*i*WAM: EFFICIENT ACTION PREDICTION FROM WORLD MODELS WITHOUT EXPLICIT FUTURE PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

World-action models (WAMs) equip a policy with a pretrained video generation model, on the premise that predicting how a scene should evolve helps decide how to act. While this simplifies action prediction, imagining future scenes is expensive for real-time control. This motivates efficient proxies for future imagination, but it remains unclear what such a proxy should preserve. Our key insight is that an action policy does not require a high-resolution future; it only needs enough information to specify a plausible future with respect to which an action should be produced. This leads to a simple design principle for ***i*WAM**: rather than denoising an entire future frame, we condition the action policy on intermediate activations from a single video-model evaluation at a latent noise vector that would otherwise initialize denoising. The key challenge is ensuring these activations correspond to the demonstrated future whose actions supervise the policy. We address this by training the video model to approximately represent an invertible flow map: integrating a demonstrated future in the clean-to-noise direction recovers a latent endpoint corresponding to that future, yielding activations aligned with the demonstrated actions. At inference, we sample this endpoint from the prior, and require only a single video-backbone forward pass for conditioning. Across simulated benchmarks and real robots, ***i*WAM** recovers the generalization benefits of future-conditioning at the inference cost of methods that discard futures entirely.

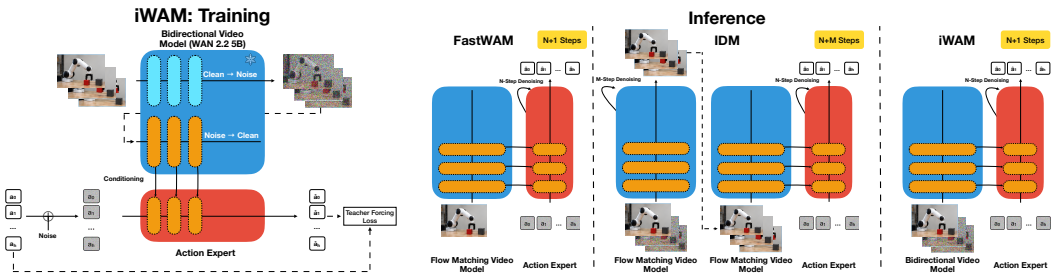
1 INTRODUCTION

Video generation models have become an attractive source of pretrained priors for robotic manipulation (Hou et al., 2026; Huang et al., 2025). Because video data is more plentiful than robotic data, pre-trained video models learn generalizable priors about objects, spatial relations, and physical world dynamics. The desire is that these features can be used to train robotic policies that exhibit far greater generalization than those trained on (comparatively limited) robot action data alone.

World action models (WAMs) extract robot actions from a video generation model by finetuning it on robot data. Some WAMs jointly predict future video and actions (Ye et al., 2026; Zhu et al., 2025; Bi et al., 2026), while others first generate a future video and provide it as conditioning to a separate action-generation network, as in inverse dynamics models (IDMs) (Du et al., 2023; Li et al., 2026). During training, IDMs condition the action policy on activations computed from the ground-truth future, but no such future is available at deployment, so both kinds of WAM must first imagine one with the video model before acting. Both approaches inherit strong generalization from the pretrained video models on which they are based, but incur substantial inference cost from generating full, high-resolution videos at deployment. This is particularly prohibitive in robotics, where real-time control demands low latency. Lighter-weight alternatives such as FastWAM (Yuan et al., 2026) take the opposite approach: they still train the video model to predict future observations alongside the action policy, but remove future observations from action conditioning entirely, supplying the policy only with intermediate-layer activations computed from the current observation. Because these activations can be obtained without generating future observations, FastWAM-style approaches substantially reduce inference cost.

In principle, explicitly generating the future may be unnecessary if the video model's intermediate activations already contain enough information about a future imagination to guide action prediction. However, flow-matching represents a distribution over possible futures: their activations encode how a particular noise sample should be transformed into one sample from this distribution. Thus, while

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



(a) ***i*WAM** training. (b) Inference, compared with baselines.

Figure 1: **Overview of *i*WAM. (a)** The frozen flow map video model maps the demonstrated future to its noise endpoint with a single clean-to-noise evaluation, then performs a single noise-to-clean evaluation from that endpoint. The action expert is conditioned on the intermediate-layer activations of this second evaluation and trained to predict the demonstrated action chunk. **(b)** FastWAM conditions the action expert on activations computed from the current observation alone, while IDM first generates a future with M video-model denoising steps and conditions on its activations. ***i*WAM** draws a noise endpoint from the prior and conditions on the activations from a single noise-to-clean evaluation, matching FastWAM's cost while conditioning on a specific future like IDM.

these activations contain useful information about the future, standard video models are designed to only expose it through the iterative denoising process that spends substantial compute. The challenge is therefore to extract the information relevant for action prediction without spending this iterative compute. Since FastWAM does not spend any iterative compute to identify a particular future, FastWAM's strong performance on standard benchmarks such as LIBERO (Liu et al., 2023) and RoboTwin (Chen et al., 2025b) is surprising. One possibility is that such conditioning suffices within the training distribution when paired with a sufficiently high-capacity video model, but the lack of an iterative process fails to preserve the same degree of generalization that explicitly generating a future would provide. We test this hypothesis in Sections 4 and 6 and show that this more limited form of conditioning indeed comes at the cost of reduced generalization.

Observing this result, we ask: ***to what extent can we forgo explicit video generation while retaining both efficient and generalizable action prediction?*** At first glance, this may appear to require a "free lunch": if the benefits of imagination arise from spending compute iteratively to generate a future at test time, eliminating this computation might inevitably sacrifice the generalization it provides. However, we note that action prediction does not require reconstructing the full high-resolution future; it only requires a sufficiently informative proxy for it. This suggests that if the computation performed by iterative generation can be even reasonably approximately amortized into substantially fewer model evaluations, we may retain its benefits for action prediction at much lower cost. Crucially, prior work (Boffi et al., 2026; Geng et al., 2026; Frans et al., 2025) suggests that such amortizations of flow-matching processes can often be represented by networks of comparable capacity, in part due to the underlying flow being deterministic. Our method exploits this structure of flow models by learning a bidirectional flow map that approximately associates each noise vector with its corresponding future in either direction. This bidirectional map allows us to extract future-specific information with just a single model evaluation, rather than explicitly integrating the full generative trajectory.

Building on this insight, we propose ***i*WAM**, an implicit world-action model that conditions the action policy directly on intermediate activations from a single evaluation of a bidirectional video flow model that we train on robot data. During training, we integrate each demonstrated future in the clean-to-noise direction to recover its corresponding noise endpoint; evaluating the model once at this endpoint yields activations associated with the same future as the actions supervising the policy. At inference, we instead sample a noise endpoint from the flow prior and perform a single model evaluation, conditioning the action policy on its intermediate activations together with features from past observations. Thus, ***i*WAM** replaces iterative video generation with a single evaluation while retaining the future-specific conditioning that enables generalization.

Across simulated and real-world manipulation benchmarks (Zhou et al., 2025; Khazatsky et al., 2024; Jain, 2026), ***i*WAM** recovers the generalization benefits of explicitly future-conditioned policies at the inference cost nearing that of approaches that discard future generation entirely. This clarifies what an action policy requires from a video model to generalize: neither a highly precise future nor

observation-conditioned activations that leave the future unresolved, but sufficient information to identify the *relevant mode of the future-video distribution*. Explicit future generation is sufficient to specify such a mode; the success of iWAM shows that it is not necessary.

2 Related Work

Video generation models for robot control. Video models pretrained on large video datasets capture how objects and scenes evolve, providing a useful prior for robot control (Hou et al., 2026; Huang et al., 2025). WAMs adapt this prior by finetuning on robot data and predicting actions from imagined futures (Du et al., 2023; Liang et al., 2024; Zhu et al., 2025; Bi et al., 2026; Li et al., 2026; Ye et al., 2026; Kim et al., 2026). WAMs show promising generalization to unseen objects, scenes, and tasks, but repeatedly evaluating a large video backbone to generate futures makes inference slow. Our goal is to retain the benefits of conditioning on imagined futures without iterative generation.

