Implicit State Estimation via Video Replanning

Po-Chen Ko [1] **Jiayuan Mao** [2] **Yu-Hsiang Fu** [1] **Hsien-Jeng Yeh** [1] **Chu-Rong Chen** [1] **Wei-Chiu Ma** [3] **Yilun Du** [4] **Shao-Hua Sun** [1]

Abstract

Video-based representations have gained prominence in planning and decision-making due to their ability to encode rich spatiotemporal dynamics and geometric relationships. These representations enable flexible and generalizable solutions for complex tasks such as object manipulation and navigation. However, existing video planning frameworks often struggle to adapt to failures at interaction time due to the incapability to reason about uncertainties in partially observed environments. To overcome these limitations, we introduce a novel framework that integrates interaction-time data into the planning process. Our approach updates model parameters online and filters out previously failed plans during generation. This enables *implicit state estimation*, allowing the system to adapt dynamically without explicitly modeling unknown state variables. We evaluate our framework through extensive experiments on a new simulated manipulation benchmark, demonstrating its ability to improve replanning performance and advance the field of video-based decision-making.

1. Introduction

Learning from videos has gained significant traction in decision-making, as videos capture rich visual and dynamic information while aligning with how humans acquire knowledge. These properties make them a powerful medium for specifying tasks and learning diverse skills across contexts. Recent work has shown the effectiveness of video-based frameworks in enabling robots to learn behaviors such as object manipulation (Li et al., 2024) and navigation (Zhang et al., 2024), highlighting the value of video as a flexible and expressive representation.

This paper focuses on video as a planning representation. Given a goal and current observation, video planning systems generate imagined task executions and convert them into robot actions. Unlike symbolic or latent representations, videos naturally encode both perceptual and action information and generalize across tasks and environments. Prior works (Chang et al., 2020; Du et al., 2024a;b) leverage these properties to train universal agents using video-based predictions.

Despite promising results, existing video planning frameworks suffer from a crucial limitation: they lack mechanisms to integrate past interactions with the environment and cannot effectively reason about uncertainty due to partial observability. Consider the task of opening a door without knowing whether it should be pushed or pulled. If pushing fails, a human will naturally infer that pulling is the correct action and adjust accordingly. In contrast, current video planning frameworks simply generate a new plan, disregarding the knowledge gained from prior attempts. This inability to incorporate feedback from interactions significantly limits their effectiveness, particularly in unstructured environments where uncertainty is inherent.

A common solution introduces explicit belief models trained in simulation to infer hidden parameters (Memmel et al., 2024; Qi et al., 2023). Yet these methods require prior knowledge of relevant parameters and may struggle when interaction data fails to disambiguate uncertainties. In contrast, humans can effectively resolve these challenges by drawing on past experiences and employing adaptive trial-and-error strategies when prior knowledge is lacking. In the context of video planning, failed interactions are not merely setbacks — they offer critical information about the environment and system parameters.

Our goal is to develop a video planning framework that effectively encodes and retrieves interaction data, adapting plans dynamically based on past experiences—without requiring additional simulation data or prior knowledge of the parameters. We propose a novel problem formulation that formalizes the challenge of generating video plans while implicitly incorporating past interactions, as illustrated in Figure 1. To address this, we introduce im-

[1]National Taiwan University, Taipei, Taiwan [2]Massachusetts Institute of Technology, MA, USA [3]Cornell University, NY, USA [4]Harvard University, MA, USA. Correspondence to: Shao-Hua Sun <shaohuas@ntu.edu.tw>.

Workshop on Building Physically Plausible World Models at the 42^{nd} International Conference on Machine Learning, Vancouver, Canada.

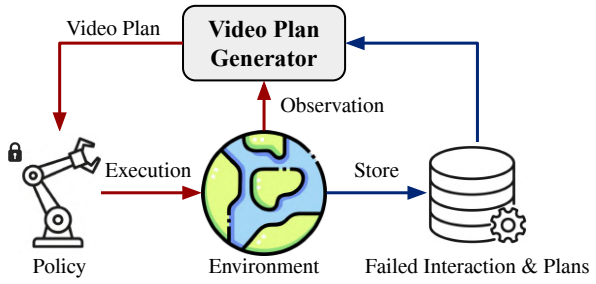


Figure 1. **Implicit state estimation via video replanning.** Previous video planning methods rely on the first frame (in red). We propose to incorporate past failures (in blue) to improve future plans.

plicit state estimation (ISE), a framework that integrates interaction-time data into the planning process without explicitly defined system parameters. Instead of relying on hand-specified beliefs, our approach enables rapid adaptation by optimizing a subset of model parameters that *implicitly* encode unobservable state features. ISE also incorporates a plan-rejection mechanism that filters out previously failed plans, preventing over-reliance on incorrect inferences and encouraging exploration. By adopting a video planning framework, our approach leverages the latest advancements in generative models, paving the way for future applications such as out-of-distribution detection.

To validate the effectiveness of our framework, we introduce the Meta-World System Identification Benchmark, a new dataset designed to evaluate online adaptation in decision-making across diverse robotic tasks with unknown system parameters. Experiments show that our method allows for rapid adaptation without explicit system identification. Compared to baselines, including RL methods and other methods for video planning, our approach significantly reduces replanning failures and generates more accurate video plans in dynamic, uncertain environments.

2. Related Works

Video planning. Videos have emerged as a popular medium for planning in the physical world. (Yang et al., 2024b). For planning actions, people have explored using video to train dynamics model (Yang et al., 2024a; Ajay et al., 2023; Zhou et al., 2024; Qin et al., 2024) or reward model (Ma et al., 2023; Chen et al., 2021a; Escontrela et al., 2023). Another popular approach involves directly predicting video sequence (Du et al., 2024a; Ko et al., 2024; Black et al., 2024) or derived representations such as 3D-correspondence or point tracks (Yuan et al., 2024; Wen et al., 2024; Bharadhwaj et al., 2025) to guide downstream policy. However, most of these frameworks do not consider environmental uncertainty, which limits their ability to efficiently adapt and replan, often resulting in suboptimal performance in such environments.

System identification. System identification focuses on learning system dynamics and estimating parameters using experimental data, with early work on input selection and the Fisher Information Matrix (Schön et al., 2011; Ljung, 1998; Menda et al., 2020). This research laid the groundwork for active identification strategies, which aim to maximize the amount of information gained from system inputs, a key element in classical system identification (Hjalmarsson et al., 1996; Lindqvist & Hjalmarsson, 2001; Gerencser & Hjalmarsson, 2005). Recent advances in the field have applied these methods to real-world systems, such as identifying physical parameters (Xu et al., 2019; Kumar et al., 2019; Mavrakis et al., 2020; Gao et al., 2020; 2022; Memmel et al., 2024). Our work can be seen as bridging classical system identification and modern video planning method by injecting the ability to implicitly infer physical parameters from past video data to the model.

Imitation learning. Imitation learning (Zare et al., 2024) is a research field of robot learning. Literally, imitation learning means that the objective is to train an agent to behave like an expert, given the dataset of the interaction of the export and the environment. Imitation learning has been proven to be an effective and data-efficient approach for robot learning (Ho & Ermon, 2016; Wang et al., 2017). Behavioral cloning (BC; (Torabi et al., 2018)) formulates imitating an expert as a supervised learning problem and achieves success on various tasks (Florence et al., 2021; Chen et al., 2024; Shridhar et al., 2022; Chi et al., 2024; Brohan et al., 2023). Imitation learning uses expert demonstration, while we aim to leverage failed interactions with environments.

