Inference-Time Planning with Action-Conditioned Video Models for Generalizable Robot Manipulation

Zhiting Mei†, Yanbo Xu, Tenny Yin, Ola Shorinwa†, Anirudha Majumdar

***Abstract*— We propose VidPlan, an inference-time planning framework that uses action-conditioned video models as high-fidelity world models for robot manipulation. In contrast to state-of-the-art (SOTA) generalist policies learned via behavior cloning, a world modeling recipe enables counterfactual reasoning with context-specific information at inference time, leading to stronger zero-shot generalization. Moreover, world models can enable the optimization of action sequences that maximize predicted rewards. However, we demonstrate that naively utilizing a video model for inference-time optimization can result in *world model hacking*, where optimized action sequences exploit hallucinations in the world model without leading to successful real-world executions. To mitigate this challenge, we develop a novel *uncertainty quantification* method that constrains action optimization to high-confidence regions. Further, we introduce a novel *hierarchical reward prediction* model to capture both semantic and fine-grained task progress, designed within a *flow map framework* for real-time optimization of video plans. Through extensive experiments on manipulation tasks, we demonstrate that VidPlan achieves improved generalization, higher task success rates, and stronger instruction following compared to SOTA vision-language-action baselines.**

I. Introduction

How can we train generalist policies for robot manipulation that have the ability to *plan*, i.e., to optimize actions at inference time by making counterfactual predictions about how the world will evolve? Planning is a core component of both human cognition [1] and artificial intelligence [2], [3]. However, SOTA robot policies [4], [5] are dominated by behavior cloning [6], [7], which trains *reactive* mappings from observations to actions using human demonstrations. In principle, planning affords several key advantages over behavior cloning for manipulation. For example, the planner can evaluate multiple different strategies using context-specific information *at inference time* for stronger generalization than behavior-cloning. Further, using its "world model" [8], the robot can optimize actions, considering fine-grained effects beyond the immediate future.

In this paper, we demonstrate the use of action-conditioned video generation models [9], [10] as high-fidelity world models for planning in robot manipulation. In theory, video models serve as ideal world models; through internet-scale training on video data, they can capture complex physical interactions (e.g., contact dynamics and deformations), produce highly photorealistic outputs with fine-grained details, and be fine-tuned with domain-specific robotics data. However, to date, the use of video models for inference-time planning in manipulation has been extremely limited. Prior and concurrent work focus on planning with task-specific reward functions [11], using video models to generate subgoal images for an independent low-level goal-conditioned policy [12], using text-to-video models to generate a video plan which is executed without inference-time optimization with respect to a reward function [13], [14], or selecting from a small number of generated plans without further local (e.g., gradient-based) optimization [15].

To our knowledge, there is *no prior work* that demonstrates a generalist (language-conditioned) policy for robot manipulation which uses a video model for inference-time, gradient-based planning. We posit that realizing the full potential of this vision requires solutions to three key challenges. First, gradient-based optimization of plans can result in *world model hacking*: optimized actions exploit hallucinations in the world model to predict high rewards without successful real-world execution. Second, *inference speed* is a major bottleneck: modern diffusion-based video models induce high latency, hindering real-time execution. Third, successful planning requires a *general-purpose reward model* that takes into account both high-level semantic features of plans (e.g., whether the robot is moving towards the object implied by the language instruction) and fine-grained details (e.g., precisely how the robot is approaching an object).

Contributions. We introduce **VidPlan**, an inference-time planning framework for generalist robot manipulation using high-fidelity video models (see Figure 1). VidPlan consists of three technical innovations that address the aforementioned challenges. **(1)** We propose a novel method for *calibrated uncertainty estimation* for action-conditioned video models to ensure that optimized actions are constrained to high-confidence world model predictions. **(2)** We demonstrate the use of *flow maps* [16] to achieve significant (e.g., $20 \sim 50\times$) speedups in video generation without degrading plan quality. **(3)** We develop a *hierarchical reward modeling* approach that captures both high-level semantic features of plans and fine-grained details. Extensive experiments demonstrate significant improvements in instruction following, overall success rates, and efficiency compared to SOTA generalist policies [4]. To our knowledge, VidPlan represents the first work to showcase generalist manipulation capabilities using physically-grounded, uncertainty-aware video models for gradient-based planning.

II. Related Work

Video Generation Models as Embodied World Models. Advances in controllable video synthesis [17], [18], [19] has catalyzed recent work in robotics exploring the use of

†Equal contribution

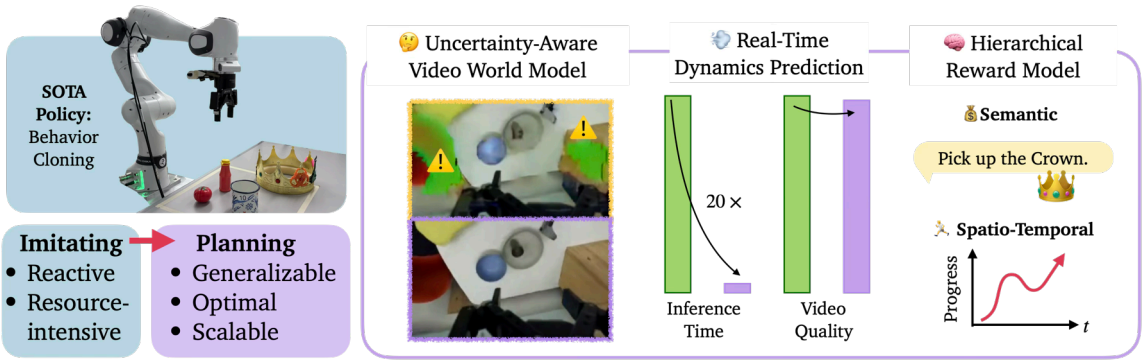

Fig. 1: **VidPlan** enables robots to *plan* at inference time, leveraging context-specific information and counterfactual reasoning to achieve stronger generalization compared to SOTA behavior-cloned policies.

video models as embodied world models, which predict future states conditioned on robot actions (see [20] for a survey). However, video models remain highly susceptible to hallucinations, such as objects that appear or disappear, and unrealistic contact dynamics and deformations [21], [22], [23]. Additionally, video generation incurs prohibitive inference-time compute costs, e.g., one future prediction takes up to 10 seconds on A100 GPUs [10]. For faster inference, recent work has sought to accelerate the sampling process through architectural improvements [24], [25], [26] and distillation-based methods [27], [28], [29], [30], [31], [32]. Other approaches leverage flow matching models, offering higher inference speeds [33]. In this work, we propose using a flow map for diffusion-to-flow-matching distillation to achieve up to $50\times$ faster planning.

Uncertainty Quantification for Video Models. Generative models have a strong propensity to hallucinate [34], [35], [36], [37], [38], [39]. Prior work [40], [41] has explored uncertainty quantification (UQ) as a mechanistic tool to identify and mitigate hallucinations. The effectiveness of these methods has driven a rapid growth in uncertainty quantification to improve the explainability of the often obscure decision-making processes of large generative models [42]. In general, UQ methods for generative models avoid traditional approaches, such as Bayesian Neural Networks [43], variational inference [44], and Deep Ensembles [45], due to their prohibitive computation costs, opting for more efficient strategies for estimating the model's confidence. Notably, while a large body of work [46], [47], [42] exists for UQ of LLMs and image generation models, only a few papers have examined UQ of video models. Prior work S-QUBED [48] estimates the total predictive uncertainty of video models, disentangling its aleatoric and epistemic components. However, S-QUBED is limited to text-to-video models. Another work C^3 [49] explores dense confidence prediction but relies on the selection of a prediction accuracy threshold, which can be challenging to specify *a priori*. To address these limitations, we introduce a novel UQ method that learns the distribution of prediction errors to derive calibrated confidence estimates [50], [51].

World Models for Planning in Robot Manipulation. Latent world models have been adopted to predict future states for inference-time planning [52], [53], [54], but are limited to goal-conditioned and localization tasks. Prior work [11] employs video models as dynamics predictors to compute action trajectories by minimizing the photometric error between generated and goal images or image keypoints. However, image reconstruction error generally fails to capture semantic notions of task completion, which is determined by salient interactions that are crucial for evaluating true success. Other methods [12] utilize VLMs as commonsense evaluators for video plans but rely on goal-conditioned policies for low-level control. Subsequent work [13], [14], [15] directly generates robot actions within the video plan, eliminating the need for separate low-level policies. However, these methods either do not optimize the resulting actions at inference-time [13], [14] or only select actions from a small set of candidate actions without gradient-based optimization [15], limiting their effectiveness. In contrast, our method utilizes a general-purpose reward model to identify promising candidate actions, which is subsequently refined these actions through gradient-based optimization.

Reward Modeling. Designing an appropriate objective function has been a longstanding challenge for both planning and reinforcement learning. Prior work has explored learned reward models that track task progress as value signals [55]. Given the significant cost of manual reward engineering and the challenge of generalization to new tasks, recent methods have explored leveraging the commonsense understanding of VLMs to evaluate robot trajectories [56], [57], [58], [59]. These methods often aim to track the progress of the task by finetuning on both successful and (synthetic) failure robot trajectories. Another line of work directly utilizes video models as reward models, for example, using video prediction likelihood or conditional entropy as reward signals [60], [61].

III. Problem Formulation

We formulate the robot manipulation task as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, f, \mathcal{R} \rangle$, representing the state space, action space, observation space, dynamics model, and reward model. To enable robot planning, we parameterize the environment dynamics model using an action-conditioned video model $\mathcal{V}_\theta : \mathcal{O}^T \times \mathcal{A}^H \to \mathcal{O}^H$ that stochastically generates future

frames given a sequence of actions $a_{t:t+H-1} \in \mathcal{A}^H$ of horizon H, as well as the history and current robot observations $o_{t-T+1:t} \in \mathcal{O}^T$ of length T. A state transition results in a reward $r_{t+1:t+H} \sim \mathcal{R}(\cdot \mid o_{t+1:t+H}, a_{t:t+H-1}, \ell)$, given the future observations, actions, and task instruction ℓ.

IV. Inference-Time Planning with Video Models

We introduce VidPlan, our inference-time planning framework for robot manipulation, which consists of (i) an *uncertainty-aware* video generation model for high-fidelity dynamics prediction with calibrated confidence estimates (Section IV-A), (ii) an efficient *flow map* for deriving compatible flow-matching models from diffusion models for faster inference (Section IV-B), and (iii) a hierarchical reward model (Section IV-C) that captures *semantic* and *spatio-temporal* progress, fusing the commonsense knowledge of vision-language foundation models with video models. Together, these components enable inference-time counterfactual reasoning and robot action optimization, as illustrated in Figure 2.

We formulate video generation using a diffusion process[1]. Given a current observation $o_{t-T+1:t}$ and an action chunk $a_{t:t+H-1}$ over a prediction horizon of length H, the video model $\mathcal{V}_\theta$ predicts the future evolution $o_{t+1:t+H}$ of the environment: $o_{t:t+H} \sim \mathcal{V}_\theta(\cdot \mid o_{t-T+1:t}, a_{t:t+H})$. We assume that the robot's observations at each timestep consist of three camera views (left and right scene cameras and a wrist camera) stacked into a single frame, while the robot's actions consist of the 6-degree-of-freedom cartesian end-effector pose (linear position and Euler angles) and the gripper position. Notably, video models are limited by two critical challenges when applied to robot planning: (i) they *hallucinate* often, confidently generating inaccurate future predictions that could degrade the real-world performance of computed plans, and (ii) they *do not* run in real-time, making online planning infeasible. We address these outstanding challenges via uncertainty-aware video generation and flow maps, respectively.

VidPlan uses an optimization-based paradigm for inference-time robot planning, defined by the optimization problem:

$$\begin{aligned} \underset{a_{t:t+H-1}}{\text{maximize}} \quad & \mathcal{J}\left(\mathcal{R}(o_{t+1:t+H}, a_{t:t+H-1}), h_{t+1:t+H}\right) \\ \text{subject to} \quad & (o_{t+1:t+H}, h_{t+1:t+H}) = \mathcal{V}_\theta(\cdot; \mathcal{U}_\beta), \end{aligned} \tag{1}$$

with dynamics model $\mathcal{V}_\theta$, UQ head $\mathcal{U}_\beta$, and objective $\mathcal{J}$, which is a function of the predicted rewards $\mathcal{R}(\cdot)$ and the video model's uncertainty $h_{t+1:t+H}$. Directly optimizing the problem in (1) is generally infeasible in most practical situations. Hence, we adopt a two-stage optimization process that fits elegantly with our hierarchical reward model. Specifically, given observations $o_{t-T+1:t}$ and an instruction ℓ, VidPlan first generates a suite of candidate actions $\{a^i_{t:t+H-1}\}_{i=1}^n$ through an action proposer u. Subsequently, each action is rolled out with the video model $\mathcal{V}_\theta$, yielding future states $\{o^i_{t+1:t+H}\}_{i=1}^n$, which are then evaluated with the semantic reward model $\mathcal{R}_u$. However, the planner may exploit dynamics and rewards prediction errors to maximize $\mathcal{J}$. To mitigate this occurrence, we modulate the reward predictions using the video model's uncertainty, limiting the effects of physically-inconsistent future predictions. Further, we refine the action associated with the highest reward to obtain $a^\star_{t:t+H-1}$ through uncertainty-aware, gradient-based optimization of predicted rewards from the spatio-temporal reward model $\mathcal{R}_l$. Thereafter, we execute the resulting action on the robot and repeat the planning procedure. (We summarize the key components of our framework in Appendix I-F.) Next, we discuss each component of VidPlan in greater detail.