Efficient action prediction from video models. Several methods avoid generating the future at inference. FastWAM (Yuan et al., 2026) and UVA (Li et al., 2025) train the video model to predict the future alongside actions, but at test time predict actions from representations of the current observation alone. VPP (Hu et al., 2025) and VidMan (Wen et al., 2024) condition the action policy on intermediate features from a single forward pass of a video diffusion model, rather than on a generated video. We argue that such approaches are limited: their features are computed either from the current observation alone or from independently sampled noise, so during training they need not correspond to the future whose actions supervise the policy. Section 4 shows that removing the future from the conditioning hurts generalization to unseen environments, and Section 5 shows that conditioning on independently sampled noise fits diverse data such as DROID poorly. ***i*WAM** also uses a single forward pass of the video backbone, but during training its activations correspond to the same future as the demonstrated actions.

Few-step and bidirectional flow maps. A large body of work reduces the cost of sampling from diffusion and flow models by learning to traverse the probability flow in a few large steps, including consistency models (Song et al., 2023), shortcut models (Frans et al., 2025), MeanFlow (Geng et al., 2026), terminal velocity matching (Zhou et al., 2026), and flow maps more generally (Boffi et al., 2026). The flow can also be run in reverse to recover the noise that generates a given sample: DDIM inversion (Song et al., 2021) does so by integrating backward in many steps, and bidirectional consistency models (Li & He, 2024) learn to do so in a few. These methods aim to generate or edit samples more cheaply. We use a bidirectional flow map for a different purpose, and never decode the future at all: during training, the clean-to-noise direction recovers the noise endpoint of the demonstrated future, and the noise-to-clean direction provides, in a single evaluation, the activations on which the action policy is conditioned.

3 Preliminaries and Problem Statement

Flow-Matching. Flow Matching (Lipman et al., 2022) is a family of generative models that learn a time-dependent velocity field that transports a simple prior distribution, such as Gaussian noise, to the data distribution. Formally, given samples $z_1 \sim p_{\text{prior}}$ and $z_0 \sim p_{\text{data}}$, let $z_\tau = a_\tau z_0 + b_\tau z_1$ denote a point along the flow path indexed by flow time $\tau \in [0, 1]$, where a_τ and b_τ are the chosen time schedules. The corresponding conditional velocity (Lipman et al., 2022) is $u_\tau = z'_\tau = a'_\tau z_0 + b'_\tau z_1$, where $'$ denotes the derivative with respect to flow time. For conditional generation with conditioning information y, a neural network parameterized by θ is trained to predict the marginal velocity field $v^\tau_\theta(z_\tau \mid y) = \mathbb{E}\left[u_\tau \mid z_\tau, y\right]$ using the flow-matching objective

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\tau, z_0, z_1, y}\left[\left\| v^\tau_\theta(z_\tau \mid y) - u_\tau \right\|_2^2\right]. \tag{1}$$

At inference time, a sample $z_1 \sim p_{\text{prior}}$ is drawn and the learned velocity field is integrated from the noise endpoint to the data endpoint to obtain a generated sample.

Flow maps and shortcut models. Flow map models (Boffi et al., 2026) instead learn to directly transport a sample between two flow times $t, s \in [0, 1]$. We parameterize the flow map through its average velocity $v^{t\to s}_\theta(z_t \mid y)$ (Geng et al., 2026), such that

$$z_s = z_t + (s - t)\, v^{t\to s}_\theta(z_t \mid y). \tag{2}$$

Unlike the marginal velocity in Flow Matching, the average velocity describes the displacement over the entire interval from t to s. A flow map can therefore move directly between distant points along

the flow without integrating the marginal velocity field over many intermediate steps. We consider flow maps defined over arbitrary time pairs $t, s \in [0, 1]$, allowing both noise-to-clean ($s < t$) and clean-to-noise ($s > t$) transport, and refer to this as a *bidirectional flow map*. As an instantiation of flow maps, we use **shortcut models** (Frans et al., 2025), although in principle, other approaches like Lagrangian self-distillation (Boffi et al., 2026) and MeanFlow (Geng et al., 2026) training are also feasible. We discuss the trade-offs between different flow-map instantiations in Appendix 5.3.

Problem formulation. Given the current observation o_0 and language instruction ℓ, let $c = (o_0, \ell)$ denote the conditioning context. A video model represents the conditional distribution of the ground truth future observations $o_{1:N} = (o_1, \ldots, o_N)$. We parameterize this distribution using a conditional flow model with vector field $v^\tau_{\theta_v}(z_\tau \mid c)$, where $\tau \in [0, 1]$ denotes flow time, with $z_0 = o_{1:N}$ at the data endpoint and $z_1 \sim p_{\text{prior}}$ at the noise endpoint[1]. At inference, future observations are generated by sampling $z_1 \sim p_{\text{prior}}$ and integrating the flow from $\tau = 1$ to $\tau = 0$, yielding the predicted future observations $\hat{o}_{1:N} = z_0$. We denote the action policy by π, which can either be a branch of a joint video-action prediction model or a separate downstream network, as in IDMs. Further, we let $h_{\theta_v}(\cdot)$ denote the intermediate-layer activations of the video model while computing $v^0_{\theta_v}$ on the given input. The various approaches below differ only in the inputs provided to h_{θ_v}.

IDM conditions on activations computed from the future observations and the action policy predicts:

$$a_{1:N} \sim \pi(\cdot \mid h_{\theta_v}(c, \tilde{o}_{1:N})), \qquad \tilde{o}_{1:N} = \begin{cases} o_{1:N}, & \text{during training,} \\ \hat{o}_{1:N}, & \text{at inference.} \end{cases} \tag{3}$$

In contrast, FastWAM-style approaches omit the future-observation tokens:

$$a_{1:N} \sim \pi(\cdot \mid h_{\theta_v}(c)). \tag{4}$$

The key distinction is therefore in the choice of h_{θ_v} used by the action policy. These two formulations highlight a key trade-off. IDM ties action prediction to the generated future $\tilde{o}_{1:N}$, which is obtained by integrating the flow at inference time. FastWAM-style approaches avoid this overhead, but do not explicitly condition on any specific future. In this paper, we will propose a choice of h_{θ_v} that ensures the generalization performance of IDMs, but with similar inference costs to FastWAM. Our goal is therefore twofold: first, to determine whether conditioning the action policy on the future is important for action policies to generalize; and second, to determine whether this conditioning can be provided without explicitly generating that future. We study the former question in Section 4, and then use these findings to develop a more inference-efficient WAM.

4 Does Removing Future Information Hurt Generalization?

IDMs explicitly condition the action policy on a predicted future, which provides a strong signal for action prediction, but obtaining this future requires running an expensive iterative video generation process at inference time. FastWAM-style approaches avoid this cost by conditioning the action policy only on $h_{\theta_v}(c)$, and prior work has shown that this can perform well on standard in-distribution benchmarks (Yuan et al., 2026). This raises a natural question: when does explicitly conditioning on a specific future actually matter, and in particular, when does removing it begin to hurt performance? We study this question by comparing the performance of policies conditioned on $h_{\theta_v}(c)$ against those conditioned on $h_{\theta_v}(c, \tilde{o}_{1:N})$ (IDM-style) in unseen tasks, both in simulation and on real robots.

Table 1: **Simulation manipulation performance.** Success rates (%) over 20 trials per task at replanning horizons of 16 and 24 action steps. The average is computed across all reported task–horizon combinations.

Method	Cube → Bowl		Can → Mug		Banana → Bin		Avg.
	16	24	16	24	16	24	
FastWAM	0%	0%	0%	0%	20%	35%	9.2%
IDM	**95%**	**100%**	95%	90%	**100%**	**100%**	**96.7%**
$\pi_{0.5}$	**95%**	—	55%	—	45%	—	65.0%
IDM-flow map	90%	80%	95%	**95%**	**100%**	95%	92.5%
iWAM	80%	85%	**100%**	**95%**	**100%**	**100%**	93.3%
iWAM (MIP)	90%	95%	85%	85%	**100%**	95%	91.7%

Setup. We compare FastWAM and IDM in two settings, training a separate set of models for each. In both, the two methods share the same Wan2.2-5B video backbone (Team Wan et al., 2025) and 1B-parameter action expert. In our main setting, we train on DROID (Khazatsky et al., 2024), a large and diverse dataset of real-world manipulation demonstrations, and evaluate the resulting policies in environments not seen during training: three pick-and-place tasks in the DROID simulation

[1]We follow the time convention of MeanFlow (Geng et al., 2026)

benchmark (Jain, 2026), and three real-world tasks on a physical robot. In the second setting, we train on LIBERO (Liu et al., 2023) and evaluate on LIBERO-PRO (Zhou et al., 2025), which tests the policies under controlled perturbations to objects, initial states, and instructions. Full training and evaluation details are provided in Section 6.

