Extracting Visual Plans from Unlabeled Videos via Symbolic Guidance

Wenyan Yang[1], **Ahmet Tikna**[2], **Yi Zhao**[1], **Yuying Zhang**[1], **Luigi Palopoli**[2], **Marco Roveri**[2], **Joni Pajarinen**[1]

[1]Department of Electrical Engineering and Automation, Aalto University
[2]Department of Engineering and Computer Science, University of Trento

Abstract: Visual planning, by offering a sequence of intermediate visual subgoals to a goal-conditioned low-level policy, achieves promising performance on long-horizon manipulation tasks. To obtain the subgoals, existing methods typically resort to video generation models but suffer from model hallucination and computational cost. We present Vis2Plan, an efficient, explainable, and white-box visual planning framework powered by symbolic guidance. From raw, unlabeled play data, Vis2Plan harnesses vision foundation models to automatically extract a compact set of task symbols, which allows building a high-level symbolic transition graph for multi-goal, multi-stage planning. At test time, given a desired task goal, our planner conducts planning at the symbolic level and assembles a sequence of physically consistent intermediate subgoal images grounded by the underlying symbolic representation. Our Vis2Plan outperforms strong diffusion video generation-based visual planners by delivering a 53% higher aggregate success rate in real robot settings while generating visual plans 35$\times$ faster. The results indicate that Vis2Plan is able to generate physically consistent image goals while offering fully inspectable reasoning steps. More details are available on here: vis2plan.

Keywords: Offline Imitation Learning, Planning

1 Introduction

Learning versatile and long-horizon manipulation skills from uncurated and unlabeled play data remains challenging for robot learning. The play dataset usually consists of long trajectories, which are composed of several unknown sub-skills that solve complex and multi-stage tasks without task labels. Such task-agnostic datasets contain rich information for solving multi-stage and versatile tasks. However, extracting long-horizon visual plans from play data is challenging: 1) the planner needs to discover and stitch sub-skills from unlabeled data; 2) the planner should generate physically reachable subgoals for multi-stage tasks. A practical approach for learning versatile skills is through semantically grounding tasks and applying hierarchical imitation learning. Recent advances in video generative models and vision-language models have shown promise as high-level planners by synthesizing intermediate visual subgoals based on language descriptions or specifying visual targets [1, 2, 3, 4, 5]. However, three core challenges limit their applicability in real-world robot learning. First, these video generative models or vision-language models conduct planning directly in pixel space or dense latent space by forecasting the future in a black-box way [6, 7, 8, 9], which struggle to reason reliably over long horizons due to model hallucination [10, 11, 12, 13, 14, 15]. Second, planning via video generative models or vision-language models is extremely slow, making these methods expensive to apply in real-world applications, especially real-time systems. Third, these approaches require heavy human effort to label unstructured play datasets: video demonstrations must be manually labeled with natural language descriptions. Despite recent vision-language models (VLMs) showing great video understanding ability, they still face challenges in reliably generating correct task descriptions [10, 11, 12, 16].

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

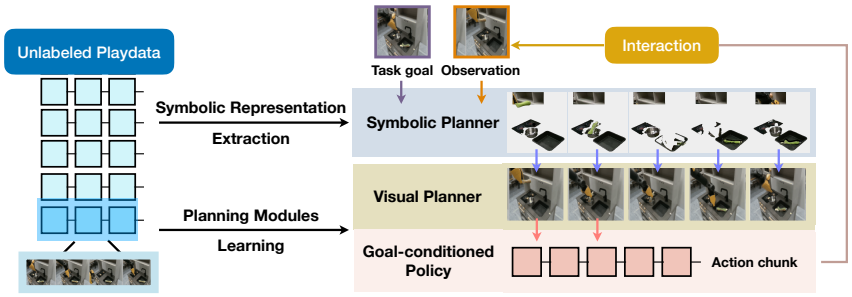


Figure 1: **Vis2Plan**: Given an unlabeled play dataset, Vis2Plan first constructs a discrete transition graph, where nodes are represented by a combination of object-centric symbols extracted from the dataset. At inference time, given a task goal specification, a symbolic planner plans a sequence of symbolic states, which guides visual planning via image retrieval and reachability estimation. Finally, the visual subgoals will guide a low-level policy to generate action chunks to accomplish the task.

To overcome these limitations, we introduce *Vis2Plan*, a symbolic-guided visual planning framework that combines the representational power of vision foundation models with the efficiency and interpretability of white-box planning. Instead of training video generators, Vis2Plan first extracts a compact set of task symbols from raw, unlabeled human play data using pretrained vision foundation models (VFMs). VFMs serve as object detectors and provide object-centric representations to invent symbols to describe robot task states. These combinations of object-centric symbols serve as nodes in a discrete transition graph, capturing stable object states and the effects of primitive manipulation skills. At test time, given a desired goal symbol extracted from a goal image specified by users, Vis2Plan then performs a search over the symbolic graph to find a sequence of symbolic state transitions. Based on the symbolic plan, Vis2Plan retrieves a batch of images based on each symbolic subgoal's label from the dataset and filters them through a reachability estimator to ensure physical achievability. The selected sequential images form a visual plan for the goal-conditioned low-level controller. Finally, a goal-conditioned controller generates action chunks to accomplish subgoals. The whole hierarchical planning is executed in a closed-loop manner as shown in Figure 1.

Our approach offers several key advantages. 1)By decoupling high-level visual planning and low-level control, we reduce computational cost and eliminate the need for large, task-specific action annotations. 2)Symbolic search guarantees correctness and interpretability of the plan, and the combination of symbolic image sampling and a reachability estimator prevents hallucinations and preserves photo-realism. Moreover, 3) training relies only on unlabeled play data; no human-provided task descriptions or reward annotations are required. We evaluate Vis2Plan in both simulation and real robotic manipulation scenarios, comparing against end-to-end goal-conditioned policies, generative video planners, and hybrid search-learn methods. In experiments with a real robot, across a suite of single- and multi-stage tasks, our framework consistently outperforms the diffusion video generative planning baseline in both subgoal completion and long-horizon success rate by 53%. Regarding inference speed, Vis2Plan is capable of generating visual plans with an average of 0.03 seconds – demonstrating robustness, efficiency, and fully inspectable reasoning.

2 Related Work

Learning from unlabeled play data One alternative imitation learning paradigm is multi-task learning from uncurated play data, a form of teleoperated robot demonstration provided without a specific goal. With play data, one typically assumes that the collected demonstrations are from a rational teleoperator with possibly some latent intent to explore the environment [17]. Note that, unlike standard offline-RL datasets [18], play-like behavior datasets neither contain fully random actions, task descriptions, nor have rewards associated with the demonstrations [17, 19, 20]. Previous work mostly focuses on training an end-to-end or hierarchical goal-conditioned visuomotor policy to learn the versatile behaviours [17, 21]. Recent work proposes to leverage Vision-Language foundation models to label play data to train a policy [22] or extract planning domains [23, 24, 25, 13]. However,

these methods still face the trouble of long-horizon reasoning, requiring fine-tuning with specific human labelling and requiring high amounts of computational resources.

Learning Abstractions from Demonstrations Learning abstractions reduces the complexity of long-horizon planning in high-dimensional domains [26, 27, 28]. Traditional methods rely on hand-designed abstractions [29, 30], while the PDDL format [31] represents actions as planning operators with preconditions and effects. Recent data-driven approaches automatically discover abstractions from interactions [32, 33, 34, 35, 36] or demonstrations [23, 37, 38, 39, 40, 41], but assume given predicates or low-level skill generators, limiting skill extraction from unlabeled multimodal data. Ahmetoglu et al. [42] learns predicates from raw data, yet only in simple environments. LLMs and VLMs enable high-level planning with minimal or no demonstrations [43, 44, 45, 46, 47, 48, 23], leveraging commonsense knowledge for efficient plan generation. However, (1) LLM-generated plans are challenging to refine reliably into low-level actions [44, 47]; (2) VLM-based methods [25, 43, 46, 49, 50, 24] can produce unreliable image descriptions in certain domains; and (3) their scale prohibits real-time planning. In this work, we integrate symbolic planning into hierarchical visual planning by using relatively lightweight VFMs to (i) extract symbolic transitions offline without prior knowledge of predicates or controllers, and (ii) leverage the symbolic skills to guide visual plan generation instead of learning an accurate neuro-symbolic domain.

Hierarchical Planning from Visual Demonstrations Hierarchical frameworks have shown promise in long-horizon tasks. Recent work combines hierarchical frameworks with model-free reinforcement learning and imitation learning to learn flexible skills from unlabeled offline data [51, 52, 53, 54, 55, 56, 57, 21, 58, 59]. These approaches apply unsupervised learning (e.g., a VAE [51, 52, 53, 21]) to discover abstract skill representations. However, these approaches require large datasets to learn a suitable skill prior and still need deep planning models, which limits the scalability and interoperability of their plans. Some methods combine conventional white-box planning with learning-based approaches [60, 61, 62, 63] by performing search-based planning over a learned low-dimensional goal space and using a goal-conditioned policy to achieve subgoals. Yet these methods often rely on simple goal spaces or short-horizon tasks (e.g., Atari games [61]; short-horizon environments [62]), or demand large datasets [64, 65]. From a vision-planning perspective, several works identify temporal action abstractions based on visual similarity to detect meaningful visual skills [66, 67], but they cannot integrate directly with traditional search planning. Other studies employ compositional large language foundation models and generative models to generate image sequences (videos) based on task specifications [68, 69, 70]. Despite impressive advances in foundation models, these approaches demand extensive data and computational resources, yet may still fail due to generating unrealistic subgoals. The key idea of Vis2Plan's hierarchical framework is to discover symbolic abstractions, enabling physically reachable visual planning.

3 Method: Vis2Plan

Vis2Plan comprises three key stages: 1) Coarse symbolic skill extraction, where we leverage VFMs to extract symbolic skills; 2) Planning module learning, where we utilize the symbolic representations and dataset to train planning modules; and 3) Symbolic-guided Visual Planning, the inference stage where we do closed-loop planning and control.