[1]We provide a brief overview of video diffusion models in Appendix II.

A. Uncertainty-Aware Diffusion Modeling

To derive calibrated uncertainty estimators for video generation, we recall that diffusion models can be formulated using probability flow ordinary differential equations (PF-ODEs) [51]: $\mathrm{d}\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))\mathrm{d}t$, with a predefined noise level scheduler σ. Diffusion models are trained to model the score $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))$. Following [51], we train $\mathcal{V}_\theta$ to optimize the loss function:

$$\begin{aligned} \mathbb{E}_{\sigma, \mathbf{x}_1, \mathbf{x}_0}\Big[w(\sigma) \Big\| & \mathcal{V}_\theta(c_{\text{in}}(\sigma) \cdot \mathbf{x}_c; c_{\text{noise}}(\sigma)) \\ & - \frac{1}{c_{\text{out}}(\sigma)}(\mathbf{x}_1 - c_{\text{skip}}(\sigma) \cdot \mathbf{x}_c) \Big\|_2^2 \Big], \end{aligned} \tag{2}$$

where $w(\sigma) = \lambda(\sigma) c_{\text{out}}(\sigma)^2$, $\sigma \sim p_d$, $\mathbf{x}_c = \mathbf{x}_1 + \sigma \mathbf{x}_0$, $\mathbf{x}_1 \sim p_{\text{data}}$, and $\mathbf{x_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. $(c_{\text{in}}, c_{\text{out}}, c_{\text{skip}})$ and c_{noise} are scaling factors and conditioning inputs, respectively.

In Equation (2), note that no video predictor can perfectly recover the denoised data sample $\mathbf{x}_1$, given only partial observability of the random variables $\mathbf{x}_1$ and $\mathbf{x}_0$ through $\mathbf{x}_1 + \sigma \mathbf{x}_0$. Concretely, for $\mathbb{E}_{\sigma, \mathbf{x}_1, \mathbf{x}_0}[\cdot] = 0$, the predicted output $\mathcal{V}_\theta(\cdot)$ must be identically equal to the target $\frac{1}{c_{\text{out}}(\sigma)}(\mathbf{x}_1 - c_{\text{skip}}(\sigma) \cdot \mathbf{x}_c)$, or more generally, both quantities must be almost surely equal in a probabilistic sense. However, training a video predictor to satisfy this objective is infeasible in practice, without making restrictive assumptions about the distribution of $\mathbf{x}_1$. Furthermore, given the stochasticity of the target, deterministic video predictors generally underperform stochastic predictors, as shown in previous work [62], [63], [51]. As a result, many SOTA diffusion models [50], [51], [63] utilize stochastic sampling processes to learn the mean of the target data, noise, or velocity. This design choice induces a distribution p_ℓ over the prediction error:

$$\begin{aligned} \ell(\cdot) = \mathcal{V}_\theta & (c_{\text{in}}(\sigma) \cdot (\mathbf{x}_1 + \sigma \mathbf{x}_0)) \\ & - \frac{1}{c_{\text{out}}(\sigma)}(\mathbf{x}_1 - c_{\text{skip}}(\sigma) \cdot (\mathbf{x}_1 + \sigma \mathbf{x}_0)). \end{aligned} \tag{3}$$

where we have omitted the conditioning input $c_{\text{noise}}(\sigma)$ for notational simplicity. The randomness of ℓ results from the stochasticity of $\mathbf{x}_1$ and $\mathbf{x}_0$. We posit that the distribution of p_ℓ provides useful insights into the knowledge boundaries of $\mathcal{V}_\theta$. In particular, the entropy of the error distribution represents an interpretable measure of uncertainty in the video model's generated outputs. For example, highly stochastic dynamic interactions would result in a higher-entropy error distribution.

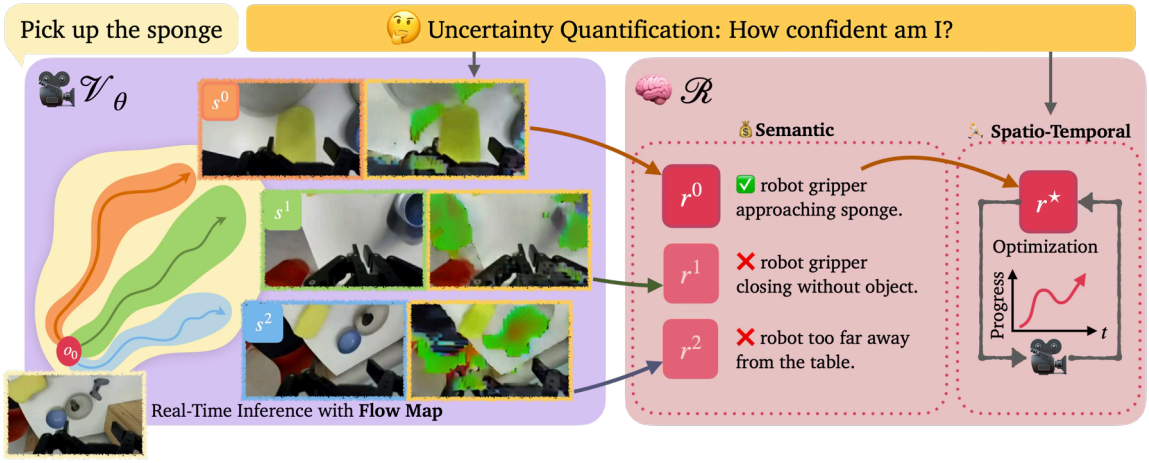

Fig. 2: **VidPlan Architecture.** VidPlan is composed of three key components: (i) an uncertainty-aware video generation model for dynamics prediction, (ii) a novel flow map for diffusion-to-flow-matching distillation for faster inference, and (iii) a hierarchical reward model for capturing semantic and spatio-temporal progress from videos.

Likewise, the expected error of the distribution provides a measure of the predicted accuracy of the generated video. We leverage these metrics during action optimization to ensure that dynamics and rewards predictions remain physically consistent with the real-world. Inspired by prior work [49], we discretize the latent error space into bins and predict the probability mass associated with each bin. This design choice preserves the generality of our proposed approach without the restrictions that arise from simplifying assumptions on the true distribution of the latent error. However, in contrast to [49], our method does not require the selection of an error threshold for broader effectiveness.

We utilize a U-Net as the backbone for the UQ model $\mathcal{U}_\beta$. To learn temporal relationships, we augment the U-Net with a self-attention transformer, incorporated in the bottleneck layer. We train the UQ model using the cross-entropy loss function for calibrated probabilistic predictions, in line with prior work. We discuss further implementation details in Appendix I-F.

B. Real-time Video Generation via Flow Map

Although diffusion models excel as probabilistic models for complex distributions such as world dynamics, their slow inference speed remains a major bottleneck for real-time planning. Leveraging the computation efficiency of flow matching [33], we develop an approach that can fine-tune diffusion world models for faster inference. We adopt the flow map framework [16], which provides a unified perspective over various distillation methods while offering flexibility in design and stability in training. Among the algorithmic families it introduces, we select Progressive Self-Distillation (PSD) because it neither requires Jacobian–vector product (JVP) operations as in [31], nor does it require maintaining additional copies of the base network as in distribution-matching approaches [32], [64], [26]. This design choice enables efficient stable training for large-scale world models.

A flow map $\mathbf{X}_{s,t}(\mathbf{x})$ [16] maps a noisy input $\mathbf{x}$ at time s to a less noisy prediction at time t, where $\mathbf{x}_s$ and $\mathbf{x}_t$ are solutions to the probability flow satisfying $\frac{d\mathbf{x}_t}{dt} = b_t(\mathbf{x}_t)$, where b_t is the velocity field as in flow matching. A commonly used parameterization is $\mathbf{X}_{s,t}(\mathbf{x}) = \mathbf{x} + (t-s)\mathbf{v}_{s,t}(x)$, where $\mathbf{v}_{s,t}$ denotes the mapping field at time s with the desired ending time t that we need to model. Directly training the model with randomly sampled pairs of s and t leads to unstable predictions. To improve training stability, we leverage the property: $\lim_{t\to s} \partial_t \mathbf{X}_{s,t}(\mathbf{x}_s) = \mathbf{v}_{s,s}(\mathbf{x}) = b_s(\mathbf{x}_s)$ [16] to fine-tune the flow map model using the loss function $L = L_{\text{flow}}(v) + L_{\text{distill}}(v)$, where $\mathbf{v}_{s,s}$ recovers the flow matching velocity field b_s.

To prevent training collapse, the term $\mathbf{X}_{u,t}(\mathbf{X}_{s,u}(\mathbf{x}_s))$ is evaluated without gradients, and the ratio of samples used for L_{flow} and L_{distill} is treated as a hyperparameter. In our experiments, we use a 2:1 split, allocating two-thirds of the data to L_{flow} and one-third to L_{distill}. To finetune a pretrained video diffusion model using a flow map while preserving the pretrained knowledge, we propose to warp the diffusion prediction to a flow map (similar to [33]). In particular, we convert noisy data in the flow space $\mathbf{x}_t^{\text{flow}} = t\mathbf{x_1} + (1-t)\mathbf{x}_0$ to the EDM space [51]: $\mathbf{x}_t^{\text{edm}} = \frac{\mathbf{x}_t^{\text{flow}}}{t} = \mathbf{x_1} + \frac{1-t}{t}\mathbf{x}_0$. With the diffusion noise level $\sigma = \frac{1-t}{t}$, we utilize the pretrained diffusion model to predict both $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}_0}$, and compute the predicted vector field for finetuning: $\hat{v}_{t,t} = \hat{\mathbf{x}}_1(\mathbf{x_t}) - \hat{\mathbf{x}_0}(\mathbf{x_t})$.

C. Semantic Spatio-Temporal Reward Modeling

Reward models are crucial in planning to identify and steer promising candidate actions towards successful task completion. However, effective reward model design is challenging, due to the complexity of real-world tasks and the sparsity of task progress signals [55], [59]. While existing methods typically utilize proxies for spatio-temporal progress (e.g., time-based reward signals [55]) or semantic progress (e.g., using object detectors or VLMs [54], [56]). We emphasize that both reward measures are essential to generalizable planning.

Consequently, we introduce a hierarchical reward modeling paradigm that evaluates candidate actions semantically and spatio-temporally for dense reward labeling. Our hierarchical reward model consists of: (i) a higher-level semantic reward

model and (ii) a lower-level spatio-temporal reward model. The semantic model $\mathcal{R}_u$ uses the vision–language foundation model CLIP/SigLIP [65], [66] to score per-frame alignment between predicted observations and the task instruction, leveraging its broad pretrained knowledge. For spatio-temporal reward modeling, we incorporate a reward head $\mathcal{R}_l$ to the video model $\mathcal{V}_\theta$ to simultaneously predict per-frame rewards during video generation. By conditioning the predicted reward on both spatial and temporal dependencies, $\mathcal{R}_l$ captures correlation between geometric progress towards relevant objects and overall task completion.

Given the propensity for video models to hallucinate, we train the reward head using proper scoring rules [67] to compute calibrated probability estimates over the value function, given by: $V(o) = \mathbb{E}_{\xi\sim u}\left[\sum_{\tau=t}^{F} \gamma^{\tau-F} R(o_\tau, a_\tau) \mid o_\tau = o\right]$, where $R(\cdot)$ represents the ground-truth rewards, with the terminal reward given by the binary success flag at the end of the task (at time F). As is standard in prior work, we use the success annotations in open-source robot datasets. We bin the resulting value function and formulate the low-level reward modeling problem as a multi-class classification problem. Concretely, we train the reward model $\mathcal{R}_l$ to predict a probability distribution over N bins using the cross-entropy loss function, given by: $L_r(o) = -\sum_{i=1}^{N} p_{b_i} \log(q_{b_i})$, where p_{b_i} equals zero except when the ground-truth value function lies in bin i, and q_{b_i} represents the predicted probability by $\mathcal{R}_l$ that the value function lies in bin i. Note that the resulting probability distribution serves as a calibrated uncertainty measure for the reward model, i.e., the probability associated with each bin represents the model's confidence in the corresponding reward prediction. We parameterize $\mathcal{R}_l$ using a U-Net architecture for training efficiency. Next, we discuss the integration of uncertainty for robust planning.