Observations. On DROID, removing future conditioning substantially hurts performance. In simulation (Table 1), FastWAM succeeds in 9.2% of trials, compared to 96.7% for IDM, and fails entirely on two of the three tasks. The gap persists on real robots (Table 2): FastWAM averages 13.3% success against 80.0% for IDM. On Blocks → Bowl, FastWAM places at least one block in 80% of trials but never completes all three placements, whereas IDM does so in 85% of trials.

On LIBERO-PRO, the gap is smaller but consistent (Table 3): averaged over the four perturbation types, IDM succeeds in 51.6% of trials against 44.5% for FastWAM with joint training, and 54.4% against 51.3% with two-stage training. Additional LIBERO-PRO results, including the effect of joint versus two-stage training, are provided in Appendix B. These results suggest that conditioning on current-observation activations, as in FastWAM, may suffice when evaluation closely matches the training distribution, but becomes limiting on diverse data and in unseen environments, where features from an imagined future generalize better by drawing on the video model's large-scale pretraining.

Table 2: **Real-world performance on DROID.** Success rates (%) over 20 trials per task. For Blocks → Bowl, columns report the fraction of trials completing at least one, at least two, or all three placements.

Method	Blocks → Bowl			Cup $\xrightarrow{\text{Pen}}$ Table	Fold Towel	Avg.
	≥ 1	≥ 2	3	Success	Success	
FastWAM	80%	15%	0%	30%	10%	13.3%
IDM	**100%**	**90%**	**85%**	85%	70%	80.0%
iWAM (MIP)	**100%**	**90%**	75%	**95%**	**85%**	**85.0%**

5 *i*WAM: Efficient WAMs Without Future Prediction

Our goal is to retain the generalization benefits of future-conditioned action prediction without the cost of iterative future generation. At first glance, this may appear to require a "free lunch": if these benefits arise from spending substantial computation generating a future, replacing this process with a single model evaluation might inevitably discard useful information. However, action prediction does not require reconstructing the full high-resolution future; it only requires a sufficiently informative proxy for it. This suggests a different possibility: rather than reproducing the full generation process, we can *amortize* enough of its computation into a learned model to expose the information relevant for action prediction.

Table 3: **Performance under LIBERO-PRO perturbations.** *joint* denotes joint training of the video model and action policy, whereas *disj* denotes the two-stage training procedure.

Approach	**None**	**Obj.**	**Pos.**	**Sem.**	**Task**
IDM (joint)	97.9	83.7	2.7	96.3	**23.7**
FastWAM (joint)	96.9	78.0	0.9	83.7	15.3
IDM (disj)	98.5	84.9	10.9	98.0	23.6
FastWAM (disj)	97.6	79.5	10.5	94.7	20.4
***i*WAM** (prior noise)	**98.8**	**85.2**	**12.3**	**98.5**	23.5

Flow-based video models provide a natural starting point for doing so. Note that for a fixed conditioning c, the noise endpoint z_1 determines a unique trajectory of the flow and therefore the future reached at its clean endpoint. The expensive part is repeatedly evaluating the video model to transport z_1 into that future. If we could instead learn a direct approximation to this transport, we could sample the same noise endpoint but extract future-specific information from a single model evaluation. Prior work suggests that such amortizations of flow-based generation can often be represented using models of comparable capacity by training a flow map (Boffi et al., 2026; Geng et al., 2026; Frans et al., 2025), making this a plausible route to reducing inference computation.

5.1 Amortizing Generation with Bidirectional Flow Map

Our goal is therefore to condition the action policy on the video model's intermediate-layer activations while keeping inference cost as low as possible. Ideally, this cost would be a single forward pass of the video backbone, rather than the many passes required to iteratively generate a future with standard flow-matching. The simplest way to realize this is to train the action policy exactly as it will be used at inference: for each training example $(c, o_{1:N}, a_{1:N})$, sample a noise endpoint $z_1 \sim p_{\text{prior}}$, evaluate the video model once at z_1, and train the policy to predict the demonstrated actions $a_{1:N}$ from the

resulting activations. We refer to this variant as ***i*WAM** (prior noise). However, this approach fits the training data poorly: as shown in Figure 2, it reaches a substantially higher action-training loss than IDM, even though both use the same amount of computation during training. Why is this the case?

The difficulty is that a single evaluation of a standard flow model does not reveal the future associated with a particular noise endpoint. Under the full integrated flow, each z_1 deterministically maps to a particular clean future. Recovering that future, however, requires integrating the learned vector field over many steps. At the noise endpoint itself, the optimal one-step prediction instead collapses to the *average* clean future conditioned on c, and is therefore identical across noise endpoints (Proposition A.1). Thus, although different noise samples ultimately map to different futures, a single standard-flow evaluation contains little information about which future a particular sample will produce. The computation spent on this evaluation therefore cannot serve as useful test-time computation for action selection.

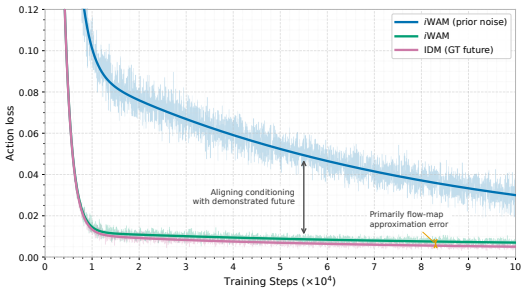


Figure 2: **Aligning the conditioning with the demonstrated future lowers action-training loss on DROID.** Sampling noise endpoints from the prior during training (***i*WAM** (prior noise)) results in substantially higher training loss. ***i*WAM** aligns its predictive conditioning with the demonstrated future through the corresponding noise endpoint, largely closing the gap to IDM, which directly conditions on ground-truth futures during training. The small remaining gap may reflect approximation error in the learned bidirectional flow map.

Training the action expert introduces a second requirement: *alignment* between the represented future and the actions used for supervision. Each demonstration pairs a realized future $o_{1:N}$ with actions $a_{1:N}$ that produce it, so the representation supplied to the action policy should correspond to that same future. An independently sampled z_1 need not satisfy this requirement: under the full flow, it may correspond to a different plausible future requiring different actions. Ideally, we would instead use the particular noise endpoint associated with $o_{1:N}$. Although the integrated flow is invertible, recovering this endpoint with a standard flow model requires integrating the velocity field in reverse, repeatedly noising $o_{1:N}$, and is therefore as expensive as generating a video in the forward direction during training.

These two problems share the same underlying cause: in standard flow matching, the correspondence between a noise endpoint and its future is represented implicitly by the *trajectory of the flow*, rather than being accessible through a single model evaluation call. At training time, we need to map the demonstrated future to its corresponding noise endpoint in one clean-to-noise evaluation; at inference time, we need one noise-to-clean evaluation to expose information about the future associated with a sampled endpoint. We address both problems by learning a ***bidirectional flow map*** that predicts finite transport between arbitrary points in flow time, in either direction. During training, the clean-to-noise direction maps the demonstrated future $o_{1:N}$ directly to its corresponding endpoint $z_1^{o_{1:N}}$ in a single evaluation. The noise-to-clean direction then learns to map this endpoint directly toward its corresponding future, making a single evaluation at $z_1^{o_{1:N}}$ informative about the particular future that generated the supervision. This gives the action policy future-aligned representations without requiring iterative flow integration.

At inference time, we no longer need the clean-to-noise direction: we simply sample $z_1 \sim p_{\text{prior}}$ and evaluate the noise-to-clean map once. Its intermediate activations provide a representation of the particular future mode associated with that noise sample, which is then used to predict actions. Crucially, this single-step map need not reconstruct the high-resolution future perfectly. Our action policy only requires that its intermediate representations preserve the future-specific information relevant for choosing actions, allowing us to amortize the expensive iterative generation process into a single video-model evaluation.

