WORLD MODEL SELF-DISTILLATION: TURNING VIDEO GENERATORS INTO TASK SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained video generators are promising visual world models for planning, as they can depict how actions change a scene, but solving tasks from high-level goals remains difficult without detailed execution descriptions. Common approaches outsource execution reasoning to language or vision-language models (VLMs), or require costly paired task-execution videos. We introduce World Model Self-Distillation (WMSD), combining self-distillation and reinforcement learning without curated task-video supervision. From an unlabeled scene image, a VLM generates a task and a step-by-step execution description. This description guides a frozen pretrained video generator, the *Demonstrator*; self-distillation transfers its behavior into an *Executor* conditioned only on the image and a short task prompt. VLM feedback further improves the Executor, exploiting the asymmetry between generating a solution and judging a sampled execution. Inference requires only the Executor. On *WorldTasks-Bench*, eight-step LTX-2 reaches 60.5% task completion, surpassing the Demonstrator's 49.5% and Base's 28.5% under VLM evaluation. Transfer to robotic video generation suggests generalization without robotics-specific fine-tuning. Independent and human evaluations support the task-solving gains. Verified maze solving shows how lightweight Demonstrator adaptation can enable transfer to new domains.

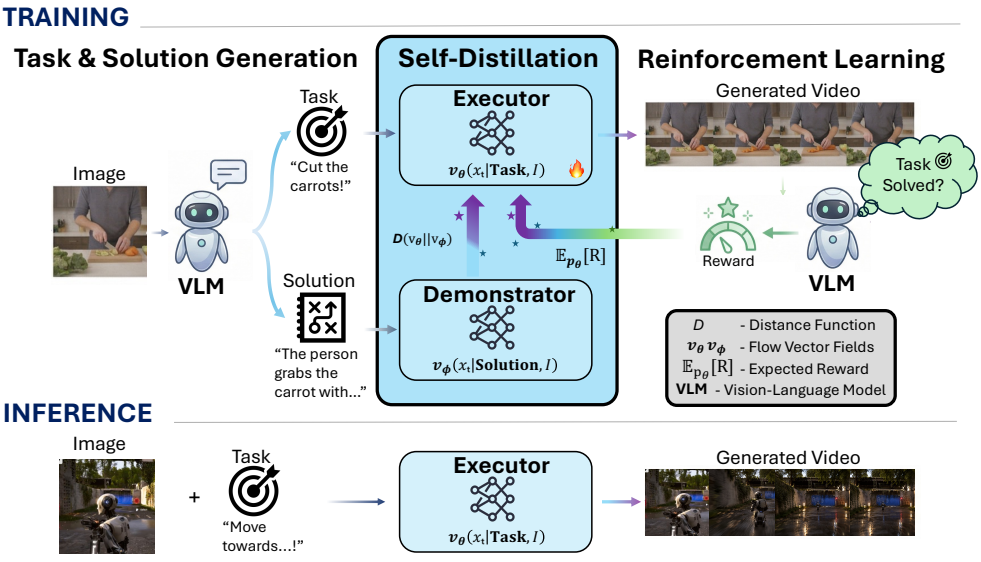


Figure 1: WMSD training and inference. Training (top): A VLM proposes a task and detailed execution description from a scene image. The frozen Demonstrator receives the image and description; the Executor receives only the image and short task. The Executor learns from Demonstrator denoising predictions and VLM rewards for task completion and consistency. **Inference (bottom):** Only the trained Executor is used, generating a video directly from the image and task without a Demonstrator or VLM call.

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.

1 INTRODUCTION

World models aim to help agents reason about their environment by simulating possible action sequences and selecting those that achieve a goal (Ha & Schmidhuber, 2018; Sutton, 1991; Hafner et al., 2019; Schrittwieser et al., 2020). Recent video generative models exhibit emergent capabilities that suggest a promising foundation for such reasoning (Guo et al., 2025; Wiedemer et al., 2025; Acuaviva et al., 2025; Bruce et al., 2024). However, pretrained text- and image-to-video generators typically rely on descriptions of the desired scene or action (Ho et al., 2022; Blattmann et al., 2023; He et al., 2025; Hassan et al., 2025; Hong et al., 2025; Stapf et al., 2026). For task solving, an external language or vision-language model (VLM) often supplies the execution steps. Ideally, the video model would accept an initial observation and a high-level instruction, then use its pretrained knowledge to generate a plausible execution directly.

Supervised fine-tuning offers a direct route: collect task instructions paired with successful execution videos and train the model to reproduce those trajectories. Yet broad task coverage requires diverse demonstrations across environments, objects, and levels of abstraction. Collecting these videos is costly. Large-scale video platforms and robot-learning datasets help, but scalable task supervision remains a challenge (NVIDIA et al., 2025; Chi et al., 2023; Zitkovich et al., 2023; O'Neill et al., 2024; Bjorck et al., 2025).

Reinforcement learning (RL) offers a complementary route: sample candidate executions, evaluate their outcomes, and increase the probability of successful generations. This approach has driven preference-based language-model training (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022) and is increasingly applied to diffusion and flow-based generators (Black et al., 2024; Fan et al., 2023; Wallace et al., 2024; Liu et al., 2025). However, obtaining reliable reward signals is challenging, and RL training can be unstable.

To solve these issues, we use self-distillation to transfer execution knowledge into a task-conditioned video model. The *Executor* receives an initial image and a high-level instruction, such as "cut the carrots," while a *Demonstrator* initialized from the same backbone receives a detailed execution description. The frozen Demonstrator guides denoising on states derived from the Executor's own generations, teaching it to map instructions to action sequences. A VLM supplies task–description pairs from scene images, removing the need to collect paired task-execution videos. At inference, only the Executor is needed, with no latency overhead relative to the base video model.