D. Action Proposal and Gradient-Based Optimization

We adopt a two-step approach using a global planner that broadly searches for good initializations, and a local planner that further optimizes these actions. We explore two action proposal methods for planning, namely: (i) a biased proposal scheme centered around nominal actions generated by behavior-cloned policies, and (ii) a general-purpose proposal scheme that explores all spatial directions. With the first approach, we seek to leverage the pretrained knowledge of imitation-learned policies to provide a good initialization for the search problem. In our work, we perturb nominal actions from $\pi_{0.5}$ [4] to generate more candidate actions; however, in preliminary experiments, $\pi_{0.5}$ often struggled with language instruction following, in line with findings in prior work [68], limiting the diversity of generated actions. In contrast to the behavior-cloning-based scheme, the second action proposal scheme generates candidate actions that span the joint velocity space of the robot by constructing feasible joint velocities from the canonical set of basis spatial velocities using the robot's Jacobian. We construct a set of six actions in the robot's base frame. Note that both action proposal schemes generate actions in velocity space; however, the video model takes in cartesian end-effector poses as actions. To address this discrepancy, we map joint velocity actions to joint positions using a dynamics action adapter (e.g., [10]), which is subsequently mapped to cartesian poses via inverse kinematics.

Combining the semantic and spatio-temporal reward predictions, we frame the global planning problem as multi-objective optimization, jointly optimizing semantic relevancy r_s while centering the relevant region: $r \coloneqq w(\cdot)(r_s - \lambda d(c, c_r))$, where $w(\cdot)$ represents the weighting function based on the video model's uncertainty, d represents the distance function between the desired target center c and the centroid of the region of highest reward c_r, and λ is the weight on the second objective. To define a net semantic reward for a predicted trajectory of horizon H, we compute the maximum reward across all frames in the trajectory and select the action proposal with the highest reward. Thereafter, we refine the action trajectory associated with the highest semantic reward via gradient-based optimization. At each iteration, we compute the expected reward using $\mathcal{R}_l$ and optimize the entropy-weighted expected reward by back-propagating gradients through $\mathcal{R}_l$ to mitigate the effects of hallucinations.

V. Experiments

We evaluate the effectiveness of VidPlan via the following questions: (**Q1**) Does VidPlan provide stronger generalization and better instruction following compared to SOTA behavior cloning? (**Q2**) Can VidPlan mitigate world model hacking during inference-time planning? (**Q3**) Does VidPlan's produce calibrated uncertainty-aware video world models? (**Q4**) Does VidPlan's flow map enable real-time high-quality video generation? We discuss our findings here and provide additional experiments and ablations in Appendix I.

Implementation Details. We construct our flow map model from Stable Video Diffusion [69] adapted for action-conditioned future prediction [10], pretrained on the DROID dataset [70]. We run real-world experiments on the Franka Emika robot in the DROID configuration [70], given the vast amount of available training data. We provide additional implementation details in Appendix I-F.

Baselines. We compare against the SOTA policy $\pi_{0.5}$ [4] as the main baseline. Beyond being a frontier generalist policy, $\pi_{0.5}$ provides a pretrained DROID checkpoint that is suitable for zero-shot deployment on the real-world DROID setup. Other alternatives such as MolmoAct [71] and GR00T-N1.5 [72] would require finetuning with significant data collection and compute costs, which is beyond the scope of this work. Hence, we do not compare against these policies, including UVA [73] and Diffusion Policy [74]. Additionally, we could not compare against the concurrent work Cosmos Policy [15], since it did not provide an open-source implementation at the time of preparation of our work.

A. Does VidPlan provide stronger generalization and better instruction following compared to SOTA behavior cloning?

In this section, we evaluate VidPlan against $\pi_{0.5}$ in five real-world manipulation tasks on a DROID robot setup, visualized in Figure 3. Specifically, we ask the robot to manipulate

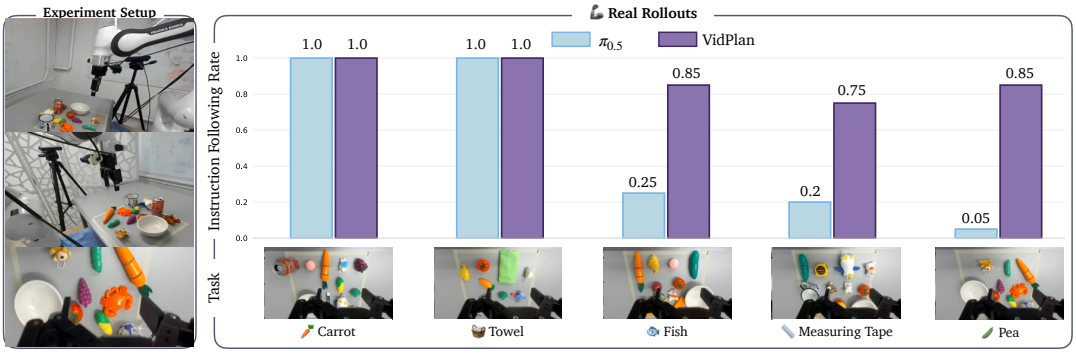

Fig. 3: **Real-world comparisons on a DROID robot setup.** On a real-world DROID setup (with left, right, and wrist camera views), VidPlan and the SOTA policy $\pi_{0.5}$ achieve perfect instruction following rates in in-distribution tasks (e.g., *carrot* and *towel* tasks). However, $\pi_{0.5}$ suffers a significant performance drop in OOD tasks, unlike VidPlan, which achieves 55% to 80% higher success rates.

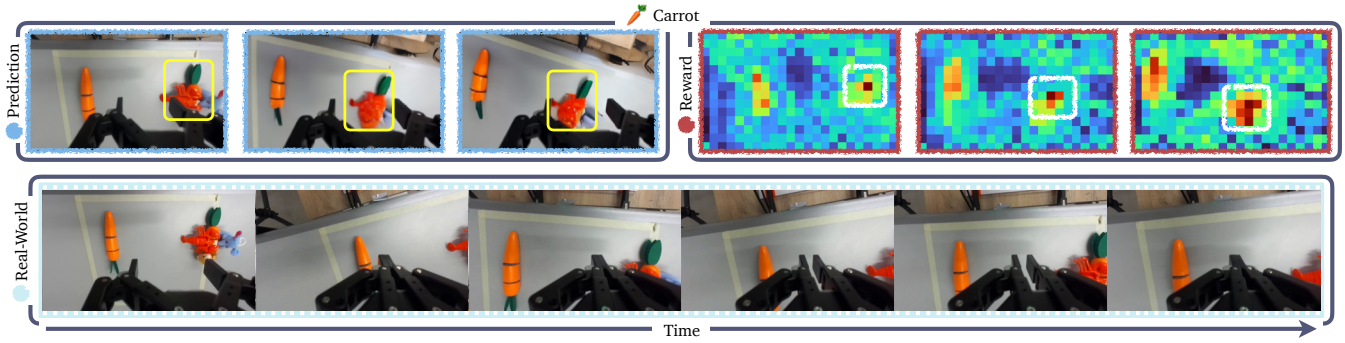

Fig. 4: **Video model-based planning is prone to world model hacking.** (Top) When asked to grasp the carrot, the video model deforms the object in the yellow box into a carrot, leading to high predicted rewards which are exploited by the planner. (Bottom) The optimized actions produce oscillatory motion between the true carrot and the hallucinated object when executed in the real-world, resulting in failure.

common objects, such as picking up a measuring tape, pea, fish, and carrot, or folding a towel — objects that one would expect a behavior-cloned policy to manipulate without difficulty. We provide the complete list of tasks in Table I, and measure the performance of each method in 20 trials per task, reporting the instruction following rate in Figure 3.

TABLE I: Real-world policy performance comparison (in $\%$).

Task	$\pi_{0.5}$	VidPlan (**ours**)
Put the *measuring tape* in the bowl	20	**75**
Put the *fish* in the bowl	25	**85**
Put the *pea* in the bowl	5	**85**
Put the *carrot* in the bowl (ID)	**100**	**100**
Fold the *towel* (ID)	**100**	**100**

First, we observe that both policies achieve a perfect success rate in the *carrot* and *towel* tasks. This finding is not surprising, given that these tasks are within the training distribution, e.g., Bridge [75] and OX-E [76]. More surprisingly, we see a drastic drop in performance of $\pi_{0.5}$ in the remaining tasks. Although intuition suggests that the target objects in our experiments should be *easy* to manipulate, our experiments show the contrary. We posit that this performance degradation is due to the inability of behavior-cloned policies to generalize beyond their training distribution. For example, many robot datasets do not have a lot of expert demonstrations with measuring tapes, fish, etc. Moreover, many SOTA policies are trained with simple uncluttered scenes, which makes the scenes in our experiments out of distribution for these policies. Consequently, even though the pea is not remarkably different from a carrot visually and structurally, $\pi_{0.5}$ struggles to manipulate the pea.

In contrast to imitation-learned policies, VidPlan evaluates the effects of multiple candidate actions and reasons counterfactually about the alignment between these actions and the specified task using the reward model, empowering it to ultimately compute a more optimal sequence of actions for the task. This planning paradigm enables strong out-of-distribution generalization, explaining the superior performance of VidPlan in Figure 3. Concretely, through inference-time planning, VidPlan outperforms $\pi_{0.5}$ by about 55% to 80% in our experiments.

B. Can VidPlan mitigate inference-time world model hacking?

Higher rewards in planning-based methods may not translate to real-world improvements due to the tendency for optimizers to exploit artifacts in the dynamics and rewards

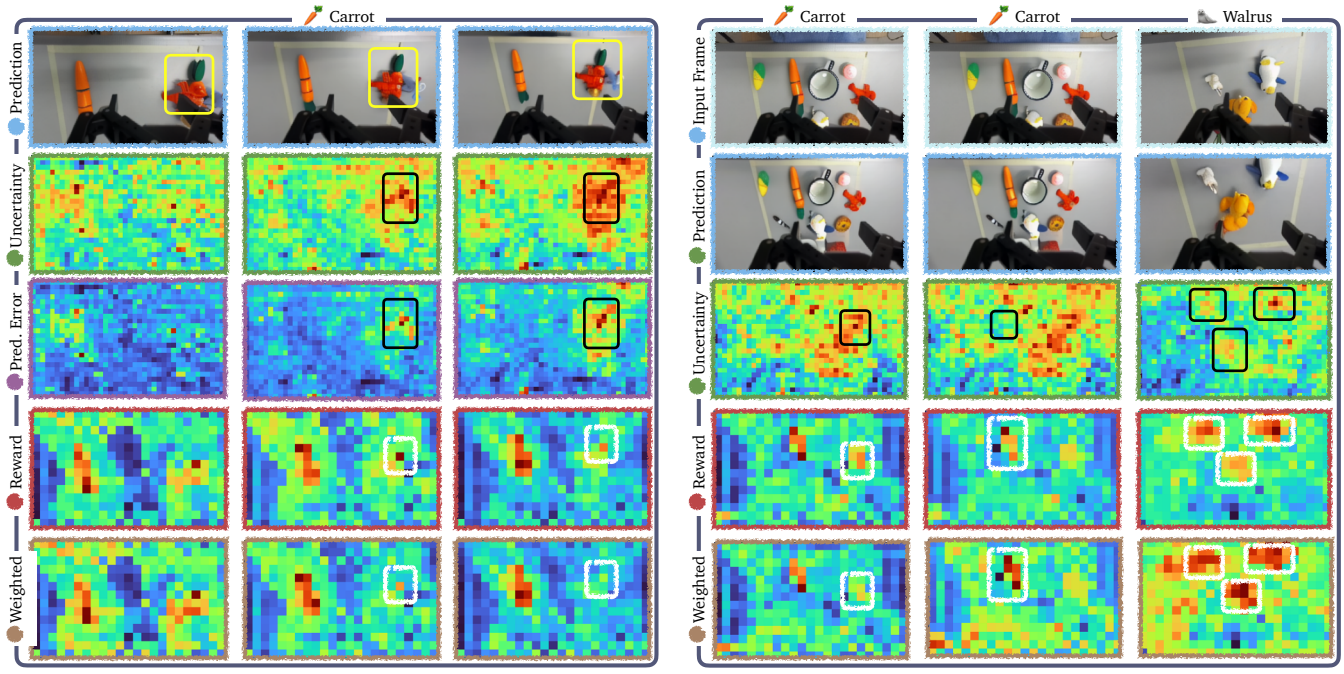

Fig. 5: **Mitigating World Model Hacking with UQ.** (Left) Via UQ, VidPlan mitigates world model hacking by modulating hallucinated rewards (for the object in the yellow box) to achieve real-world success. (Right) From Column 1 to 3, VidPlan suppresses hallucinated rewards, reinforces true rewards, and promotes exploration when necessary using its uncertainty signal.

