). The adopted configuration uses $D = 3$ and $\beta = 0.2$; the DCT is a fixed transform (Ahmed et al., 1974).

C.7 Retrieval and Selection

Table 15 summarizes the adopted configuration. The Phase 1 pre-filtered set is constructed using cosine similarity between semantic segment features. Phase 2 sequentially balances target relevance, redundancy with the already-selected set, and task-level language consistency. Task instructions are encoded using the frozen CLIP ViT-L/14 text encoder (Radford et al., 2021) (`openai/clip-vit-large-patch14`).

More precisely, the prior pool contains every valid re-encoded LIBERO-90 segment. For each segment of target demonstrations 1, 7, 15, 24, and 45, Phase 1 retains the $N_c = 30$ prior segments with the largest Sim_1. Language is not used in this phase for the adopted method. Phase 2 uses the Sim_2 and MMR selection rule defined in the main paper without modification. The first item has the largest Sim_1; selection then proceeds until $K = 5$ items have been chosen for the query. The language-consistency term is cosine similarity between L2-normalized CLIP embeddings of

Table 15: Adopted embedding, segmentation, and retrieval settings used for RVSR. Table 14 separately lists the alternative retrieval values examined in the additional analyses.

Setting	Value
Embedding	
Video encoder	Frozen V-JEPA 2 ViT-g/16 (Assran et al., 2025)
Input	Agent-view RGB, 384×384 center crop
Temporal tokens	Two-frame tubelets, spatial token average
Segmentation	
Method and input	KTS on full-trajectory temporal tokens
Kernel	RBF, trajectory-adaptive bandwidth
Penalty	1.2 per segment
Segment length	10–200 video tokens
Segment encoding	
Encoding context	Independent encoding within each KTS segment
Semantic feature	L2-normalized temporal mean, z_{sem}
DCT block	First three non-DC DCT-II components ($D = 3$)
Retrieval	
Pre-filtered set size	$N_c = 30$ per target query
Per-query retrieval budget	$K = 5$ segments per query
Selection weights	$\lambda = 0.5$, $\beta = 0.2$, $\nu = 0.3$
Language encoder	Frozen CLIP ViT-L/14 (Radford et al., 2021), Phase 2 only

the prior and target task instructions. Each instruction is encoded once and shared by all segments from that task.

C.8 Retrieved Dataset Construction and Policy Inputs

Each selected range is represented as a half-open interval $[a, b)$ in the source demonstration. We follow STRAP's dataset construction procedure to provide observation histories and action targets for the retrieved segments.

Scope of redundancy suppression. RVSR suppresses redundancy within the K-segment set selected for each target query. Selection is independent across target queries, and a source interval is retained with multiplicity when selected by different queries. RVSR therefore does not explicitly optimize global redundancy over the complete retrieved training dataset.

During policy training, target demonstrations and retrieved segments are treated as two data sources. Both receive weight 1.0 and are normalized by dataset size, yielding equal target/retrieved sampling probability.

The policy consumes agent-view and eye-in-hand RGB together with end-effector position and orientation, end-effector state, joint state, and gripper state. Each camera is encoded by a ResNet-18 backbone with feature-wise linear modulation (FiLM), trained from scratch, followed by spatial softmax with 32 keypoints; training uses a random 116×116 crop from the stored 128×128 image. The policy Transformer is non-causal, uses a context of five, and predicts the five-step action sequence with a five-component GMM. The Transformer embedding, attention, and output dropouts are all 0.1. The remaining optimization and schedule parameters are listed in Table 11.

D SENSITIVITY ANALYSES

D.1 REPORTING PROTOCOL

Each sensitivity experiment changes one factor while holding the remaining retrieval settings, data, retrieval seed, policy architecture, training steps, policy seeds, evaluation horizon, and checkpoint-selection rule fixed. We report $\Delta\mathrm{SR}$, the difference in three-seed mean success rate from the adopted setting rerun in the same environment. Thus, the tables show within-environment sensitivity rather than absolute success rates; the adopted row in each table is the corresponding same-environment reference. The differences are measured in percentage points. The results summarize mean success across the ten target tasks within each sweep environment; task-level trends can differ. We interpret them as sensitivity around the adopted configuration in those environments, rather than as a ranking that necessarily transfers across computing environments. The adopted configuration was fixed before these sensitivity analyses and shared across all tasks.

D.2 PHASE 1 PRE-FILTERED SET SIZE N_c

Table 16: Sensitivity to the number of candidates retained per target query after Phase 1 relevance-based pre-filtering. Deltas are relative to the same-environment reference (Section D.1).