Offline reinforcement learning. Offline RL assumes datasets composed of state-action-reward sequences (Levine et al., 2020; Figueiredo Prudencio et al., 2024). In contrast to online RL, during the stage of training, the agent tries to learn to perform the task by leveraging the dataset, without any interaction to the environment. The paradigm of offline RL has existed for dacades. However, with the rise of deep learning, most modern offline RL algorithms utilize deep neural networks to express the value functions and policies (Kidambi et al., 2020; Yu et al., 2020; Kumar et al., 2020b; Yu et al., 2021; Yang et al., 2022). In addition, with the development of large language model and transformer, there are also some research about using transformer for reinforcement learning (Chen et al., 2021b; Furuta et al., 2022). In contrast to offline RL, our framework do not assume dense reward signals.

3. Problem Formulation

We consider a setting where certain system parameters θ, such as mass, friction, or utility — essential for solving the task — are unknown. Unlike existing work for explicit state estimation, we do not assume access to the parameterization of θ. We only assume that the underlying θ can be inferred from interaction data. For example, in a bar-pushing task, the center of mass can be implicitly inferred from an interaction video, regardless of whether the push is successful.

We aim to design a system capable of rapidly solving tasks by planning efficiently and leveraging interactions in the environment. For each environment, we assume access to an experience dataset $\mathcal{D}$, consisting of data tuples $d_i = (v_i, o_i, s_i) \in \mathcal{D}$, where v_i represents an interaction video, o_i denotes the object ID, and s_i is a binary success indicator. The dataset contains both successful interactions where $s_i = 1$ and failed interactions where $s_i = 0$. Our work focuses on evaluating whether the selected plans can guide successful action sequences. We complement our focus on plan evaluation by using a fixed action prediction module to convert videos into actions for execution.

4. Approach

We tackle manipulation tasks in environments with unknown hidden parameters (e.g., mass or friction) by leveraging prior interaction videos. Our framework incrementally refines its belief about the environment through trial-and-error and video-based planning. Given the current scene and past interactions, it retrieves and refines a latent state embedding representing a hypothesis of the hidden parameters, then generates a video plan based on this embedding (Figure 2).

The system maintains two buffers: failed interactions and failed plans. At each round, it generates candidate plans and selects one that differs most from past failures using a rejection strategy (Section 4.2). This plan is converted into actions using an Action Module, which may be based on inverse dynamics, goal-conditioned policies, or trajectory tracking. We use AVDC's training-free point tracking method (Ko et al., 2024). Failed attempts are added to the buffers, guiding future updates to the state belief and plan selection.

Our framework consists of three components: (1) a **Video Plan Generator** conditioned on the scene and state embedding, (2) a **Rejection Module** that promotes diverse planning, and (3) an **Action Module** that translates videos into actions. Detailed descriptions are provided in Sections 4.1, 4.2, and 4.3.

4.1. Video plan generator and retrieval module

To generate meaningful plans in environments with unknown dynamics, our system first hypothesizes the hidden parameters of the environment via the retrieval and refinement of a latent state embedding, which serves as a proxy for the current environment configuration. We first discuss our *Video Plan Generator* and *Retrieval Module*, which generate candidate plans based on observed images and state embeddings. The *Video Plan Generator* predicts future frames conditioned on the current observation and, when available, the state embedding. For new objects without prior interactions, the generator operates solely on the observed image by conditioning on a learned "null" state embedding.

The *Retrieval Module* extracts state embeddings from prior interactions by encoding past videos and selecting a representative embedding for each object ID. At inference time, it encodes the current interaction video, computes distances to past embeddings from dataset $\mathcal{D}$, and uses a softmax function to sample the most relevant state embedding. In addition, we use an additional identification model to continuously refine the selected state embedding to best fit current interaction videos.

4.1.1. State embedding

To represent the hidden system parameters, we encode interaction videos from previous action executions with the same object using a context encoder E. The video embedding for each video v_j is obtained as $e^v_j = E(v_j)$. In the dataset $\mathcal{D}$, multiple data tuples may share the same object ID o_j. Naively encoding each video independently would result in multiple, different embeddings for the same object. To address this, we preprocess the dataset by grouping all entries with the same object ID and selecting the encoding from one of the successful executions (where $s_j = 1$) to represent the object, resulting in a state embedding e^o_j. We assume that success is monotonic — all successful videos are similar and, therefore, yield similar embeddings.

4.1.2. Video plan generator

The Video Plan Generator is used to generate a video containing M future frames conditioned on the current observation and a state embedding. We follow the implementation of AVDC's video planner, which is a first-frame conditioned video diffusion model utilizing a 3D U-Net architecture. The original implementation conditions on task features were obtained from encoding natural language task descriptions with text encoders such as CLIP-Text, rather than state embeddings. To adapt the method to our needs, we project the state embedding into the hidden dimension of the language features and treat the projected embedding as an additional language token. The parameters of the pro-

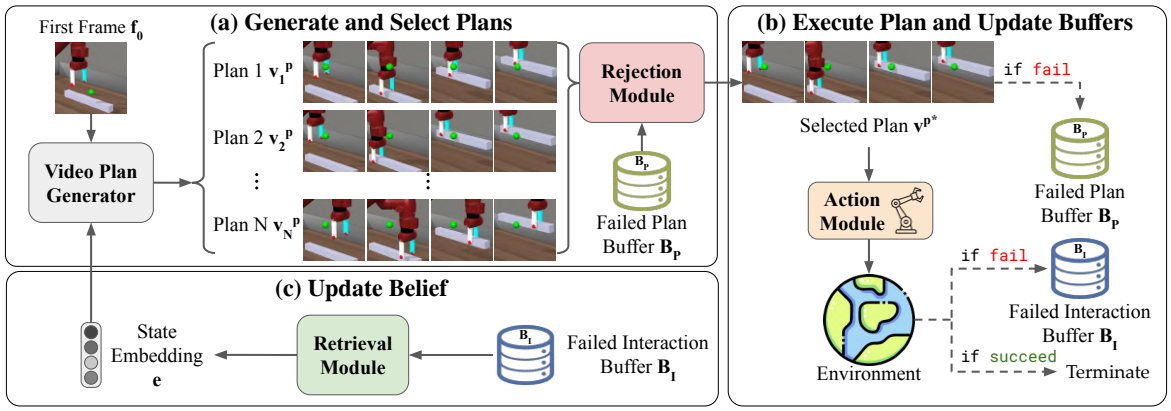


Figure 2. **Framework overview. (a) Generate and select plans** The Video Plan Generator first generates a set of candidate video plans conditioned on first-frame f_0 and state embedding e. The Rejection Module then selects the plan that is least similar to the plans in the Failed Plan Buffer B_P. **(b) Execute plan and update buffers.** The Action Module interacts with the environment by following the selected plan v^{p*}. If the interaction trial is not successful, we update the Failed Plan Buffer B_P and the Failed Interaction Buffer B_I. **(c) Update belief.** The state embedding e is updated using the Retrieval Module based on the Failed Interaction Buffer B_I. Then, we generate the next batch of video plans according to the updated belief.

jection layer are trained end-to-end along with the video planner, which is optimized using a standard denoising diffusion loss. For all experiments in this paper, we choose the resolution of generated frames to be 128x128 and $M = 7$. Each time we invoke the Video Plan Generator, $M = 7$ future frames are generated, we then concatenate them with the first frame, resulting in a $M + 1 = 8$ frames video plan. For training details, please refer to Appendix C.1.

4.1.3. Retrieval Module

During inference, the Video Plan Generator needs to take in a state embedding extracted from prior interactions. To ensure that the method is general across different encoders E, we first normalize and align the dimensionality of the feature space constructed by video features e_j^v by applying PCA to all extracted video features e_j^v, yielding a PCA projection P and PCA-transformed features $e_j^{vp} = P(e_j^v)$ and $e_j^{op} = P(e_j^o)$.