5.2 Implicit World-Action Model

We instantiate this idea in ***i*WAM**, an *implicit world-action model* that conditions the action policy directly on intermediate activations from the bidirectional video flow model. During training, given a demonstrated future $o_{1:N}$, we first evaluate the bidirectional flow map in the clean-to-noise direction

Algorithm 1 Implicit-WAM

Require: Flow-map model θ_v, action model θ_a

Training

repeat
  $(c, o_{1:N}, a_{1:N}) \sim \mathcal{D}$
  $z_1^{o_{1:N}} \leftarrow o_{1:N} + v_{\theta_v}^{0\to 1}(o_{1:N} \mid c)$
  $h^{\text{iWAM}} \leftarrow h_{\theta_v}^{1\to 0}(c, z_1^{o_{1:N}})$
  $\theta_a \leftarrow \text{Update}\big(\theta_a,\ \mathcal{L}_{\text{act}}(\theta_a; a_{1:N}, h^{\text{iWAM}})\big)$
until converged

Inference

$z_1 \sim p_{\text{prior}}$
$h^{\text{iWAM}} \leftarrow h_{\theta_v}^{1\to 0}(c, z_1)$
$\hat{a}_{1:N} \sim \pi_{\theta_a}(\cdot \mid h^{\text{iWAM}})$
return $\hat{a}_{1:N}$

to recover its corresponding noise endpoint $z_1^{o_{1:N}}$. We then evaluate the model once in the reverse, noise-to-clean direction at this endpoint, and use the intermediate-layer activations to condition the action policy. At inference, no demonstrated future is needed: we sample z_1 from the noise prior, perform a single noise-to-clean model evaluation, and condition the action policy on its intermediate activations. Thus, ***i*WAM** requires only one video-model evaluation at deployment.

We parameterize the bidirectional flow map through an average velocity $v_{\theta_v}^{t\to s}(z_t \mid c)$ that directly transports a state from flow time t to flow time s:

$$z_s = z_t + (s - t)\, v_{\theta_v}^{t\to s}(z_t \mid c). \tag{5}$$

Unlike standard flow-map training, which samples time pairs only in the noise-to-clean direction (Boffi et al., 2026; Geng et al., 2026; Frans et al., 2025), we train over arbitrary time pairs (t, s), covering both directions. Given a demonstrated future $z_0 = o_{1:N}$, the clean-to-noise direction directly produces its corresponding noise endpoint, and the noise-to-clean direction predicts transport from this endpoint back toward the same demonstrated future:

$$z_1^{o_{1:N}} = o_{1:N} + v_{\theta_v}^{0\to 1}(o_{1:N} \mid c)\,, \qquad o_{1:N} = z_1^{o_{1:N}} - v_{\theta_v}^{1\to 0}(z_1^{o_{1:N}} \mid c)\,. \tag{6}$$

Thus, the recovered noise endpoint $z_1^{o_{1:N}}$ and its corresponding noise-to-clean prediction are directly tied to the demonstrated future $o_{1:N}$.

Let $h_{\theta_v}^{t\to s}(c, z_t)$ denote the intermediate-layer activations used to predict $v_{\theta_v}^{t\to s}(z_t \mid c)$. Because $h_{\theta_v}^{1\to 0}(c, z_1^{o_{1:N}})$ is used to predict transport from the recovered noise endpoint toward $o_{1:N}$, these activations are trained to encode information about the demonstrated future without requiring that future to be explicitly generated. At inference, we instead sample $z_1 \sim p_{\text{prior}}$ and compute $h_{\theta_v}^{1\to 0}(c, z_1)$ with a single video-model evaluation. The action policy then predicts actions as

$$a_{1:N} \sim \pi\big(\cdot \mid h_{\theta_v}^{1\to 0}(c, \tilde{z}_1)\big)\,, \qquad \tilde{z}_1 = \begin{cases} z_1^{o_{1:N}}, & \text{during training,} \\ z_1, & \text{at inference,} \end{cases} \tag{7}$$

without explicitly generating the future associated with z_1. Algorithm 1 summarizes the complete ***i*WAM** training and inference procedure. In this way, ***i*WAM** replaces iterative high-resolution future generation with a single video-model evaluation while retaining future-specific information that can guide action prediction.

5.3 Implementing *i*WAM in Practice

Flow maps can be trained with several self-distillation objectives (Boffi et al., 2026). Methods such as MeanFlow (Geng et al., 2026) and TVM (Zhou et al., 2026) achieve state-of-the-art one-step image generation, but require network derivatives, which are costly and can destabilize training at video scale. We instead use shortcut-style training (Frans et al., 2025), which bootstraps larger steps from compositions of smaller ones using only forward evaluations of the network. Although shortcut-style objectives can yield lower one-step sample quality than derivative-based alternatives (Geng et al., 2026; Boffi et al., 2026), they provide stable training when predicting many frames jointly, and ***i*WAM** requires the activations to expose future-specific information that can guide action prediction, rather than to produce high-fidelity one-step generations. Unlike the original Shortcut formulation, we train over arbitrary time pairs (t, s), enabling both the clean-to-noise and noise-to-clean transport. We train ***i*WAM** in two stages: we first learn this bidirectional flow map, then freeze the video model and train the action policy on its intermediate activations.

Prompt guidance. Explicit prompt guidance is important for video generation to generalize to unseen tasks and environments, and IDM-style approaches typically use classifier-free guidance (CFG) for this purpose. CFG, however, constructs the guided field by combining two separate model evaluations (Appendix A), so neither evaluation produces intermediate-layer activations corresponding to the guided field that ***i*WAM** requires. Following Guidance-Free Training (GFT) (Chen et al., 2025a), we instead condition the flow map on a guidance strength w and train it to directly predict the corresponding prompt-guided average velocity in a single evaluation. When training the action policy, we use the same guidance strength as at inference, ensuring that its conditioning activations are computed under the same guidance strength in both settings.

***i*WAM Action Policies: Flow and MIP.** We train ***i*WAM** with both a flow-based action policy and the Minimal Iterative Policy (MIP) objective (Pan et al., 2025). MIP predicts actions in two denoising steps—an initial regression step followed by refinement. It nearly matches the performance of the flow-based policy, which uses multiple denoising steps at evaluation (Section 6).

6 Experiments

We now evaluate whether ***i*WAM** preserves the generalization benefits of future conditioning established in Section 4 while reducing action-generation cost. Our experiments address three questions:

1. Does the approximate computation provided by a shortcut model retain the performance and generalization of policies that explicitly generate the future?
2. What performance-latency trade-off does ***i*WAM** achieve compared with FastWAM and IDM?
3. Why does ***i*WAM** work well, while being efficient?

Experimental setup. We evaluate on DROID (Khazatsky et al., 2024) and LIBERO-PRO (Zhou et al., 2025) to study generalization in diverse manipulation environments and under controlled distribution shifts. DROID is a large-scale, real-world manipulation dataset containing 95K trajectories collected across 564 scenes using a Franka Panda arm with a Robotiq gripper. It captures similar manipulation behaviors across different objects, environments, and operators. We evaluate DROID-trained policies on three simulated pick-and-place tasks and three physical-robot tasks in environments not seen during training. LIBERO-PRO evaluates policies trained on LIBERO (Liu et al., 2023), which comprises four 10-task suites covering spatial relationships, object variation, task goals, and long-horizon manipulation. It tests generalization through perturbations to object attributes, initial object positions, instruction phrasing, and task intent. We train a separate set of policies on each dataset, using the same pretrained Wan2.2-5B video backbone and 1B-parameter action expert for all WAMs. DROID policies use the filtered training data described in Appendix C.1; LIBERO policies are trained on the original demonstrations and evaluated on LIBERO-PRO without further fine-tuning. Full training and evaluation details are provided in Appendices B and C.1.

Baselines and Comparison. For LIBERO-PRO, along with testing the original FastWAM implementation (Yuan et al., 2026) that jointly trains the action policy and the video model, we also adapt the two-stage training strategy of Liang et al. (2025) to FastWAM and IDM, where we first fine-tune the video model independently, then freeze its parameters while training the action policy. In addition, we also consider the simplified ***i*WAM** (prior noise) variant, which samples $z_1 \sim p_{\text{prior}}$ independently during training rather than using the noise endpoint associated with the demonstrated future. Full training and evaluation details are provided in Appendix B.