N_c	$\Delta\mathrm{SR}$ vs. $N_c = 30$
30 (adopted)	0.0
50	-2.8
100	-5.3

As shown in Table 16, increasing N_c from 30 to 50 or 100 reduces mean success by 2.8 and 5.3 percentage points, respectively. Retaining more candidates broadens the set available to Phase 2 but relaxes the Phase 1 pre-filter. Among the evaluated values, $N_c = 30$ has the highest mean success across the ten tasks.

D.3 NUMBER OF RETAINED DCT COMPONENTS

Here D denotes the number of retained non-DC components $u_1, \ldots, u_D$. A DCT-II of T temporal tokens produces one DC coefficient and $T - 1$ non-DC coefficients (Ahmed et al., 1974). The minimum KTS segment has 10 V-JEPA 2 tokens (20 input frames), so $D = 9$ is the largest fixed depth available to every segment. We compare a single non-DC component ($D = 1$), the adopted $D = 3$, and $D = 9$, the maximum available to every segment.

Table 17: Sensitivity to the number of retained non-DC DCT components. Deltas are relative to the same-environment reference (Section D.1).

D	$\Delta\mathrm{SR}$ vs. $D = 3$
1	-3.5
3 (adopted)	0.0
9	-1.3

Table 17 shows that $D = 1$ and $D = 9$ yield mean success rates 3.5 and 1.3 percentage points below $D = 3$, respectively. The adopted $D = 3$ thus achieves the highest mean among the three evaluated settings.

D.4 TEMPORAL-PATTERN WEIGHT β

In Table 18, mean success at $\beta = 0.4$ is 0.3 percentage points below the adopted $\beta = 0.2$. It decreases further as the evaluated temporal-pattern weight increases from 0.4 to 1.0.

Table 18: Sensitivity to the temporal-pattern weight β. The $\beta = 0$ case corresponds to the main paper's ablation without the non-DC DCT block and is not repeated here. Deltas are relative to the same-environment reference (Section D.1).

β	ΔSR vs. $\beta = 0.2$
0.2 (adopted)	0.0
0.4	-0.3
0.5	-1.9
0.6	-2.2
0.8	-2.4
1.0	-3.3

Table 19: Sensitivity to the relevance–redundancy balance λ. Deltas are relative to the same-environment reference (Section D.1).

λ	ΔSR vs. $\lambda = 0.5$
0.3	-1.7
0.5 (adopted)	0.0
0.7	-0.3
1.0	-3.5

D.5 MMR Relevance Weight λ

In Table 19, mean success at $\lambda = 0.7$ is 0.3 percentage points below the adopted $\lambda = 0.5$. Removing the redundancy penalty ($\lambda = 1$) lowers mean success by 3.5 percentage points relative to $\lambda = 0.5$. The $+\nu \, \mathrm{lc}(c, q)$ language-consistency term remains active at $\lambda = 1$, so this setting differs from the relevance-only Top-K ablation.

D.6 Per-Query Retrieval Budget K

Table 20: Sensitivity to the number of segments selected per target query. Deltas are relative to the same-environment reference (Section D.1).

K	ΔSR vs. $K = 5$
3	-5.6
5 (adopted)	0.0
10	-2.9
20	-5.3

Among the evaluated budgets, Table 20 shows the highest mean success at the adopted $K = 5$. Selecting only three segments provides less retrieved data, whereas increasing the budget to 10 or 20 does not improve success. This pattern indicates that success does not increase monotonically with the retrieval budget in the evaluated range.

E Ablation Studies

E.1 Independent Segment Re-encoding

V-JEPA 2 uses bidirectional attention, so tokens obtained by encoding a full trajectory are conditioned on frames outside a subsequently sliced segment. For every KTS segment, we compare (i) the tokens corresponding to its frame range, sliced from the full-trajectory encoding, and (ii) tokens obtained by independently encoding the raw frames within the same range. The comparison uses all 22,417 KTS segments in LIBERO-10 and LIBERO-90, with the KTS boundaries and source intervals held fixed.

We compare the two encodings at the feature and retrieval levels. For each segment, semantic-feature similarity is the cosine between the two z_{sem} vectors. Temporal-pattern similarity is the cosine between the two complete z_{tp} vectors, computed using $D = 3$ and $\beta = 0.2$ exactly as in RVSR. For each of the 336 target queries, selected-segment overlap is $|\mathcal{S}_{\mathrm{slice}} \cap \mathcal{S}_{\mathrm{reenc}}|/K$ with $K = 5$. Source-task Jaccard is $|T_{\mathrm{slice}} \cap T_{\mathrm{reenc}}|/|T_{\mathrm{slice}} \cup T_{\mathrm{reenc}}|$, where each T is the set of unique source-task labels in the corresponding retrieval result.