Given the interaction video from a prior execution, v^e, we first encode it using the encoder E and apply the PCA transformation, resulting in $e^{ep} = P(E(v^e))$. We then compute the distance between e^{ep} and the video encodings from $\mathcal{D}$ as $logits_j = -dist_j = -K(e^{ep}, e_j^{vp})$, where K represents the distance function. For the choice of K, we used Cosine Distance for CLIP and DINO-based methods and L2 Distance for other methods. After calculating the logits, we compute the sampling probability of each output embedding by passing the logits through a softmax function: $p_j = \text{softmax}(logits_j/\tau)$, where τ is the temperature parameter. Finally, we sample the output state embedding $e = e_k^{op}$ according to the resulting probability distribution $k \sim \text{Categorical}(p_1, p_2, \ldots, p_n)$. While such retrieval-based state estimation formulation necessitates the data being available during test time, it enables training-free state estimation and guarantees that the Video Plan Generator never accepts out-of-distribution state embeddings during inference time.

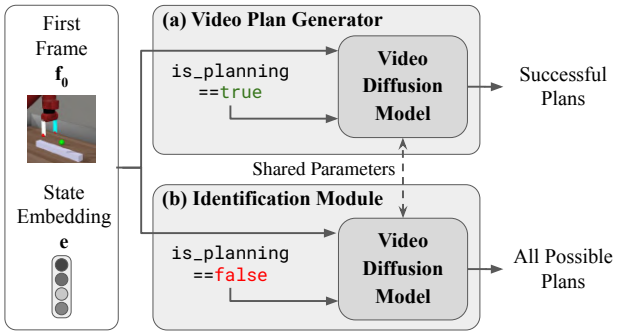


Figure 3. **Video Plan Generator and Identification Module.** We implement the Video Plan Generator and the Identification Module as two video generative models with shared parameters, switched with a binary trigger `is_planning`. The Video Plan Generator is trained with only successful videos from the experience dataset D, while the Identification Module is trained with all videos from D, including the failures, enabling the state-embedding refining process.

4.1.4. Refining State Embedding

In addition to using retrieval to find a relevant state embedding for a video, we also use an optimization-based approach to find a refined state embedding corresponding to a specific interaction video. Specifically, we introduce another generative model, referred to as the Identification model or the ID module, trained with the parameters shared with the Video Plan Generator, as shown in Figure 3.

The ID module is trained to generate all possible interaction outcomes, including both successful and unsuccessful ones. During replanning, we freeze its parameter and only optimize the state embedding to maximize the probability of generating the given interaction video.

$$e = \arg\min_{e} MSE(v^e, v)$$

Here, e is the optimized state embedding, v^e is the interaction video, and v is the videos sampled from the ID module conditioned on e.

Since the ID module and the Video Plan Generator share the same embedding space for state embeddings, the optimized state embedding can be used directly to generate video plans. In the experimental section of the paper, we analyze the effect of this refinement procedure for state estimation when 1) this approach optimizes state embedding from randomly initialized vectors drawn from unit variance Gaussian distribution, 2) the state embedding is set to the output of the Retrieval Module, and 3) these two approaches are combined, where this approach optimizes the output of the Retrieval Module.

4.1.5. Latent feature space

Given the experience dataset D, context encoder E decides the feature space that the Retrieval Module and the Video Plan Generator operate on. To implement E, we experiment with taking pre-trained vision encoders that are known to give good semantic representations, such as CLIP (Radford et al., 2021) and DINOv2 (Oquab et al., 2024). To encode videos with these image encoders, we used a simple strategy that first encodes each frame independently and then concatenates the per-frame features, resulting in a single-vector video feature.

4.2. Rejection module

While the modules introduced in Section 4.1 alone are sufficient to perform system identification essential for solving the tasks given a set of videos of interactions in the environment, the identified system parameters are often not perfect. This is especially the case when we only have a limited number of interactions with the environment. Thus, to effectively estimate the precise state of the environment, it is important that the system repeatedly actively interacts with the environment.

To actively interact with the environment, we use a video plan generator to generate plans for execution. To encourage the planner to explore novel plans and avoid being overly reliant on suboptimal beliefs, we employed a simple rejection-based sampling method. In the Rejection Module, the previously failed plans are stored in a data buffer. At each planning round, we generate N plans $(v_1^p, v_2^p, \ldots, v_N^p)$ instead of one by conditioning the Video Plan Generator on N independently sampled state embeddings from the Retrieval Module, yielding N different plans with potentially different beliefs on environment parameter θ. Let $\mathcal{F}$ be the set of previously failed plans stored in the data buffer, we first calculate the distance to the nearest failed plan for each plan P_i:

$$d(v_i^p, \mathcal{F}) = \min_{F \in \mathcal{F}} K(v_i^p, F)$$

where K is the distance function. Empirically, we found that L2 distance on raw pixel space works surprisingly well already.

We then select the plan v^{p*} whose distance to its nearest failed plan is the largest (out of a set of generated video plans):

$$v^{p*} = \arg\max_{v_i^p} d(v_i^p, \mathcal{F})$$

The selected v^{p*} is later converted to low-level control signals through the Action Module.

The combination of Section 4.1 and Section 4.2 form the core of our Implicit State Estimation procedure. Interactions with the environment through the rejection module in Section 4.2 obtain a buffer of interaction videos that help describe the implicit state of the environment. The retrieval module in Section 4.1 then obtains an explicit latent that represents the implicit state of the environment.

4.3. Action module

Given a generated video plan, to convert the video into continuous actions to execute, we follow the concept of dense object-tracking from AVDC to recover actions from video plans. On certain interaction tasks, the object of interest is not applicable. For example, in a bar-picking task where the robot is required to grasp around the center of mass to succeed, knowing the future trajectory of the bar is not sufficient to decide the grasp location. In these tasks, we track the robot's wrist instead. After obtaining the object/robot wrist trajectory from the tracking results, we convert the trajectory into actions using hand-crafted heuristic policies.

5. Experiment

5.1. Meta-World system identification benchmark

To evaluate the performance of different methods for online adaptation in decision-making, we propose the Meta-World System Identification Benchmark, which contains an offline experience dataset paired with environments each exhibiting unknown system parameter θ. The benchmark includes tasks like manipulating objects (e.g., push bar, pick

Table 1. **Meta-World System Identification Benchmark.** Each task involves interaction-based inference of system parameters θ, which are either continuous or discrete with varying mode counts.

Task	Identification Type
Push Bar	Continuous
Pick Bar	Continuous
Slide Brick	Continuous
Open Box	Discrete, 2 modes
Turn Faucet	Discrete, 2 modes

bar) and solving puzzles (e.g., turn faucet), where the underlying environmental rules must be identified from the interactions. Table 1 shows the set of 5 tasks. They evaluate a robot's ability to manipulate objects with varying physical properties and constraints, requiring adaptability to dynamic conditions such as randomized centers of mass, uncertain interaction modes, and external forces.

Push bar. The objective of this task is to push a bar toward a designated target with the center of mass of the bar randomized.

Pick bar. This task requires the robot to grasp a bar and transport it to a specified target. Similarly, the center of mass of the bar is randomly assigned.

Slide brick. This task consists of two distinct stages. In the first stage, the robot must push a brick up an inclined surface. In the second stage, the brick slides freely down the slope. The objective is to control the initial push such that the brick comes to rest within a predefined target region on the slope.

Open box. The objective of this task is to open a box. The box cover can be manipulated in one of two modes: it can either be lifted or slid open.

Turn faucet. In this task, the robot must rotate a faucet. The faucet's direction of rotation is either clockwise or counterclockwise.

In the above five tasks, the system parameters cannot be inferred without physical interaction. The details of the tasks are provided in Appendix B.