For DROID, we benchmark against two-stage IDM, which uses 8 flow-denoising steps at inference to generate the future. We also evaluate *IDM-flow map*, which uses the flow map learned by ***i*WAM** to approximate the future in a single step. For FastWAM, we train both joint and two-stage variants, but omit the latter from the main results due to lower performance. All flow-matching action policies use 8 action-denoising steps, while ***i*WAM** (MIP) uses 2. For simulation experiments, we additionally evaluate the official DROID-finetuned $\pi_{0.5}$ checkpoint. Complete evaluation details are provided in Appendix C.3.

Does approximate computation preserve performance and generalization? *i*WAM retains competitive performance across our evaluations. On LIBERO-PRO, the prior-noise variant matches IDM and improves over FastWAM across perturbations (Table 3). In unseen DROID simulation environments, the full ***i*WAM** achieves 93.3% success compared with IDM's 96.7%, while substantially

outperforming FastWAM and $\pi_{0.5}$ (Table 1). On the physical robot, ***i*WAM** (MIP) achieves 85.0% success compared with 80.0% for IDM (Table 2). Thus, ***i*WAM** shows that we can approximate the iterative computation of generating futures and retain its generalization benefits.

What performance–latency trade-off does *i*WAM achieve? Figure 3 compares success and inference latency in DROID simulation. Averaged across the three tasks and both replanning horizons, ***i*WAM** (MIP) achieves 91.7% success, approaching IDM's 96.7% at approximately 6× lower measured latency. FastWAM is faster but achieves only 9.2% success. IDM-flow map achieves 92.5%, but requires two video-backbone evaluations: one to predict the future and another to extract the conditioning activations. ***i*WAM** obtains its conditioning activations in one evaluation, achieving similar success with lower latency.

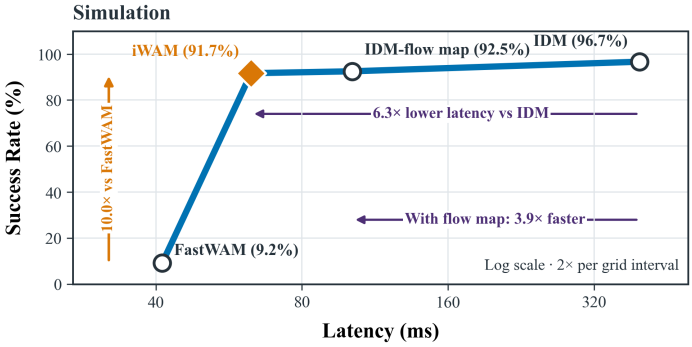


Figure 3: **Success–latency trade-off** on DROID simulation evaluations. Success is averaged across three tasks and replanning horizons of 16 and 24 steps.

We measure latency on a single NVIDIA A100 80 GB PCIe GPU with batch size one. To compare video-model costs under a common action-denoising budget, we time all methods using two action-denoising steps. Success rates come from the main evaluations, which use eight action-denoising steps for flow-based policies and two for ***i*WAM** (MIP); they are not recomputed under the common timing budget.

Why is *i*WAM both effective and efficient? Our results support the two requirements identified in Section 5: exposing future-specific information with a single model evaluation, and aligning that information with demonstrated actions during training. On LIBERO-PRO, ***i*WAM** (prior noise) matches IDM and improves over FastWAM (Table 3). This suggests that a single evaluation can provide useful predictive information even without explicitly matching the noise endpoint to the demonstration.

The DROID results show the limits of this approach. Prior-noise conditioning yields substantially higher action-training loss and negligible task success, suggesting that access to predictive information alone is insufficient. An independently sampled noise endpoint may correspond to a plausible future that requires different actions from those in the demonstration. The action expert would then receive supervision inconsistent with the future represented by its conditioning. We hypothesize that DROID's greater diversity and generalization make this mismatch more consequential than in LIBERO.

The bidirectional flow map addresses this mismatch during training by approximately recovering the noise endpoint associated with the demonstrated future. A single noise-to-clean evaluation at this endpoint supplies activations aligned with the action supervision, which should make the correspondence between conditioning and actions easier to learn. The reduction in training loss toward IDM's loss (Figure 2), together with the recovery in task success, supports this explanation. ***i*WAM**'s competitive performance suggests that these activations retain the future-specific information needed for action prediction without requiring an accurate reconstruction of the full future. Training with aligned conditioning thus lets the action expert use the predictive information available from a single evaluation to generalize and perform in more challenging settings.

7 Conclusion

We presented ***i*WAM**, which conditions an action policy on the intermediate activations of a single video-model forward pass instead of on an explicitly generated future. Across simulated and real-world manipulation, ***i*WAM** matches the performance of world action models that explicitly generate the future at a fraction of their inference cost, and substantially outperforms methods that condition only on the current observation. These results suggest that policies can benefit from a video model's ability to imagine the future without paying the cost of generating it.

AI USAGE DISCLOSURE

We used large language models to assist with writing and polishing the manuscript and to support literature retrieval and discovery, including identifying relevant related work. We also used LLMs as coding assistants. The authors take full responsibility for the final content of the paper.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38: 33346–33382, 2026.

Huayu Chen, Kai Jiang, Kaiwen Zheng, Jianfei Chen, Hang Su, and Jun Zhu. Visual generation without guidance. *arXiv preprint arXiv:2501.15420*, 2025a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026.

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, Oier Mees, Marc Pollefeys, Zhuang Liu, Jiajun Wu, Pieter Abbeel, Jitendra Malik, Yilun Du, and Jianfei Yang. World model for robot learning: A comprehensive survey. 2026.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346. PMLR, 2025.

Siqiao Huang, Jialong Wu, Qixing Zhou, Shangchen Miao, and Mingsheng Long. Vid2world: Crafting video diffusion models to interactive world models. *arXiv preprint arXiv:2505.14357*, 2025.

Arhan Jain. Droid sim evaluation, 2026. URL https://github.com/arhanjain/sim-evals.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted

Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Liangchen Li and Jiajun He. Bidirectional consistency models. *arXiv preprint arXiv:2403.18035*, 2024.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

Junbang Liang, Ruoshi Liu, Ege Ozguroglu, Sruthi Sudhakar, Achal Dave, Pavel Tokmakov, Shuran Song, and Carl Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation, 2024.

Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

NVIDIA. Isaac sim. URL https://github.com/isaac-sim/IsaacSim.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control. 2025. URL https://arxiv.org/abs/2512.01809.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Youpeng Wen, Junfan Lin, Yi Zhu, Jianhua Han, Hang Xu, Shen Zhao, and Xiaodan Liang. Vidman: Exploiting implicit dynamics from video diffusion model for effective robot manipulation. *Advances in Neural Information Processing Systems*, 37:41051–41075, 2024.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Linqi Zhou, Mathias Parger, Ayaan Haque, and Jiaming Song. Terminal velocity matching. In *International Conference on Learning Representations*, volume 2026, pp. 87506–87530, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

A PRELIMINARIES: SHORTCUT MODELS & GUIDANCE

Classifier-free guidance and guidance-free training. Conditional generation commonly uses classifier-free guidance (CFG) to strengthen adherence to the conditioning input y, providing explicit prompt guidance that can be important for test-time generalization. Let $v^{\tau}_{\theta_v}(z_\tau \mid \varnothing)$ denote the unconditional marginal velocity, where $\varnothing$ denotes dropped conditioning. At guidance strength $w \geq 1$, CFG forms the guided velocity

$$v^{\tau}_{\theta_v}(z_\tau \mid y, w) = w\, v^{\tau}_{\theta_v}(z_\tau \mid y) - (w-1)\, v^{\tau}_{\theta_v}(z_\tau \mid \varnothing). \tag{8}$$

Therefore, it requires separate conditional and unconditional model evaluations at each sampling step. Guidance-Free Training (GFT) (Chen et al., 2025a) instead trains a single network to directly predict the prompt-guided marginal velocity. Rearranging Eq. equation 8 gives

$$v^{\tau}_{\theta_v}(z_\tau \mid y) = \frac{1}{w}\, v^{\tau}_{\theta_v}(z_\tau \mid y, w) + \left(1 - \frac{1}{w}\right) v^{\tau}_{\theta_v}(z_\tau \mid \varnothing), \tag{9}$$

which allows the guided field to be trained using the original flow-matching target:

$$\mathcal{L}_{\mathrm{GFT}} = \mathbb{E}\left[\left\| \frac{1}{w}\, v^{\tau}_{\theta_v}(z_\tau \mid y, w) + \left(1 - \frac{1}{w}\right) \mathrm{sg}\big[v^{\tau}_{\theta_v}(z_\tau \mid \varnothing)\big] - u_\tau \right\|_2^2\right], \tag{10}$$

where $\mathrm{sg}[\cdot]$ denotes stop-gradient. Sampling w during training and providing it as an input allows a single network to predict guided velocities across guidance strengths. Importantly for our setting, GFT also produces internal activations corresponding directly to the guided velocity in a single forward pass, which we exploit in Section 5.3.