Table 21: Full-trajectory slicing versus independent segment re-encoding with identical KTS boundaries. Diagnostics report mean±sample standard deviation over 22,417 segments (feature similarity) or 336 queries (retrieval overlap). Policy success reports mean±standard deviation over three policy seeds.

(a) Full-trajectory slice vs. independent re-encoding

Diagnostic	Value
Semantic-feature cosine	0.940 ± 0.026
Temporal-pattern cosine	0.826 ± 0.049
Selected-segment overlap (%)	17.3 ± 21.5
Source-task Jaccard (%)	55.1 ± 38.0

(b) RVSR policy success

Segment representation	Success (%)
Full-trajectory slice	56.2 ± 5.6
Independent re-encoding	66.4 ± 1.3

Table 21 shows that the same segments retain semantic-feature similarity of 0.940, while their complete temporal-pattern feature has similarity 0.826. Within the latter, the non-DC DCT block $\bar{u}_{1:3}$ has similarity 0.372 ± 0.227, indicating a larger change in this block than in the semantic feature. The two encodings share only 17.3% of their selected segment identities, although the Jaccard overlap between their source-task sets is 55.1%. Together with the concentrated semantic-similarity distribution in Section F, these changes show that retrieval selections can differ even when z_{sem} remains similar.

Within RVSR, independent re-encoding raises the three-seed mean success rate by 10.2 percentage points. Re-encoding also ensures that each segment feature depends only on frames inside its stated KTS boundary.

E.2 Interaction between Retrieval Encoder and Selection

To examine the interaction between the retrieval encoder and selection rule, Table 22 evaluates a 2×2 design with framewise DINOv2 image features or temporally contextualized V-JEPA 2 video features, crossed with Top-K or RVSR selection. Each encoder supplies the features used for both KTS and retrieval, so each condition includes the segment boundaries induced by that encoder.

Table 22: End-to-end encoder–selector interaction (success rate, %; mean ± standard deviation over three policy seeds).

Retrieval encoder	Top-K	RVSR selector
DINOv2	57.5 ± 5.3	57.0 ± 3.0
V-JEPA 2	56.5 ± 8.5	$\mathbf{66.4 \pm 1.3}$

Top-K ranks segments using cosine similarity between mean-pooled encoder features and does not use the temporal-pattern feature z_{tp}. Replacing Top-K with the RVSR selector changes mean success by -0.5 percentage points with DINOv2 and $+9.9$ percentage points with V-JEPA 2. This difference in observed gains supports the benefit of combining the video representation with redundancy-aware selection in the evaluated pipelines.

E.3 LANGUAGE IN RELEVANCE-BASED PRE-FILTERING

We test the main-paper design choice to exclude coarse task-level language similarity from Phase 1 relevance-based pre-filtering. For this ablation, we define

$$\begin{aligned}\mathrm{Sim}_1^{\mathrm{lang}}(c,q) = (1-\gamma_{\mathrm{lang}})\cos(z_{\mathrm{sem}}(c), z_{\mathrm{sem}}(q)) \\ + \gamma_{\mathrm{lang}}\cos(\psi(\ell_c), \psi(\ell_q)),\end{aligned} \tag{10}$$

with $\gamma_{\mathrm{lang}} = 0.2$. We use a separate symbol because β denotes the temporal-pattern weight in the Phase 2 redundancy score.

We analyze all 336 target queries, with a Phase 1 pre-filtered set of $N_c = 30$ segments and a Phase 2 selected set of $K = 5$ segments per query. The diagnostics are the number of distinct source tasks, the fraction drawn from a different LIBERO scene, and the mean language-consistency score $\mathrm{lc}(c, q)$. For the Phase 2 selected set, we also report mean semantic cosine. Each value is averaged over queries within a task and then across the ten tasks. These diagnostics do not label behavioral correctness.

Table 23: The 2×2 comparison of language use in Phase 1 pre-filtering and Phase 2 redundancy-aware selection. Within each pair of rows, the Phase 1 pre-filtered set is identical and only the Phase 2 language-consistency term changes. Success rates use one policy seed (42); retrieval diagnostics use the fixed retrieval seed 42.