5.2. Evaluation metrics

Our study focuses on two key aspects. First, we hypothesize that our replanning mechanism can reduce the number of replans required for successful execution. To quantitatively measure the rate of adaptation, we report the average number of replans needed per successful attempt in our main experiment. Specifically, the agent must repeatedly replan until it either succeeds or reaches a predefined maximum number of replan trials, M. For all tasks, we set $M = 14$.

Second, instead of only evaluating task success, we also assess the effectiveness of the replanning mechanism by comparing the replanned videos to groundtruth interaction videos. To quantify this similarity, we use standard video comparison metrics: PSNR, SSIM, and LPIPS (Zhang et al., 2018).

5.3. Implementation details and baselines

We evaluate our framework on the 5 tasks in the Meta-World System Identification Benchmark. For our model, the Video Plan Generator is trained on the videos in the offline experience dataset as described in 4.1.2. Each task is evaluated over 400 trials. We compare our approach with the original AVDC implementation for video planning which does not online adapt, variants of our method that use different context encoders E, and other reinforcement learning baselines.

- **AVDC** is the baseline with only the Action Module, *i.e.*, without the Retrieval Module and Rejection Module introduced in this paper.
- **Ours** is our proposed method with E based on DINOv2 feature as described in 4.1.5.
- **Ours (Refine)** is a variant of our method with refining state embeddings from scratch as mentioned in 4.1.4.

Our problem formulation for state estimation can be viewed as a single-step multi-task reinforcement learning (RL) problem, where the success signal s_i from the dataset serves as the reward, by additionally assuming privileged access to action labels (i.e., ground-truth system parameters). We also implement such explicit state estimation baselines with popular RL algorithms to calibrate the difficulty of the tasks.

- **BC** uses a state embedding-conditioned behavior cloning framework where the Policy takes in state and state embedding directly. The state embedding is obtained with E based on DINOv2.
- **CQL** (Kumar et al., 2020a) is a state embedding-conditioned actor-critic framework. The Q-function takes in state, action, and state embedding, while the Policy takes in state and state embedding, which is obtained with E based on DINOv2.

5.4. Results

The replanning efficiency of all methods is presented in Table 2. Our method and its variants consistently outperform baselines in video planning (AVDC), imitation learning (BC), and offline RL (CQL). AVDC performs better than BC and CQL in terms of trial efficiency, demonstrating

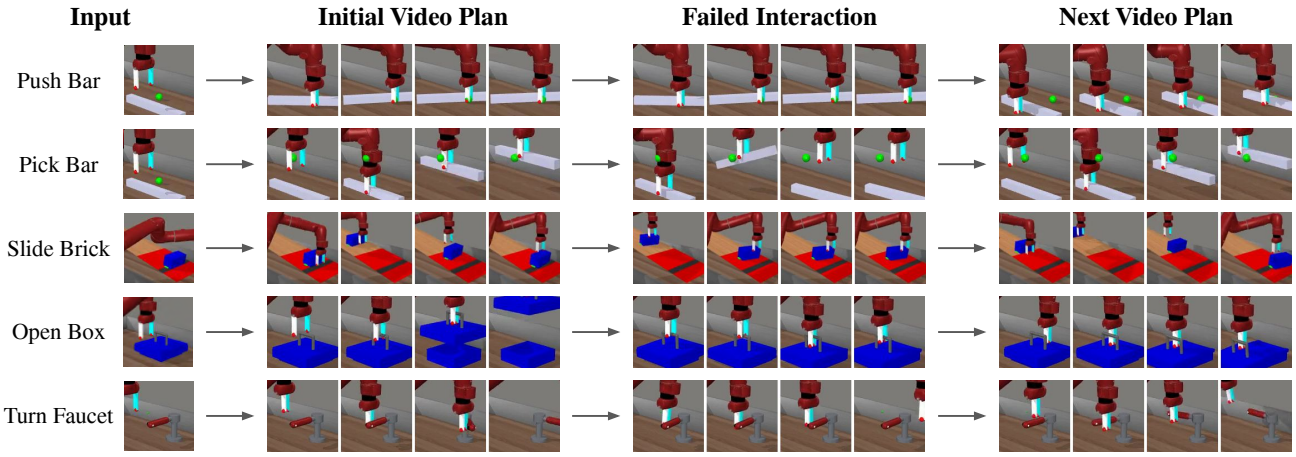


Figure 4. **Qualitative results of adapted video plans.** We show qualitative examples of adaptation trials for each task in the Meta-World Identification Benchmark to illustrate the effectiveness of the proposed method. The framework generates new plans based on previously failed interaction and video plans.

Table 2. **Number of replans until success.** Video planning methods, AVDC and ours, outperform baselines in imitation learning (BC) and offline RL (CQL) on most tasks. Our method and its variant consistently outperform the video planning baseline. The error terms show the standard error of the mean value. The reported error terms represent the standard error of the mean across 400 trials. For the CQL baseline, performance is averaged over three independently initialized and trained policies for more robust evaluation.

Method	Push Bar	Pick Bar	Slide Brick	Open Box	Turn Faucet	All (Normalized)
CQL	9.96± 0.20	9.22± 0.19	8.94± 0.18	3.32 ± 0.18	5.63 ± 0.21	2.56± 0.10
BC	10.26±0.30	8.79±0.32	9.24±0.33	1.59± 0.19	3.54± 0.27	1.98± 0.05
AVDC	6.83± 0.27	4.41± 0.22	7.36± 0.27	1.82± 0.16	1.67± 0.16	1.31± 0.04
Ours (Refine)	4.66±0.23	3.87±0.21	6.72±0.26	1.73±0.16	1.54±0.15	1.12± 0.03
Ours	**4.26**± 0.20	**3.84** ± 0.18	**6.69** ± 0.26	**1.25**± 0.10	**1.39**± 0.14	**1.00** ± 0.03

the benefits of video-based planning. BC and CQL struggle significantly on these tasks and often fail to complete them, frequently repeating similar predictions—suggesting overfitting or limited flexibility. Overall, our approach achieves the strongest performance, showing its broad applicability across both discrete and continuous parameter settings.

Our variants. Among the variations of our method, using the Rejection Module (Ours) alone to generate state embeddings achieves the best overall performance. In contrast, the refinement process introduced in Section 4.1.4 alone, Ours (Refine+FS), yields similar or slightly worse performance but does not assume data availability at test time, *i.e.*, do not require access to the dataset used to train the diffusion model. Additionally, its formulation as a learned model offers the potential for improved generalization. Finally, we show qualitative results for adaptations in Figure 4.

Video plan evaluation. In Table 3, we report the evaluations of video plans generated by the baseline (AVDC) and our methods with various context encoders E, including R3M (Nair et al., 2022), CLIP (Radford et al., 2021), and DINOv2 (Oquab et al., 2024). Our method produces significantly higher-quality videos compared to AVDC. Our method with DINOv2 outperforms its counterpart using R3M or CLIP, validating our design choices.

Table 3. **Evaluation on replanned videos.** We quantitatively assess the accuracy of the generated video by measuring the similarity between the framework's output plan and the ground-truth plan produced by the GT scripted policy.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
AVDC -	19.426	0.700	0.271
Ours (R3M)	19.632	0.704	0.261
Ours (CLIP)	19.584	0.703	0.262
Ours (DINOv2)	**19.817**	**0.708**	**0.255**

5.5. Ablation studies

Rejection and retrieval modules. To investigate the effectiveness of the Rejection and Retrieval Modules, we compare AVDC (Action Module only), AVDC+Rejection, AVDC+Retrieval, and our method (with both Rejection and Retrieval). The result in Figure 5 shows the normalized numbers of replans across all the tasks and individual task performance is shown in Appendix C. Both AVDC+Rejection and AVDC+Retrieval outperform

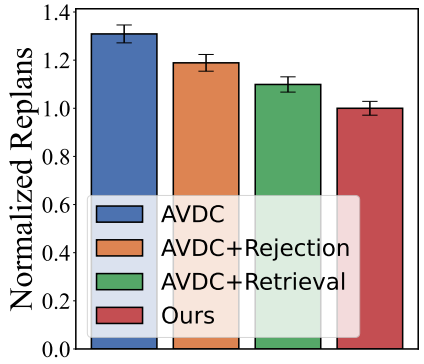


Figure 5. **Rejection and retrieval modules.** Both the Rejection and Retrieval Modules improve the performance from AVDC (Aciton Module only) and combining them (ours) yields the best performance. In this figure, the numbers of replans are normalized aggregated among the five tasks.