Self-distillation transfers the Demonstrator's behavior (Hinton et al., 2015; Furlanello et al., 2018); outcome rewards allow the Executor to improve task completion beyond imitation alone. We therefore evaluate generated rollouts with a VLM and combine its feedback with teacher guidance. This uses a generation–verification asymmetry: finding a solution can be harder than checking a proposed one (Song et al., 2025). VLMs provide a practical verifier for language-specified visual tasks (Rocamonde et al., 2023; Wang et al., 2024; Jiang et al., 2025). Their judgments can be noisy, especially for ambiguous visual events (Amodei et al., 2016; Huang et al., 2024; Bansal et al., 2025), so we combine task and quality feedback with a teacher anchor. Together, these signals encourage successful, coherent execution while retaining the Demonstrator's guidance. We call this approach *World Model Self-Distillation* (WMSD; Fig. 1).

Our **main contributions** are:

1. A self-distillation method that turns pretrained video generators into instruction-conditioned task solvers without paired task-execution videos.
2. RL with VLM feedback that improves task completion beyond the Demonstrator under our benchmark evaluation and yields transfer competitive with task-specific supervised fine-tuning.
3. *WorldTasks*, a dataset of tasks and detailed execution descriptions synthesized from unlabeled scene images, and *WorldTasks-Bench*, a benchmark for evaluating general task solving in generated videos.

2 Related Work

Prior work shows how video models can generate future events and how teacher models can guide training. We build on these ideas to train a video model that needs an image and task, but no execution description, at inference.

Video models and task interfaces. Video generators simulate visual futures (Ho et al., 2022; Blattmann et al., 2023; Bruce et al., 2024; NVIDIA et al., 2025), and planning systems often use language or vision-language models to guide them (Du et al., 2023; 2024; Pan et al., 2024). Robotics methods additionally learn from demonstrations (Chi et al., 2023; Zitkovich et al., 2023; O'Neill et al., 2024). WMSD uses execution descriptions during training so that a language model does not need to write them for every inference request.

Distillation and reward optimization. Knowledge and on-policy distillation transfer teacher predictions, including on student-generated states (Hinton et al., 2015; Furlanello et al., 2018; Agarwal et al., 2024). DMD and consistency methods target efficient generation (Yin et al., 2024; Song et al., 2023; Yin et al., 2025); DMDR (Jiang et al., 2025) combines DMD with reward optimization and reports gains over its teacher. Our contribution is to extract task-solving ability from pretrained video models through privileged description-to-task transfer: the Demonstrator sees execution details, while the Executor uses only the image and task. We transfer this guidance on states derived from Executor-generated videos, using the same teacher-flow model mismatch as a detached reward and a regression anchor; AWM and FlowGRPO supply the optimization algorithms (Xue et al., 2025; Liu et al., 2025). An unsuccessful DMD alternative is documented in Appendix D.1. VLMs can score many generated videos without human annotation (Rocamonde et al., 2023; Wang et al., 2024). Because these scores can be wrong, we also use an independent model, human judgments, and a fixed Maze verifier to check the results.

3 World Model Self-Distillation

For scene image $\mathcal{I}$, task $\mathcal{T}$, and execution description $\mathcal{D}$, the Executor uses $c_{\mathrm{E}} = (\mathcal{I}, \mathcal{T})$ and the frozen Demonstrator uses $c_{\mathrm{D}} = (\mathcal{I}, \mathcal{D})$. We want the Executor to learn the intermediate actions supplied by the description while still being rewarded for completing the task. Each update therefore samples Executor videos, obtains Demonstrator predictions on noisy states derived from those videos, and combines prediction matching with task and quality rewards.

Sample from the Executor; query the Demonstrator. The Executor needs guidance on the videos it currently generates, including its mistakes. We therefore sample G videos from the Executor using only the image and task, then ask the Demonstrator for predictions on noisy states derived from those videos. Let $\tau = (x_{t_0}, \ldots, x_{t_K})$ denote the latent *sampler path* with K denoising steps. Flow time $t \in [0, 1]$ indexes denoising, not the video frames in Y. The deterministic inference sampler discretizes

$$\dot{x}_t = v_\theta(x_t, t \mid c_{\mathrm{E}}), \qquad x_0 \sim \mathcal{N}(0, I). \tag{1}$$

Let z_t denote a noisy latent at time t sampled or reconstructed according to the RL algorithm (Appendix E.2). At the same z_t, the Demonstrator's solution-conditioned field targets the Executor's task-conditioned prediction. For M regression timesteps sampled per video, define

$$\ell_\theta(z_t, t) = \|v_\theta(z_t, t \mid c_{\mathrm{E}}) - v_{\theta'}(z_t, t \mid c_{\mathrm{D}})\|^2, \qquad \widehat{C}_\theta(Y) = \frac{1}{M} \sum_{m=1}^{M} \ell_\theta(z_{t_m}, t_m). \tag{2}$$

Squared errors are also averaged over latent elements. Let μ_θ denote this distribution of noisy states derived from Executor outputs. Its two teacher-guidance terms are

$$r_{\mathrm{distill}} = -\,\mathrm{sg}[\widehat{C}_\theta(Y)], \qquad \mathcal{L}_{\mathrm{anchor}} = \mathbb{E}_{(z_t,t)\sim\mu_\theta} \ell_\theta(\mathrm{sg}[z_t], t). \tag{3}$$

The detached reward favors teacher-aligned videos; the anchor regresses the Executor field to the Demonstrator on fixed states. Only Executor parameters change. On-policy refers to videos sampled from the current Executor.

Algorithm 1 WMSD with a frozen, privileged Demonstrator

Require: Precomputed triples $(\mathcal{I}, \mathcal{T}, \mathcal{D})$; Executor θ; frozen Demonstrator θ'

1: **for** each update **do**
2: Sample triples and G Executor paths per task using $c_{\mathrm{E}} = (\mathcal{I}, \mathcal{T})$
3: Decode videos; evaluate task, consistency, and quality rewards
4: Obtain noisy latents z_t; query Demonstrator fields using $c_{\mathrm{D}} = (\mathcal{I}, \mathcal{D})$
5: Compute detached distillation rewards and normalized advantages (Eqs. 3 and 7)
6: Update θ with the optimizer surrogate and fixed-state anchor (Eq. 4)
7: **end for**

Compute task and quality feedback. Matching the Demonstrator does not by itself check whether a video completes the task or remains visually consistent. We normalize the task-completion, consistency, visual-quality, and detached distillation rewards within each task group and combine them as a weighted sum. Standardizing and clipping this sum gives per-video advantages A_i for the optimizer. Section 4 describes the reward models; Appendix A.2 gives the weights and normalization details.