Configuration	Phase 1 pre-filtering	Phase 2 language term	Success (%)
Phase 2 only (RVSR)	Semantic similarity	On (ν=0.3)	67.8
Neither phase	Semantic similarity	Off (ν=0)	67.0
Both phases	Semantic + language (γ_{lang}=0.2)	On (ν=0.3)	61.8
Phase 1 only	Semantic + language (γ_{lang}=0.2)	Off (ν=0)	60.6

	Phase 1 pre-filtered set ($N_c = 30$)			Phase 2 selected set ($K = 5$)			
Configuration	Source tasks/query	Cross-scene (%)	Language consistency	Source tasks/query	Cross-scene (%)	Language consistency	Semantic cosine
Phase 2 only (RVSR)	3.15	25.1	0.854	1.77	19.1	0.883	0.9873
Neither phase	3.15	25.1	0.854	2.46	25.3	0.850	0.9871
Both phases	1.56	18.6	0.924	1.47	18.5	0.924	0.9824
Phase 1 only	1.56	18.6	0.924	1.48	17.1	0.922	0.9828

Table 23 shows that adding language in Phase 1 raises mean language consistency in the pre-filtered set from 0.854 to 0.924, but reduces source tasks per query from 3.15 to 1.56 and cross-scene retrieval from 25.1% to 18.6%. Mean success is 6.0 percentage points lower with Phase 2 language enabled and 6.4 percentage points lower without it. Adding the language-consistency term in Phase 2 increases success by 0.8 percentage points for semantic-only Phase 1 and by 1.2 percentage points when Phase 1 already uses language. Its effect on the selected set is largest for the semantic-only Phase 1 setting, where language consistency rises from 0.850 to 0.883. This pattern is consistent with the main-paper design of using task-level language as a consistency term during Phase 2 redundancy-aware selection rather than as a Phase 1 relevance signal. Because policy success uses one seed and the retrieval measures are diagnostic proxies, the comparison is descriptive rather than a multi-seed performance estimate.

E.4 DCT VERSUS DTW FOR REDUNDANCY ESTIMATION

This experiment tests whether a temporal similarity derived from sequence alignment can replace the DCT-cosine term in Phase 2. We replace the DCT-cosine term with a DTW-derived similarity while keeping the V-JEPA 2 tokens, Phase 1 pre-filtered sets, semantic term, temporal-pattern weight, retrieval budget, language term, policy architecture, and evaluation protocol fixed.

The S-DTW operator used for the baseline sub-trajectory retrievers in Section C.1 locates a short target sub-trajectory within a longer prior trajectory. Phase 2 serves a different role: it compares pairs of candidates that have already been localized by KTS and retained by Phase 1. There is no long trajectory in which to search for an unknown subsequence, so we use ordinary DTW to align both candidate segments in their entirety. This ablation is a controlled replacement of RVSR's temporal redundancy term, not a reproduction of the STRAP or COLLAGE retrieval pipeline.

For V-JEPA 2 token sequences $X = (x_1, \ldots, x_L)$ and $Y = (y_1, \ldots, y_M)$ of lengths L and M, respectively, we use the DTW recurrence defined in Section C.1 with local Euclidean cost $\delta(x_i, y_j) = \|x_i - y_j\|_2$. Let $D(L, M)$ denote the final accumulated cost. The implementation normalizes it by the sum of the sequence lengths:

$$d_{\mathrm{DTW}}(X, Y) = \frac{D(L, M)}{L + M}. \tag{11}$$

The division by $L + M$ reduces the dependence of the accumulated cost on the durations of the two segments. Thus, variable-length segments can be compared without temporal resampling. For each query q, let $d_q^{\min}$ and $d_q^{\max}$ be the minimum and maximum distance over distinct pairs in its pre-filtered set $\mathcal{C}_q$. We convert distance to a query-local similarity so that larger values indicate greater redundancy, as required by the selection objective:

$$\mathrm{Sim}_{\mathrm{DTW}}^{(q)}(c, c') = 1 - \frac{d_{\mathrm{DTW}}(\Phi_c, \Phi_{c'}) - d_q^{\min}}{d_q^{\max} - d_q^{\min} + \epsilon}, \tag{12}$$

with $\epsilon = 10^{-9}$ and diagonal entries set to one. The DTW variant then replaces the main paper's Sim_2 by

$$\begin{aligned} \mathrm{Sim}_2^{\mathrm{DTW}}(c, c') = (1 - \beta) \cos\big(z_{\mathrm{sem}}(c), z_{\mathrm{sem}}(c')\big) \\ + \beta \, \mathrm{Sim}_{\mathrm{DTW}}^{(q)}(c, c'), \end{aligned} \tag{13}$$

using the adopted $\beta = 0.2$, with the DCT cosine replaced by the DTW similarity defined above.