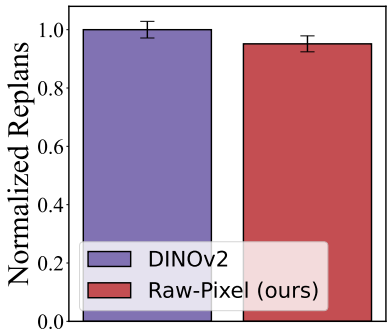


Figure 6. **Rejection module distance.** We compare using raw-pixel distance and DINOv2 embedding distance to reject and the former yields better performance. The numbers of replans are normalized and aggregated among the five tasks.

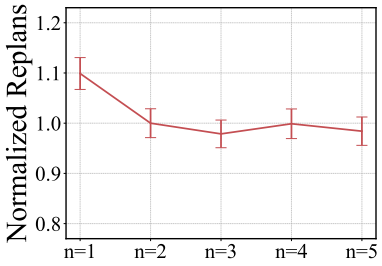


Figure 7. **Number of generated video plans.** We experiment with generating different numbers of video plans $n \in [1, 5]$ and set $n = 2$. In this figure, the numbers of replans are normalized and aggregated among the five tasks.

AVDC, justifying the effectiveness of the Rejection and Retrieval modules. Our full method (with both the Rejection and Retrieval modules) achieves the best performance, validating the combinations.

Rejection module distance metric. We compare pixel-wise distance and DINOv2-based similarity for selecting video plan candidates, as described in Section 4.2. Figure 6 shows that pixel-wise distance performs better, likely because DINOv2 struggles to distinguish these plans due to the nature of its pretraining data.

Number of generated video plans. To study how the number of video plans generated to be rejected affects the performance, we vary the number from 1 to 5 and show the result in Figure 7. The result aggregates the performance across all the tasks, and individual task performance is shown in Appendix C. From $n = 1$ (AVDC+Retrieval) to $n = 2$ (ours), around 10% of replans are saved. However, from $n = 2$ to $n = 5$, the numbers of replans until success are near the same. An increase in video plan candidates generated by the video generator results in an increase in the computational cost. Therefore, since the performances of $n \in \{2, 3, 4, 5\}$ are comparable, we set $n = 2$.

5.6. Evaluation on real world data

To demonstrate the method's ability to plan efficiently, we evaluated the video-planning framework on a real-world door-opening task. The door can be either pushed or pulled open, which can't be distinguished from the appearance and must be revealed through interaction. The setup is shown in Figure 8.

We collected 20 real-world videos (10 successes, 10 failures) to train the video model. We address the challenge of training on such a small dataset by adopting a joint training with simulation data: 40% of samples came from real-world data and 60% from the Meta-World System Identification Benchmark. To reduce overfitting to the first frames, which often leads to monotonic plans, we injected Gaussian noise into the first-frames during inference. Furthermore, we set $n = 4$ to promote diverse plan generation.

Figure 8. **Real-world setup.** A WidowX arm attempts to open a door that can either be pulled or pushed open.

For simplicity, we evaluate the single-episode replanning success rate, conditioned on an initial failure—different from the main evaluation metric. A perfect random policy, which assumes knowledge of the correct modes (e.g., push or pull) but does not use past interaction, would succeed with 50% probability. Our method achieved 15/20 successes, compared to 8/20 for a naive video planner without using past information, demonstrating the effectiveness of our pipeline. Full results and qualitative examples are in Appendix C.6.

6. Conclusion

Our framework enhances video-based decision-making by adapting to interaction-time uncertainties. Integrating real-time data enables implicit state estimation and effective replanning. Experiments on a simulated manipulation benchmark validate its ability to filter failed plans and update models online. Experimental results demonstrate improved robustness in uncertain environments, and ablation studies justify our design choices.

Impact Statements

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Ajay, A., Han, S., Du, Y., Li, S., Gupta, A., Jaakkola, T. S., Tenenbaum, J. B., Kaelbling, L. P., Srivastava, A., and Agrawal, P. Compositional foundation models for hierarchical planning. In *Neural Information Processing Systems*, 2023.

Bharadhwaj, H., Mottaghi, R., Gupta, A., and Tulsiani, S. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, 2025.

Black, K., Nakamoto, M., Atreya, P., Walke, H. R., Finn, C., Kumar, A., and Levine, S. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, 2024.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-1: Robotics transformer for real-world control at scale, 2023.

Chang, C.-Y., Huang, D.-A., Xu, D., Adeli, E., Fei-Fei, L., and Niebles, J. C. Procedure planning in instructional videos. In *European Conference on Computer Vision*, 2020.

Chen, A. S., Nair, S., and Finn, C. Learning generalizable robotic reward functions from" in-the-wild" human videos. *Robotics: Science and Systems*, 2021a.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021b.

Chen, S.-F., Wang, H.-C., Hsu, M.-H., Lai, C.-M., and Sun, S.-H. Diffusion model-augmented behavioral cloning. In *International Conference on Machine Learning*, 2024.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2024.

Du, Y., Yang, S., Dai, B., Dai, H., Nachum, O., Tenenbaum, J., Schuurmans, D., and Abbeel, P. Learning universal policies via text-guided video generation. In *Neural Information Processing Systems*, 2024a.

Du, Y., Yang, S., Dai, B., Dai, H., Nachum, O., Tenenbaum, J., Schuurmans, D., and Abbeel, P. Learning universal policies via text-guided video generation. In *Neural Information Processing Systems*, 2024b.

Escontrela, A., Adeniji, A., Yan, W., Jain, A., Peng, X. B., Goldberg, K., Lee, Y., Hafner, D., and Abbeel, P. Video prediction models as rewards for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Figueiredo Prudencio, R., Maximo, M. R. O. A., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

Florence, P., Lynch, C., Zeng, A., Ramirez, O., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, 2021.

Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022.

Gao, Z., Elibol, A., and Chong, N. Y. A 2-stage framework for learning to push unknown objects. In *International Conference on Development and Learning and Epigenetic Robotics*, 2020.

Gao, Z., Elibol, A., and Chong, N. Y. Estimating the center of mass of an unknown object for nonprehensile manipulation. In *IEEE International Conference on Mechatronics and Automation*, 2022.

Gerencser, L. and Hjalmarsson, H. Adaptive input design in system identification. In *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005.

Hjalmarsson, H., Gevers, M., and de Bruyne, F. For model-based control design, closed-loop identification gives better performance. *Automatica*, 1996.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Neural Information Processing Systems*, 2016.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823. Curran Associates, Inc., 2020.

Ko, P.-C., Mao, J., Du, Y., Sun, S.-H., and Tenenbaum, J. B. Learning to act from actionless videos through dense correspondences. In *International Conference on Learning Representations*, 2024.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Neural Information Processing Systems*, 2020a.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020b.