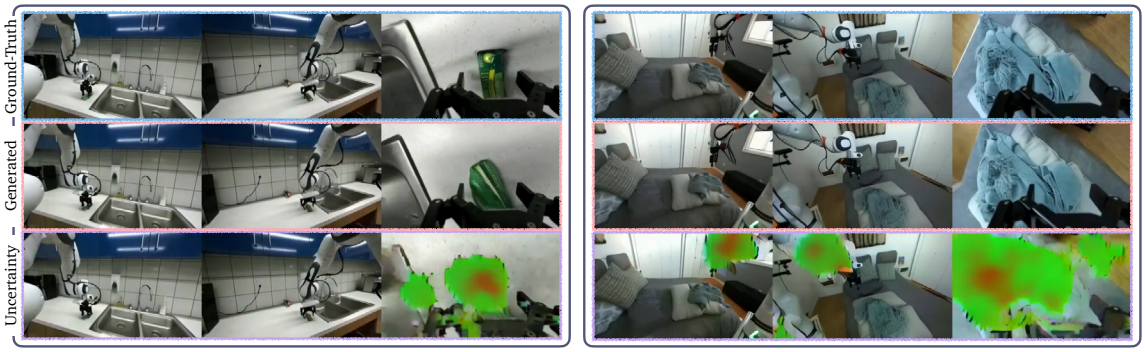

Fig. 6: **Interpretable UQ with VidPlan.** (Left) The video model is uncertain about object deformations that are inconsistent with the laws of physics. (Right) Despite the highly accurate generated video, the video model is still uncertain due to the highly stochastic dynamics of deformable objects, e.g., the cloth.

prediction in world models. Here, we consider a real-world task where the robot is asked to pick up a carrot, as shown in Figure 4. The video model hallucinates the unfamiliar object in the yellow box into a carrot, leading to higher predicted rewards for the hallucinated object. Consequently, the planner selects actions that move the robot to this object to exploit these artificially higher rewards. However, these actions do not result in real-world success. In particular, real-world execution produces oscillatory behaviors as the robot moves back and forth between the true carrot and the hallucinated object.

To address this challenge, VidPlan integrates UQ into inference-time planning, which enables the planner to generate risk-aware trajectories that mitigate the effects of world model hacking. In Figure 5, we demonstrate the effectiveness of uncertainty-aware planning for the same task in Figure 4. First, in Figure 5 (left), we highlight that VidPlan identifies that the video model is uncertain about its predictions of the object in the yellow (in Row 2). In addition, the video model estimates higher prediction errors for these regions (in Row 3). The planner utilizes the estimated uncertainty to downweight the corresponding hallucinated rewards to focus on the true carrot (in Row 4 to 5), ultimately resulting in real-world success. Similarly, in Figure 5 (right), we show that uncertainty-guided planning produces desirable robust behaviors. In Column 1, VidPlan suppresses the reward associated with the lobster to focus on picking up the carrot, since the video model is uncertain about the lobster. Likewise, in Column 2, VidPlan boosts the reward associated with the carrot, since the video model is confident about the

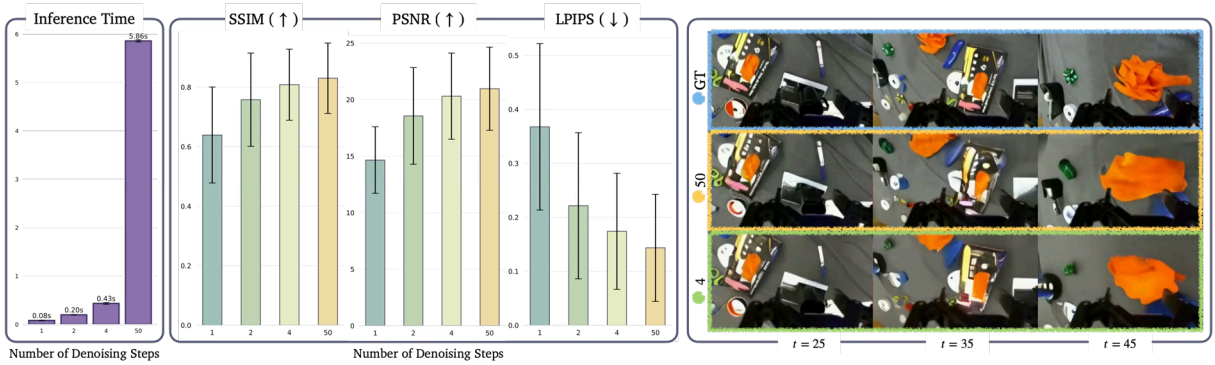

Fig. 7: **Flow Map Video Model.** (Left) Our flow map provides faster inference, e.g., $70\times$, $29\times$, and $13\times$ speedups with the $1, 2$ and 4-step models. (Middle) Perceptual quality does not degrade significantly; the 4-step and the original diffusion models attain almost the same perceptual scores. (Right) Across different video lengths, the 4-step and the original diffusion models achieve similar video generation quality.

carrot. In Column 3, given that the video model is uncertain about all objects, VidPlan modulates all reward predictions appropriately, resulting in more exploratory behaviors.

C. Does VidPlan produce calibrated uncertainty-aware video world models?

Here, we examine the calibration of the video model's uncertainty estimates produced by VidPlan across 100 tasks in the DROID dataset. We obtain near perfect calibration, with low expected calibration error (ECE) at 0.06. Additionally, in Appendix I-C, we show that entropy of the predicted error distribution is strongly positively correlated with the LPIPS and ground-truth latent error.

Further, we examine the interpretability of the latent uncertainty estimates computed by VidPlan. In Figure 6, we show two sample tasks from three camera views: *right*, *left*, and *wrist*, where green-to-red regions represent areas of increasing uncertainty. In Figure 6 (left), we see that the video model is uncertain about the robot-object interaction (third column) as the object deforms in ways that are inconsistent with the laws of physics. Similarly, in the second task, the video model is uncertain about the dynamics of the deformable object in the wrist-camera view, as one would expect. These two examples highlight the fine-grained expressiveness of VidPlan's uncertainty estimates. First, we observe that the uncertainty in the first task is associated with hallucinations, which can be reduced by training with additional data; making it epistemic in nature. In contrast, in the second task, the video model is still uncertain despite its highly accurate generated video. This uncertainty is associated with the highly stochastic dynamics of deformable bodies, which is generally irreducible, rendering it aleatoric.

D. Does VidPlan's flow map enable real-time video generation?

We explore the inference speedups afforded by VidPlan's flow map, which is summarized in Figure 7. We observe significantly faster inference with the flow map video model. Concretely, inference speed scales linearly with the number of denoising steps, resulting in $70\times$, $29\times$, and $13\times$ speedups with the $1, 2$ and 4-step models, respectively. At these inference speeds (2Hz to 12Hz), the flow map video model is fast enough for real-time planning. Next, we examine the tradeoffs between inference speeds and video generation quality. From Figure 7, we observe negligible differences in video quality between the 4-step flow map model and the original diffusion model, and as expected, video generation quality decreases more significantly for the 1 and 2-step models. Further, we explore video generation quality with respect to the video length, shown in Figure 7. We find that the 4-step and the original diffusion models achieve qualitatively the same video quality across a broad range of video durations. These results highlight the effectiveness of our flow map in real-time, high-fidelity video generation.

VI. Conclusion and Future Work

We introduce VidPlan, an inference-time uncertain-aware planning framework that uses video models as high-fidelity world models for context-specific counterfactual reasoning. VidPlan consists of three core innovations: (i) an uncertainty-aware video model that jointly generates future predictions and associated confidence, (ii) a flow map for diffusion-to-flow-matching distillation for real-time inference, and (iii) an hierarchical reward model that evaluates candidate actions based on semantic and spatio-temporal signals. Together, these components enable robust gradient-based optimization that mitigates world model hacking to achieve strong zero-shot generalization compared to SOTA imitation-learned policies.

Although broadly effective, our hierarchical reward model does not leverage synergies between semantic and spatio-temporal. Training more generalizable reward models for joint semantic and spatio-temporal reward modeling to harness shared dependencies would be an interesting area for future work. Further, better informed action proposals and faster video generation with lower compute overhead could significantly speed up the planning process, constituting an exciting direction for future research. Looking forward, we believe that inference-time planning could unlock the remarkable potential of generalist robot policies.

REFERENCES

[1] R. Morris and G. Ward, *The cognitive psychology of planning*. Psychology Press, 2004.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[4] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[5] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[6] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[7] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 4950–4957.

[8] D. Hafner, W. Yan, and T. Lillicrap, "Training agents inside of scalable world models," *arXiv preprint arXiv:2509.24527*, 2025.

[9] J. Quevedo, A. K. Sharma, Y. Sun, V. Suryavanshi, P. Liang, and S. Yang, "Worldgym: World model as an environment for policy evaluation," 2025.

[10] Y. Guo, L. X. Shi, J. Chen, and C. Finn, "Ctrl-world: A controllable generative world model for robot manipulation," *arXiv preprint arXiv:2510.10125*, 2025.

[11] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[12] Y. Du, M. Yang, P. Florence, F. Xia, A. Wahid, B. Ichter, P. Sermanet, T. Yu, P. Abbeel, J. B. Tenenbaum, L. Kaelbling, A. Zeng, and J. Tompson, "Video language planning," 2023.

[13] B. Chen, T. Zhang, H. Geng, K. Song, C. Zhang, P. Li, W. T. Freeman, J. Malik, P. Abbeel, R. Tedrake, *et al.*, "Large video planner enables generalizable robot control," *arXiv preprint arXiv:2512.15840*, 2025.

[14] J. Pai, L. Achenbach, V. Montesinos, B. Forrai, O. Mees, and E. Nava, "mimic-video: Video-action models for generalizable robot control beyond vlas," *arXiv preprint arXiv:2512.15692*, 2025.

[15] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, *et al.*, "Cosmos policy: Fine-tuning video models for visuomotor control and planning," *arXiv preprint arXiv:2601.16163*, 2026.

[16] N. M. Boffi, M. S. Albergo, and E. Vanden-Eijnden, "How to build a consistency model: Learning flow maps via self-distillation, 2025."

[17] DeepMind, "Veo-3: A text-to-video generation system with audio," DeepMind / Google, Tech. Rep. Tech Report, 2025.

[18] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, *et al.*, "Wan: Open and advanced large-scale video generative models," *arXiv preprint arXiv:2503.20314*, 2025.

[19] N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding, *et al.*, "Cosmos world foundation model platform for physical ai," *arXiv preprint arXiv:2501.03575*, 2025.

[20] Z. Mei, T. Yin, O. Shorinwa, A. Badithela, Z. Zheng, J. Bruno, M. Bland, L. Zha, A. Hancock, J. F. Fisac, *et al.*, "Video generation models in robotics-applications, research challenges, future directions," *arXiv preprint arXiv:2601.07823*, 2026.

[21] J. Quevedo, P. Liang, and S. Yang, "Evaluating robot policies in a world model," *arXiv preprint arXiv:2506.00613*, 2025.

[22] F. Zhu, H. Wu, S. Guo, Y. Liu, C. Cheang, and T. Kong, "Irasim: Learning interactive real-robot action simulators," *arXiv preprint arXiv:2406.14540*, 2024.

[23] G. R. Team, K. Choromanski, C. Devin, Y. Du, D. Dwibedi, R. Gao, A. Jindal, T. Kipf, S. Kirmani, I. Leal, F. Liu, A. Majumdar, A. Marmon, C. Parada, Y. Rubanova, D. Shah, V. Sindhwani, J. Tan, F. Xia, T. Xiao, S. Yang, W. Yu, and A. Zhou, "Evaluating Gemini robotics policies in a Veo world simulator," 2025.

[24] J. Zhang, K. Zheng, K. Jiang, H. Wang, I. Stoica, J. E. Gonzalez, J. Chen, and J. Zhu, "Turbodiffusion: Accelerating video diffusion models by 100-200 times," *arXiv preprint arXiv:2512.16093*, 2025.

[25] Y. HaCohen, N. Chiprut, B. Brazowski, D. Shalem, D. Moshe, E. Richardson, E. Levin, G. Shiran, N. Zabari, O. Gordon, P. Panet, S. Weissbuch, V. Kulikov, Y. Bitterman, Z. Melumian, and O. Bibi, "Ltx-video: Realtime video latent diffusion," *arXiv preprint arXiv:2501.00103*, 2024.