Update the Executor. We combine reward optimization with Demonstrator anchoring:

$$\mathcal{L}_{\mathrm{final}} = \mathcal{L}_{\mathrm{RL}}(\theta; \mathrm{sg}[A]) + \beta_d \mathcal{L}_{\mathrm{anchor}}. \tag{4}$$

Here $\mathcal{L}_{\mathrm{RL}}$ can be instantiated with FlowGRPO (Liu et al., 2025), AWM (Xue et al., 2025), or NFT (Zheng et al., 2025), using detached advantages $\mathrm{sg}[A]$; Appendix E.2 describes these choices. The RL term favors successful videos, while the anchor preserves the Demonstrator's execution guidance. Algorithm 1 summarizes training for image-and-task inference.

Complementary roles of reward and anchor. Updating the Executor changes both sampled trajectories and fixed-trajectory prediction errors. A stochastic identity separates these effects: for the idealized path cost $C_\theta(\tau) = \int_0^1 \ell_\theta(x_t, t)\, dt$ and a finite-step *stochastic* sampler with differentiable densities on common support and integrable derivatives, the expected mismatch obeys

$$\nabla_\theta \mathbb{E}_{p_\theta}[C_\theta(\tau)] = \mathbb{E}_{p_\theta}[C_\theta(\tau) \nabla_\theta \log p_\theta(\tau)] + \mathbb{E}_{p_\theta}[\partial_\theta C_\theta(\tau)]. \tag{5}$$

The score-function term motivates rewarding trajectories with lower mismatch; the fixed-path partial derivative motivates regression on detached states. Appendix E presents additional mathematical results and their proofs.

4 Setting and Training Data

At inference, the Executor generates a video $Y = (y_1, \ldots, y_F)$ from the scene image and task alone; privileged execution descriptions are used only during training. We use *world model* for this conditional video generator.

Synthetic supervision. Training requires an image, a task for an agent in that image, and a description of how to carry out the task. We construct *WorldTasks* to supply these examples without collecting videos of people performing each task. From approximately 20,000 scene images, largely from MiraData (Ju et al., 2024), quality and suitability filters retain 18,305 image IDs with eight Qwen3.5-27B-generated task–description pairs each (146,440 pairs). Further prompt filtering yields 8,430 images and 67,440 pairs, a distinct pipeline stage. Instructions identify the actor through `[Agent]: Task`. We also reward task completion and video consistency, so that training does not rely only on matching imperfect descriptions; Appendix A.4 documents filtering and composition.

Reward configuration. For LTX-2, we use Qwen3.5-27B-FP8 to score task completion and video consistency from the image, task, and generated video. The corresponding reward prompts are given in Appendix A.4.2 (Figs. 8 and 9). Each reward is a yes/no log-probability difference, $r_j = \log p(\text{yes} \mid Y, \mathcal{I}, \mathcal{T}; j) - \log p(\text{no} \mid Y, \mathcal{I}, \mathcal{T}; j)$. PickScore (Kirstain et al., 2023) adds a per frame visual-quality signal which is averaged per video. Appendix A.2 gives the reward weights and advantage normalization.

Evaluation boundary and task-defining constraints. We want to evaluate on image–task pairs that were not used for training. *WorldTasks-Bench* contains 200 such pairs with distinct image IDs. An audit finds no benchmark image ID or exact full prompt in either training manifest, establishing disjointness at those levels. Appendix A.1 gives the evaluation details; the separate 200-example ablation set is analyzed independently. The task prompt $\mathcal{T}$ must retain the object identities, exclusions, action order, and spatial constraints needed to specify the requested outcome. The description $\mathcal{D}$ supplies intermediate steps that can be inferred from the image and task.

5 Main Results

We want to know whether a video model can learn from detailed execution descriptions during training and then complete more tasks without those descriptions at inference. **Base** is our unadapted pretrained eight-step distilled LTX-2 video model conditioned on the scene image and task. **Demonstrator** uses the same pretrained model with identical weights, conditioned on the scene image and a VLM-written execution description. **WMSD** is the Executor obtained by adapting Base with our training procedure; it uses the scene image and task at inference. We evaluate all three with eight sampling steps on the 200-task *WorldTasks-Bench*. We measure **whether the generated video completes the task**, whether the correct agent acts, and whether the execution is physically and temporally consistent. An independent model and human participants also judge the videos so that the evidence extends beyond the model used to supply training rewards (Appendices A.1 and C.1).

Table 1: Eight-step LTX-2: the instruction/description interface. Task completion is primary. Panel A is the archived benchmark export (valid-case denominators); panel B judges a set of videos fixed before independent evaluation. Each panel compares methods on its own set of videos, using eight sampling steps. Best values are bold and second-best values are underlined, separately within each panel.

Condition	Inference input	**Task**	Agent	Consistency
A. Benchmark evaluator: Qwen3.5-27B-FP8				
Base	Image + task	0.285	0.391	0.694
Demonstrator (+VLM)	Image + VLM description	<u>0.495</u>	<u>0.572</u>	<u>0.732</u>
WMSD	Image + task	**0.605**	**0.691**	**0.882**
B. Independent evaluator: GPT-5.6 Luna; $N = 200$ per method				
Base	Image + task	0.345	0.505	0.585
Demonstrator	Image + VLM description	<u>0.535</u>	<u>0.670</u>	<u>0.735</u>
WMSD	Image + task	**0.580**	**0.725**	**0.770**

Benchmark task completion rises from 28.5% for Base to 49.5% with a VLM description and 60.5% for WMSD using only the task. Agent and consistency scores also improve. Independent judging confirms the gain over Base: WMSD reaches 58.0% versus 34.5% for Base and 53.5% for the Demonstrator. WMSD has higher observed scores on all three independently judged criteria; the paired task analysis examines retained Demonstrator successes and additional rescues (Fig. 2b; Appendix B.3).