Kumar, K. N., Essa, I., Ha, S., and Liu, C. K. Estimating mass distribution of articulated objects using non-prehensile manipulation. *arXiv preprint arXiv:1907.03964*, 2019.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, J., Zhu, Y., Xie, Y., Jiang, Z., Seo, M., Pavlakos, G., and Zhu, Y. Okami: Teaching humanoid robots manipulation skills through single video imitation. In *Conference on Robot Learning*, 2024.

Lindqvist, K. and Hjalmarsson, H. Identification for control: adaptive input design using convex optimization. In *IEEE Conference on Decision and Control*, 2001.

Ljung, L. System identification. In *Signal analysis and prediction*. 1998.

Ma, Y. J., Kumar, V., Zhang, A., Bastani, O., and Jayaraman, D. LIV: Language-image representations and rewards for robotic control. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023.

Mavrakis, N., Stolkin, R., et al. Estimating an object's inertial parameters by robotic pushing: a data-driven approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

Memmel, M., Wagenmaker, A., Zhu, C., Fox, D., and Gupta, A. ASID: Active exploration for system identification in robotic manipulation. In *International Conference on Learning Representations*, 2024.

Menda, K., De Becdelievre, J., Gupta, J., Kroo, I., Kochenderfer, M., and Manchester, Z. Scalable identification of partially observed systems with certainty-equivalent em. In *International Conference on Machine Learning*, 2020.

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

Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Qi, H., Kumar, A., Calandra, R., Ma, Y., and Malik, J. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, 2023.

Qin, Y., Shi, Z., Yu, J., Wang, X., Zhou, E., Li, L., fei Yin, Z., Liu, X., Sheng, L., Shao, J., Bai, L., Ouyang, W., and Zhang, R. Worldsimbench: Towards video generation models as world simulators. *ArXiv*, 2024.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Schön, T. B., Wills, A., and Ninness, B. System identification of nonlinear state-space models. *Automatica*, 2011.

Shridhar, M., Manuelli, L., and Fox, D. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2022.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018.

Wang, Z., Merel, J. S., Reed, S. E., de Freitas, N., Wayne, G., and Heess, N. Robust imitation of diverse behaviors. In *Neural Information Processing Systems*, 2017.

Wen, C., Lin, X., So, J., Chen, K., Dou, Q., Gao, Y., and Abbeel, P. Any-point trajectory modeling for policy learning. In *Robotics: Science and Systems*, 2024.

Xu, Z., Wu, J., Zeng, A., Tenenbaum, J. B., and Song, S. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

Yang, R., Bai, C., Ma, X., Wang, Z., Zhang, C., and Han, L. Rorl: Robust offline reinforcement learning via conservative smoothing. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 23851–23866. Curran Associates, Inc., 2022.

Yang, S., Du, Y., Ghasemipour, S. K. S., Tompson, J., Kaelbling, L. P., Schuurmans, D., and Abbeel, P. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024a.

Yang, S., Walker, J. C., Parker-Holder, J., Du, Y., Bruce, J., Barreto, A., Abbeel, P., and Schuurmans, D. Position: Video as the new language for real-world decision making. In *International Conference on Machine Learning*, 2024b.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14129–14142. Curran Associates, Inc., 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 28954–28967. Curran Associates, Inc., 2021.

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

Zare, M., Kebria, P. M., Khosravi, A., and Nahavandi, S. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 2024.

Zhang, J., Wang, K., Xu, R., Zhou, G., Hong, Y., Fang, X., Wu, Q., Zhang, Z., and Wang, H. Navid: Video-based vlm plans the next step for vision-and-language navigation. In *Robotics: Science and Systems*, 2024.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Zhou, S., Du, Y., Chen, J., LI, Y., Yeung, D.-Y., and Gan, C. Robodreamer: Learning compositional world models for robot imagination. In *International Conference on Machine Learning*, 2024.

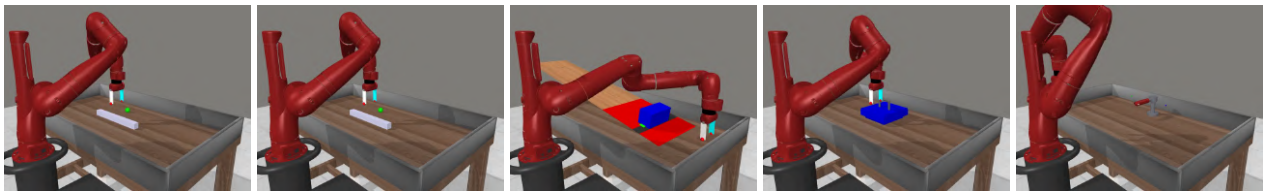
Appendix

A. Supplementary materials

A.1. Website and Code

We present extensive qualitative results and release our code in our website.

B. Benchmark Details of Simulator Tasks



Push Bar | Pick Bar | Slide Brick | Open Box | Turn Faucet

Figure 9. Settings of five task environments.

The environments of the five tasks are shown in Figure 9.

B.1. Push Bar

The objective of this task is to push a bar towards the target. Physically, the bar is combined with a mass point and a mass-free bar. Therefore, if the robot does not push the bar exact at the center of mass, torque is imposed on the bar, resulting in the deflection of the bar.

The bar has a length of 0.4 meters. In the pretraining stage the video generation model, the distances between the center of mass and geometric center of the bar are shown in Table 4.

Table 4. **Center of mass positions.** The distances between the center of mass and the geometric center of the bar in the pretraining stage of the video generation model. The sign represents the direction.

-0.18	-0.165	-0.15	-0.135	-0.12
-0.105	-0.09	-0.075	-0.06	-0.05
-0.03	-0.015	0.0	0.015	0.03
0.045	0.06	0.075	0.09	0.105
0.12	0.135	0.15	0.18	

The visualization of this task is shown in Figure 10.

B.2. Pick Bar

The objective of this task is to grasp a bar towards the target. Physically, like push-bar mentioned above, the bar is combined with a mass point and a mass-free bar. Therefore, if the robot does not grasp the bar exact at the center of the mass, torque is imposed on the bar, resulting in the tilt of the bar.

The properties of the bar in this task are the same as the bar in push-bar. And the distances between the center of mass and geometric center of the bar in the pretraining stage are also the same as above.

The visualization of this task is shown in Figure 11.

B.3. Slide Brick

There are two stages in this task. In the first stage, the robot pushes a brick up an inclined surface. In the second stage, the brick slides freely down the slope. The objective is to push the brick at a proper height so that the brick can slide down and stop at the gray band.

Note that the wooden slope has a relatively low friction coefficient and that the red slope has a relatively high friction coefficient. Therefore, the brick can slide down and accelerate in the wooden slope and decelerate in the red slope. The friction coefficient of the wooden slope is fixed and is set to 0, while the friction coefficient of the red slope is various, and those in the pretraining stage are shown in Table 5.

The visualization of this task is shown in Figure 12.

Table 5. **Friction coefficients.** The friction coefficients of the red slope in the pretraining stage of the video generation model.

0.24	0.25	0.26	0.27	0.28	0.3	0.32
0.34	0.35	0.36	0.38	0.39	0.4	

B.4. Open Box

The objective of this task is to open a box whose lid operates in one of two possible ways: it either needs to be lifted or slid open. The specific mode is chosen at random and cannot be identified visually, so the robot must physically interact with the box and adjust its strategy based on the observed response.

The visualization of the task is shown in Figure 13.

B.5. Turn Faucet

The objective of this task is to rotate a faucet. The direction of rotation—clockwise or counterclockwise—is randomly assigned and cannot be determined in advance through observation. The robot must interact with the faucet to identify the correct direction and execute the appropriate motion.

The visualization of the task is shown in Figure 14.