[26] P. Zhang, H. Huang, Y. Chen, W. Lin, Z. Liu, I. Stoica, E. P. Xing, and H. Zhang, "Faster video diffusion with trainable sparse attention," *arXiv e-prints*, pp. arXiv–2505, 2025.

[27] K. Frans, D. Hafner, S. Levine, and P. Abbeel, "One step diffusion via shortcut models," *arXiv preprint arXiv:2410.12557*, 2024.

[28] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," 2023.

[29] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[30] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[31] C. Lu and Y. Song, "Simplifying, stabilizing and scaling continuous-time consistency models," *arXiv preprint arXiv:2410.11081*, 2024.

[32] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park, "One-step diffusion with distribution matching distillation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 6613–6623.

[33] J. Schusterbauer, M. Gui, F. Fundel, and B. Ommer, "Diff2flow: Training flow matching models via diffusion model alignment," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 28 347–28 357.

[34] J.-Y. Yao, K.-P. Ning, Z.-H. Liu, M.-N. Ning, Y.-Y. Liu, and L. Yuan, "Llm lies: Hallucinations are not bugs, but features as adversarial examples," *arXiv preprint arXiv:2310.01469*, 2023.

[35] C. Chen, K. Liu, Z. Chen, Y. Gu, Y. Wu, M. Tao, Z. Fu, and J. Ye, "Inside: Llms' internal states retain the power of hallucination detection," *arXiv preprint arXiv:2402.03744*, 2024.

[36] Y. Lim and H. Shim, "Addressing image hallucination in text-to-image generation through factual image retrieval," *arXiv preprint arXiv:2407.10683*, 2024.

[37] S. K. Aithal, P. Maini, Z. Lipton, and J. Z. Kolter, "Understanding hallucinations in diffusion models through mode interpolation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 134 614–134 644, 2024.

[38] V. Rawte, S. Jain, A. Sinha, G. Kaushik, A. Bansal, *et al.*, "Vibe: A text-to-video benchmark for evaluating hallucination in large multimodal models," in *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, 2025, pp. 232–246.

[39] Z. Chu, L. Zhang, Y. Sun, S. Xue, Z. Wang, Z. Qin, and K. Ren, "Sora detector: A unified hallucination detection for large text-to-video models," *arxiv:2405.04180*, 2024.

[40] V. Rawte, A. Sheth, and A. Das, "A survey of hallucination in large foundation models," *arXiv preprint arXiv:2309.05922*, 2023.

[41] L. Kuhn, Y. Gal, and S. Farquhar, "Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation," *arXiv preprint arXiv:2302.09664*, 2023.

[42] O. Shorinwa, Z. Mei, J. Lidard, A. Z. Ren, and A. Majumdar, "A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions," *ACM Computing Surveys*, 2025.

[43] I. Kononenko, "Bayesian neural networks," *Biological Cybernetics*, vol. 61, no. 5, pp. 361–370, 1989.

[44] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 2008–2026, 2018.

[45] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[46] M. Chan, M. Molina, and C. Metzler, "Estimating epistemic and aleatoric uncertainty with a single model," *Advances in Neural Information Processing Systems*, vol. 37, pp. 109 845–109 870, 2024.

[47] L. Berry, A. Brando, and D. Meger, "Shedding light on large generative networks: Estimating epistemic uncertainty in diffusion models," in *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[48] Z. Mei, O. Shorinwa, and A. Majumdar, "How confident are video models? empowering video models to express their uncertainty," *arXiv preprint arXiv:2510.02571*, 2025.

[49] Z. Mei, T. Yin, M. Baker, O. Shorinwa, and A. Majumdar, "World

models that know when they don't know: Controllable video generation with calibrated uncertainty," *arXiv preprint arXiv:2512.05927*, 2025.

[50] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[51] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," 2022.

[52] G. Zhou, H. Pan, Y. LeCun, and L. Pinto, "Dino-wm: World models on pre-trained visual features enable zero-shot planning," *arXiv preprint arXiv:2411.04983*, 2024.

[53] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus, *et al.*, "V-jepa 2: Self-supervised video models enable understanding, prediction and planning," *arXiv preprint arXiv:2506.09985*, 2025.

[54] T. Yin, Z. Mei, T. Sun, L. Zha, E. Zhou, J. Bao, M. Yamane, O. Sho, and A. Majumdar, "Womap: World models for embodied open-vocabulary object localization," in *9th Annual Conference on Robot Learning*, 2025.

[55] J. Zhang, Y. Luo, A. Anwar, S. A. Sontakke, J. J. Lim, J. Thomason, E. Biyik, and J. Zhang, "Rewind: Language-guided rewards teach robot policies without new demonstrations," *arXiv preprint arXiv:2505.10911*, 2025.

[56] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu, S. Lin, and J. Pang, "A vision-language-action-critic model for robotic real-world reinforcement learning," *arXiv preprint arXiv:2509.15937*, 2025.

[57] T. Lee, A. Wagenmaker, K. Pertsch, P. Liang, S. Levine, and C. Finn, "Roboreward: General-purpose vision-language reward models for robotics," *arXiv preprint arXiv:2601.00675*, 2026.

[58] J. Xiao, Y. Yang, X. Chang, R. Chen, F. Xiong, M. Xu, W.-S. Zheng, and Q. Zhang, "World-env: Leveraging world model as a virtual environment for vla post-training," *arXiv preprint arXiv:2509.24948*, 2025.

[59] Y. J. Ma, J. Hejna, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, *et al.*, "Vision language models are in-context value learners," in *The Thirteenth International Conference on Learning Representations*, 2024.

[60] A. Escontrela, A. Adeniji, W. Yan, A. Jain, X. B. Peng, K. Goldberg, Y. Lee, D. Hafner, and P. Abbeel, "Video prediction models as rewards for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 760–68 783, 2023.

[61] T. Huang, G. Jiang, Y. Ze, and H. Xu, "Diffusion reward: Learning rewards via conditional video diffusion," in *European Conference on Computer Vision*. Springer, 2024, pp. 478–495.

[62] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022.

[63] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2021.

[64] T. Yin, Q. Zhang, R. Zhang, W. T. Freeman, F. Durand, E. Shechtman, and X. Huang, "From slow bidirectional to fast autoregressive video diffusion models," 2025.

[65] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[66] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 11 975–11 986.

[67] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[68] C. Glossop, W. Chen, A. Bhorkar, D. Shah, and S. Levine, "Cast: Counterfactual labels improve instruction following in vision-language-action models," *arXiv preprint arXiv:2508.13446*, 2025.

[69] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, *et al.*, "Stable video diffusion: Scaling latent video diffusion models to large datasets," *arXiv preprint arXiv:2311.15127*, 2023.

[70] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," 2025.

[71] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, *et al.*, "Molmoact: Action reasoning models that can reason in space," *arXiv preprint arXiv:2508.07917*, 2025.

[72] J. Bjorck, F. Castañeda, *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[73] S. Li, Y. Gao, D. Sadigh, and S. Song, "Unified video action model," *arXiv preprint arXiv:2503.00200*, 2025.

[74] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[75] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, "Bridge data: Boosting generalization of robotic skills with cross-domain datasets," *arXiv preprint arXiv:2109.13396*, 2021.

[76] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[77] J. A. Vincent, H. Nishimura, M. Itkina, P. Shah, M. Schwager, and T. Kollar, "How generalizable is my behavior cloning policy? a statistical approach to trustworthy performance evaluation," *IEEE Robotics and Automation Letters*, 2024.

[78] J. Wang, M. Leonard, K. Daniilidis, D. Jayaraman, and E. Hu, "Evaluating pi0 in the wild: Strengths, problems, and the future of generalist robot policies," 2025.

[79] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[80] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

APPENDIX I
ADDITIONAL EXPERIMENTS

A. Additional Discussion of Results in Section V-A

We discuss our baseline comparison experiments. For each task, we create cluttered scenes by including random objects around the specified target object for manipulation. In preliminary experiments, we found $\pi_{0.5}$ to be notably sensitive to the format of the task description. For example, $\pi_{0.5}$ often failed when prompted with "pick up <object>"; however, when prompted with "put the carrot in the bowl" $\pi_{0.5}$ succeeded virtually all the time. Hence, we use the task description format "Put the <object> in the bowl" or "Fold the <object>" in our experiments.

In Figure 8, we provide an illustration of the robot trajectory with $\pi_{0.5}$ and VidPlan in the *measuring tape* task. With $\pi_{0.5}$, the robot wanders around the scene, visiting different objects, before picking up the wrong object (the mug). This convoluted path is suboptimal for the task.

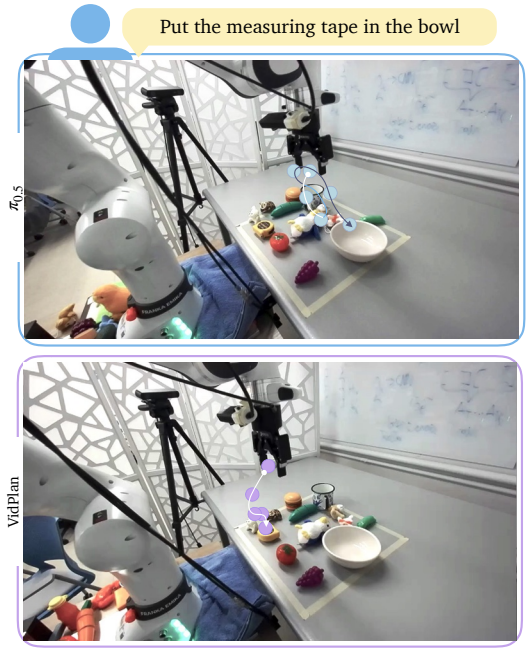

Fig. 8: **Sample action trajectory from $\pi_{0.5}$ and VidPlan in the *measuring tape* task.** (Top) $\pi_{0.5}$ does not follow the instruction and takes a convoluted path that first pushes the penguin before putting the mug in the bowl. (Bottom) Via inference-time planning, VidPlan goes directly towards the measuring tape guided by the reward model.

VidPlan addresses these challenges by first predicting the dynamics of the scene with the video model and subsequently optimizing the candidate actions based on the predicted rewards. Consequently, with VidPlan, the robot takes a more direct path to the target object. Next, we explore the degradation in performance of $\pi_{0.5}$ as the environment moves further away from the training distribution.

B. Behavior-Cloned Policies Struggle to Generalize

Behavior cloning yields policies that perform impressively on tasks and environments within the training distribution. However, we observe large drops in performance when deploying these policies in less familiar settings, as discussed in Section V-A. Here, we examine the generalizability of $\pi_{0.5}$ as we incrementally increase the complexity/clutter in the scene for the task "put the measuring tape in the bowl."

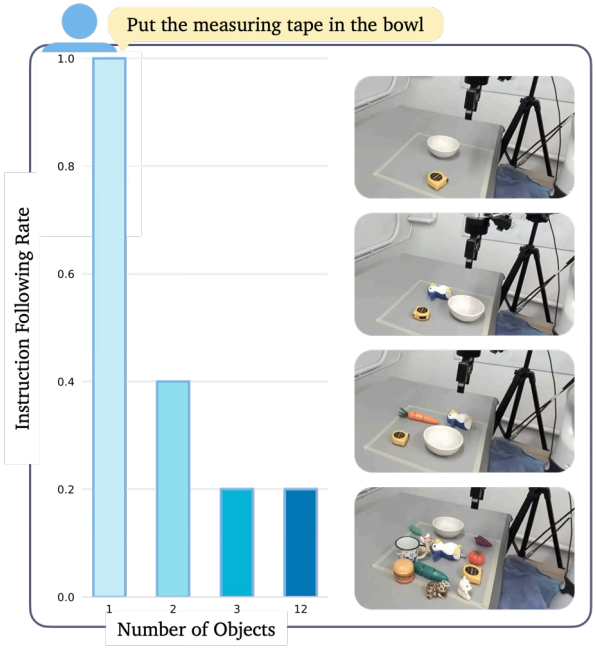

Fig. 9: **Generalization of $\pi_{0.5}$ with increasing distance from training distribution.** The instruction following rate of $\pi_{0.5}$ degrades significantly as the environment moves further away from the training distribution. After adding three objects to the scene, the performance of $\pi_{0.5}$ drops by 80%, when the robot is asked to put a measuring tape in a bowl.

Figure 9 summarizes the results, showing a progressive decline in the instruction following capability of $\pi_{0.5}$ as more objects are added to the scene. When only a single object is present in the scene, $\pi_{0.5}$ picks up the measuring tape in every trial, achieving a perfect success rate. This scenario represents the in-distribution case, i.e., the nominal environment where there are only target objects on the table. However, when another object is added to the scene, the performance of $\pi_{0.5}$ drops by 60%, with a further performance drop of 20% when a third object is added to the scene. These results suggests that: (i) $\pi_{0.5}$ can generalize to some extent, as the success rate remains moderate when there are only two objects in the scene, and that (ii) it fails catastrophically when trying to generalize to more unfamiliar environment conditions, e.g., a cluttered scene. These findings highlight the limited generalizability of behavior-cloned policies, as demonstrated in prior work [77], [78].