Gains across task families. We want to know whether WMSD helps with several kinds of tasks or whether most of the improvement comes from one kind. We therefore compare Base, Demonstrator, and WMSD task completion separately for each task category (detailed in Appendix B.2) using the independent evaluator. *Object manipulation improves from 6/34 for Base to 19/34 for WMSD ($+38.2$ points), navigation/traversal from 27/51 to 38/51 ($+21.6$ points), and spatial positioning from 7/44 to 17/44.* WMSD and the Demonstrator each complete 19/34 manipulation tasks; their perception/state counts are 21/33 and 22/33. Appendix Table 13 gives the full breakdown.

Human judgments favor WMSD. We want to check whether people also judge WMSD videos as completing the requested tasks more successfully. Participants compare videos generated for the same task without knowing which method produced them, and rate task completion, the acting agent, physical plausibility, and overall preference. Task-completion net preference favors WMSD by 47.0 points over Base (117 ratings) and 10.3 over the Demonstrator (116 ratings); overall preference is $+38.1$ and $+20.2$ points (105 and 104 ratings). *Participants favor WMSD across all four criteria (Table 2).* Appendix C.1 provides further details on the human evaluation protocol.

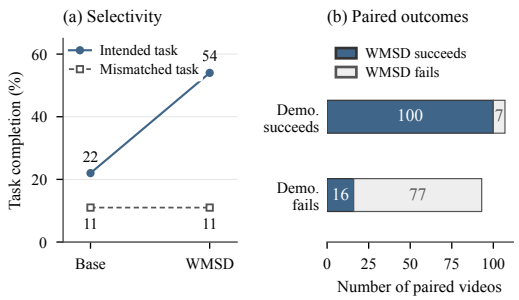


Figure 2: Selectivity and paired outcomes. (a) Intended versus mismatched tasks (100 videos per method). (b) Updated paired task evaluation: 100 retained successes, 16 rescued failures, and seven lost successes across 200 pairs (Appendix B.3).

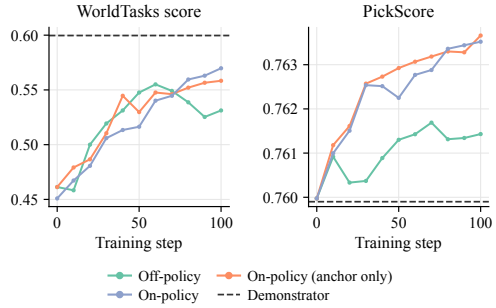


Figure 3: Distillation-only guidance. WorldTasks score and PickScore on the 200-example ablation set. On-policy variants use Executor states; off-policy uses Demonstrator states. No external rewards.

Table 2: Blinded human preferences. Net preference (percentage points) is wins minus losses divided by all available ratings, including ties and neither. N counts available ratings for each comparison and criterion.

	WMSD vs. Base		WMSD vs. Demonstrator	
Criterion	N	Net pref.	N	Net pref.
Task completion	117	+47.0	116	+10.3
Correct agent	117	+24.8	115	+8.7
Physical plausibility	105	+18.1	104	+16.3
Overall preference	105	+38.1	104	+20.2

Removing the planning call. *Once trained, WMSD has essentially unchanged measured sampling latency and avoids the Demonstrator's external VLM planning call.* Appendix A.6 reports the timing measurements and protocols.

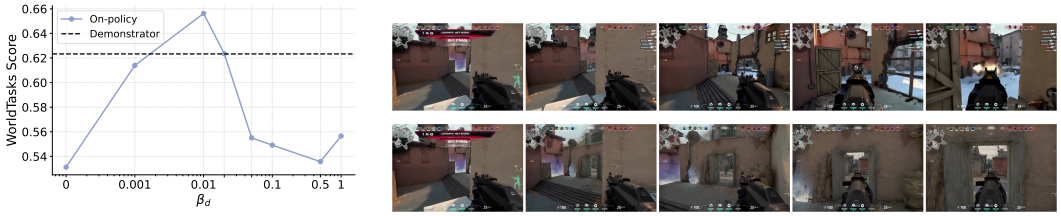
6 Instruction Selectivity and Mechanism

Higher task-completion scores leave several questions open. We first check whether WMSD follows the particular task it was given, then test which training components help and whether the model can still follow detailed prompts.

Improvement follows the intended instruction. We want to rule out the possibility that WMSD simply generates videos that receive positive judgments regardless of the instruction. For each scene, we generate videos for two different tasks and judge every video twice: once against the task used to generate it and once against the other task from that scene. *Intended-task success rises from 0.220 for Base to 0.540 for WMSD, while mismatched-task success remains 0.110 (Fig. 2a).* The intended-minus-mismatched gap grows from 0.110 to 0.430, showing that WMSD improves completion of the requested task without increasing success judgments for the mismatched task.

Retaining Demonstrator successes. *WMSD retains 100 of 107 Demonstrator task successes (93.5%) and rescues 16 of its 93 failures (17.2%; Fig. 2b).* Seven Demonstrator successes are lost, yielding an observed net gain of nine tasks (+4.5 percentage points). This pattern shows high retention with some additional rescues; the net advantage does not reach the 5% significance threshold (two-sided exact McNemar $p = 0.093$; Appendix B.3, Table 14).

Guidance on Executor states. We want to know whether the Executor learns better when the Demonstrator guides it on states derived from its own generated videos. We compare matching on Demonstrator-generated videos with matching on Executor-generated videos, and test whether adding the distillation reward helps further. We evaluate these variants on the separate 200-example ablation set. None uses task, consistency, or PickScore rewards (Table 3). Both forms of matching improve the reported scores. *The variants using Executor-generated states continue improving after the variant*



(a) Anchor-strength sensitivity.

(b) Consistency reward (top) versus no consistency reward (bottom).

Figure 4: Regularization diagnostics. (a) The archived anchor sweep favors a small nonzero coefficient. (b) For *Aim the weapon at the doorway entrance*, removing the consistency reward permits an implausible doorway appearance. The panels examine teacher anchoring and the consistency reward, respectively.

using Demonstrator-generated states levels off (Fig. 3). This supports using the Executor's own generations to decide where teacher guidance is needed. The *On-policy* curve includes the distillation reward; full WMSD additionally uses task and quality rewards.

Table 3: What each ablation activates. External denotes task/quality rewards. All teachers in these principal variants are frozen. Exploratory teacher updates are specified in Appendix D.2.