C. Additional Experiment Details

C.1. Video diffusion model

Our video diffusion model adopts a U-Net architecture following (Ko et al., 2024). The original implementation conditions on the first frame and the task name tokens, where the task tokens are encoded using CLIP-Text, to predict future frames. Building on this framework, we additionally condition the video generation process on past interaction videos. Specifically, we encode the past interaction video as described in Section 4.1.5, and concatenate the resulting embedding with the encoded task tokens after projecting it to match the CLIP-Text output dimensionality (512 channels) using a trainable linear layer.

To improve computational efficiency, the video diffusion model generates a low-resolution video plan through the denoising process. This low-resolution plan is then upsampled to the target resolution using a separate super-resolution model that predicts residuals over the upsampled frames. The diffusion model is trained using the v-parameterization approach with a learning rate of 1e-4. Key hyperparameters for the video diffusion model are summarized in Table 6.

Table 6. **Hyperparameters.** Comparison of configuration parameters for the Meta-World benchmark.

num_parameters	125M
diffusion_resolution	(32, 32)
target_resolution	(128, 128)
base_channels	128
num_res_block	2
attention_resolutions	(2, 4, 8)
channel_mult	(1, 2, 3, 4)
batch_size	128
training_timesteps	12k
denoising_timestep	100
sampling_timestep	25

C.2. Rejection and retrieval modules

Tables 7 and 8 show the per-task performance for the experiment in Figure 5.

Table 7. Number of replans until success.

Method	Push Bar	Pick Bar	Slide Brick	Open Box	Turn Faucet	All (Normalized)
AVDC	6.83± 0.27	4.41± 0.22	7.36± 0.27	1.82± 0.16	1.67± 0.16	1.30± 0.4
AVDC+Rejetion	6.21± 0.25	**3.77**± 0.19	7.16± 0.26	1.62± 0.15	1.52± 0.15	1.18± 0.03
AVDC+Retrieval	4.91± 0.22	4.1± 0.20	6.86± 0.25	1.38± 0.13	1.60± 0.14	1.10± 0.03
Ours	**4.26**± 0.20	3.84± 0.18	**6.69**± 0.26	**1.25**± 0.10	**1.39**± 0.14	**1.00**± 0.03

Table 8. Evaluation for video replan videos

Method	PSNR ↑	SSIM ↑	LPIPS ↓
AVDC	19.426	0.700	0.271
AVDC+Rejection	19.459	0.700	0.265
AVDC+Retrieval	19.521	0.703	0.264
Ours	**19.817**	**0.708**	**0.255**

C.3. Rejection module distance

Tables 9 and 10 show the per-task performance for the experiment in Figure 6.

Table 9. Number of Replans until Success.

Distance Metric	Push Bar	Pick Bar	Slide Brick	Open Box	Turn Faucet	All (Normalized)
Raw-Pixel (ours)	**4.26**± 0.20	**3.83**± 0.18	6.69± 0.26	**1.25**± 0.10	**1.39**± 0.14	**1.00**± 0.03
DINOv2	4.83± 0.22	4.08± 0.20	**6.62**± 0.26	1.27± 0.11	1.47± 0.14	1.05± 0.03

Table 10. Evaluation for video replan videos

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Raw-Pixel (ours)	**19.817**	**0.708**	**0.255**
DINOv2	19.787	**0.708**	0.256

C.4. Number of generated video plans

Tables 11 and 12 show the per-task performance for the experiment in Figure 7.

Table 11. Number of replans until success.

n	Push Bar	Pick Bar	Slide Brick	Open Box	Turn Faucet
n=1	4.91± 0.22	4.1± 0.20	6.86± 0.25	1.38± 0.13	1.60± 0.14
n=2	4.26± 0.20	3.83± 0.18	**6.69**± 0.26	1.25± 0.10	1.39± 0.14
n=3	4.18± 0.19	3.63± 0.18	7.23± 0.26	**1.13**± 0.10	1.36± 0.13
n=4	4.19± 0.20	**3.47**± 0.18	6.91± 0.24	1.22± 0.11	1.53± 0.15
n=5	**4.15**± 0.19	3.60± 0.19	7.24± 0.26	1.22± 0.11	**1.33**± 0.13

Table 12. Evaluation for video replan videos

n	PSNR ↑	SSIM ↑	LPIPS ↓
n=1	19.459	0.700	0.271
n=2	**19.817**	**0.708**	**0.255**
n=3	19.723	0.706	0.258
n=4	19.617	0.703	0.262
n=5	19.483	0.700	0.266

C.5. Runtime analysis

We measured the average computation time over 100 iterations on a machine with an Intel Xeon W-2255 CPU and an NVIDIA RTX 3080 Ti GPU. Each iteration includes embedding retrieval, video generation (DDIM), plan selection (reject), and action decoding via optical flow. As shown in Table 13, planning takes only a few seconds per episode, making our framework practical for real-time interaction.

Table 13. **Computation time (in seconds)** averaged over 100 trials.

Component	AVDC	Ours	Ours + Refine
Retrieve	N/A	0.487 ± 0.008	11.876 ± 0.036
DDIM	0.777 ± 0.001	1.573 ± 0.004	1.599 ± 0.006
Reject	N/A	1.276 ± 0.075	1.230 ± 0.068
Optical Flow	1.564 ± 0.021	1.661 ± 0.030	1.761 ± 0.028
Total	2.341 ± 0.021	4.997 ± 0.081	16.466 ± 0.082

C.6. Real-world experiment

Table 14. **Real-World results. AVDC** represents naive video planning frameworks without our proposed replanning and rejection strategies. **Ours** shows the performance of our full video generation pipeline.

Method	Success Rate
AVDC	8/20
Ours	15/20

Figure 16 shows the qualitative rollout for our real-world experiment. We followed most of the hyperparameters as used in the Meta-World System Identification Benchmark. Additionally, we changed $n = 4$ and injected Gaussian noise of standard deviation 0.35 into first-frames with pixel values normalized to $[0, 1]$ to mitigate the first-frame bias issue.

D. Limitation

To isolate the challenge of video-based belief refinement, our work assumes a reliable action module, attributing failures solely to planning errors. While this abstraction simplifies evaluation, it omits real-world execution noise, which future work could address through uncertainty-aware control. Our visually driven approach enables interpretability but struggles with visually ambiguous tasks (e.g., subtle directional differences); incorporating tactile or proprioceptive sensing could improve disambiguation. We also observe a "first-frame bias" in the video generator under low-data regimes, where plans become overly deterministic. Although we have shown in Section 5.6 that such bias can be mitigated via increasing the number of candidate plans n and noise injection to the first-frame image, future methods may benefit from diversity-promoting strategies or explicit mode learning.

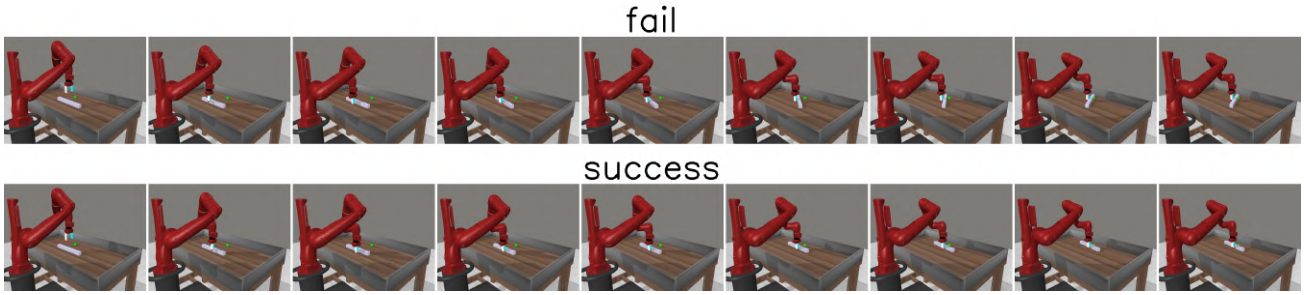


Figure 10. Push Bar. The above video shows a failed trial, because the deflection of the bar is too large so that the bar cannot reach the target.