This comparison therefore replaces the complete temporal redundancy term, including its similarity construction; it is not an isolation of the DCT transform alone.

Table 24: Controlled comparison of the temporal redundancy term. Policy success is mean±standard deviation over three policy seeds. All other retrieval and policy settings are fixed. Retrieval time is wall-clock mean±standard deviation over three runs.

Temporal term	Success (%)	Retrieval time (s)
DCT cosine (adopted)	66.4 ± 1.3	37.81 ± 0.70
DTW	59.9 ± 2.1	223.77 ± 1.25

Replacing DCT cosine with DTW reduces mean success by 6.5 percentage points in this controlled comparison (Table 24). This result supports the adopted DCT-cosine term in the evaluated RVSR configuration. The operators also have different retrieval costs. We measure the retrieval stage on the same CPU host (two Intel Xeon Gold 6234 processors), using the same 19,000-segment prior pool, 336 queries, $N_c = 30$, and $K = 5$. Each run loads the stored segment representations and produces the retrieval CSVs for all ten LIBERO-10 tasks; retrieved-training-dataset construction, embedding, and policy training are excluded. Section I separately reports target-side pipeline times that include output-dataset construction. DTW takes $5.92\times$ the wall-clock time of DCT in this setting. DCT features can be computed once per segment and compared by cosine, whereas DTW performs pairwise dynamic programming over the variable-length token sequences in each pre-filtered set.

F Retrieval Diagnostics

The segmentation-level low-activity analysis appears with the KTS implementation in Section C.5. Here we first characterize the Phase 1 pre-filtered set and then compare relevance-only V-JEPA 2

Top-K selection with RVSR. Both selectors use the same encoder, KTS segments, prior pool, 336 target queries, and $K = 5$ retrieval budget. The following measures are diagnostics of the query-wise selected sets, not ground-truth labels of behavioral correctness.

F.1 Semantic Similarity within the Pre-filtered Set

The prior pool contains 19,000 segments. The target set contains 336 query segments from five demonstrations for each of the ten LIBERO-10 tasks. For each query, Phase 1 retains $N_c = 30$ candidates. We compute candidate–candidate cosine similarities within this set and then average the per-query summary statistics over all queries.

For z_{sem}, the mean similarity, within-query standard deviation, and mean minimum are 0.988, 0.006, and 0.970, respectively. For the non-DC DCT block $\bar{u}_{1:D}$, the corresponding values are 0.817, 0.101, and 0.515. Within the Phase 1 pre-filtered sets, candidate pairs have tightly concentrated semantic similarities, while their non-DC DCT-block similarities vary more widely. The DCT block thus provides an additional source of variation for redundancy estimation.

F.2 Action-DTW Near-Duplicate Rate

For this diagnostic, each segment is represented by an action sequence $A = (a_1, \ldots, a_T)$, where $a_t \in \mathbb{R}^7$. To bound the pairwise diagnostic computation, sequences longer than 16 steps are uniformly subsampled to 16; shorter sequences are retained. Let m_j and M_j be the minimum and maximum of action dimension j in the prior pool. We scale that dimension to $[-1, 1]$ as

$$\widetilde{a}_{t,j} = 2\frac{a_{t,j} - m_j}{M_j - m_j} - 1. \tag{14}$$

For scaled sequences X and Y of lengths L and M, we use the DTW recurrence in Section C.1 with Euclidean local cost and define

$$d_A(X, Y) = \frac{D(L, M)}{L + M}. \tag{15}$$

Thus, the accumulated cost is normalized by the sum of the sequence lengths, not by the selected warping-path length.