Variant	Distillation states	External	Distill. reward	Anchor
Off-policy	Demonstrator	No	No	Regression
On-policy anchor only	Executor	No	No	Yes
On-policy distillation	Executor	No	Yes	Yes
RL only	—	Yes	No	No
WMSD	Executor	Yes	Yes	Yes

Outcome rewards complement teacher guidance. We want to know whether rewarding successful, visually coherent videos improves task completion beyond matching the Demonstrator alone. We therefore compare exploratory runs that add task/PickScore rewards, using different sources of teacher guidance and different teacher-update schedules (Appendix D.2). *Task completion improves beyond distillation alone, and guided RL continues improving after RL-only gains flatten.* This supports combining teacher guidance with task and quality rewards. We also vary the anchor strength to check how strongly the Executor should be constrained to match the Demonstrator. This sweep favors a small nonzero coefficient near 0.01. Separately, videos generated with the consistency reward show fewer implausible appearances and disappearances (Appendix Fig. 17).

Longer, multi-step tasks. We test Base and WMSD on 50 additional examples whose detailed descriptions contain multiple action clauses, using short and detailed prompts with matched images and generation seeds. Besides the average benchmark score, we measure atomic-step coverage: the fraction of requested actions visibly completed. *With identical detailed prompts, WMSD exceeds Base on average score and atomic-step coverage (Table 4).* Within WMSD, detailed prompting raises coverage from 0.5076 to 0.6224 ($+0.1148$), while the average task/agent/consistency score is 0.700 versus 0.793 with short prompts. These results provide a proof of concept for longer instructions; short prompts must still preserve essential task constraints (Appendix C.1.1).

Table 4: Detailed-prompt retention diagnostic. Fifty additional examples with matched images and generation seeds. Average score is the mean of task, agent, and consistency rates. Atomic-step coverage judges individual actions in each detailed description.

Condition	**Average score**	**Atomic-step coverage**
Base, short prompt	0.553	0.4282
Base, detailed prompt	0.647	<u>0.5708</u>
WMSD, short prompt	**0.793**	0.5076
WMSD, detailed prompt	<u>0.700</u>	**0.6224**

7 TRANSFER AND OPERATING CONDITIONS

We next ask whether the improvements also occur with other video-model configurations and on robotic tasks that were not used for fine-tuning. Finally, maze-solving tasks let us check success with a verifiable reward and test adaptation of the Demonstrator to a new task domain.

Gains across model configurations. We want to check whether the improvement is specific to the eight-step LTX-2 setup. We therefore apply WMSD to HY1.5 with FlowGRPO and to full-step LTX-2 (Table 5). *Task completion rises from 0.464 to 0.574 for HY1.5 (+11.0 points) and from 0.315 to 0.452 for full-step LTX-2 (+13.7).* Main eight-step WMSD also exceeds the Demonstrator's benchmark rates: 0.605 versus 0.495 for task completion, 0.691 versus 0.572 for agent correctness, and 0.882 versus 0.732 for consistency. For the LTX-2 SFT baseline, we use videos from the pool used to construct WorldTasks, automatically annotate the agent and task shown, and fine-tune on these video–annotation pairs (Appendix A.2). *Task completion is 0.292 versus 0.315 before fine-tuning.* This lack of improvement may reflect the limited task-directed action in many routine internet videos. Task completion improves in both additional configurations; the HY1.5 comparison changes the backbone, optimizer, and reward model together. Appendix A.3 reports training compute.

Table 5: WorldTasks results across backbones and sampling budgets. Benchmark-evaluator success rates. The main eight-step LTX-2 results match panel A of Table 1; additional settings use HY1.5/FlowGRPO and full-step LTX-2. Best values are bold within each configuration.

Configuration	**Task**	Agent	Consistency
LTX-2 (8-step)	0.285	0.391	0.694
LTX-2 + WMSD (8-step)	**0.605**	**0.691**	**0.882**
HY1.5	0.464	0.540	0.780
HY1.5 + WMSD (FlowGRPO)	**0.574**	**0.630**	**0.828**
LTX-2 (full-step)	0.315	0.395	0.690
LTX-2 + automatically annotated-video SFT	0.292	0.389	0.682
LTX-2 + WMSD (full-step)	**0.452**	**0.500**	**0.693**

Transfer to robotic-task videos. We want to know whether learning from WorldTasks also helps the model generate videos of robots completing tasks. We evaluate the same WorldTasks-trained LTX-2 model on DreamGen (Jang et al., 2025), without any robotics-specific fine-tuning. *WMSD improves Object, Behavior, and Environment scores by 50.0, 27.6, and 17.2 percentage points over unadapted LTX-2, respectively, and remains competitive with robotics-SFT models (Table 6).* Supervised models lead on Behavior and Environment. LTX-2 improves on all three criteria despite receiving no robotics-specific fine-tuning; Fig. 5 illustrates a bottle-manipulation rollout. DreamGen reports a positive correlation between benchmark scores and robot-policy success (Jang et al., 2025), suggesting that WMSD-generated videos may support robot-policy training.

Table 6: DreamGen robotic-task video generation. Scores in percent; higher is better. WMSD transfers WorldTasks training to LTX-2 without robotics-specific fine-tuning. Best scores are bold; second-best are underlined.

Model and training	Object	Behavior	Environment
Zero-shot video models			
HunyuanVideo	0.0	2.1	0.0
Wan	2.0	2.1	6.7
Cosmos	32.0	31.9	24.1
LTX-2	20.0	29.8	41.4
Robotics-specific supervised fine-tuning			
HunyuanVideo	26.0	10.6	27.6
Wan	58.0	55.3	**65.5**
Cosmos	<u>62.0</u>	**61.7**	**65.5**
LTX-2 + WMSD (WorldTasks only)	**70.0**	<u>57.4</u>	<u>58.6</u>

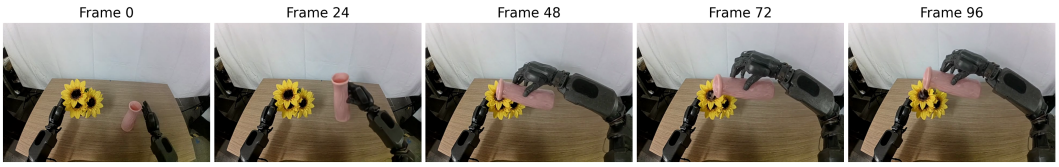


Figure 5: Transfer to robotic-task video generation. WorldTasks-trained WMSD with LTX-2, without robotics-specific fine-tuning. Task: *Use the right hand to pick up the pink bottle and pour water on the flower.* The strip shows the approach, grasp, and bottle tilt.