Shortcut models. Shortcut Models (Frans et al., 2025) learn to transport samples between distant flow times by conditioning a single network on flow time and step size, thereby supporting variable sampling budgets. They can be viewed as a progressive self-distillation scheme, where flow matching provides supervision for infinitesimal steps and larger steps are bootstrapped from smaller ones (Boffi et al., 2026). Existing shortcut model formulations, however, focus on noise-to-clean transport and do not provide the reverse clean-to-noise mapping needed to associate a clean sample with its noise endpoint. Moreover, Shortcut Models report that standard CFG becomes unreliable for large steps and requires the guidance scale to be specified during training. In our work, we will extend this formulation to arbitrary two-time transport and directly learned guided velocities, enabling bidirectional transport and arbitrary guidance strengths in the single-step regime.

Proposition A.1 (Marginal velocity at the noise endpoint)**.** *Let $c = (o_0, \ell)$ denote the conditioning context and consider the linear interpolant $z_\tau = (1-\tau)o_{1:N} + \tau\epsilon$, where $\epsilon \sim p_{\mathrm{prior}}$ is sampled independently of $(c, o_{1:N})$. Then, the marginal vector field at the noise endpoint $\tau = 1$ satisfies*

$$v^1(z_1 \mid o_0, \ell) = z_1 - \mathbb{E}[o_{1:N} \mid o_0, \ell]\,. \tag{11}$$

Consequently, a one-step denoising estimate obtained from the endpoint velocity is

$$z_1 - v^1(z_1 \mid c) = \mathbb{E}[o_{1:N} \mid c]\,, \tag{12}$$

which is independent of the sampled noise z_1.

Proof. For the linear interpolant, the conditional velocity is

$$u_\tau = \frac{dz_\tau}{d\tau} = \epsilon - o_{1:N}. \tag{13}$$

Hence, the marginal vector field is

$$v^\tau(z_\tau \mid c) = \mathbb{E}[\epsilon - o_{1:N} \mid z_\tau, c]\,. \tag{14}$$

At $\tau = 1$, $z_1 = \epsilon$. Since ϵ is independent of $(c, o_{1:N})$,

$$v^1(z_1 \mid c) = \mathbb{E}[z_1 - o_{1:N} \mid z_1, c] \tag{15}$$

$$= z_1 - \mathbb{E}[o_{1:N} \mid c]\,. \tag{16}$$

Rearranging gives Eq. equation 12. □

B Detailed Analysis on LIBERO-PRO

B.1 Evaluation on LIBERO-PRO.

As in Section 4, we train all models only on the original LIBERO demonstrations and evaluate them zero-shot, without further fine-tuning, on both the unperturbed tasks and the four LIBERO-PRO perturbation types defined in Table 3. For each suite and evaluation condition, we run 50 trials for each of the 10 tasks using the benchmark's fixed initial states, yielding 500 trials per table entry. The Average columns aggregate results across the four suites, corresponding to 2,000 trials per evaluation condition. Following Yuan et al. (2026), we use a batch size of 128 and train all models for 10 epochs. For two-stage variants, both the video model and the action policy are trained for 10 epochs each. At evaluation, we likewise adopt FastWAM's inference configuration and apply it identically across all methods.

The original FastWAM implementation (Yuan et al., 2026) jointly trains the action policy and the video model, allowing gradients from the action loss to update the pretrained video model. Due to the narrow action-training data distribution of LIBERO, we find that this can substantially degrade the generalization performance inherited from video pretraining. We adapt the two-stage training strategy of Liang et al. (2025) to FastWAM and IDM, where we first fine-tune the video model independently, then freeze its parameters while training the action policy. This substantially improves FastWAM and IDM's performance across perturbations. However, even with this training strategy, across perturbation categories (see Table 4), we find that FastWAM exhibits significantly worse generalization performance than IDM. In addition, we consider the simplified ***i*WAM** (prior noise) variant, which samples $z_1 \sim p_{\text{prior}}$ independently during training rather than using the noise endpoint associated with the demonstrated future, while using the same Stage 1 video backbone.

B.2 Performance on LIBERO-PRO.

As shown in Table 4, FastWAM generalizes worse than IDM across all perturbation categories. In particular, there is a significant gap under object-size perturbations on LIBERO-Goal, where IDM achieves 60.8% compared to 44.4% for FastWAM (Table 5); detailed task-level results are provided in Table 6. When the grasped object is enlarged, IDM's imagined future correctly captures the gripper interacting with the larger object, while FastWAM's action chunk does not appear to appropriately account for the increase in object size. This suggests that future-predictive information provides geometric cues important for generalization under distribution shift.

Table 4: **Performance under LIBERO-PRO perturbations.** Models are trained on LIBERO and evaluated under perturbations to object attributes (Obj.), initial object positions (Pos.), instruction phrasing while preserving task intent (Sem.), and instructions specifying a different target object or action (Task); None denotes the unperturbed setting. *joint* denotes joint training of the video model and action policy, whereas *disj* denotes the two-stage training procedure.

Approach	LIBERO-10					LIBERO-Goal					LIBERO-Object					LIBERO-Spatial					Average				
	None	Obj.	Pos.	Sem.	Task	None	Obj.	Pos.	Sem.	Task	None	Obj.	Pos.	Sem.	Task	None	Obj.	Pos.	Sem.	Task	None	Obj.	Pos.	Sem.	Task
IDM (joint)	96.4	66.6	0.4	95.4	16.2	97.4	79.0	8.2	97.4	9.0	98.6	92.4	0.0	95.0	**10.2**	**99.4**	96.6	2.2	97.4	**59.4**	97.9	83.7	2.7	96.3	23.7
FastWAM (joint)	94.6	51.6	0.0	80.8	10.6	96.6	71.2	3.0	93.4	8.6	99.4	93.4	0.0	99.2	9.4	97.0	95.8	0.6	61.4	32.6	96.9	78.0	0.9	83.7	15.3
IDM ($p = 0$)	97.2	64.8	6.6	96.4	**20.8**	98.4	79.0	11.4	98.0	**12.8**	99.2	**96.2**	**9.8**	97.8	10.0	98.4	98.2	10.8	96.6	53.0	98.3	84.5	9.7	97.2	**24.1**
IDM (disj)	97.2	**68.6**	4.2	97.2	20.0	98.0	79.8	17.2	**98.4**	10.6	**99.6**	94.0	8.2	98.6	10.0	**99.4**	97.0	14.2	97.6	53.8	98.5	84.9	10.9	98.0	23.6
FastWAM (disj)	96.0	57.0	4.8	92.2	15.6	97.4	73.4	11.8	95.8	10.4	99.4	91.0	3.2	**99.8**	10.0	97.8	96.6	**22.2**	91.0	45.4	97.6	79.5	10.5	94.7	20.4
***i*WAM (prior noise)**	**98.0**	64.4	**8.2**	**98.6**	19.2	**98.6**	**83.0**	**17.4**	98.0	11.0	99.4	94.8	8.0	99.6	10.0	**99.4**	**98.4**	15.6	**97.8**	53.8	**98.8**	**85.2**	**12.3**	**98.5**	23.5

Furthermore, FastWAM and several IDM variants jointly train the action expert and video model, allowing the action loss to backpropagate through the video model. Fine-tuning a pretrained video model in this manner may degrade its generalization, particularly when the action-training data are narrowly distributed. To investigate this hypothesis, we decouple the two training stages: we first fine-tune the video model independently, freeze its parameters, and then train the action expert using the action loss. We apply this disjoint training procedure to both FastWAM and IDM, denoted FastWAM (disj) and IDM (disj), respectively, with results reported in Table 4.