In Figure 10, we show a sample trajectory using $\pi_{0.5}$. Notice that the robot visits many objects during the task, e.g., the plushy toy, burger, and penguin, before ultimately picking up

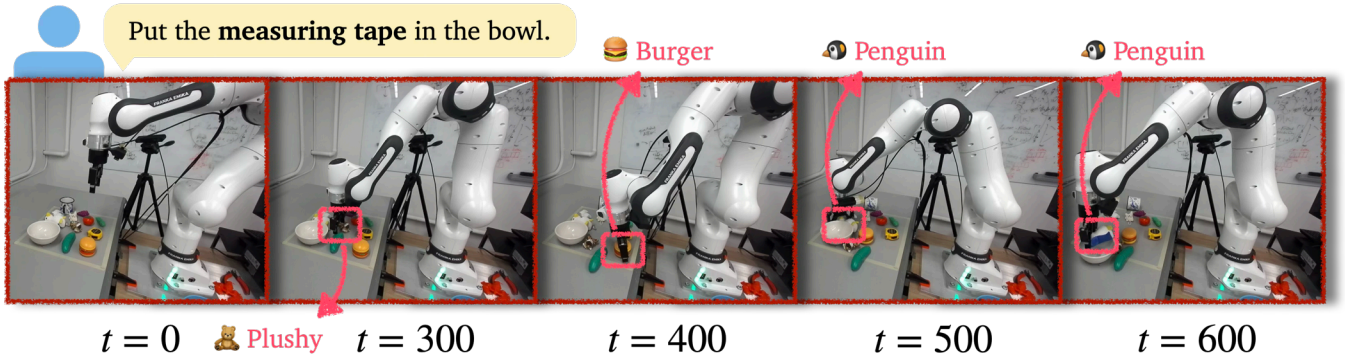

Fig. 10: **Sample action trajectory showing limited generalization of $\pi_{0.5}$ in the *measuring tape* task.** $\pi_{0.5}$ struggles to generalize to out-of-distribution scenes. When asked to put the measuring tape in the bowl in a cluttered scene, $\pi_{0.5}$ hesitates for about 200 timesteps, then attempts to pick up various objects, e.g., the plushy toy, burger, and penguin.

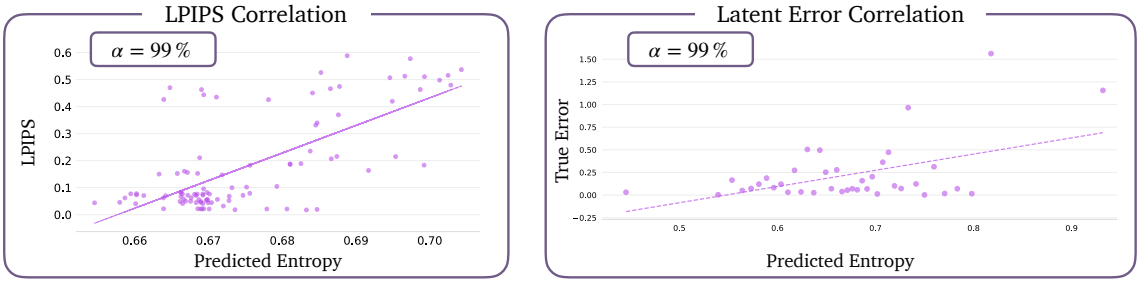

Fig. 11: **Correlation between predicted entropy and LPIPS/absolute latent error in video prediction.** We observe a positive correlation between the video model's uncertainty and the LPIPS/absolute latent error at a statistically significant level of 99%. The video model's uncertainty is well-aligned with the observed error.

the wrong target object (the penguin), showing a fundamental lack of task understanding. As noted earlier, $\pi_{0.5}$ is extremely sensitive to the language instruction. Concretely, semantically-similar language instructions produce significantly different outcomes. For example, the policy often fails if the robot is asked to "pick up <target object>" but succeeds when asked to "put <target object> in a container." These findings suggest that there is limited semantic understanding within $\pi_{0.5}$, i.e., the policy fails to learn semantic abstractions that unify language and visual inputs. Semantic understanding is essential for generalization, presenting a promising direction for future research.

C. Calibration of VidPlan's UQ Estimates

We further discuss the calibration of the video model's uncertainty estimates produced by VidPlan. In Figure 11, we show that the entropy of the predicted error distribution is strongly positively correlated with the LPIPS [79] and the ground-truth latent error, with correlation coefficients of 0.38 and 0.32 at the 99% significance level, respectively. These results show that entropy is well-aligned with video error, with increasing entropy associated with increasing (observed) video error.

D. Interpretability of VidPlan's UQ Estimates

Here, we provide additional results showing the interpretability of the video model's uncertainty. In Figure 12 (left), we observe that the video model hallucinates the bag of blocks on the table and changes the originally cylindrical blocks into planar pieces. VidPlan expresses the model's uncertainty about its hallucinations, as shown by the green and red regions in the uncertainty map. Similarly, in the task on the right, the video model hallucinates the lid of the coffee maker disappearing in the generated wrist-camera video. VidPlan correctly identifies these hallucinations as regions of high uncertainty.

In Figure 13, we show that VidPlan is able to capture uncertainty that arises from dynamic interactions, e.g., in a cable manipulation task (left) and a towel manipulation task (right). We draw the reader's attention to the wrist camera view on the left, where the video model deforms the charger connected to the cable. VidPlan designates the corresponding regions of the video as areas of high uncertainty, which is aligned with the relatively higher error between these regions of the generated video and the ground-truth video. In the towel manipulation task, VidPlan highlights regions associated with the towel as high-uncertainty regions, which is well-aligned with human intuition. Non-rigid bodies exhibit complex dynamics, making future prediction particularly challenging, a phenomenon that is captured by higher uncertainty of the video model. Moreover, the generated video deviates significantly from the ground-truth video in regions corresponding to the towel.

Beyond identifying dynamics uncertainty, VidPlan localizes areas of generated videos with non-physically-consistent spatio-temporal changes. For example, in Figure 14, we show

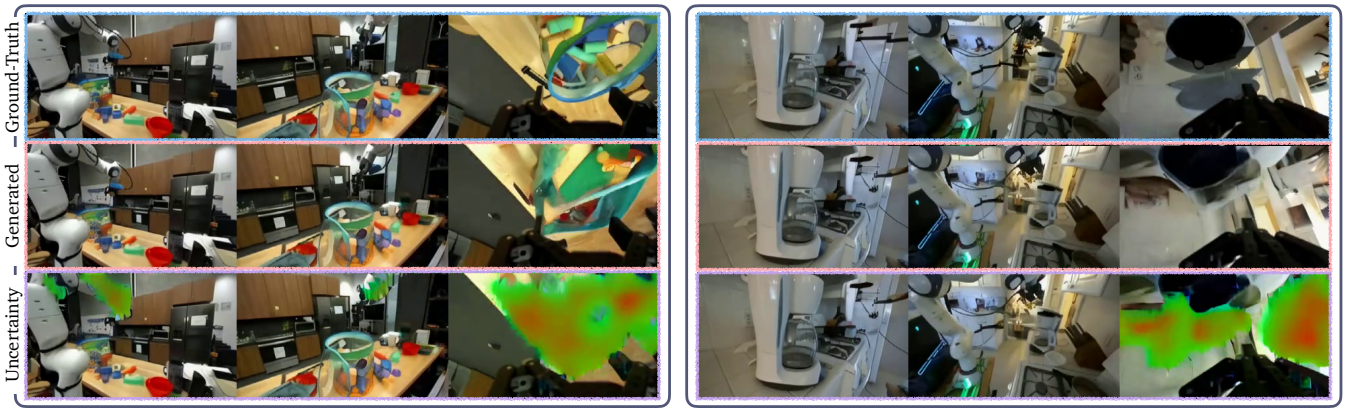

Fig. 12: **VidPlan localizes uncertain regions with objects morphing or disappearing.** (Left) The video model deforms the cylindrical bars and other objects in the bag, which corresponds to areas of high uncertainty. (Right) Similarly, the video model is uncertain about its hallucinations of the lid of the coffee maker disappearing in the wrist-camera video.

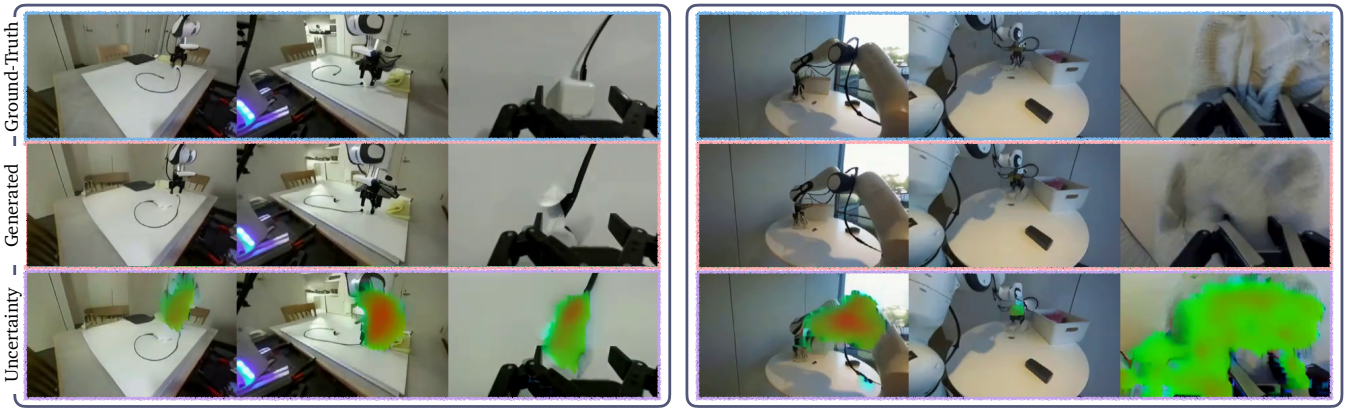

Fig. 13: **VidPlan captures uncertainty from robot-object interactions.** As shown in the composited heatmaps, the video model is uncertain about its predictions of the state of the charger (left) and the towel (right). These predictions are aligned with the error in the video prediction as well as human intuition.

Fig. 14: **VidPlan identifies non-physically-consistent future predictions.** In the generated videos, the video model morphs the soda can and the drawer chest. VidPlan localizes these hallucinations in the composited uncertainty maps.

a pair of tasks, where the video model deforms a can of soda (left) and a drawer chest (right) in ways that violate the laws of physics. VidPlan correctly associates these regions with higher uncertainty. Likewise, in Figure 15, the video model hallucinates grasping a cereal bag (left). VidPlan identifies the corresponding regions as areas of high uncertainty, which is consistent with the larger ground-truth error. In the second task (right), many of the objects in the drawer are occluded, making future prediction difficult. VidPlan's uncertainty estimates capture this inherent uncertainty.

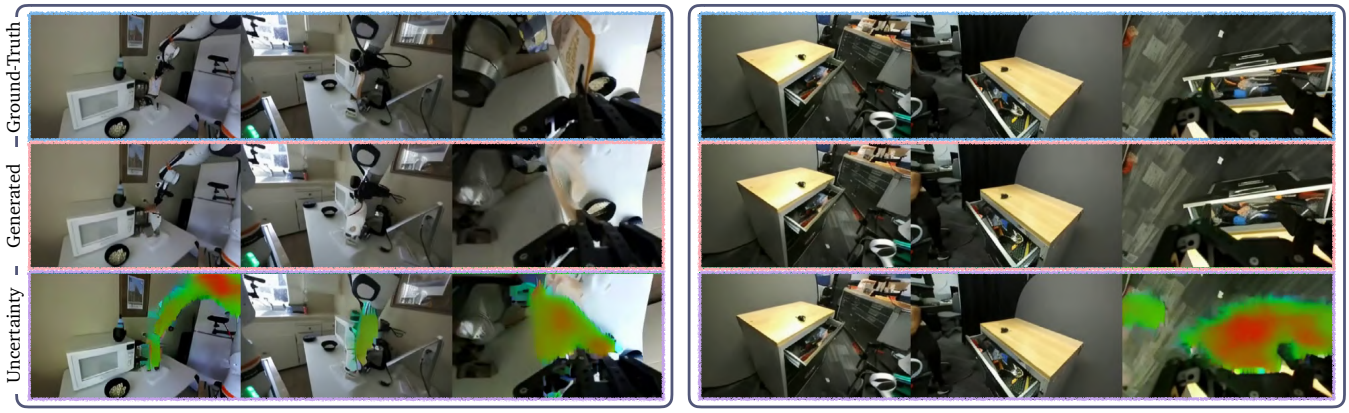

Fig. 15: **Additional visualization of VidPlan's uncertainty estimates.** VidPlan identifies the hallucinated cereal bag (left) and captures the uncertainty associated with the objects in the drawer due to occlusions (right).