Problem Setting: Learning from unlabeled demonstrations We leverage human play data, where a teleoperator explores the scene (e.g., opening an oven, picking up a pot) driven by curiosity without instructions [71, 17]. More importantly, there is no human labeling work to identify or describe tasks. The unlabeled play dataset is formulated as $\mathcal{D} = \{(o, a)\} \subset \mathcal{O} \times \mathcal{A}$ of observation–action sequences, our goal is to extract a hierarchical planner capable of discovering sub-skills with symbolic abstractions, enabling it to "controllably generate" desired plans.

3.1 Stage 1: Coarse Symbolic Skill Extraction

Extracting long-horizon visual plans from unlabeled play data is challenging due to: 1) each play sequence representing a long-horizon manipulation composed of several unknown sub-skills in

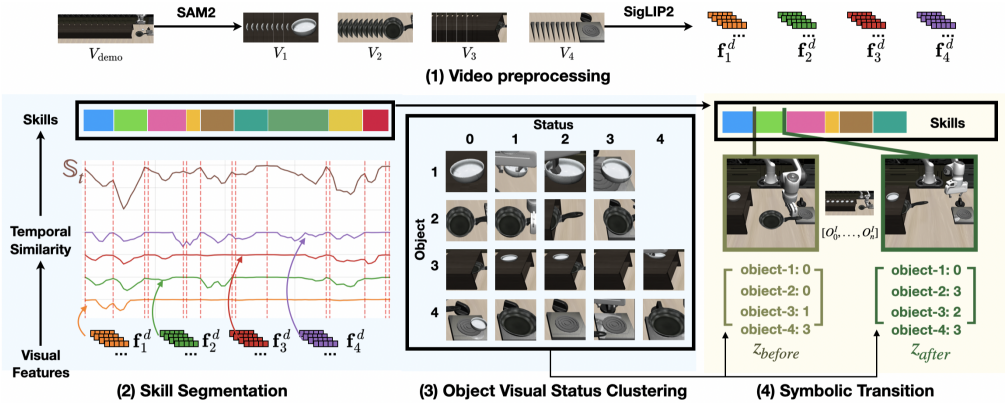


Figure 2: **Processing: symbolic skill extraction**. First, we convert the demonstration videos into object-centric feature sequences (top row). Then we segment the demonstration into sub-skills based on object-centric temporal similarities. We discover the representative status of each object. Finally, we can convert the demonstration video into symbolic transitions.

random orders, and the planner needs to extract and stitch subgoals from different trajectories to form a feasible plan. 2) The observation space being continuous and high-dimensional. To handle these challenges, we should a) reduce the planning horizon by discovering sub-skills and b) simplifying and discretizing the observation space. Thus, we designed a two-step process to extract the symbolic representation of the task: 1) **Stable state identification**, where we aim to divide long-horizon unlabeled demonstrations into sub-skills by identifying key stable states, and 2) **Symbolic state labelling**, where we convert key image observations into discrete states described by symbols.

Stable State Identification First, we preprocess demonstration videos by detecting and tracking task-relevant objects using Qwen [72] and SAM2 [73], creating object-centric feature sequences (encoded with SigLIP2 [74]). Figure 2 (1) illustrates this procedure. We then identify the stable states, where all objects' visual status remain static or similar, and a skill is a transition between two stable states. We compute the temporal similarity between object-centric consecutive frames using cosine similarity of object-centric features, and selecting peaks through non-maximum suppression (Figure 2 (2)) [75]. The peaks identify the representative stable states and naturally divide long-horizon demonstrations into manageable sub-skills. We refer to the appendix for more details.

Symbolic Transition Labelling Next, we want to convert the identified stable states into discrete symbolic transitions. Recent studies have shown that VFMs can serve as a foundation for clustering, yielding informative groupings of embeddings [76, 77, 78]. These findings suggest VFMs can discover representative object states in an unsupervised manner. We apply Agglomerative Clustering [79] with cosine similarity to extract each object's representative visual states. The number of clusters is determined via the Silhouette score [80]. Figure 2 (3) illustrates examples of each object's representative states. Once the unsupervised representative states are discovered, we then train KNN classifiers to label the image frames: we take each sub-skill's precondition (start) and effect (end) frames, and use KNN classifiers to describe each frame's state. Consequently, we obtain several symbolic transitions $(z_{\text{before}}, z_{\text{after}})$ (Figure 2 (4)), which can be used to construct a directed graph for in-domain planning.

3.2 Stage 2: Planning Modules Learning

Based on the symbolic transitions extracted from Stage 1, we can construct a symbolic graph that enables the agent to plan multistage, long-horizon tasks. However, it cannot yet support closed-loop control because: 1) the symbolic extraction stage omits intermediate frames between key states, which contain important information for closed-loop planning; 2) in practice, we found visual subgoals are critical for the low-level policy to execute the plan. Therefore, we train two modules in this stage: i) a next-symbolic-state predictor that predicts the possible next-step symbolic subgoals based on real-time visual feedback, and ii) a reachability estimator that finds sequential images matching symbolic-subgoals to construct a visual plan.

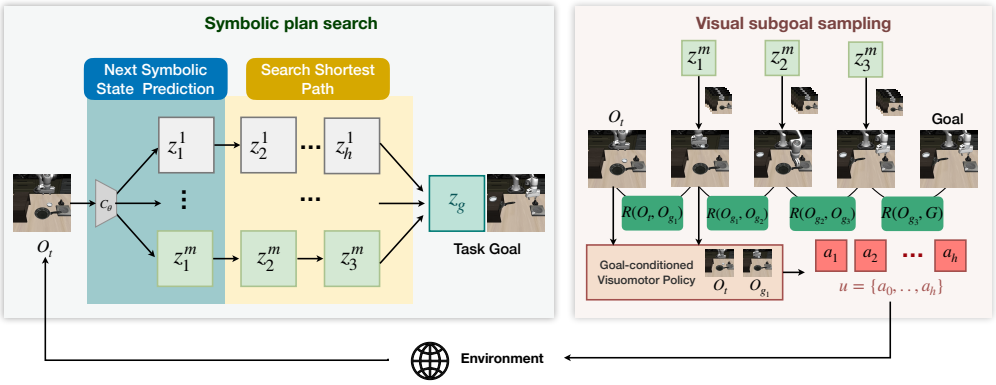


Figure 3: **Symbolic-Guided Visual Planning Framework:** Given an observation O_t, the state predictor $C(O_t)$ outputs a candidate set of next symbolic states $\mathcal{Z} = \{z^1, \ldots, z^m\}$. Using the user-specified task goal z_g, Vis2Plan searches $\mathcal{Z}$ to find the shortest symbolic path from the current state to z_g. Conditioned on this path, Vis2Plan then samples reachable subgoal images from the dataset to assemble a visual plan, which guides the goal-conditioned controller to execute the task.

Next-Symbolic-State Predictor After Stage 1, the planner only has coarse symbolic transitions and cannot yet plan in image space: it must map image observations into symbolic states. Prior works [81, 82, 83] typically train classifiers on images to predict the action phase: "precondition" (state before action execution), "effect" (state after action execution) or "execution" (intermediate states) of each symbolic transition [83]. Yet these methods can be inaccurate. Here, given an image observation O_t, we train a classifier-like predictor C_θ to predict the next symbolic state. During training, we sample symbolic transitions $\{z_{\text{before}}, z_{\text{after}}\}$ and their intermediate frames $[O_0^I, \ldots, O_n^I]$. We optimize C_θ with a cross-entropy loss to predict z_{after} from any intermediate O_i^I:

$$\min_\theta \; -\mathbb{E}_{z_{\text{after}}, O_i^I}\big[\log C_\theta(z_{\text{after}} \mid O_i^I)\big].$$

Reachability Estimator To ensure image subgoals are physically feasible, we train a reachability estimation function R_ψ to measure reachability between visual subgoals. We adopt contrastive reinforcement learning (CRL) [84, 85, 86], which leverages contrastive learning to estimate the discounted state occupancy measure. We modify the MC-InfoNCE loss [84] to the state-only case:

$$\mathcal{L}_{\text{MC-InfoNCE}}(R_\psi)\mathbb{E}_{(o)\sim p(o), o^{(1)}\sim p^\pi(o_{t+}|o), o^{(i)}\sim p(o)}\left[\log \frac{\exp\big(R_\psi(o, o^{(1)})\big)}{\sum_{i=1}^N \exp\big(R_\psi(o, o^{(i)})\big)}\right], \tag{1}$$

where $p(o)$ is the visual observation distribution of unlabeled play dataset $\mathcal{D}$, and $o^{(1)} \sim p^\pi(o_{t+} \mid o)$ denotes hindsight sampling of future observations. Since our π_ψ imitates the offline dataset policy, this state-only on-policy MC-InfoNCE loss can estimate the state-occupancy measure of π_ϕ. Consequently, $R_\psi(o_i, o_j)$ serves as a reachability score: higher values indicate better physical reachability from o_i to o_j for π_ϕ based on offline dataset. We also calculate a reachability threshold δ here: $\delta = \min(R_\psi(o_t, o_{t+}))$.