Adapting to different reward signals. We test whether WMSD improves task success with both VLM feedback and verifiable rewards, and whether the Demonstrator must first learn to use its solution description. In the Maze task from VideoRLVR (Zhu et al., 2026), the model must generate a video drawing a route from the start to the target without changing the maze walls. We compare rule-based and VLM training rewards, and compare WMSD teachers with and without training to follow the supplied route. Only the Demonstrator receives the exact route; Appendix B.4 gives the training, adaptation, and verification details. *The lightweight Demonstrator adaptation enables WMSD to raise rule-reward success from 0.700 to 0.750 and F1 from 0.8386 to 0.8663 over RL only (Table 7).* These results show that adapting the Demonstrator makes its solution guidance effective in this new domain.

Table 7: Maze adaptation and teacher controls. All results use the same deterministic verifier. Rule/VLM identify training rewards. Additional teacher and VLM-reward controls are reported in Appendix B.4.

Condition	Reward	Success	F1
RL only	Rule	0.700	0.8386
RL only	VLM	0.700	0.8377
WMSD, unadapted teacher	Rule	0.580	0.7048
WMSD, unadapted teacher	VLM	0.560	0.7151
WMSD, trained Demonstrator	Rule	**0.750**	**0.8663**

8 Discussion & Limitations

Generalizability. Training with *WMSD* leads to substantial improvements on *WorldTasks-Bench* and transfers to robotics-related tasks without robotics-specific fine-tuning (Sec. 7). On DreamGen, the resulting performance is competitive with supervised fine-tuning, while the backbone comparisons show gains for both HY1.5 and LTX-2 (Table 5). Few-step video distillation also provides an efficient basis for RL-based optimization: our main LTX-2 results use only eight denoising steps per generation. WMSD thus combines the sampling efficiency of distilled generators with substantial improvements in task completion.

Limits of training without target-domain videos. WMSD learns from synthetic task–description pairs without requiring curated execution videos for WorldTasks training or robotics-specific demonstrations for DreamGen transfer. This enables adaptation through execution guidance and outcome feedback. However, plausible task solutions can still contain errors in the appearance and motion of a particular robotic platform (Fig. 5). The initial frame specifies the visible scene, while subsequent motion depends on the pretrained generator's knowledge. Additional platform-specific context may help preserve these details. Extending WMSD to video continuation and in-context learning could supply motion history or reference demonstrations to guide generation.

Out-of-distribution tasks. The Maze study shows that WMSD can extend to a new domain with a small adaptation of the Demonstrator. Solution-conditioned maze generation is out of distribution for the unadapted teacher, but five epochs of rank-32 LoRA fine-tuning provide effective execution guidance. With this fixed, adapted Demonstrator, WMSD reaches 0.750 success and 0.8663 F1, improving on RL only at 0.700 and 0.8386 (Table 7). The Executor uses only the maze image and task at inference. These results show that modest Demonstrator adaptation can extend WMSD to out of distribution data.

AI USE STATEMENT

Generative AI tools assisted with experiment implementation, data annotation, learned reward and evaluation models, and manuscript drafting and editing. The method and evaluation sections identify their experimental roles. The authors are responsible for verifying the implementation, analysis, and final claims.

ETHICS STATEMENT

WMSD studies task solving in generated videos, not validated real-world execution. Potential risks include inherited model biases and misleading or harmful video generation. Task-completion and consistency rewards do not establish instruction safety. Our robotic-video results do not establish safe robot control; deployment would require separate safety validation.

REFERENCES

Pablo Acuaviva, Aram Davtyan, Mariam Hassan, Sebastian Stapf, Ahmad Rahimi, Alexandre Alahi, and Paolo Favaro. From generation to generalization: Emergent few-shot learning in video diffusion models, 2025. URL https://arxiv.org/abs/2506.07280.

R Agarwal, N Vieillard, Y Zhou, and P Stanczyk. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

D Amodei, C Olah, J Steinhardt, and P Christiano. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. URL https://arxiv.org/abs/1606.06565.

Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. Videophy: Evaluating physical commonsense for video generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9D2QvO1uWj.

J Bjorck, F Castañeda, N Cherniadev, X Da, and R Ding. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

K Black, M Janner, Y Du, I Kostrikov, and S Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

A Blattmann, T Dockhorn, S Kulal, and D Mendelevitch. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. URL https://arxiv.org/abs/2311.15127.

J Bruce, MD Dennis, A Edwards, J Parker-Holder, and Y Shi. Genie: Generative interactive environments. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=bJbSbJskOS.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023.

PF Christiano, J Leike, T Brown, M Martic, and S Legg. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Y Du, S Yang, P Florence, F Xia, A Wahid, and P Sermanet. Video language planning. In *International Conference on Learning Representations*, 2024.

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. In *NeurIPS*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1d5b9233ad716a43be5c0d3023cb82d0-Abstract-Conference.html.

Y Fan, O Watkins, Y Du, H Liu, M Ryu, and C Boutilier. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

T Furlanello, Z Lipton, M Tschannen, and L Itti. Born again neural networks. In *Proceedings of Machine Learning Research*, 2018.

Ziyu Guo, Xinyan Chen, Renrui Zhang, Ruichuan An, Yu Qi, Dongzhi Jiang, Xiangtai Li, Manyuan Zhang, Hongsheng Li, and Pheng-Ann Heng. Are video models ready as zero-shot reasoners? an empirical study with the mme-cof benchmark. *ArXiv*, abs/2510.26802, 2025.

D Ha and J Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. URL https://arxiv.org/abs/1803.10122.

D Hafner, T Lillicrap, J Ba, and M Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. URL https://arxiv.org/abs/1912.01603.