We calibrate the near-duplicate threshold using 10,000 random pairs of distinct prior segments from the same source task, with source tasks sampled uniformly. The primary threshold is the 10th percentile of this reference distribution, $\tau = 0.1795$. A selected pair is a near duplicate when its normalized Action-DTW distance is at most τ. For each query, we measure the fraction of the $\binom{K}{2}$ selected pairs satisfying this condition, average queries within each target task, and then macro-average the ten tasks. For the selected set $\mathcal{S}_q = \{c_1, \ldots, c_K\}$, let A_{c_i} denote the processed action sequence of segment c_i. The three reported action diagnostics are

$$\begin{aligned} D_{\text{mean}}(q) &= \frac{1}{\binom{K}{2}} \sum_{i<j} d_A(A_{c_i}, A_{c_j}), \\ D_{\min}(q) &= \min_{i<j} d_A(A_{c_i}, A_{c_j}), \\ R_{\text{ND}}(q) &= \frac{1}{\binom{K}{2}} \sum_{i<j} \mathbf{1}\left[d_A(A_{c_i}, A_{c_j}) \le \tau\right]. \end{aligned} \tag{16}$$

Table 25: Phase 2 selected-set diagnostics. The near-duplicate rate uses the calibrated 10th-percentile Action-DTW threshold. Values are macro-averaged over the ten LIBERO-10 tasks.

Metric	V-JEPA 2 Top-K	RVSR
Action-space redundancy		
Near-duplicate rate (%)	43.1	23.9
Mean pairwise DTW distance	0.247	0.330
Minimum pairwise DTW distance	0.136	0.169
Semantic relevance		
Mean query–selected semantic cosine	0.990	0.987

As shown in Table 25, RVSR reduces the near-duplicate rate from 43.1% to 23.9%, a difference of 19.2 percentage points (task-level bootstrap 95% interval: 15.6–23.1 points). The percentile interval uses 10,000 bootstrap resamples of the ten paired task-level differences. The same direction holds when the 5th- or 20th-percentile calibration threshold is used: the Top-K/RVSR rates are 23.0/9.2% and 78.6/62.5%, respectively. We also repeat the complete calibration and comparison with 8-step, 32-step, and uncapped action sequences. The corresponding Top-K/RVSR near-duplicate rates are 44.3/24.9%, 43.9/24.0%, and 44.7/24.0%, preserving the same directional difference. Mean query–selected semantic cosine changes only from 0.990 to 0.987. Action-DTW remains an operational action-similarity proxy; larger distances are not inherently preferable without considering relevance. This comparison evaluates the complete RVSR selector, including language consistency, rather than isolating the DCT component.

G Retrieval Distributions and Visual Context for Mug-Mic

Overall distribution (left). We compute each source task's share using the training sampling probabilities over the retrieved data and normalize the shares to 100%. For COLLAGE, we calculate these shares within each modality and combine them using its modality sampling weights. For Mug-Mic, these are 0.1799 (DINO), 0.6436 (Flow), 0.0781 (PointNet++), and 0.0984 (Language). Tasks with shares below 5% in every method are grouped as Other.

Distribution across target progress (right). We group target queries into five equal progress bins using their midpoints and plot the source distribution retrieved for each bin. Shares use the same sampling weights and are normalized within each bin. COLLAGE's language retrieval has no query-level correspondence, so this panel includes only DINO, Flow, and PointNet++. The behavior labels approximately describe target progress rather than annotated action boundaries.

Figures 5 and 6 provide representative observations of the target trajectory and source tasks.

H Real-Robot Experimental Setup

We use a Rainbow Robotics RB3-730 six-degree-of-freedom (6-DoF) arm (Rainbow Robotics, n.d.) with a third-person Intel RealSense D455 and a wrist-mounted D405. Demonstrations are collected with a 3Dconnexion SpaceMouse, and both collection and policy execution run at 10 Hz. Figure 7 shows the robot platform.

We collect target and prior demonstrations on the same RB3-730 platform and control interface, allowing the retrieval comparison to use a common action space without cross-embodiment mapping.

The target set contains three demonstrations for each of three pick-and-place tasks. The prior pool contains ten demonstrations for each of ten tasks (100 trajectories total). Each trajectory records both RGB views, robot states, actions, and the task instruction. Figures 8 and 9 visualize the target and prior tasks, respectively; Table 26 lists their full instructions.

Two target tasks—Blue Pen in White Basket and Pencil Eraser in Red Cup—have task-matched demonstrations in the prior pool: their instructions and object–destination pairs are the same. The target and prior demonstrations were collected independently with different surrounding-object configurations, rather than repeated in an identical scene. The USB Adapter on Green Plate task has no exact object–destination match in the prior pool. Target and prior trajectories remain disjoint in every case. The study therefore covers both familiar tasks with scene variation and a new object–destination combination.

Policy Training. All real-robot methods use the same BC Transformer architecture and optimization settings as in simulation. We train each policy for 100 epochs and evaluate the checkpoint from epoch 100; unlike the simulation protocol, there is no best-checkpoint selection across epochs.

Table 26: Language instructions for the self-collected RB3 prior and target tasks.