For the low-level policy, we implement a goal-conditioned visuomotor policy π_g [87, 71] as our low-level controller. Given the current observation O_t and the next image goal O_{t+h} from a video plan, π_g outputs an action chunk $a_{t:t+h}$ that advances toward O_{t+h}, which the agent then executes for a fixed number of timesteps. We train π_g on paired snippets $(O_{i:i+h}, a_{i:i+h})$ by sampling a random start index i and horizon h, and optimizing:

$$\max_\phi \; \mathbb{E}_{(O_{i:i+h}, a_{i:i+h})\sim D}\big[\log \pi_\phi(a_{i:i+h} \mid O_i, O_{i+h})\big]. \tag{2}$$

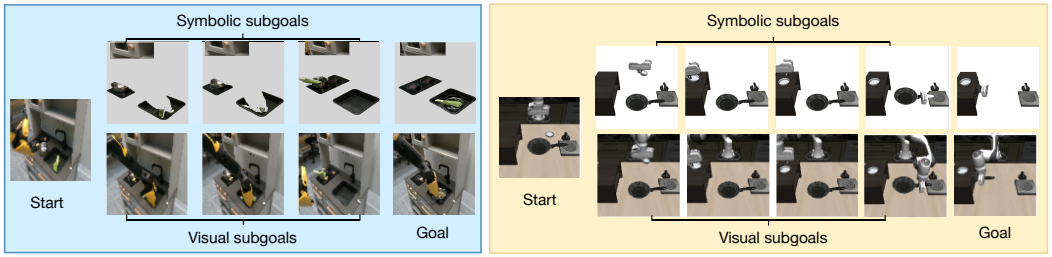
3.3 Stage 3: Symbolic-guided Visual Planning

Combining the modules learned from stage 2, we now have all the ingredients for Vis2Plan. To plan to achieve a certain specified task goal, Vis2Plan will make three-stage planning: **1) symbolic**

planning: given current image observation O_t and task symbolic goal, C_θ will predict a set of next symbolic state candidates $Z_{\text{next}} = \{z_1^1, ..., z_1^m\}$. We refer to the symbolic example in Fig.2. By applying A* search over the symbolic graph for each candidate in Z_{next}, we choose the shortest path plan as symbolic subgoal sequence $\tau_z = \{z_1, ..., z_H, z_g\}$. **2) Visual subgoal generation:** For each symbolic subgoal z_i in τ_z, we can sample z_i's corresponding visual observation based on the relabelled dataset $O_i \sim P_D\big(\cdot \mid z_i\big)$. Given the sampled symbolic subgoal related images, the planner uses beam-search to find a sequence of visual subgoals $\tau_O = \{O_1, .., O_h\}$ that can maximize this objective

$$\arg\max_{\tau_O} R(\tau_O) \;=\; -\sum_{i=0}^{H-1} R_\psi(O_i, O_{i+1}) \quad \text{s.t.} \quad R_\psi(O_i, O_{i+1}) \leq \delta,\ \forall i, \tag{3}$$

3) Goal-conditioned action chunk generation: Based on the visual plan, π_g will generate an h-horizon action chunk $u = \{a_0, .., a_h\}$ based on the current visual observation O_t and the next visual subgoal O_g: $u = \pi_g(O_t, O_g)$. Figure 4 presents examples of visual plans.



(a) pot green onion in pot and put pot in sink (b) pot bowel on shelf and pan in shelf

Figure 4: Examples of Vis2Plan's visual plan: a real robot kitchen task (left) and a LIBERO simulation kitchen task (right).

4 Experiments

Setting We evaluate the methods in both simulation (LIBERO Mimicplay kitchen [88, 71]) and a real world task (kitchen manipulations). Tasks are categorized based on the number of subgoals: single-stage tasks (e.g., grasp the bowel) and multi-stage tasks (e.g., grasp the bowel and put it on the stove). The sub-task horizon is between 140 and 400 action steps (short-horizon), and approximately between 500 and 1400 action steps for multi-stage tasks (long-horizon) with a 10Hz control frequency. We use the play data from [71] as offline dataset for simulation tasks. For the real robot experiments, we collected 15 teleoperated human play trajectories, each trajectory potentially contains 5 to 15 sub-tasks. We use the task success rate as an evaluation metric. For more details about the environments and simulation results, please refer to the Appendix.

Baselines **1) End-to-end goal-conditioned policies:** Goal-conditioned (GC) policies naturally handle multiple tasks and can be trained via goal-conditioned behavior cloning with hindsight relabeling. These serve as our low-level controllers. We evaluate four GC-BC variants: *GC-RNN*, *GC-Transformer*,*GC-Diffusion* from [71] and *GC-ACT*, an image goal-conditioned variant of [89]. **2) Hierarchical planners:** each of these methods will have a high-level visual planner generating image subgoals and a low-level goal-conditioned visuomotor policy (GC-diffusion). We implemented a) *AVDC* [1], which trains a video generative diffusion model to make a visual plan. b) graph-search retrieval (*GSR*) [90], which builds a graph over image latents and connects two vertices if their latent features are similar. c) *UVD-graph*, which uses UVD [91] to segment long-horizon tasks into subgoals, and then builds a graph similarly to GSR. For hierarchical planner baselines, we adopt Adaflow [87], a flow-based generative model, as low-level policy for LIBERO simulation tasks; and we use Mimicplay policy [71] for real robot tasks. Additionally, we note that methods in [37, 6, 92], despite sharing similar hierarchical control methodologies, failed on all tasks and are thus omitted from further comparisons.

Table 1: Performance comparison in MimicPlay LIBERO simulation environment. Results show success rates for both single-goal (short-horizon) and multi-goal (long-horizon) tasks with standard deviations across 5 seeds. Bold values indicate the highest performance. Our Vis2Plan consistently outperforms baselines, particularly excelling in challenging long-horizon tasks where end-to-end and other hierarchical approaches struggle.

	Short Horizon (1 goal)				**Long Horizon ($\geq$ 2 subgoals)**				
	Task-1	Task-2	Task-3	Task-4	Task-1	Task-2	Task-3	Task-4	Task-5
GC-RNN	$\mathbf{0.96}_{\pm 0.02}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
GC-Transformer	$0.95_{\pm 0.03}$	$0.01_{\pm 0.02}$	$0.0_{\pm 0.0}$	$0.01_{\pm 0.02}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
GC-Diffusion	$0.70_{\pm 0.05}$	$0.81_{\pm 0.03}$	$0.92_{\pm 0.03}$	$0.34_{\pm 0.03}$	$0.02_{\pm 0.00}$	$0.04_{\pm 0.00}$	$0.14_{\pm 0.00}$	$0.28_{\pm 0.00}$	$0.14_{\pm 0.00}$
GC-ACT	$0.95_{\pm 0.03}$	$0.01_{\pm 0.02}$	$0.0_{\pm 0.0}$	$0.01_{\pm 0.02}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
GSR	$0.0_{\pm 0.0}$	$0.81_{\pm 0.04}$	$0.00_{\pm 0.0}$	$0.35_{\pm 0.10}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
UVD-graph	$0.58_{\pm 0.15}$	$0.36_{\pm 0.10}$	$0.00_{\pm 0.04}$	$\mathbf{0.36}_{\pm 0.05}$	$\mathbf{0.18}_{\pm \mathbf{0.03}}$	$\mathbf{0.24}_{\pm 0.09}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$
AVDC	$0.71_{\pm 0.11}$	$0.26_{\pm 0.12}$	$0.48_{\pm 0.09}$	$\mathbf{0.36}_{\pm 0.05}$	$0.00_{\pm 0.11}$	$0.04_{\pm 0.15}$	$0.66_{\pm 0.10}$	$0.54_{\pm 0.08}$	$0.46_{\pm 0.09}$
Vis2Plan (ours)	$0.78_{\pm 0.10}$	$\mathbf{0.94}_{\pm 0.04}$	$\mathbf{0.94}_{\pm 0.05}$	$0.32_{\pm 0.11}$	$0.14_{\pm 0.10}$	$\mathbf{0.24}_{\pm 0.09}$	$\mathbf{0.72}_{\pm 0.10}$	$\mathbf{0.56}_{\pm 0.09}$	$\mathbf{0.82}_{\pm 0.08}$

Performance in Simulation We first compare Vis2Plan with baseline approaches in the LIBERO simulation environment [71, 88]. Table 1 reports the task success rate of our approach Vis2Plan and baselines. First, the results indicate that end-to-end goal-conditioned approaches such as GC-RNN and GC-Transformer generally struggle, only accomplishing the short-horizon Task-1 (turning on the stove knob). However, GC-Diffusion achieves modest performance on some tasks: 0.81 on Task-2 subgoal and up to 0.28 on a long-horizon task. Based on the end-to-end baseline results, we opt to use GC-diffusion as the low-level goal-conditioned visuomotor policy for all hierarchical frameworks. Hierarchical solutions such as GSR and UVD-graph show reasonable subgoal-level results but suffer from inaccurate high-level subgoal planning, leading to a noticeable drop in success rates for longer tasks. In contrast, both mimicplay and diffusion variants of Vis2Plan consistently outperform other baselines, attaining high success rates across most subgoal tasks: 0.95 and 0.94 on Task-1 and Task-3. Vis2Plan especially excels on challenging long-horizon scenarios, yielding 0.59 and 0.82 on Task-4 and Task-5, respectively. Comparing the end-to-end GC-Diffusion with GC-Diffusion-integrated hierarchical planners, we can see that AVDC and our Vis2Plan improve the task performance over solely using GC-Diffusion, which indicates meaningful visual subgoal plans can lead to better task success rates. In the following paragraph, we will conduct visual plan quality analysis which discovers how high-level planners affect short/long-horizon task performances.

Real Robot Experiments Table 2 reports real-world performance in short-horizon (Tasks 1–3) and long-horizon (Tasks 4–6) scenarios. In this real setup, we select the hierarchical planning frameworks GSR, UVD-graph, AVDC, and GC-diffusion as baselines based on their higher simulation success rates. However, we noticed that the GC-diffusion policy faces challenges in low-level control due to potentially limited training data. Instead, we use Mimicplay policy [71] with sub-trajectory retrieval strategy: it retrieves a sequence of end-effector poses based on the visual subgoals, and feeds the visual-subgoals and the retrieved end-effector pose trajectory to Mimicplay policy for control. We present the real experiment results in Table 2. GC-Diffuser, which performed moderately well in simulation, fails to complete any task in the real world, likely due to limited demonstration data. In contrast, our Vis2Plan approach either matches or exceeds the best-performing methods in five out of six tasks, achieving perfect success (1.0) on Task 1 and Task 6 while outperforming all baselines on Task 5. Although our method achieves slightly lower results for Task 3, it remains competitive and demonstrates overall robustness and scalability for both short and long task horizons.

Table 2: **Real robot experiment.** Success rate (over 11 trials) for each method across six manipulation tasks, grouped by short-horizon (Tasks 1–3) and long-horizon (Tasks 4–6) scenarios. Vis2Plan achieves the highest average performance.