Disjoint training yields substantial gains under initial-state perturbations, improving IDM from 2.7% to 10.9% and FastWAM from 0.9% to 10.5%, while also improving performance under the other perturbation categories. Qualitative analysis of the predicted videos indicates that these gains stem

Table 5: Aggregate success rates under LIBERO-Goal object perturbations. Results are averaged over 5 tasks per perturbation type, with 50 trials per task. The resize setting combines a $1.5\times$ bowl size change with color/appearance changes.

Perturbation Type	**FastWAM-joint**	**IDM-joint**	**FastWAM-disj**	**IDM-disj**
Color / appearance only	**98.0**	97.2	93.2	90.4
Object resize ($1.5\times$ bowl) + color / appearance	44.4	60.8	53.6	**66.4**
Average	71.2	**79.0**	73.4	78.4

Table 6: Success rates under object perturbations on LIBERO-Goal. Object substitutions modify either appearance only or both appearance and object size. Higher values indicate better performance.

#	Task	Object Substitution	Size Change	FastWAM-joint	IDM-joint	FastWAM-disj	IDM-disj
0	open_the_middle_drawer_of_the_cabinet	wooden cabinet → yellow cabinet	No	**100.0**	96.0	**100.0**	82.0
1	put_the_bowl_on_the_stove	bowl → bigger bowl; stove → yellow stove	Bowl 1.5×	64.0	92.0	82.0	**96.0**
2	put_the_wine_bottle_on_top_of_the_cabinet	wine bottle → white bottle; cabinet → yellow cabinet	No	**96.0**	92.0	74.0	80.0
3	open_the_top_drawer_and_put_the_bowl_inside	cabinet → yellow cabinet; bowl → bigger bowl	Bowl 1.5×	10.0	18.0	14.0	**36.0**
4	put_the_bowl_on_top_of_the_cabinet	bowl → bigger bowl; cabinet → yellow cabinet	Bowl 1.5×	32.0	**46.0**	44.0	44.0
5	push_the_plate_to_the_front_of_the_stove	plate → yellow plate	No	98.0	**100.0**	**100.0**	96.0
6	put_the_cream_cheese_in_the_bowl	bowl → bigger bowl; cream cheese → red cream cheese	Bowl 1.5×	56.0	**78.0**	68.0	74.0
7	turn_on_the_stove	stove → yellow stove	No	**100.0**	**100.0**	98.0	98.0
8	put_the_bowl_on_the_plate	bowl → bigger bowl; plate → yellow plate	Bowl 1.5×	60.0	70.0	60.0	**82.0**
9	put_the_wine_bottle_on_the_rack	wine bottle → white bottle; wine rack → brown rack	No	96.0	**98.0**	94.0	96.0

from more accurate future predictions, particularly for initial-state perturbations, where the jointly trained IDM video model tends to overfit to the training-time initial object positions. FastWAM also benefits substantially under object-size and language perturbations, improving from 83.7% to 94.7% under language perturbations and from 44.4% to 53.6% on LIBERO-Goal object-resize perturbations.

To better understand why joint training hurts IDM, we examine the role of future-observation augmentation during action training. In the standard joint setting, the future observations provided to the action expert are randomly noised to a sampled noise level with probability 0.5. When we remove this augmentation ($p = 0$) and always provide clean futures, joint IDM recovers performance comparable to disjoint training across perturbations (Table 4). This suggests that, when clean futures are available, the action expert can fit the action objective with substantially less modification to the shared video-model representations, thereby better preserving the generalization inherited from pretraining despite action-loss backpropagation.

This observation also helps explain why FastWAM continues to benefit from disjoint training. Without future-derived conditioning, its action expert must infer the action sequence from the current observation and instruction alone. Optimizing the action objective therefore places greater pressure on the shared video-model representations, making them more susceptible to specialization toward the narrow action-training distribution. Freezing the video model prevents this adaptation and preserves more of its pretrained generalization, leading to the stronger robustness observed for FastWAM (disj).

Together, these results highlight the importance of preserving the predictive capabilities of the pretrained video model and, more critically, exposing the action expert to future-predictive information at test time for robust generalization.

C Detailed Experimental Setup of DROID

C.1 Training Details

Data and preprocessing. We train on DROID (Khazatsky et al., 2024) using the DreamZero release (Ye et al., 2026), which removes idle frames and unlabeled or unsuccessful episodes, yielding 57,774 episodes and 14.7M frames at 15 Hz. Each training example consists of the current observation, a 32-step action chunk ($\approx$2.1 s), and the corresponding future video. We subsample the future by $2\times$ in time to obtain 16 future frames; together with the current frame, this gives 17 frames, which the Wan2.2 VAE compresses into 5 latent frames. We use the wrist and first exterior camera, resize each view to 160×320, and concatenate them side by side before VAE encoding. Actions contain seven joint-position targets, expressed as deltas relative to the current joint state, and one absolute gripper command. We normalize actions to $[-1, 1]$ using the 1st and 99th percentiles, with separate statistics

for each action dimension and each of the 32 positions within the chunk. Proprioception consists of the current seven joint positions and gripper state and is provided only to the action expert; the video model receives no proprioceptive input. We apply temporally consistent color jitter following DreamZero (Ye et al., 2026), sample one language annotation per episode when multiple are available, and precompute instruction embeddings using the frozen UMT5 encoder from Wan2.2.

Training procedure. We initialize the video model from Wan2.2-TI2V-5B, keeping the VAE and text encoder frozen. Training proceeds in two stages. In Stage 1, we fine-tune the full video DiT for 100K steps with a global batch size of 256 ($\sim$1.7 epochs) using the bidirectional shortcut objective described in Section 5, together with Guidance-Free Training (Chen et al., 2025a). We use a maximum number of sampling steps $K_{\max} = 32$ for the shortcut training objective, sample the inverse-guidance parameter $\beta \sim \mathcal{U}(0, 1)$, and apply 10% conditioning dropout. Three quarters of each batch use the standard flow-matching loss, while the remaining quarter uses the Shortcut bootstrap loss, with noise-to-clean and clean-to-noise transport sampled equally. Bootstrap targets are constructed from two half-steps using a Polyak-EMA target network with update rate 10^{-3}. We additionally apply a round-trip consistency loss to one eighth of each batch, with its weight warmed up to 0.2 following the bidirectional training principle of Bidirectional Consistency Models (Li & He, 2024). Flow times are sampled using the shifted Wan2.2 schedule with shift factor 5, which biases training toward higher-noise flow times. When initialized without GFT, shortcut-style training incurs only about 16% additional cost over an equivalent flow model (Frans et al., 2025).

In Stage 2, we initialize the frozen video model from the Stage 1 EMA weights and train a freshly initialized action policy for 100K steps with a global batch size of 256. Unlike Stage 1, we do not use a shifted flow-time schedule for action training; instead, flow times are sampled uniformly from $[0, 1]$. The action policy uses 30 transformer layers, hidden dimension $d_a = 1024$, and FFN dimension 4096, totaling approximately 1B trainable parameters. Since the learned flow map also serves as a standard flow model, we reuse the same trained video model for both ***i*WAM** and IDM to ensure a controlled comparison. For both methods, we freeze the video model and train their respective action policies for 100K steps; during IDM evaluation, the shared video model is used with standard flow integration. We can also use the learned flow map to approximate the future in a single step during evaluation, which we denote as *IDM -flow map*.

For FastWAM, we instead train a separate video model and action policy jointly for 200K steps with a global batch size of 256. All flow-based action policies are trained with the flow-matching formulation. We additionally train a ***i*WAM** variant using the Minimal Iterative Policy (MIP) objective (Pan et al., 2025), which predicts actions in two denoising steps: an initial regression step followed by refinement; we denote this variant as ***i*WAM** (MIP). We also evaluated a two-stage FastWAM variant, but it underperformed the jointly trained version and is therefore omitted from the main results.

Optimization. Both stages use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-5}, gradient clipping at norm 1.0, and a global batch size of 256. For Stage 1 bidirectional Shortcut training, the learning rate is linearly warmed up for 5k steps to 10^{-5} and then kept constant. For Stage 2, the learning rate warms up for 5k steps to 10^{-4} and then follows cosine decay to 10^{-6}. Unless otherwise noted, Stage 2 models are trained for 100k steps; the FastWAM baseline is trained for 150k steps with the cosine schedule extended accordingly. Training uses bfloat16 computation with fp32 master weights and optimizer states.