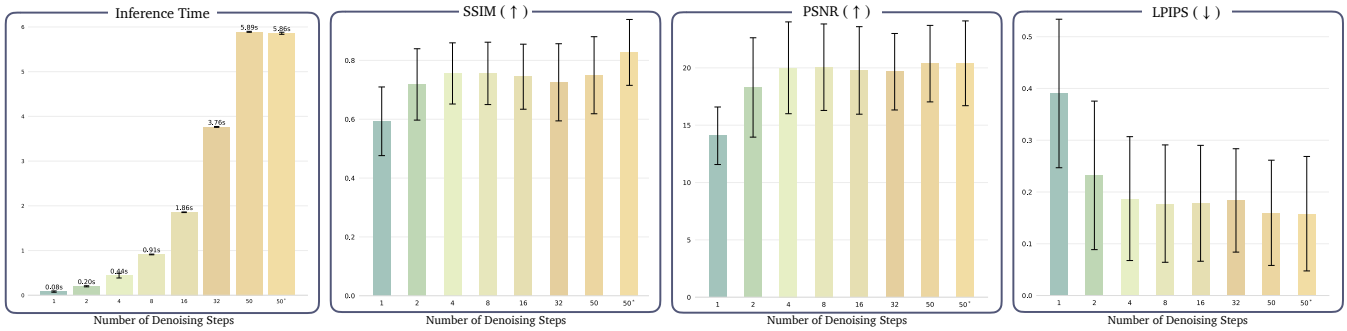

Fig. 16: **Inference speed and visual quality of the flow map models for varying number of denoising step.** Inference speed increases linearly with the number of denoising steps, and importantly, the visual quality remains relatively constant till about 4 steps. The * indicates the pretrained diffusion model.

E. Ablations: Flow Map Results

We provide additional results on our flow map models derived from pretrained diffusion models. Specifically, we evaluate the inference speed and perceptual quality of the resulting flow map model across a range of denoising steps. We report the results in Figure 16. As noted in the earlier discussion, we find that inference speed scales linearly with the number of denoising steps, and importantly, the visual quality of the generated videos remains relatively constant up to about 4 denoising steps, with only marginal differences in the perceptual scores. These results underscore the significant speedup provided by the flow map model without a notable tradeoff in perceptual quality (about $10\times$ speedup for the 4-step model).

In Figures 17 and 18, we provide qualitative results showing the quality of the videos generated from the 1-step, 2-step, and 4-step models. We observe almost no perceptible difference in quality at $t = 5$ across all models. At $t = 10$, the quality of the 1-step model begins to degrade, particularly in the background regions of the video. This degradation worsens in the 1-step model with a noticeable divergence between the ground-truth and the videos generated at $t = 30$ and $t = 60$. However, the 4-step model maintains high-quality generations even at $t = 60$. This result indicates that we can vary the number of inference steps during planning based on our choice of roll-out horizon for optimal efficiency.

F. Implementation Details

We provide additional implementation details for VidPlan. We train all models on the training split of the DROID dataset. We train the flow map model on four H200 GPUs with 140GB VRAM for a maximum of four days. For simulated experiments, we use the validation dataset split of the DROID dataset but generate novel tasks for the real-world experiments. At inference time, VidPlan uses a single H200 GPU for planning, which requires about 2.6 sec for a prediction horizon of 5, optimized over 5 iterations using a 4-step flow map model (assuming parallel video generation for the action proposals). For faster planning, we demonstrate inference speeds of 12.5 Hz with our one-step flow map model, corresponding to a total planning time of 0.48 sec for the same horizon length.

Uncertainty Quantification and Reward Heads. We adopt the U-Net backbone for the video model's UQ and reward heads with two upsampling and downsampling blocks, given its remarkable efficiency in capturing spatial relationships which is crucial in vision-based learning. For spatio-temporal modeling, we introduce a self-attention layer in the U-Net's bottleneck, with a dimension of 512 channels. To learn calibrated probabilities that are consistent with observed

Fig. 17: **Flow map model rollouts at $t = 5$ and $t = 10$ for different numbers of denoising steps.** Across all models, video generation quality does not noticeably degrade at $t = 5$. However, the background quality degrades moderately for the 1-step and 2-step models at $t = 10$, unlike the 4-step model.

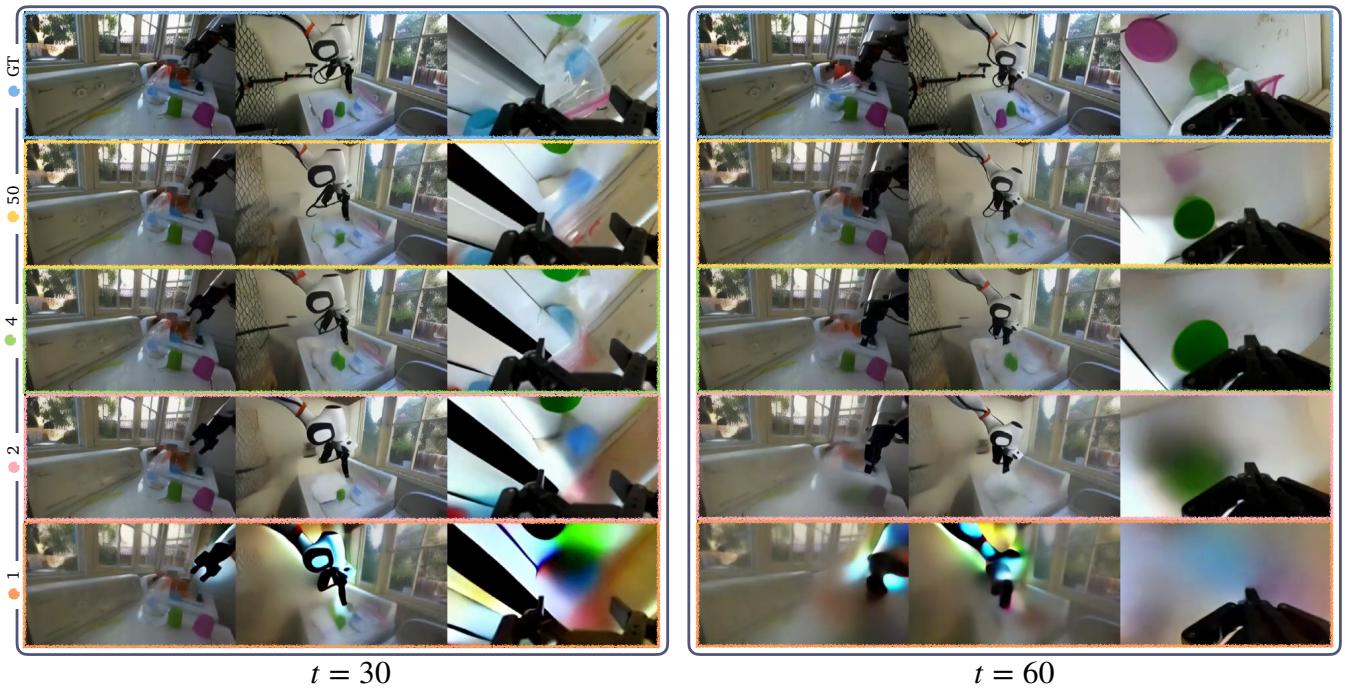

Fig. 18: **Flow map model rollouts at $t = 30$ and $t = 60$ for different numbers of denoising steps.** With fewer denoising steps, e.g., 1-step, we see much more severe video quality degradation at longer rollout horizons. However, the 4-step model generates high-quality videos even at $t = 60$.

frequencies, we train the UQ and reward heads using the cross-entropy loss function. In training the UQ head, we discretize the distribution over the latent errors into 30 bins, noting that the absolute prediction errors reside within the range $[0, 10]$. Likewise, for the UQ head, we discretize the Monte-Carlo estimated returns into 20 bins, noting that the estimated returns lie in the range $[0, 1]$. We train both heads simultaneously for a maximum of two days without

TABLE II: Flow Map Training Hyperparameter

Hyperparameter	Value
LR	5e-6
Batch size flow	16
Batch size distill	8

backpropagating gradients to the flow-matching pipeline, as discussed in Section IV.

For meaningful visualization, we map the latent uncertainty estimates to the RGB color space by compositing three canonical latent-space colormaps. Specifically, we instantiate a latent-space colormap by transforming red, green, and blue solid-color videos to the (SVD) latent space. To define a heatmap for each generated video frame, we blend the basis colors of the colormap using the estimated uncertainty of each latent channel as weights. Recall that VidPlan's uncertainty estimates are bounded between zero and one. We map zero-valued uncertainty channels to blue latents and one-valued uncertainty estimates to red latents, with green latents assigned to values in the middle of the range. To further aid understanding, we overlay the uncertainty heatmaps on the generated RGB video frames to create a composited image that highlights higher-uncertainty regions.

Flow Map Generation. For real-time video generation, we subsequently transform our video diffusion model to a flow map model using the hyperparameters in Table II, as discussed in Section IV-B. To utilize pretrained information in the diffusion model, similar to [33], we transform the input from the flow space to the diffusion space and obtain an estimated vector field from the diffusion model, as shown in Algorithm 1. We summarize the finetuning process in Algorithm 2, with the loss functions given by:

$$L_{\text{flow}}(v) = \mathbb{E}_{\mathbf{x}_1,\mathbf{x}_0,t}\left[\left\|\mathbf{X}_{t,t}(\mathbf{x}_t) - (\mathbf{x}_d)\right\|^2\right],$$
$$L_{\text{distill}}(v) = \mathbb{E}_{\mathbf{x}_1,\mathbf{x}_0,s,u,t}\left[\left\|\mathbf{X}_{s,t}(\mathbf{x}_s) - \mathbf{X}_{u,t}(\mathbf{X}_{s,u}(\mathbf{x}_s))\right\|^2\right]. \tag{4}$$

Note that the algorithm is agnostic to the loss formulation of the original diffusion model, because we can obtain the estimated noise and clean data from any diffusion formulation (see Equation (8) to Equation (12)). In addition, note that the video model is conditioned on action sequences represented by Cartesian-space end-effector poses, as discussed in Section IV. The overall planning pipeline is summarized in Algorithm 3.

Algorithm 1: DIFF2FLOW: Warp Pretrained EDM Diffusion Predictors into a Flow-Map Vector Field

Require : A pretrained diffusion video model $\mathcal{V}_\theta$ (EDM noise-level conditioning); flow start and end times $s, t \in (0, 1]$, conditions $\mathbf{c}$.