	GC-Diffusion	**GSR**	**UVD-graph**	**AVDC**	**Vis2Plan**
Task-1 (Short)	0.0	0.45	0.09	0.0	1.0
Task-2 (Short)	0.0	0.65	0.18	0.09	0.64
Task-3 (Short)	0.0	0.55	0.55	0.55	0.45
Task-4 (Long)	0.0	0.45	0.36	0.18	0.55
Task-5 (Long)	0.0	0.55	0.55	0.27	0.63
Task-6 (Long)	0.0	0.09	0.09	0.0	1.0
Average	0.0	0.47	0.30	0.18	**0.71**

Visual Plan Quality Analysis We now analyze why the baseline hierarchical planners fail. One assumption is the generated subgoal plans may not be physically achievable or photo-unrealistic.

In our evaluation protocol, we allowed each planner to generate up to 20 plan sequences (represented as image frames) and manually inspected whether each sequence was "meaningful." We use the reachability estimator R_ψ and the reachability threshold δ in Eq. 3 to determine if the plan is meaningful: every adjacent subgoal pair should be physically achievable. The primary evaluation metric was the high-level plan success rate (the ratio of making "meaningful" visual plans). Table 3 shows a comparative analysis of different methods' high-level plan quality in the real world setting, distinguishing between single-task (1Task) and multi-task ($\geq$2Tasks) scenarios. While GSR achieved perfect scores (1.0) for single-task plans, it often incorrectly connected visually similar states, resulting in lower performance (0.57) in multi-task settings. AVDC suffered from generative inconsistencies – such as nonexistent objects including imagining two robot arms instead of the actual single arm, or missing items – causing its score to drop significantly when tasked with multiple goals. UVD-graph demonstrated a reasonable overall success rate (0.75 single-task, 0.86 multi-task); however, it frequently planned overly complicated subgoals, producing prolonged image sequences where lower-level goal-conditioned policies were prone to failure. In contrast, our Vis2Plan approach consistently attained perfect scores (1.0) across all tasks, underscoring its robustness in producing precise, feasible, and coherent high-level plans – a crucial requirement for reliable execution in real-world robotics applications. Figure 5 presents some typical failure cases observed in the baseline visual planning results.

Table 3: **High-level plan quality analysis.** Success rates for single- and multi-task scenarios in simulation and real setups.

	GSR	UVD-graph	AVDC	Ours
Sim. 1 Task	0.23	0.25	0.74	**1.00**
Sim. $\geq$2 Tasks	0.00	0.50	0.48	**1.00**
Real 1 Task	1.00	0.75	0.48	**1.00**
Real $\geq$2 Tasks	0.57	0.86	0.16	**1.00**

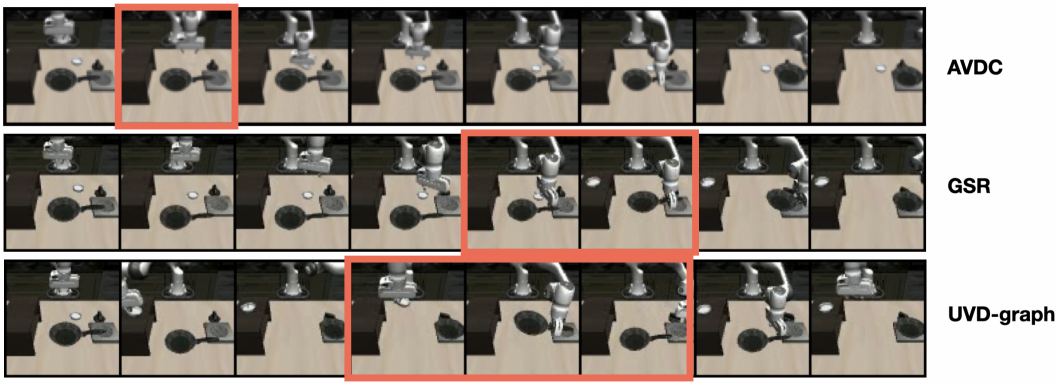


Figure 5: Baseline subgoal generation failure examples. The red bounding box highlights the physically unreachable frames: the bowel disappeared in AVDC generation and UVD-graph; For the GSR plan, the bowel's location changed while the gripper is manipulating the pan.

5 Conclusion

In this paper, we introduced Vis2Plan, a symbol-guided visual planning framework for multi-stage robotic manipulation from unlabeled play data. Given unlabeled play data, Vis2Plan first extracts discrete symbolic-level task abstractions and constructs a symbolic transition graph with pre-trained vision foundation models. The symbolic transition graph enables A* search to derive high-level plans over these abstractions. Conditioning on the planned symbol trajectories, we retrieve and filter images from the dataset to construct physically feasible visual subgoals to be followed by a low-level goal-conditioned policy. We demonstrate that combining unsupervised symbolic extraction from video-based play data with white-box planning leads to strong performance in multi-stage robot tasks. Our experiments – both on the LIBERO simulation benchmark and on a real robotic manipulator – demonstrate robust, interpretable, and efficient performance, paving the way for scalable, inspectable robot learning from unlabeled data.

6 Limitations

From the high-level planning perspective, one limitation is that task-related object names in play data need to be specified by humans. Additionally, the extracted symbolic information still lacks the semantic richness of natural language descriptions, making it unable to generalize to out-of-domain settings. However, a promising research direction would be combining this framework with Vision-Language Models (VLMs) to address their complementary weaknesses: enhancing the generalization ability of Vis2Plan while mitigating the hallucination issues of VLMs. From the robot control perspective, Vis2Plan's performance bottleneck lies in the data-driven low-level goal-conditioned policy. Improving performance would require either collecting more training data or specifically designing task-appropriate low-level skill controllers.

Acknowledgments

This research was supported by the European Union's Horizon Europe research and innovation program under grant agreement (No 101189836) for the XsCAVE project, and the Research Council of Finland for the MARL project (357301). We acknowledge the computational resources provided by the Aalto Science-IT project through the Triton cluster.

References

[1] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum. Learning to act from actionless videos through dense correspondences. *arXiv preprint arXiv:2310.08576*, 2023.

[2] Y. Du and L. Kaelbling. Compositional generative modeling: A single model is not all you need. *arXiv preprint arXiv:2402.01103*, 2024.

[3] Z. Xu, Q. Qiu, and Y. She. Vilp: Imitation learning with latent video planning. *IEEE Robotics and Automation Letters*, 2025.

[4] Y. Qin, Z. Shi, J. Yu, X. Wang, E. Zhou, L. Li, Z. Yin, X. Liu, L. Sheng, J. Shao, et al. Worldsimbench: Towards video generation models as world simulators. *arXiv preprint arXiv:2410.18072*, 2024.

[5] M. Attarian, A. Gupta, Z. Zhou, W. Yu, I. Gilitschenski, and A. Garg. See, plan, predict: Language-guided cognitive planning with video prediction. *arXiv preprint arXiv:2210.03825*, 2022.

[6] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In *International Conference on Learning Representations*.

[7] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

[8] Z. Xing, Q. Dai, Z. Weng, Z. Wu, and Y.-G. Jiang. Aid: Adapting image2video diffusion models for instruction-guided video prediction. *arXiv preprint arXiv:2406.06465*, 2024.

[9] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pages 1838–1849. PMLR, 2023.

[10] A. Favero, L. Zancato, M. Trager, S. Choudhary, P. Perera, A. Achille, A. Swaminathan, and S. Soatto. Multi-modal hallucination control by visual information grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14303–14312, 2024.

[11] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[12] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan. Replanvlm: Replanning robotic tasks with visual language models. *IEEE Robotics and Automation Letters*, 9(11):10201–10208, 2024. doi: 10.1109/LRA.2024.3471457.

[13] A. Ajay, S. Han, Y. Du, S. Li, A. Gupta, T. Jaakkola, J. Tenenbaum, L. Kaelbling, A. Srivastava, and P. Agrawal. Compositional foundation models for hierarchical planning. *Advances in Neural Information Processing Systems*, 36:22304–22325, 2023.

[14] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh. Physically grounded vision-language models for robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12462–12469. IEEE, 2024.

[15] P. Sermanet, T. Ding, J. Zhao, F. Xia, D. Dwibedi, K. Gopalakrishnan, C. Chan, G. Dulac-Arnold, S. Maddineni, N. J. Joshi, et al. Robovqa: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 645–652. IEEE, 2024.

[16] B. Li, P. Wu, P. Abbeel, and J. Malik. Interactive task planning with language models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=VmfWywWuYQ.

[17] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[18] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[19] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. Pmlr, 2020.

[20] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pages 1838–1849. PMLR, 2023.

[21] S. Park, D. Ghosh, B. Eysenbach, and S. Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=cLQCCtVDuW.

[22] N. Blank, M. Reuss, M. Rühle, Ö. E. Yağmurlu, F. Wenzel, O. Mees, and R. Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. *arXiv preprint arXiv:2410.17772*, 2024.

[23] A. Athalye, N. Kumar, T. Silver, Y. Liang, T. Lozano-Pérez, and L. P. Kaelbling. Predicate invention from pixels via pretrained vision-language models. *arXiv preprint arXiv:2501.00296*, 2024.

[24] B. Wang, J. Zhang, S. Dong, I. Fang, and C. Feng. Vlm see, robot do: Human demo video to robot action plan via vision language model. *arXiv preprint arXiv:2410.08792*, 2024.

[25] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling. Guiding long-horizon task and motion planning with vision language models. *arXiv preprint arXiv:2410.02193*, 2024.

[26] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61: 215–289, 2018.

[27] M. Grounds and D. Kudenko. Combining reinforcement learning with symbolic planning. In *European Symposium on Adaptive Agents and Multi-Agent Systems*, pages 75–86. Springer, 2005.

[28] T. Silver. *Neuro-Symbolic Learning for Bilevel Robot Planning*. PhD thesis, Massachusetts Institute of Technology, 2024.

[29] D. P. Bertsekas, D. A. Castanon, et al. Adaptive aggregation methods for infinite horizon dynamic programming. 1988.