Task	Instruction
Prior tasks (10 demonstrations each)	
Battery in Blue Basket	pick up the battery and put it in the blue basket
Blue Pen on Pink Plate	pick up the blue pen and put it on the pink plate
Blue Pen in White Basket	pick up the blue pen and put it in the white basket
Pencil Eraser in Blue Cup	pick up the pencil eraser and put it in the blue cup
Pencil Eraser on Pink Plate	pick up the pencil eraser and put it on the pink plate
Pencil Eraser in Red Cup	pick up the pencil eraser and put it in the red cup
Red Pen on Green Plate	pick up the red pen and put it on the green plate
Tape in Blue Cup	pick up the tape and put it in the blue cup
USB Adapter in Blue Basket	pick up the USB adapter and put it in the blue basket
USB Adapter on Pink Plate	pick up the USB adapter and put it on the pink plate
Target tasks (3 demonstrations each)	
Blue Pen in White Basket	pick up the blue pen and put it in the white basket
Pencil Eraser in Red Cup	pick up the pencil eraser and put it in the red cup
USB Adapter on Green Plate	pick up the USB adapter and put it on the green plate



Figure 5: Representative progress states for the Mug-Mic target task: reach the mug, grasp and lift, transfer, insert the mug into the microwave, and close the door. These observations illustrate the approximate behavior labels in Figure 3.

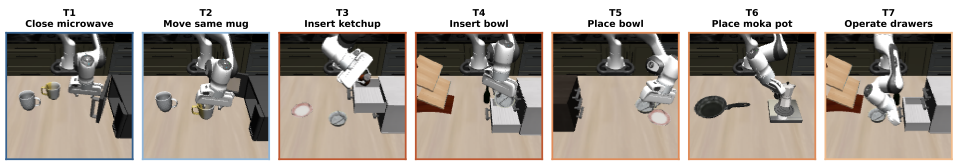


Figure 6: Representative observations from source tasks 1–7 in Figure 3. Tasks 1 and 2 are same-scene sources, whereas tasks 3–7 are from different scenes.



Figure 7: Rainbow Robotics RB3-730 arm used for real-robot data collection and policy evaluation (Rainbow Robotics, n.d.).

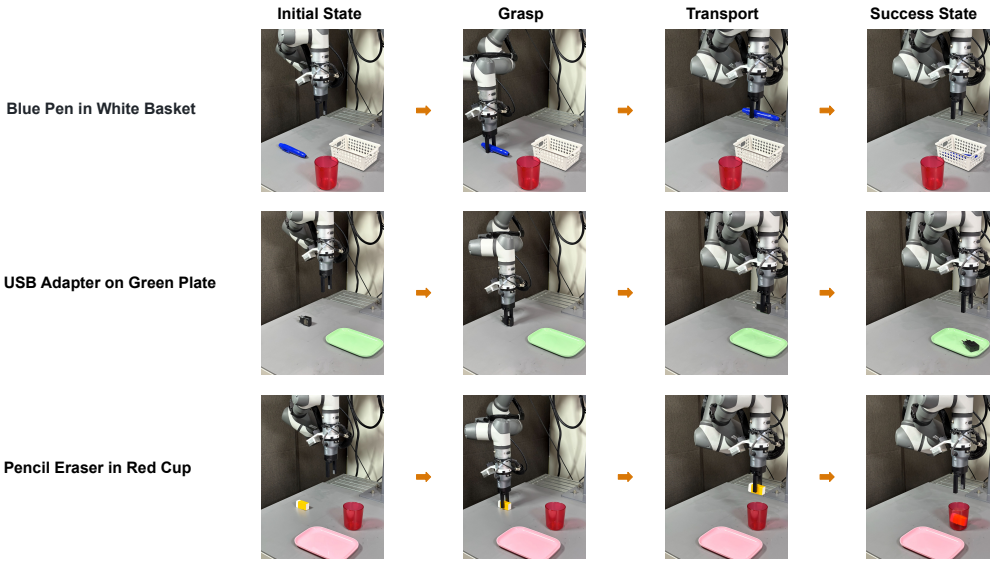


Figure 8: Real-robot target tasks. Each row shows representative initial, grasp, transport, and successful terminal states for Blue Pen in White Basket, USB Adapter on Green Plate, and Pencil Eraser in Red Cup.