C.2 Benchmarks Initialization

For DROID, we benchmark against IDM trained using the two-stage procedure, which uses 8 flow-denoising steps at inference to generate the future, requiring 9 video-backbone forward passes in total. We also evaluate a variant that uses the flow map learned by ***i*WAM** to approximate the future in a single step, which we denote as *IDM-flow map*, reducing action prediction to 2 video-backbone forward passes. For FastWAM, we train both joint and two-stage variants, but omit the latter from the main results due to its relatively low performance. For both simulation and real-world evaluations, we use a guidance strength of $w = 7$ for ***i*WAM** and both IDM variants; FastWAM does not have a corresponding guidance-strength parameter at inference. All flow-matching action policies use 8 action-denoising steps, and iWAM (MIP) uses 2 action-denoising steps. For simulation experiments,

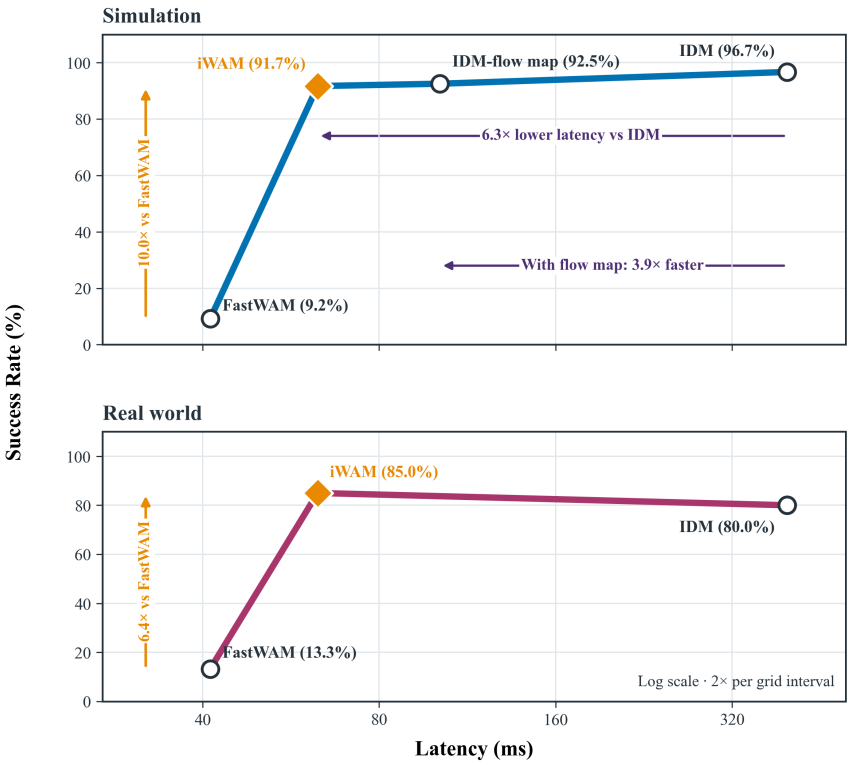


Figure 4: **Success–latency trade-off** on simulation (top) and real-world (bottom) evaluations, each using 20 trials per task. Simulation success is averaged across three tasks and replanning horizons of 16 and 24 steps; real-world success is averaged across three tasks.

we additionally evaluate the official DROID-finetuned $\pi_{0.5}$ checkpoint using its default 10-step action-denoising procedure.

Performance–Latency Pareto Frontier. We evaluate the performance–latency trade-off for DROID policies using ***i*WAM** (MIP), FastWAM, IDM-flow map, IDM, and $\pi_{0.5}$. We measure the latency on a single NVIDIA A100 80 GB PCIe GPU using batch size one, matching the setting used during policy evaluation. Since ***i*WAM** (MIP) uses two action-denoising evaluations at inference, we measure the latency of all methods under the same two-step action-denoising budget for a controlled comparison, so that latency differences arise from the video model rather than from action denoising. Their corresponding performance values are taken from the main evaluations, which use 8 action-denoising steps, and are not recomputed under the reduced two-step budget.

C.3 Evaluation Setup

We evaluate DROID-trained policies in both simulation and the real world. For simulation, we use the DROID benchmark (Jain, 2026), implemented in Isaac Lab (Mittal et al., 2025) on top of NVIDIA Isaac Sim (NVIDIA), and evaluate three pick-and-place tasks: "put the cube in the bowl," "put the can in the mug," and "put banana in the bin." Each rollout lasts 30 s (450 control steps), and we evaluate replanning horizons of 16 and 24 action steps with 20 trials per task and horizon and results are reported in Table 1. All methods that we train use an action chunk size of 32. For $\pi_{0.5}$, we instead use the official DROID-finetuned checkpoint with its native action chunk size of 15 during simulation evaluation.

For real-world evaluation, we consider three manipulation tasks: placing all three blocks into a bowl, picking a pen from the cup and placing it onto a table, and folding a towel. We fix the replanning horizon to 16 action steps and conduct 20 trials per task. Each trajectory is terminated after at most 750 control steps, i.e. 50s of executed actions at 15 Hz; time spent waiting for inference is not counted towards this budget.

Place All Blocks in Bowl. Three blocks are placed on the table together with a bowl (Figure 5), and the instruction is "put all the blocks in the bowl." A rollout is considered successful if all three blocks are placed in the bowl. We additionally report partial completion rates corresponding to placing at least one, at least two, or all three blocks.

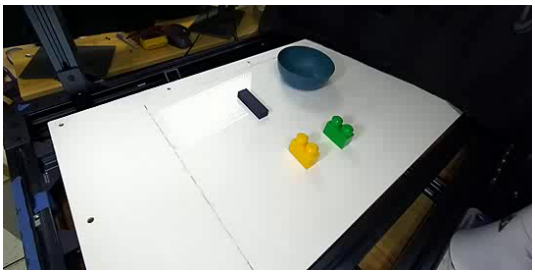
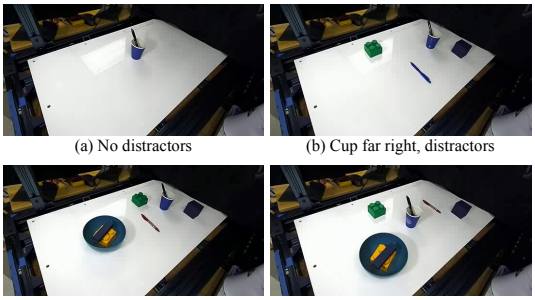


Figure 5: Initial configuration for the Blocks → Bowl task.

Pick Pen from Cup and Place on Table. A pen is initially placed inside a cup, and the instruction is "take the pen from the cup and place it on the table." We evaluate four initial configurations that vary the cup placement and the arrangement of distractor objects, including one configuration without distractors (Figure 6), with five rollouts per configuration for a total of 20 trials. A rollout is considered successful if the policy removes the pen from the cup and places it on the table.

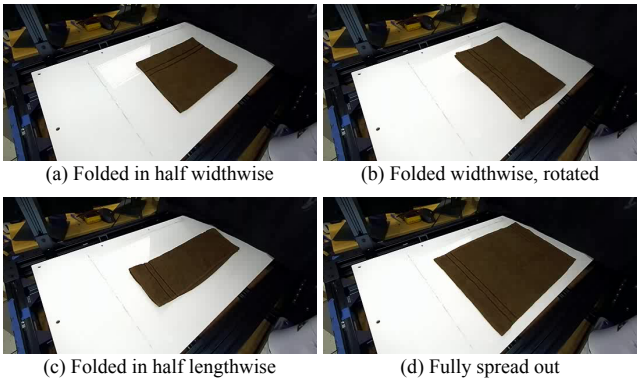


(a) No distractors (b) Cup far right, distractors

(c) Cluttered, cup far right (d) Cluttered, cup center

Figure 6: Initial configurations for the Pen → Table task.

Fold Towel. A towel is initially laid flat on the table, and the instruction is "fold the towel." We evaluate four initial configurations that vary the towel's fold state and orientation: folded in half widthwise, folded widthwise with a rotated orientation, folded in half lengthwise, and fully spread out (Figure 7). A rollout is considered successful if the robot lifts and folds the towel such that, in total, approximately 40% or more of the initially exposed towel area is flipped onto the opposite side.



(a) Folded in half widthwise (b) Folded widthwise, rotated

(c) Folded in half lengthwise (d) Fully spread out

Figure 7: Initial configurations for the towel-folding task.