[30] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[31] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson, et al. Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[32] Y. Liang, N. Kumar, H. Tang, A. Weller, J. B. Tenenbaum, T. Silver, J. F. Henriques, and K. Ellis. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. *arXiv preprint arXiv:2410.23156*, 2024.

[33] B. Li, T. Silver, S. Scherer, and A. Gray. Bilevel learning for bilevel planning. *arXiv preprint arXiv:2502.08697*, 2025.

[34] A. Ahmetoglu, M. Y. Seker, J. Piater, E. Oztop, and E. Ugur. Deepsym: Deep symbol generation and rule learning for planning from unsupervised robot interaction. *Journal of Artificial Intelligence Research*, 75:709–745, 2022.

[35] S. James, B. Rosman, and G. Konidaris. Autonomous learning of object-centric abstractions for high-level planning. In *Proceedings of the The Tenth International Conference on Learning Representations*, 2022.

[36] S. Cheng and D. Xu. Guided skill learning and abstraction for long-horizon manipulation. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[37] M. Asai and A. Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proceedings of the aaai conference on artificial intelligence*, volume 32, 2018.

[38] M. Asai and A. Fukunaga. Classical planning in deep latent space: From unlabeled images to pddl (and back). In *NeSy*, 2017.

[39] M. Asai, H. Kajino, A. Fukunaga, and C. Muise. Classical planning in deep latent space. *Journal of Artificial Intelligence Research*, 74:1599–1686, 2022.

[40] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4166–4173. IEEE, 2022.

[41] J.-J. Shao, H.-R. Hao, X.-W. Yang, and Y.-F. Li. Learning for long-horizon planning via neuro-symbolic abductive imitation. *arXiv preprint arXiv:2411.18201*, 2024.

[42] A. Ahmetoglu, E. Oztop, and E. Ugur. Symbolic manipulation planning with discovered object and relational predicates. *IEEE Robotics and Automation Letters*, 2025.

[43] Y. Liang, N. Kumar, H. Tang, A. Weller, J. B. Tenenbaum, T. Silver, J. F. Henriques, and K. Ellis. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. *arXiv preprint arXiv:2410.23156*, 2024.

[44] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu. Interpret: Interactive predicate learning from language feedback for generalizable task planning. *arXiv preprint arXiv:2405.19758*, 2024.

[45] W. Liu, N. Nie, R. Zhang, J. Mao, and J. Wu. Blade: Learning compositional behaviors from demonstration and language. In *Conference on Robot Learning (CoRL)*, 2024.

[46] X. Fang, B.-R. Huang, J. Mao, J. Shone, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Keypoint abstraction using large models for object-relative imitation learning. *arXiv preprint arXiv:2410.23254*, 2024.

[47] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. Kaelbling, and M. Katz. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 20256–20264, 2024.

[48] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

[49] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling. Guiding long-horizon task and motion planning with vision language models. *arXiv preprint arXiv:2410.02193*, 2024.

[50] Z. Wang, R. Shen, and B. C. Stadie. Wonderful team: Zero-shot physical task planning with visual llms. *Transactions on Machine Learning Research*.

[51] K. Hakhamaneshi, R. Zhao, A. Zhan, P. Abbeel, and M. Laskin. Hierarchical few-shot imitation with skill transition models. *arXiv preprint arXiv:2107.08981*, 2021.

[52] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626, 2020. doi:10.1109/IROS45743.2020.9340764.

[53] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pages 1838–1849. PMLR, 2023.

[54] W. Shin and Y. Kim. Guide to control: Offline hierarchical reinforcement learning using subgoal generation for long-horizon and sparse-reward tasks. In *IJCAI*, pages 4217–4225, 2023.

[55] C. Schmidt, D. Gammelli, J. Harrison, M. Pavone, and F. Rodrigues. Offline hierarchical reinforcement learning via inverse optimization. *arXiv preprint arXiv:2410.07933*, 2024.

[56] A. Sharma, M. Sharma, N. Rhinehart, and K. M. Kitani. Directed-info gail: Learning hierarchical policies from unsegmented demonstrations using directed information. *arXiv preprint arXiv:1810.01266*, 2018.

[57] J. Clinton and R. Lieck. Planning transformer: Long-horizon offline reinforcement learning with planning tokens, 2024. URL https://arxiv.org/abs/2409.09513.

[58] C. Wu, H. Hu, Y. Yang, N. Zhang, and C. Zhang. Planning, fast and slow: Online reinforcement learning with action-free offline data via multiscale planners. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=HwVZbPbMjw.

[59] B. Li, T. Silver, S. Scherer, and A. Gray. Bilevel learning for bilevel planning. *arXiv preprint arXiv:2502.08697*, 2025.

[60] J. Li, C. Tang, M. Tomizuka, and W. Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022.

[61] K. Kujanpää, J. Pajarinen, and A. Ilin. Hierarchical imitation learning with vector quantized models. In *International Conference on Machine Learning*, pages 17896–17919. PMLR, 2023.

[62] Z.-H. Yin and P. Abbeel. Offline imitation learning through graph search and retrieval. *arXiv preprint arXiv:2407.15403*, 2024.

[63] X. Mao, G. Giudici, C. Coppola, K. Althoefer, I. Farkhatdinov, Z. Li, and L. Jamone. Dexskills: Skill segmentation using haptic data for learning autonomous long-horizon robotic manipulation tasks. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5104–5111. IEEE, 2024.

[64] K. Pertsch, O. Rybkin, F. Ebert, S. Zhou, D. Jayaraman, C. Finn, and S. Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

[65] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829*, 2019.

[66] W. Wan, Y. Zhu, R. Shah, and Y. Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 537–544, 2024. doi:10.1109/ICRA57147.2024.10611129.

[67] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

[68] A. Ajay, S. Han, Y. Du, S. Li, A. Gupta, T. Jaakkola, J. Tenenbaum, L. Kaelbling, A. Srivastava, and P. Agrawal. Compositional foundation models for hierarchical planning. *Advances in Neural Information Processing Systems*, 36:22304–22325, 2023.

[69] S. Zhou, Y. Du, J. Chen, Y. Li, D.-Y. Yeung, and C. Gan. Robodreamer: Learning compositional world models for robot imagination. In *International Conference on Machine Learning*, pages 61885–61896. PMLR, 2024.

[70] Y. Du and L. P. Kaelbling. Position: Compositional generative modeling: A single model is not all you need. In *Forty-first International Conference on Machine Learning*, 2024.

[71] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[72] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[73] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[74] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[75] Z. Du, X. Wang, G. Zhou, and Q. Wang. Fast and unsupervised action boundary detection for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2022.

[76] E. Palumbo, M. Vandenhirtz, A. Ryser, I. Daunhawer, and J. E. Vogt. From logits to hierarchies: Hierarchical clustering made simple. *arXiv preprint arXiv:2410.07858*, 2024.

[77] N. Adaloglou, F. Michels, H. Kalisch, and M. Kollmann. Exploring the limits of deep image clustering using pretrained models. *arXiv preprint arXiv:2303.17896*, 2023.

[78] A. Gadetsky, Y. Jiang, and M. Brbic. Let go of your labels with unsupervised transfer. *arXiv preprint arXiv:2406.07236*, 2024.

[79] B. Walter, K. Bala, M. Kulkarni, and K. Pingali. Fast agglomerative clustering for rendering. In *2008 IEEE Symposium on Interactive Ray Tracing*, pages 81–86. IEEE, 2008.

[80] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[81] D. Aineto, S. Jiménez, and E. Onaindia. Learning strips action models with classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, pages 399–407, 2018.

[82] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3182–3189. IEEE, 2021.

[83] D. Sliwowski and D. Lee. Conditionnet: Learning preconditions and effects for execution monitoring. *IEEE Robotics and Automation Letters*, 2024.

[84] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 35603–35620, 2022.

[85] C. Zheng, B. Eysenbach, H. Walke, P. Yin, K. Fang, R. Salakhutdinov, and S. Levine. Stabilizing contrastive rl: Techniques for robotic goal reaching from offline data. *arXiv preprint arXiv:2306.03346*, 2023.

[86] C. Zheng, R. Salakhutdinov, and B. Eysenbach. Contrastive difference predictive coding. *arXiv preprint arXiv:2310.20141*, 2023.

[87] X. Hu, B. Liu, X. Liu, and Q. Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *arXiv preprint arXiv:2402.04292*, 2024.

[88] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[89] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[90] Z.-H. Yin and P. Abbeel. Offline imitation learning through graph search and retrieval, 2024. URL https://arxiv.org/abs/2407.15403.

[91] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs. Universal visual decomposer: Long-horizon manipulation made easy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6973–6980. IEEE, 2024.

[92] K. Kujanpää, J. Pajarinen, and A. Ilin. Hierarchical imitation learning with vector quantized models. In *International Conference on Machine Learning*, pages 17896–17919. PMLR, 2023.

[93] NaturalPoint, Inc. *OptiTrack Motion Capture System*. NaturalPoint, Inc., Corvallis, OR, USA, 2025. URL https://optitrack.com. Hardware version: *Prime 13W* (or whichever you used);.

Contents

A Experiment details

A.1 Simulation environment

A.1.1 Setups

The simulation tasks are selected from the Kitchen Scene 9 environment (KITCHEN_SCENE9) of the LIBERO benchmark [88], which is built on robosuite and MuJoCo. We choose LIBERO for its BDDL-based goal specification language, which enables concise definitions of diverse tasks and facilitates unified multitask evaluation when learning from play data.

A.1.2 Tasks

Initial state A flat stove is placed in the right region of the kitchen table, with its knob turned off. A frying pan is placed in the central region of the table, and a white bowl is placed on the tabletop just behind the pan. A wooden two-layer shelf is placed along the left edge of the table.

Short Horizon Tasks There are 4 tasks chosen for short-horizon evaluation: Task-1: turn on the stove; Task-2: put bowel on shelf ; Task-3: put bowel on stove ; Task-4: put pan on stove. The goal setups are visualized in Figure 6

Figure 6: Short horizon tasks in LIBERO simulation.