Mariam Hassan, Sebastian Stapf, Ahmad Rahimi, Pedro M. B. Rezende, Yasaman Haghighi, David Brüggemann, Isinsu Katircioglu, Lin Zhang, Xiaoran Chen, Suman Saha, Marco Cannici, Elie Aljalbout, Botao Ye, Xi Wang, Aram Davtyan, Mathieu Salzmann, Davide Scaramuzza, Marc Pollefeys, Paolo Favaro, and Alexandre Alahi. Gem: A generalizable ego-vision multimodal world model for fine-grained ego-motion, object dynamics, and scene composition control. In *CVPR*, pp. 22404–22415, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Hassan_GEM_A_Generalizable_Ego-Vision_Multimodal_World_Model_for_Fine-Grained_Ego-Motion_CVPR_2025_paper.html.

Haoran He, Yang Zhang, Liang Lin, Zhongwen Xu, and Ling Pan. Pre-trained video generative models as world simulators. *CoRR*, abs/2502.07825, February 2025. URL https://doi.org/10.48550/arXiv.2502.07825.

G Hinton, O Vinyals, and J Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

J Ho, W Chan, C Saharia, J Whang, R Gao, and A Gritsenko. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. URL https://arxiv.org/abs/2210.02303.

Yicong Hong, Yiqun Mei, Chongjian Ge, Yiran Xu, Yang Zhou, Sai Bi, Yannick Hold-Geoffroy, Mike Roberts, Matthew Fisher, Eli Shechtman, Kalyan Sunkavalli, Feng Liu, Zhengqi Li, and Hao Tan. Relic: Interactive video world model with long-horizon memory, 2025. URL https://arxiv.org/abs/2512.04040.

Z Huang, Y He, J Yu, F Zhang, C Si, Y Jiang, and Y Zhang. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, Loic Magne, Ajay Mandlekar, Avnish Narayan, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Yinzhen Xu, Xiaohui Zeng, Kaiyuan Zheng, Ruijie Zheng, Ming-Yu Liu, Luke Zettlemoyer, Dieter Fox, Jan Kautz, Scott Reed, Yuke Zhu, and Linxi Fan. Dreamgen: Unlocking generalization in robot learning through video world models, 2025. URL https://arxiv.org/abs/2505.12705.

Dengyang Jiang, Dongyang Liu, Zanyi Wang, Qilong Wu, Liuzhuozheng Li, Hengzhuang Li, Xin Jin, David Liu, Changsheng Lu, Zhen Li, Bo Zhang, Mengmeng Wang, Steven Hoi, Peng Gao, and Harry Yang. Distribution matching distillation meets reinforcement learning. *arXiv preprint arXiv:2511.13649*, 2025. URL https://arxiv.org/abs/2511.13649.

X Ju, Y Gao, Z Zhang, Z Yuan, X Wang, and A Zeng. Miradata: A large-scale video dataset with long durations and structured captions. In *Advances in Neural Information Processing Systems*, 2024.

Y Kirstain, A Polyak, U Singer, S Matiana, and J Penna. Pick-a-pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oCBKGw5HNf.

NVIDIA, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezanali, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchapmi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical ai, 2025. URL https://arxiv.org/abs/2501.03575.

A O'Neill, A Rehman, A Maddukuri, and A Gupta. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *IEEE International Conference on Robotics and Automation*, 2024.

L Ouyang, J Wu, X Jiang, D Almeida, and C Wainwright. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

C Pan, B Yaman, T Nesti, A Mallik, and AG Allievi. Vlp: Vision language planning for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *CoRR*, abs/2310.12921, 2023. URL https://doi.org/10.48550/arXiv.2310.12921.

J Schrittwieser, I Antonoglou, T Hubert, and K Simonyan. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

Y Song, H Zhang, C Eisenach, S Kakade, and D Foster. Mind the gap: Examining the self-improvement capabilities of large language models. In *International Conference on Learning Representations*, 2025.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, pp. 32211–32252, 2023.

S Stapf, PA Huertos, A Davtyan, and P Favaro. Composition of memory experts for diffusion world models. *arXiv preprint arXiv:2605.18813*, 2026. URL https://arxiv.org/abs/2605.18813.

N Stiennon, L Ouyang, J Wu, D Ziegler, and R Lowe. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, 2020.

Richard S. Sutton. First results with dyna, an integrated architecture for learning, planning and reacting. In *Neural Networks for Control*. MIT Press, 1991. URL https://doi.org/10.7551/mitpress/4939.003.0012.

B Wallace, M Dang, R Rafailov, L Zhou, and A Lou. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Y Wang, Z Sun, J Zhang, Z Xian, E Biyik, and D Held. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*, 2024. URL https://arxiv.org/abs/2402.03681.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners, 2025. URL https://arxiv.org/abs/2509.20328.

S Xue, C Ge, S Zhang, Y Li, and ZM Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv preprint arXiv:2509.25050*, 2025. URL https://arxiv.org/abs/2509.25050.

T Yin, M Gharbi, R Zhang, E Shechtman, and F Durand. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025. URL https://arxiv.org/abs/2509.16117.

Tinghui Zhu, Sheng Zhang, James Y. Huang, Selena Song, Xiaofei Wen, Yuankai Li, Hoifung Poon, and Muhao Chen. Video models can reason with verifiable rewards. *arXiv preprint arXiv:2605.15458*, 2026. URL https://arxiv.org/abs/2605.15458.

B Zitkovich, T Yu, S Xu, P Xu, T Xiao, F Xia, and J Wu. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of Machine Learning Research*, 2023.

A REPRODUCIBILITY

This supplement provides evaluation protocols and data construction (A), full results (B), evaluator validation (C), additional experiments (D), supporting theory (E), and qualitative examples (F).

A.1 EVALUATION PROTOCOL AND SPLIT AUDIT

For the evaluations in Sec. 5, we identify which set of saved videos each analysis uses and audit the training/benchmark separation described in Sec. 4. We compare image identifiers, full prompts, and filename prefixes, and distinguish the original benchmark export from the set of videos fixed before independent judging.

Panel A of Table 1 reports the benchmark export, with marginal counts in Table 12 and 121/200 WMSD task successes. In Panel B, WMSD achieves 123/200 Qwen task successes and 116/200 independent task successes. The independent and cross-evaluator analyses use these fixed videos.