Function DIFF2FLOW($\mathcal{V}_\theta$, $\tilde{o}^{s}_{\tau-T:\tau}$, $\mathbf{c}$, s, t)**:**

```
// From flow to EDM space
õ^{s,edm}_{τ−T:τ} ← õ^{s,flow}_{τ−T:τ} / s
// EDM noise level
σ ← (1 − s)/s
// Run the pretrained diffusion
   model at noise level σ
(ô_{τ:τ+H}, ê) ← V_θ(õ^{s,edm}_{τ:τ+H}; c, σ, t)
// Convert predictions to a
   flow-map vector field
v̂_{s,t} ← ô_{τ:τ+H} − ê
return v̂_{s,t}
```

Algorithm 2: Video World Model Finetuning with Flow Map

Require : Pretrained diffusion video dynamics model $\mathcal{V}_\theta$; batch size $B = B_{\text{flow}} + B_{\text{distill}}$.

while *not converged* **do**

  Input: A minibatch of trajectories $\{(o_{\tau-T:\tau}, a_{\tau:\tau+H}, o_{\tau:\tau+H})\}_{\tau=1}^{B}$

  `// Split minibatch`

  $\mathcal{B}_{\text{flow}} \leftarrow \{(o_{\tau-T:\tau}, a_{\tau:\tau+H}, o_{\tau:\tau+H})\}_{\tau=1}^{B_{\text{flow}}}$

  $\mathcal{B}_{\text{distill}} \leftarrow \{(o_{\tau-T:\tau}, a_{\tau:\tau+H}, o_{\tau:\tau+H})\}_{\tau=1}^{B_{\text{distill}}}$

  $L_{\text{flow}} \leftarrow 0, \;\; L_{\text{distill}} \leftarrow 0$

  `// Flow matching objective`

  foreach $(o_{\tau-T:\tau}, a_{\tau:\tau+H}, o_{\tau:\tau+H}) \in \mathcal{B}_{\textit{flow}}$ **do**

    Sample $t \sim \mathcal{U}(0,1)$ and $\epsilon \sim \mathcal{N}(0, I)$

    $\tilde{o}^t_{\tau:\tau+H} \leftarrow (1-t)\, o_{\tau:\tau+H} + t\,\epsilon$

    $\mathbf{c} = (o_{\tau-T:\tau}, a_{\tau:\tau+H})$

    $\mathbf{v}_{t,t} \leftarrow \textsc{Diff2Flow}(\mathcal{V}_\theta,\, \tilde{o}^t_{\tau:\tau+H},\, \mathbf{c},\, t,\, t)$

    $L_{\text{flow}} \leftarrow L_{\text{flow}} + \|\mathbf{v}_{t,t} - (o_{\tau:\tau+H} - \epsilon)\|_2^2$

  end

  `// Semigroup distillation objective`

  foreach $(o_{\tau-T:\tau}, a_{\tau:\tau+H}, o_{\tau:\tau+H}) \in \mathcal{B}_{\textit{distill}}$ **do**

    Sample $s, u, t \sim \mathcal{U}(0,1)$ such that $0 \le s < u < t \le 1$ and $\epsilon \sim \mathcal{N}(0, I)$

    $\tilde{o}^s_{\tau:\tau+H} \leftarrow (1-s)\, o_{\tau:\tau+H} + s\,\epsilon$

    $\mathbf{c} = (o_{\tau-T:\tau}, a_{\tau:\tau+H})$

    $\mathbf{v}_{s,u} \leftarrow \textsc{Diff2Flow}(\mathcal{V}_\theta,\, \tilde{o}^s_{\tau:\tau+H},\; \mathbf{c},\, s,\, u)$

    $\mathbf{X}_{s,u} \leftarrow \tilde{o}^s_{\tau:\tau+H} + (u-s)\,\mathbf{v}_{s,u}$

    $\mathbf{v}_{u,t} \leftarrow \textsc{Diff2Flow}(\mathcal{V}_\theta,\, \mathbf{X}_{s,u},\, \mathbf{c},\, u,\, t)$

    $\mathbf{X}_{s,t} \leftarrow \mathbf{X}_{s,u} + (t-u)\,\mathbf{v}_{u,t}$

    `// Direct teacher step` $X^*_{s,t}$

    $\mathbf{v}_{s,t} \leftarrow \textsc{Diff2Flow}(\mathcal{V}_\theta,\, \tilde{o}^s_{\tau:\tau+H},\, \mathbf{c},\, s,\, t)$

    $\mathbf{X}^*_{s,t} \leftarrow \tilde{o}^s_{\tau:\tau+H} + (t-s)\,\mathbf{v}_{s,t}$

    $L_{\text{distill}} \leftarrow L_{\text{distill}} + \left\|\mathbf{X}^*_{s,t} - \text{sg}(\mathbf{X}_{s,t})\right\|_2^2$

  end

  $\theta \leftarrow \theta - \eta \nabla_\theta \left(L_{\text{flow}} + L_{\text{distill}}\right)$

end

Algorithm 3: VidPlan: Inference-Time Planning with Video Models

Require *Action Proposer* u*, Video Model* $\mathcal{V}_\theta$*, Reward Model* $\mathcal{R}$*, UQ Model* $\mathcal{U}_\beta$

PlanningLoop $(o_{t-T+1:t}, \ell)$**:**

  Input: Observation History $o_{t-T+1:t}$, Task Prompt ℓ

  Output: Planned Trajectory $a_{t:t+H-1}$

  `// Action Proposal`

  $\mathcal{A}_{t:t+H-1} \coloneqq \{a^i_{t:t+H-1}\}_{i=1}^{N} \leftarrow u(o_{t-T+1:t}, \ell)$

  foreach $a^i_{t:t+H-1} \in \mathcal{A}_{t:t+H-1}$ **do**

    `// Future Prediction and Uncertainty Quantification`

    $o^i_{t+1:t+H},\; h^i_{t+1:t+H} \leftarrow \mathcal{V}_\theta(o_{t-T+1:t}, a_{t:t+H-1}),\; \mathcal{U}_\beta(\cdot)$

    `// Reward prediction`

    $r^i_{t+1:t+H} \leftarrow \mathcal{R}(o_{t+1:t+H}, a_{t:t+H-1})$

    `// Uncertainty-Guided Reward Weighting via` $w(\cdot)$

    $r^i_{t+1:t+H} \leftarrow w(h^i_{t+1:t+H}) \cdot r^i_{t+1:t+H}$

  end

  `// Global (High-Level) Planner`

  $a_{t:t+H-1} \leftarrow \arg\max_{a^i_{t:t+H-1}} \{r^i_{t+1:t+H}\}$

  foreach iter **do**

    `// Local Planner via` $\text{GradOpt}(\cdot)$

    $a_{t:t+H-1} \leftarrow \text{GradOpt}(a_{t:t+H-1}; \mathcal{R}(\cdot), h_{t+1:t+H})$

    `// Future Prediction and Uncertainty Quantification`

    $o_{t+1:t+H},\; h_{t+1:t+H} \leftarrow \mathcal{V}_\theta(\cdot),\; \mathcal{U}_\beta(\cdot)$

  end

  return $a^\star_{t:t+H-1}$

APPENDIX II
PRELIMINARIES

We provide additional background material on diffusion, flow matching, and flow map to complement Section IV-A and Section IV-B.

A. Diffusion Model

As described in Section IV-B, we adopt the EDM-framework [51] for diffusion models, since it offers a general framework unifying various diffusion formulations [62], [50]. Diffusion models learn to gradually denoise noisy data, whose marginal distribution can be described by:

$$p_t(\mathbf{x}) = \int p_{\mathbf{data}(\mathbf{x}_0)} \left[\mathcal{N}(\mathbf{x}; s(t)\mathbf{x}, s(t)^2\sigma(t)^2\mathbf{I}\right] d\mathbf{x_0} \quad (5)$$

$$= s(t)^{-d} \left[p_{\text{data}} * \mathcal{N}(0, \sigma(t)^2\mathbf{I})\right](\mathbf{x}/s(t)), \quad (6)$$

where $p_a * p_b$ denotes the convolution of probability density functions, σ is the pre-defined noise schedule and $s(t)$ is a pre-defined scaling schedule. Additionally, we have $p(\mathbf{x}; \sigma) = p_{\text{data}}/s(t) * \mathcal{N}(0, \sigma(t)^2\mathbf{I})$.

The generation process (also called the backward process) can be defined as:

$$d\mathbf{x} = \left[\frac{\dot{s}(t)}{s(t)}\mathbf{x} - s(t)^2\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\frac{\mathbf{x}}{s(t)}; \sigma(t))\right] dt, \quad (7)$$

where it evolves a sample $\mathbf{x}_{t_a} \sim p(\mathbf{x}; \sigma(t_a))$ from time t_a to time t_b, given by the sample $\mathbf{x}_{t_b} \sim p(\mathbf{x}; \sigma(t_b))$. One commonly used formulation is to set $s(t) = 1$ (c.f. Section IV-B).

The diffusion model objective is to learn the time-dependent score function $\nabla_{\mathbf{x}} \log p\left(\frac{\mathbf{x}}{s(t)}; \sigma(t)\right)$, which governs the generation dynamics in Equation (7). In practice, rather than modeling the score function directly, diffusion models are commonly trained using equivalent parameterizations corresponding to different prediction targets. We summarize the three most widely used formulations.

a) Noise ($\boldsymbol{\epsilon}$) Parameterization.: The model predicts the noise term $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$. The corresponding score function is given by:

$$\nabla_{\mathbf{x}} \log p\left(\frac{\mathbf{x}_t}{s(t)}; \sigma(t)\right) = -\frac{1}{s(t)\sigma(t)}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t). \quad (8)$$

b) Clean Data Parameterization.: Alternatively, the model may directly predict the clean data $\mathbf{x}_1$:

$$\mathbf{x}_{1,\theta} = \mathbf{x}_{1,\theta}(\mathbf{x}_t, t). \quad (9)$$

Under this parameterization, the score function becomes:

$$\nabla_{\mathbf{x}} \log p\left(\frac{\mathbf{x}_t}{s(t)}; \sigma(t)\right) = -\frac{\mathbf{x}_t - s(t)\mathbf{x}_{1,\theta}(\mathbf{x}_t, t)}{s(t)^2\sigma(t)^2}. \quad (10)$$

c) Velocity ($\mathbf{v}$) Parameterization.: The velocity parameterization predicts a linear combination of the scaled data and noise:

$$\mathbf{v}_t = \frac{\mathbf{x}_t}{s(t)\sigma(t)} - \mathbf{x_0}. \quad (11)$$

Given a model prediction $\mathbf{v}_\theta(\mathbf{x}_t, t)$, the score function can be recovered as

$$\nabla_{\mathbf{x}} \log p\left(\frac{\mathbf{x}_t}{s(t)}; \sigma(t)\right) = -\frac{1}{s(t)\sigma(t)}\left(\frac{\mathbf{x}_t}{s(t)\sigma(t)} - \mathbf{v}_\theta(\mathbf{x}_t, t)\right). \quad (12)$$

Equivalence of Parameterizations. The three parameterizations are mathematically equivalent, as they correspond to different reparameterizations of the same underlying score function in Equation (7). Consequently, we can map between these parameterizations analytically. Hence, he choice of parameterization primarily affects optimization behavior rather than model expressiveness.

B. Flow Matching

Flow matching [80] is defined via an ODE:

$$\frac{d}{dt}\mathbf{x}_t = b_t(\mathbf{x}_t). \quad (13)$$

Here, $\mathbf{x_t} = \alpha_t\mathbf{x}_0 + \beta_t\mathbf{x_1}$ and $b_t = \mathbb{E}_{\mathbf{x}_0,\mathbf{x}_1}[\frac{d}{dt}x_t]$, which is also known as the "velocity". A common choice is $\alpha_t = (1-t)$ and $\beta_t = t$, as we used in Section IV-B.

The learning objective is to model the velocity using a network:

$$L_b(\hat{b}) = \int_0^1 \mathbb{E}_{\mathbf{x}_0,\mathbf{x}_1}\left[\left|\hat{b}_t(\mathbf{x_t}) - \frac{d}{dt}\mathbf{x}_t\right|^2\right]. \quad (14)$$

C. Flow Map

A flow map [16] is defined as:

$$X_{s,t}(\mathbf{x}_s) = \mathbf{x_t} \text{ for all } (s,t) \in [0,1]^2, \quad (15)$$

where $(\mathbf{x_t})_{t\in[0,1]}$ is any solution of Equation (14). Following [16], we parameterize the flow map via:

$$X_{s,t}(\mathbf{x}_s) = \mathbf{x}_s + (t-s)v_{s,t}(x_s), \quad (16)$$

which should satisfy the condition given by:

$$v_{t,t}(x) = b_t(x). \quad (17)$$

In addition, a flow map must satisfy one of the following conditions:

$$\text{(i) (Lagrangian):} \quad \partial_t X_{s,t}(x) = v_{t,t}\big(X_{s,t}(x)\big), \quad \forall (s,t) \in [0,1]^2 \text{ and } x \in \mathbb{R}^d. \quad (18)$$

$$\text{(ii) (Eulerian):} \quad \partial_s X_{s,t}(x) + \nabla X_{s,t}(x)\, v_{s,s}(x) = 0, \quad \forall (s,t) \in [0,1]^2 \text{ and } x \in \mathbb{R}^d. \quad (19)$$

$$\text{(iii) (Semigroup):} \quad X_{u,t}\big(X_{s,u}(x)\big) = X_{s,t}(x), \quad \forall (s,t,u) \in [0,1]^3 \text{ and } x \in \mathbb{R}^d. \quad (20)$$

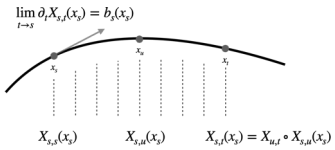

Fig. 19: **Flow map fine-tuning with PSD.** The instantaneous change of the flow map, $\lim_{t\to s} \partial_t \mathbf{X}_{s,t}(\mathbf{x}_s)$, recovers the velocity field $b_t(\mathbf{x}_t)$ in flow matching and ensures training stability. The semigroup property of the flow map, $\mathbf{X}_{s,t}(\mathbf{x}_s) = \mathbf{X}_{u,t} \circ \mathbf{X}_{s,u}(\mathbf{x}_s)$, enables accelerated generation.

We utilize the semigroup condition, since the Lagrangian and Eulerian conditions require Jacobian-vector product (JVP) operations, limiting their practical application in large models such as generative video models. Figure 19 illustrates the training objective of a flow map with the Semigroup condition.