Long Horizon Tasks There are five tasks chosen for short-horizon evaluation: Task-1: turn on stove, pan on stove, bowl on shelf; Task-2: turn on stove, bowl on stove, pan in shelf; Task-3: bowl

on shelf, pan in shelf; Task-4: bowl on stove, pan in shelf; Task-5: pan on stove, bowl on shelf. The goal setups are visualized in Figure 7

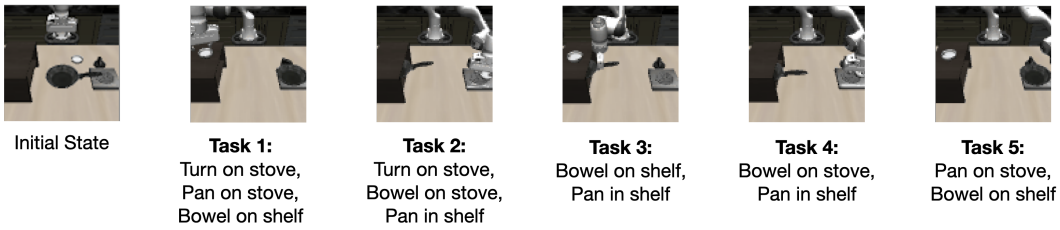


Figure 7: Long horizon tasks in LIBERO simulation.

A.2 Real World Setup

Figure 8: Real kitchen setup.

A.2.1 Setups

Figure 8 illustrates the real-world task setup. We design kitchen object manipulation tasks for the Boston Dynamics Spot robot arm with a control frequency of 7Hz. **1) Observation**: We use a Realsense D435 camera to receive a third-person view 512×512 RGB observation. **2) Action space**: The action at time step t is defined as

$$a_t = \left[\Delta x_t,\ \Delta y_t,\ \Delta z_t,\ \Delta\phi_t,\ \Delta\theta_t,\ \Delta\psi_t,\ g_t\right] \in \mathbb{R}^6 \times [0,1],$$

where

- $\Delta x_t, \Delta y_t, \Delta z_t$ are the Cartesian displacement increments of the end-effector,
- $\Delta\phi_t, \Delta\theta_t, \Delta\psi_t$ are the roll–pitch–yaw rotation increments of the end-effector,
- $g_t \in [0,1]$ is the gripper's opening degree (0 = fully closed, 1 = fully open).

All pose increments $\Delta x, \ldots, \Delta\psi$ are expressed in the Spot robot's central coordinate frame. This 6D delta-pose plus gripper-degree parameterization enables precise, relative control of the manipulator in reinforcement learning.

A.2.2 Tasks

For each task, the initial state and subgoals are pre-defined.

Initial state A green onion is put in the sink, microwave on the top-left corner is closed, a pot is placed in between the stove and the sink.

Figure 9: Short horizon tasks in real world.

Figure 10: Long horizon tasks in real world.

Short Horizon Tasks There are three tasks chosen for short-horizon evaluation in the real setup: Task-1: Put pot on stove; Task-2: Put green onion in pot; Task-3: Open microwave. The goal setups are visualized in Figure 7.

Long Horizon Tasks There are three tasks chosen for long-horizon evaluation in the real setup: Task-4: Put pot on stove, green onion in pot, open microwave; Task-5: Put pot on stove, green onion in cabinet; Task-6: Put pot in sink, green onion in cabinet. The goal setups are visualized in Figure 10. The whole task is completed if and only if all subgoals are completed in the correct order.

A.3 Demonstration collections

1) LIBERO kitchen demonstrations: We use the demonstrations collected by Mimicplay [71]. The Mimicplay dataset contains 33 human play demonstrations averaging 7 minutes each (30 frames per second). **2) Real kitchen demonstrations**: The robot teleoperation data is collected using an OptiTrack motion capture system [93], which tracks the 6D pose of a gripper as it is operated by a human demonstrator. The control frequency of the robot arm and gripper is 7Hz. We collected 15 demonstrations in total: on average, the human operator interacts with the scene for 20 minutes per demonstration (5 frames per second). In both simulation and real-world setups, the human operator is asked to explore the environment by randomly teleoperating the robot arm to conduct different tasks based on the operator's own intention.

B Vis2Plan details

B.1 Details of skill segmentation

Here we provide the additional details of segmenting skills from unlabeled demonstration videos for Section 3.1.

Video preprocessing Here we provide detailed explanation for Figure 2 (1). Given a demonstration video and a fixed set of K task-related object categories $\{o_1, \ldots, o_K\}$, we first detect each o_k in the initial frame using the open-vocabulary VLM Qwen [72] and then track these bounding boxes across time with SAM2 [73], yielding K object-centric videos. Now the demonstration video V_{demo}

can be described as a combination of object-centric videos V_k: $V_{\text{demo}} = \big(V_1, \ldots, V_k\big)$. $V_k = \big(I_{k,1}, \ldots, I_{k,T}\big)$, where $I_{k,i}$ is the object-centric frame and T is the total number of demonstration frames. Next, we extract a feature trajectory from each clip via a vision foundation model $\mathbf{F}_{\text{VFM}}$ (e.g., SigLIP2 [74]):

$$\mathbf{f}_k^d = \big(\mathbf{F}_{\text{VFM}}(I_{k,1}^d),\, \mathbf{F}_{\text{VFM}}(I_{k,2}^d),\, \ldots,\, \mathbf{F}_{\text{VFM}}(I_{k,T_k}^d)\big) \;\in\; (\mathbb{R}^D)^T,$$

where D is the feature dimensionality. In this way, the unlabeled visual demonstration is represented by the set of K feature sequences: $\mathcal{F}^d = \{\mathbf{f}_1^d, \mathbf{f}_2^d, \ldots, \mathbf{f}_K^d\}$.

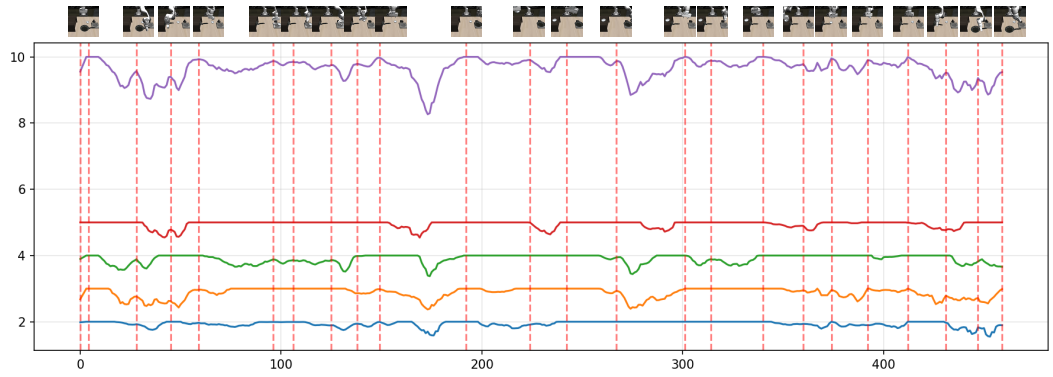


Figure 11: **Skill segmentaion visualization**: Here we provide on skill segmentation (stable state identification) result of one demonstration. The line plots are the visualization of object-centric adjacent similarities and the summation of object-centric adjacent similarities $\mathbb{S}_t$ (the top purple line). The red vertical dash lines are the NMS peak-finding results of $\mathbb{S}_t$, and the corresponding stable states (image frames of the peaks) are plot on top of the figure.

Stable State Identification Here we provide detailed explanation for Figure 2 (2): We define a stable state is where all objects' visual status remain static or similar, and a skill is a transition between two stable states. Our assumption is that the object's visual features can describe the objects' status (e.g., if it is under manipulation or not), and we can detect if object is under manipulation by tracking the objects' visual appearance. Follow this motivation, we extract segment video using equations [75]:

$$\mathbb{S}_t = \sum_{i=1}^{K} \left(\frac{\mathbf{F}_{\text{VFM}}(I_{i,t}^d) \cdot \mathbf{F}_{\text{VFM}}(I_{i,t+1}^d)}{\|\mathbf{F}_{\text{VFM}}(I_{i,t}^d)\| \|\mathbf{F}_{\text{VFM}}(I_{i,t+1}^d)\|} \right), \quad \mathbb{K} = \text{NMS}\Big(\frac{\sum_{i=t-k}^{t+k} W(\mathbb{S}_t, \mathbb{S}_i)\mathbb{S}_i}{\sum_{i=t-k}^{t+k} W(\mathbb{S}_t, \mathbb{S}_i)}\Big) \tag{4}$$

Here, $\mathbb{S}_t$ is the temporal similarity at time t, summing t and $t+1$ frame's object-centric cosine similarities. After Gaussian smoothing, we apply non-maximum suppression (NMS) within a local temporal window and select the highest peaks as key stable frames $\mathbb{K}$ (see Figure 2 (2)). Two neighboring stable state can be used to describe one sub-skill.

Symbolic Graph As described in Section 3.1, we use SigLIP2 features with unsupervised learning to extract the symbolic representations of key stable states. Using these representations, we construct a symbolic graph for high-level planning. Figure 12 presents a visualization of the symbolic graph for the LIBERO kitchen task.