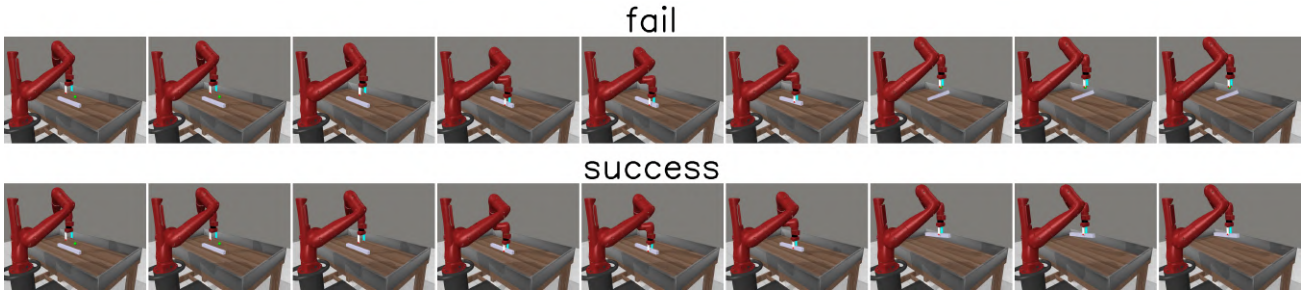


Figure 11. Pick Bar. The above video shows a failed trial, because the bar fell down.

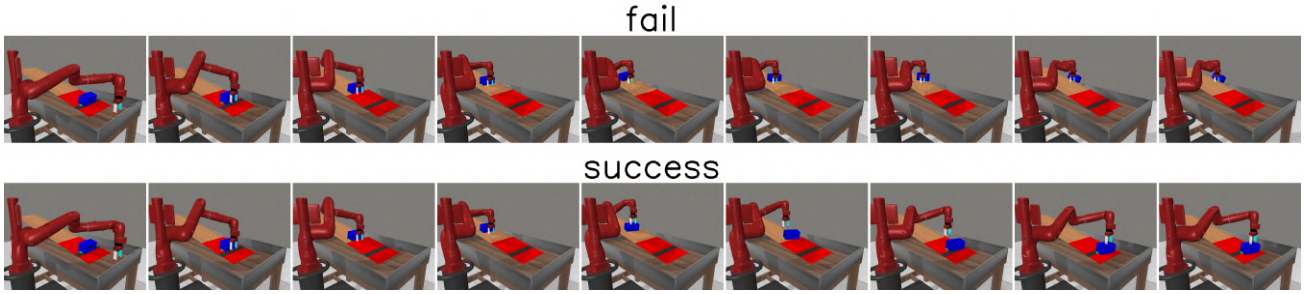


Figure 12. Slide Brick. The above video shows a failed trial, because the brick didn't slide down to the gray band.

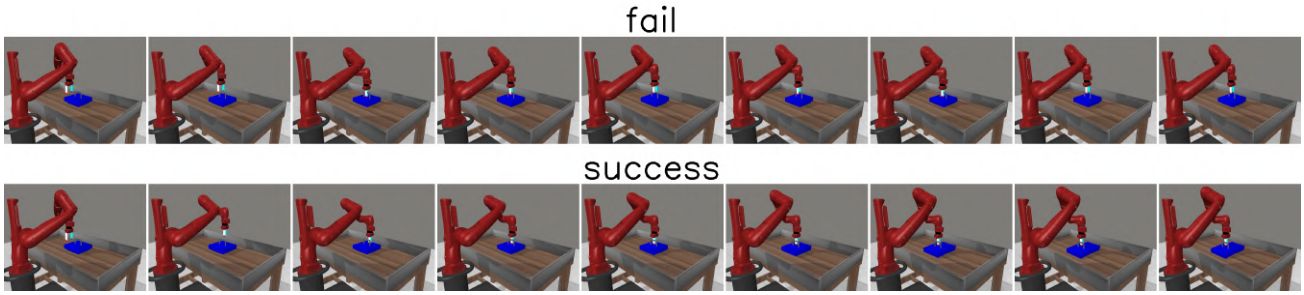
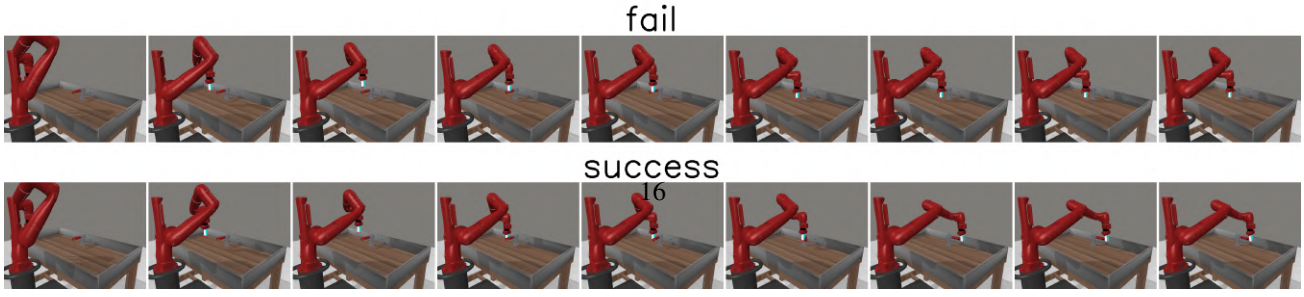


Figure 13. Open Box. The above video shows a failed trial, becuase the robot try to grasp the handle up. However, the right way to open the box is to push the handle horizontally.



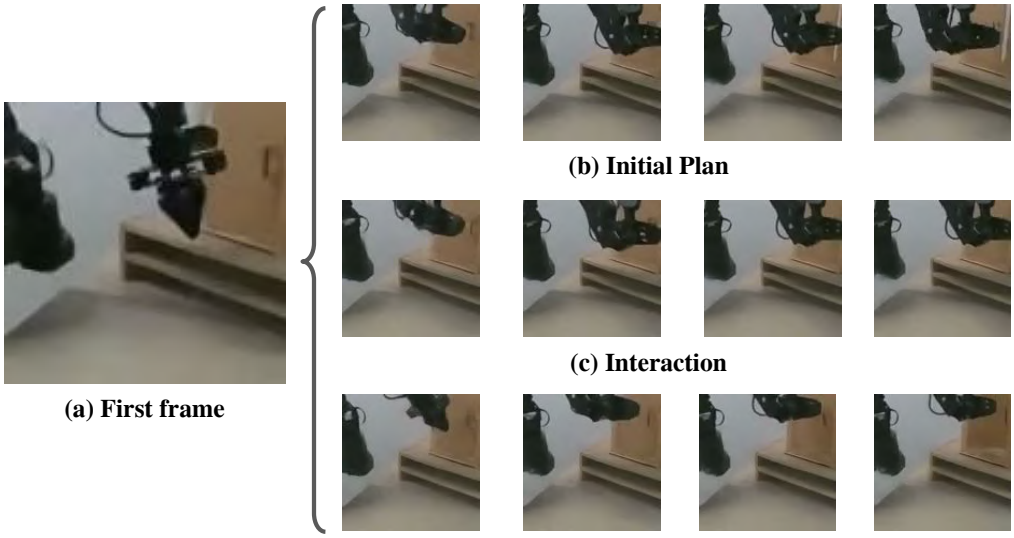


Figure 16. **Real-world qualitative results.** Given the first frame (a), the robot first attempts to push the door (b). Then, it gets stuck and fails to open the door (b). Finally, it generates a plan that pushes the door instead (c) by leveraging information from previous failure.