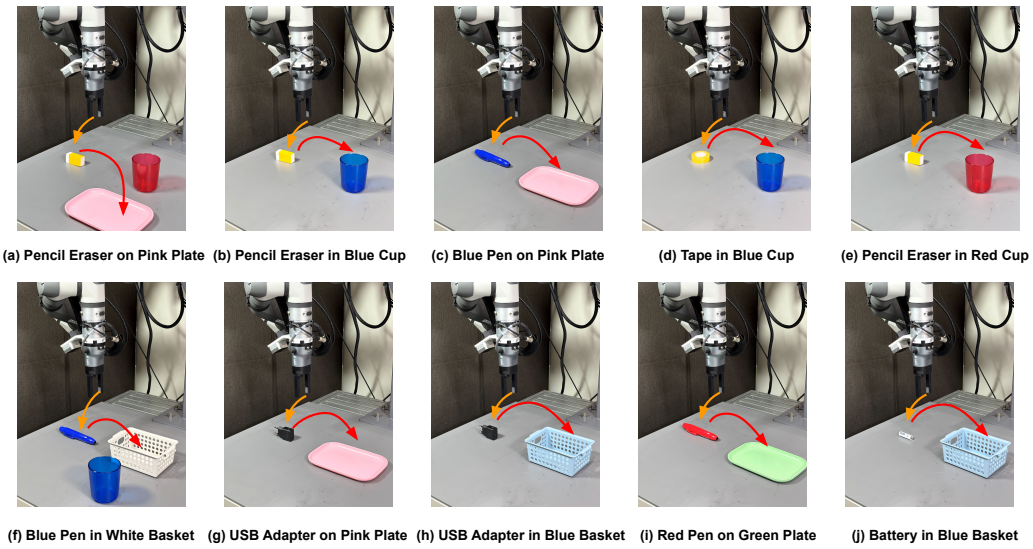


Figure 9: The ten real-robot prior tasks, with one representative observation and the corresponding object-to-destination motion for each task.

I Computational Cost and Infrastructure

I.1 Target-Side Runtime

We profile the exact target set used in the simulation experiments: five demonstrations for each of the ten LIBERO-10 tasks (50 trajectories and 13,655 frames). Retrieval searches the complete LIBERO-90 prior pool. The time spent computing prior representations is excluded because these representations are computed once and reused across target tasks. Each value in Table 27 is a single-run wall-clock measurement that includes initialization, data loading, computation, and output writing. For GPU stages, the ten tasks are divided evenly across the two RTX 4090 GPUs, and the table reports the slower shard.

Table 27: Target-side runtime over all ten LIBERO-10 tasks. Embedding stages use two RTX 4090 GPUs; retrieval uses the complete LIBERO-90 prior pool. Reusable prior-pool preprocessing is excluded.

Method and stage	Time (s)
STRAP: DINOv2 embedding	44.58
STRAP: S-DTW retrieval	480.01
RVSR: full-trajectory encoding	447.29
RVSR: KTS	1.13
RVSR: segment re-encoding	151.02
RVSR: retrieval	82.07

STRAP uses 288 EEF-pause queries and retrieves 1,440 segments; RVSR uses 336 KTS queries and retrieves 1,680 segments. Both therefore retain $K = 5$ segments per query. The measured target-side totals are 524.59 s for STRAP and 681.51 s for RVSR. RVSR spends more time on video encoding and independent segment re-encoding, while its retrieval stage is shorter in the measured pipeline. The 156.92-s total difference is an offline cost incurred before policy training.

These are pipeline measurements rather than an implementation-normalized kernel benchmark. In particular, the released STRAP path processes tasks separately, whereas the RVSR retrieval implementation processes all ten tasks in one invocation. The DCT–DTW retrieval comparison in Table 24 has a different scope: it excludes construction of the retrieved training HDF5 and compares DCT with DTW inside RVSR.

I.2 Hardware and Software

Table 28 summarizes the computing environments. Main experiments used RTX 4090 GPUs; some additional sensitivity experiments used RTX PRO 6000 Blackwell GPUs. Each sensitivity result in Section D is reported relative to the adopted configuration rerun in the same computing environment. Each job uses one GPU.

Table 28: Computing environments.

Experiments	Environment
Main	2× NVIDIA RTX 4090 (24 GB each); Xeon/EPYC hosts; Ubuntu 20.04; Python 3.10; PyTorch 2.5.1+cu124; CUDA 12.4; cuDNN 9.1
Additional sweeps	4× NVIDIA RTX PRO 6000 Blackwell (96 GB each); AMD EPYC 9654; Ubuntu 22.04.5; Python 3.10; PyTorch 2.11.0+cu128; CUDA 12.8; cuDNN 9.19