B.2 Network implementations

Reachability Estimator $R(s, s_g)$ We follow the methodology from contrastive RL work [85] to implement our reachability estimator. We modify the Contrastive RL approach to distinguish between a future state sampled from the average discounted state-occupancy measure in the state-only case:

$$s_f^+ \sim p^\pi(s_{t+} \mid s) = \int\int p^{\pi(\cdot|g)}(s_{t+} \mid s, a)\, p^\pi(g \mid s, a)\, \pi(a|s)\, \mathrm{d}g\mathrm{d}a,$$

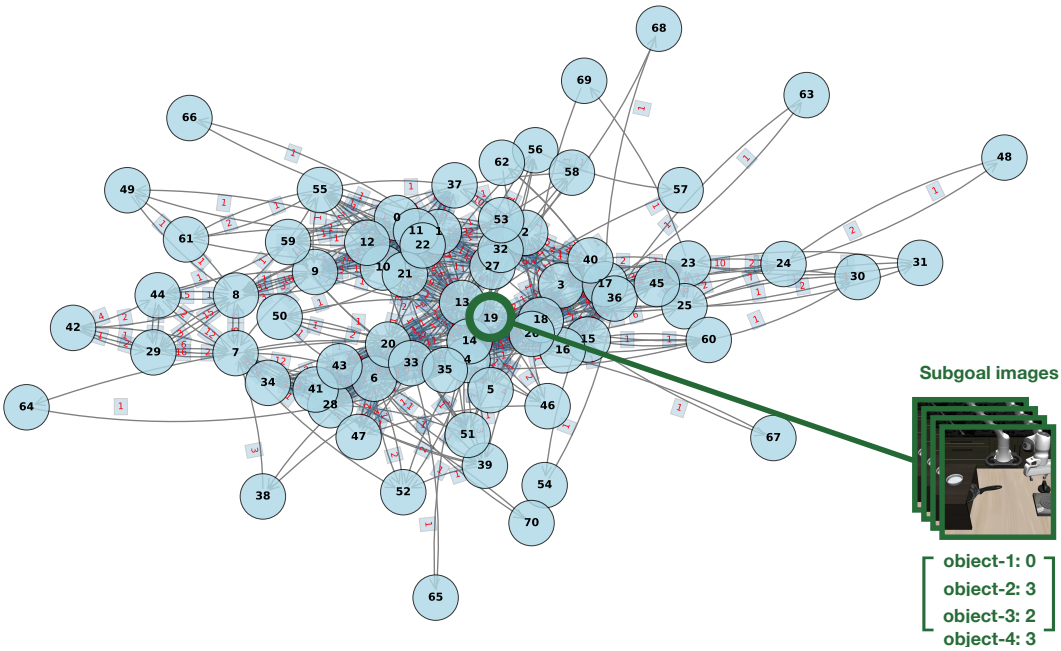


Figure 12: LIBERO kitchen symbolic graph example. Each node of the graph is described by the symbolic representation of task-relevant objects. We also provide one symbolic node example (node 19) to illustrate the node structure.

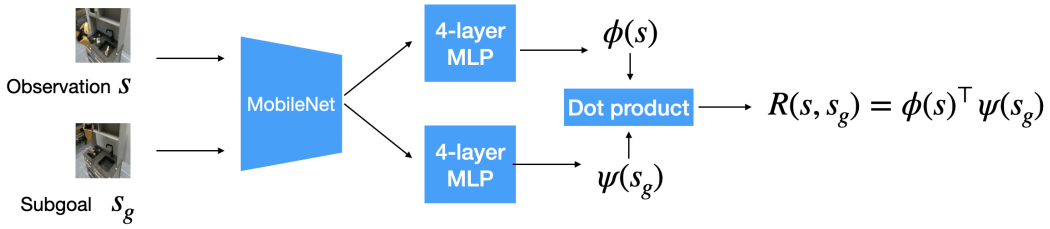


Figure 13: Overview of contrastive RL value network architecture (reachability network). We follow the implmentation suggestions in work [85] and modify it to state-only variant.

and a future state sampled from an arbitrary state–action pair:

$$s_f^- \sim p(s_{t+}) = \int p^\pi(s_{t+} \mid s)\, p(s)\, \mathrm{d}s$$

We train the estimator using the NCE-Binary objective:

$$\mathcal{L} = \mathbb{E}_{s_f^+ \sim p^\pi(s_{t+}|s)}\Big[\log \sigma\big(\phi(s)^\top \psi(s_f^+)\big)\Big] + \mathbb{E}_{s_f^- \sim p(s_{t+})}\Big[\log\big(1 - \sigma\big(\phi(s)^\top \psi(s_f^-)\big)\big)\Big]. \tag{5}$$

After training, the inner product $R(s, s_g) = \phi(s)^\top \psi(s_g)$ between the state representation $\phi(s)$ and future (goal) state $\psi(s_g)$ is proportional to the state-occupancy measure and can be used as a reachability estimator. Figure 13 illustrates the reachability network architecture. For training hyperparameters, we strictly follow those used in [85].

Low-level goal-conditioned policies The low-level policy should generate a sequence of actions to achieve the subgoal proposed by the visual planner. It receives the current image observation and a goal image to generate appropriate actions. All goal-conditioned policies are trained with a hindsight relabeling strategy: given an unlabeled video demonstration, the goal image $g_t^r \in \mathcal{V}^r$ is defined as the frame occurring H steps after the current time step in the demonstration. Here, H is drawn uniformly

Table 4: Low-level Policy Hyperparameters: AdaFlow.

Hyperparameter	AdaFlow
Learning rate	1×10^{-4}
Optimizer	AdamW
β_1	0.95
β_2	0.999
Weight decay	1×10^{-6}
Batch size	64
Epochs	500
LR scheduler	cosine
EMA decay rate	0.9999
η	1.0
$\epsilon_{\min}$	10
Action prediction horizon	16
Observation inputs	2
Action execution horizon	8
Obs. input size	128×128

Table 5: Low-level Policy Hyperparameters: Mimicplay (latent).

Hyperparameter	Default
Batch Size	16
Learning Rate (LR)	1×10^{-4}
Num Epoch	1000
LR Decay	None
KL Weights λ	1000
MLP Dims	[400, 400]
Img. Encoder – Left View	ResNet-18
Img. Encoder – Right View	ResNet-18
Image Feature Dim	64
GMM Num Modes	5
GMM Min Std	0.0001
GMM Std Activation	Softplus

Table 6: Low-level Policy Hyperparameters: Mimicplay (action).

Hyperparameter	Default
Batch Size	16
Learning Rate (LR)	1×10^{-4}
Num Epoch	1000
Train Seq Length	10
LR Decay Factor	0.1
LR Decay Epoch	[300, 600]
MLP Dims	[400, 400]
Img. Encoder – Wrist View	ResNet-18
Image Feature Dim	64
GMM Num Modes	5
GMM Min Std	0.01
GMM Std Activation	Softplus
GPT Block Size	10
GPT Num Head	4
GPT Num Layer	4
GPT Embed Size	656
GPT Dropout Rate	0.1
GPT MLP Dims	[656, 128]

from the integer range $[100, 600]$ (i.e., 5–30 seconds), serving as a data-augmentation mechanism. We adapt AdaFlow [87] for LIBERO simulation and Mimicplay [71] for real-world tasks as our low-level goal-conditioned policies.

- **Goal-conditioned AdaFlow** In LIBERO simulation, we implement a goal-conditioned variant of the imitation-flow-based generative policy from AdaFlow [87] as our low-level controller. At each step, this policy receives the current observation and the target subgoal image (both at 128×128 resolution) and predicts a horizon of 16 consecutive actions. All architecture and hyperparameters remain the same as implemented in [87]. In Table 4, we summarize all key settings of the goal-conditioned AdaFlow policy.

- **Mimicplay Policy** In real-world tasks, we use the Mimicplay policy as the low-level control policy. We maintain the implemented architecture and hyperparameters as used in [71]. At each time step t, the Mimicplay policy receives the current image observation and the next subgoal image observation (both at 128×128 resolution) and outputs an 16-step end-effector pose trajectory, which conditions the low-level policy to generate a sequence of 16-step actions for execution. Table 5 summarizes the hyperparameters used to train the Mimicplay latent planner, and Table 6 those for the robot policy π. Parameters prefixed with "GMM" refer to the MLP-based Gaussian mixture model, whereas those prefixed with "GPT" specify the transformer's configuration[71].

B.3 Pseudocode of Closed-loop Symbolic-guided Visual Planning

Here we provide pseudocode to illustrate how symbolic-guided visual planning is executed, as additional details for Section 3.3. More specifically, we optimized the visual subgoal generation step to accelerate visual plan generation. We use the pretrained reachability estimator to calculate the reachability features of all key image frames and save these features with their corresponding images. During visual subgoal sampling, instead of sampling image batches for each symbolic subgoal, we directly sample the subgoal images' reachability feature batch ($\phi(s)$ and $\psi(s)$) to avoid computationally expensive reachability estimation. This optimized sampling approach is visualized in Figure 14.

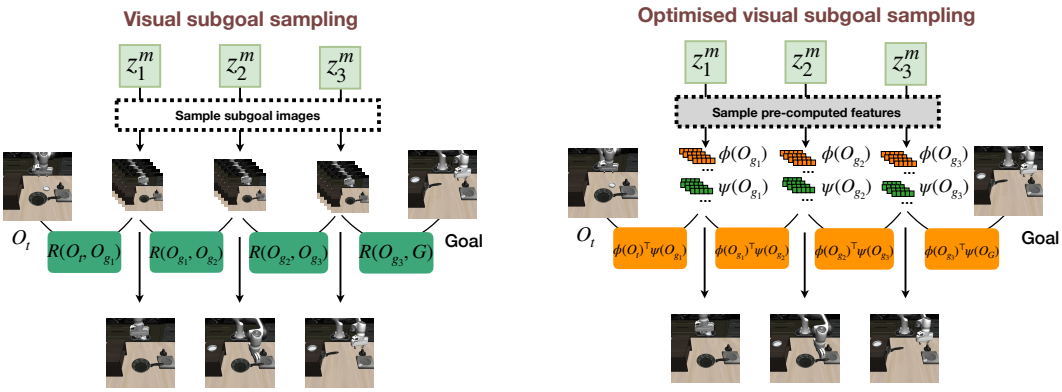


Figure 14: Optimized visual subgoal sampling (stage 2 in Algorithm 1). The right plot illustrates the standard visual subgoal sampling pipeline, while the left plot shows our optimized variant. We directly sample pre-computed reachability features of subgoal images to search for the most feasible visual plan, significantly reducing computational overhead.

Algorithm 1: Stage 3: Symbolic-guided Visual Planning with Vis2Plan

Input: Task goal z_g, state predictor C_θ, reachability estimator R_ψ, goal-conditioned policy π_g
Initialize: symbolic path $\tau_z = \emptyset$, visual plan $\tau_O = \emptyset$
while goal z_g is not reached **do**
  Receive current observation O_t
  // Stage 1: Symbolic Planning
  Predict next symbolic state candidates: $\mathcal{Z}_{\text{next}} = C_\theta(O_t)$
  for z_1 in $\mathcal{Z}_{\text{next}}$ do
    Find symbolic path: path $= \textsc{Search}(z_1, z_g)$
    Update if shorter: $\tau_z =$ path if cost(path) $<$ cost(τ_z)

  // Stage 2: Visual Subgoal Generation
  for z_i in τ_z do
    Sample visual observation: $O_i \sim P_D(\cdot \mid z_i)$
    Add to visual plan: $\tau_O = \tau_O \cup \{O_i\}$
  Optimize visual plan: $\tau_O = \arg\max_{\tau'_O} \sum_{i=0}^{H-1} R_\psi(O'_i, O'_{i+1})$ using beam search
    s.t. $R_\psi(O'_i, O'_{i+1}) \leq \delta, \forall i$

  // Stage 3: Goal-conditioned Action Generation
  Get next visual subgoal: $O_g = \tau_O[1]$
  Generate action chunk: $u = \pi_g(O_t, O_g)$
  Execute action chunk u in environment
Return: task completed

B.4 End2End Goal-Conditioned Baselines

Goal-conditioned transformers (GC-Transformer) We use the goal-conditioned transformer architecture defined in Mimicplay as the baseline implementation [71]. The Mimicplay GC-transformer embed the wrist token w_t, environment embedding e_t, and proprioceptive embedding p_t to form the sequence $s_{t:t+T} = [w_t, e_t, p_t, \ldots, w_{t+T}, e_{t+T}, p_{t+T}]$, with $T = 10$. This sequence is processed by a GPT-style transformer f_{trans} comprising $N = 4$ layers of multi-head self-attention (4 heads) and position-wise feed-forward networks with intermediate dimension 656 and output dimension 128. The model uses a block size of 10, embedding dimension of 656, and dropout rate of 0.1. Given the past $T - 1$ embeddings, f_{trans} autoregressively predicts the next action feature x_T. The final control command a_t is produced by passing x_t through a two-layer MLP with hidden dimensions [400,400]. Visual observations from wrist, left, and right camera views are encoded via ResNet-18

into 64-dimensional features. To model the multimodal action distribution, we employ an MLP-based Gaussian mixture model with 5 components and a minimum standard deviation of 0.01 (Softplus activation). We train the network for 1000 epochs with a batch size of 16 and initial learning rate 1×10^{-4}, decayed by a factor of 0.1 at epochs 300 and 600.

Goal-conditioned Diffusion Policy (GC-Diffusion) We adapt the Goal-conditioned variant diffusion policy from work Adaflow [87] as one baseline. The diffusion-based robot policy is trained with the AdamW optimizer (learning rate 1×10^{-4}, $\beta_1 = 0.95$, $\beta_2 = 0.999$, weight decay 1×10^{-6}), using a batch size of 64. We train for 500 epochs in the likelihood-weighting phase and 3000 epochs in the reverse-mapping phase, employing a cosine learning-rate scheduler and an exponential moving-average decay of 0.9999. The denoising diffusion model uses 100 forward diffusion steps and 100 inference steps under the DDPM framework. At test time, the policy predicts an action horizon of 16 steps conditioned on the two most recent observations, executes 8 actions per cycle, and processes visual inputs at a resolution of 128×128.

Goal-conditioned Action-Chunk Transformer (GC-ACT) Action-chunk transformer (ACT) was proposed in [89]. For our work, we implemented a goal-conditioned variant of ACT based on the code repository available at `https://github.com/Shaka-Labs/ACT`. Our GC-ACT receives the current image observation and a goal image observation as inputs, and generates a 16-step action chunk for execution.

B.5 High-level Planner Baselines

AVDC We adapt the video diffusion generation model from [1] as one of our visual planner baselines. We strictly follow the configuration provided in the AVDC public code repository (`https://github.com/flow-diffusion/AVDC`). Specifically, we modify the original implementation to use an image goal instead of a task text description. The AVDC diffusion video generation model receives the current image observation and a goal image observation (both at 128×128 resolution) and outputs a 9-step horizon image plan, with each subgoal image at 64×64 resolution. We train the AVDC video generation model similarly to training a goal-conditioned policy: we hindsight sample the future image of the current trajectory as the goal image, and subsample the intermediate frames to 9 frames as the subgoals (target video frames). Given an unlabeled video demonstration, the goal image $g_t^r \in \mathcal{V}^r$ is defined as the frame occurring H steps after the current time step in the demonstration. Here, H is drawn uniformly from the integer range $[100, 1500]$ (i.e., 5–75 seconds).

GSR We implemented the baseline GSR follow the instruction from [90], where the author build a directed graph from offline demonstrations. Based on GSR implementation, each image observation from demonstrations is treated as a vertex of a graph, and the directed edge between vertexes indicate the image state transition. There are two types of edges built in GSR: 1) Dataset edge: represents ground truth transitions on each demonstration trajectory $(v_0, v_1), (v_1, v_2), ...$, serving as an approximation of the world dynamics; 2) Augmented edge: a bidirectional edge between nodes u and v is added if they both lie in the tolerance range of each other in the pretrained representation space, bridging similar states across trajectories. In this work, we use our trained reachability estimator $R(s, s_g)$ as the pretrained representation network to construct the augmented edges.

UVD-graph Here, we develop a graph-based approach using the UVD implementation. We finetune the UVD feature extraction following the instructions in [91] (`https://github.com/zcczhang/UVD/`). We then use UVD to segment the demonstration into sub-tasks. Similar to GSR, we use our pretrained reachability estimator $R(s, s_g)$ to build a graph based on the segmented subgoals. This implementation creates a more sparse graph compared to GSR. The key difference between UVD-graph and our Vis2Plan is that Vis2Plan uses object-centric features to identify subgoals and creates a symbolic description of key states, enabling more structured and interpretable planning.

Table 7: Average visual plan generation speed (seconds) for different planners across task types. "Sim" refers to LIBERO simulation and "Real" refers to real-world tasks. AVDC shows consistent visual plan generation times across tasks because it generates a fixed number of subgoals (9) regardless of task complexity.

Planner	Sim Short	Sim Long	Real Short	Real Long
GSR	0.10	15.12	0.11	10.13
UVD-graph	1.2×10^{-4}	2.8×10^{-4}	1.1×10^{-4}	1.8×10^{-4}
AVDC	1.42	1.42	1.42	1.42
Vis2Plan (ours)	0.03	0.04	0.03	0.05

C Visual Planning Results

C.0.1 Visual Planning Inference Speed Comparison

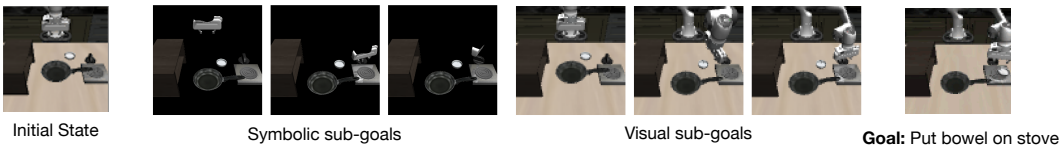
Here we compare the visual plan generation inference speed across different methods (Table 7). GSR builds a graph based on one-step temporal transitions, where each vertex represents an image frame from the demonstration. This creates an extremely complicated graph that is time-consuming to search, resulting in visual plan generation times of several seconds for long-horizon tasks. AVDC uses a diffusion generative model to generate image sequences, but the diffusion process is computationally expensive, making real-time planning infeasible.

Despite having similar graph complexity to GSR, the UVD-graph approach is the fastest planner as it does not require a visual goal sampling procedure. However, our Vis2Plan framework still achieves near real-time planning performance while maintaining high plan quality and physical feasibility.

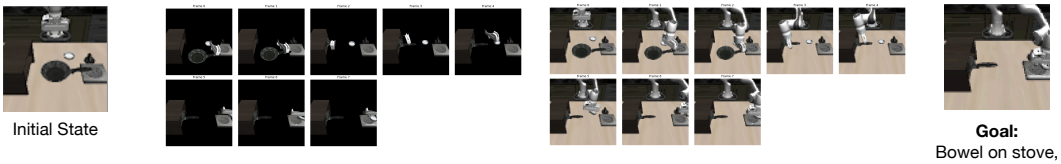
C.1 Visual Plan results

Here we provide some representative planning results of Vis2Plan in both simulation and real-world tasks. For additional visualization results, please refer to the second supplementary material.

The choice metric is simple: the feature extracted by the VFM shall have good ability to distinguish in an unsupervised manner We design the experiment in this way:

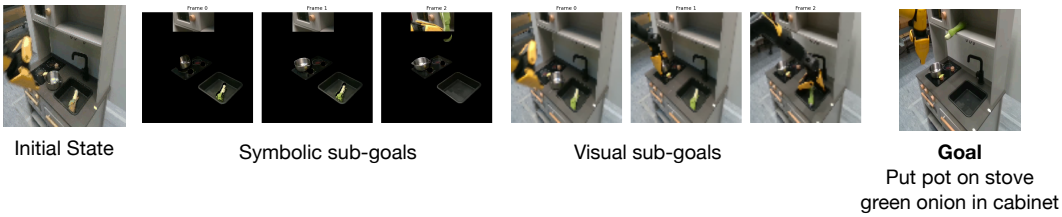


(a) Simulation short-horizon Vis2Plan visual planning example: put bowl on stove



(b) Simulation long-horizon Vis2Plan visual planning example: put bowl on stove and pan in shelf

(c) Real world short-horizon Vis2Plan visual planning example: put pot on stove



(d) Real world long-horizon Vis2Plan visual planning example: put pot on stove and green onion in cabinet

Figure 15: Representative Vis2Plan visual planning examples in both simulation and real-world environments. (a)-(b) Simulation examples showing short and long-horizon tasks. (c)-(d) Real-world examples demonstrating similar capabilities in physical environments.