The available manifests contain 200 benchmark image IDs, all distinct from those in the 18,305-image inventory and its 8,430-image filtered subset. Exact full prompts are also disjoint. Archived ablations use a separate 200-example set, while the detailed-prompt diagnostic uses 50 additional examples.

Each set of videos is then scored with fixed model weights. Paired comparisons use the same evaluation examples across methods; the retained uncertainty analyses resample evaluation examples.

Judges and sampling. For the main LTX-2 WorldTasks experiments, training rewards and the Qwen benchmark evaluation use `Qwen/Qwen3.5-27B-FP8`. We also investigate other VLMs: the HY1.5 WorldTasks training reward uses `Qwen/Qwen3-VL-8B-Instruct`, independent WorldTasks judging uses `gpt-5.6-luna` with reasoning disabled, and DreamGen evaluation uses `Qwen/Qwen2.5-VL-7B-Instruct` through its benchmark evaluator. Thus, the experiments span several VLMs across WorldTasks and DreamGen. Qwen's evaluation configuration uses at most 21 uniformly sampled frames in chronological order, video-input FPS 4.0. The independent runner uses 12 rounded, uniformly spaced frame indices including first and last frames, a maximum edge of 448 pixels, JPEG quality 75, and a maximum of 64 completion tokens. Task, agent, and consistency each receive a separate request with the instruction; method identities and detailed solutions are hidden. The frozen evaluation contains 1,800 valid judgments. The temporal evaluation contains 1,500 valid unique method/example/criterion keys. Under the fixed-denominator protocol, any unresolved key receives a failure label.

Metrics and missing judgments. Task completion, correct-agent attribution, and physical/temporal consistency are binary endpoints. For each criterion, the benchmark export divides successful judgments by all valid judgments. If an evaluator call fails or its response cannot be parsed into a score, that judgment is excluded from the rate, so the denominator may be below 200 (Table 12). The secondary mean averages the three marginal success rates. Frozen-export rates use a fixed denominator of 200 per method. All tables distinguish these evaluation success rates from the continuous rewards used during training.

A.2 TRAINING CONFIGURATION

We detail the fine-tuning, sampling, and reward settings that instantiate the method in Sec. 3 for the LTX-2 and HunyuanVideo-1.5 experiments in Secs. 5 and 7. These configurations specify how each model learns from task rewards and solution-conditioned Demonstrator guidance.

Table 8 summarizes the LTX-2 and HunyuanVideo-1.5 training settings used in Sec. 5.

Table 8: Main training hyperparameters used for fine-tuning LTX-2 and HunyuanVideo-1.5. For details on implementation-specific parameters, we refer to the official codebase.

Category	LTX-2	HunyuanVideo-1.5
Base model	`LTX-2-19b-distilled`	`HunyuanVideo-1.5-480p_i2v`
Model type	`ltx2_i2v`	`hy15_i2v`
Fine-tuning method	LoRA	LoRA
LoRA rank	64	64
LoRA alpha	128	128
Trainer type	`AWM-demo`	`GRPO-demo`
Loss type	`exp_first`	–
Advantage weighting	`ghuber`	–
Advantage aggregation	`gdpo`	–
Learning rate	2×10^{-4}	1×10^{-4}
Optimizer	Adam	Adam
Adam betas	(0.9, 0.999)	(0.9, 0.999)
Adam epsilon	10^{-8}	10^{-8}
Weight decay	10^{-4}	10^{-4}
Max grad norm	1.0	1.0
EMA decay	0.96	0.9
EMA decay schedule	constant	power
EMA update interval	1	4
Number of inner epochs	1	1
Unique samples per epoch	32	8
Group size	24	16
Training resolution	384×576	480×848
Evaluation resolution	512×768	480×848
Number of frames	121	121
Inference steps (train)	8	10
Inference steps (eval)	8	40
Training timesteps	4	10
Timestep range	[0, 1]	[0, 0.9]
Clip range	$[-1, 1]$	$[-3\times10^{-4}, 3\times10^{-4}]$
Advantage clip range	$[-5, 5]$	$[-5, 5]$
β_d	0.008	1.0
	Reward	
Reward model	`Qwen3.5-27B-FP8`	`Qwen3-VL-8B-Instruct`
Reward type	multi (task/consistency/pick)	task+consistency
Reward weight	0.5 / 0.225 / 0.225	0.95
Distillation reward weight	0.05	0.05

Reward mixture and advantages. For LTX-2, the raw reward mixture and the normalized advantages are

$$R_i = 0.50\, r_{\text{task},i} + 0.225\, r_{\text{cons},i} + 0.225\, r_{\text{pick},i} + 0.05\, r_{\text{distill},i}, \tag{6}$$

$$u_i = \sum_j \lambda_j \frac{r_{j,i} - \mu_{j,g(i)}}{\max(\sigma_{j,g(i)}, 10^{-6})}, \qquad A_i = \text{clip}\left(\frac{u_i - \mu_u}{\max(\sigma_u, 10^{-6})}, -5, 5\right). \tag{7}$$

Here $g(i)$ identifies the same-task group. Each reward component is normalized within its group before weighting; batch normalization yields the combined advantages (Eq. 7). This makes feedback relative to competing generations rather than simply standardizing raw R_i. Reward prompts are provided in Appendix A.4.2.

Annotated-video SFT baseline. For the LTX-2 SFT baseline in Sec. 7, we use 10,000 videos from the source pool used to construct *WorldTasks*. We annotate the agent and task shown using a procedure similar to the image-based annotation pipeline in Sec. 4, with a different prompt for video inputs. We fine-tune on these video–annotation pairs for 12 hours, matching the wall-clock duration of our main WMSD training (Appendix A.3).

A.3 Compute resources and generated-video accounting

We report training resource use and generated-video counts for the experiments in Secs. 5–7. GPU-hours are calculated from allocated devices and run duration, while rollout-video counts follow the task batch size, generation group size, and number of rollout updates.

Every training run described by the primary/ablation resource allocation lasted 12 hours. Thus a primary run using 128 GH200 GPUs consumes $128 \times 12 = 1{,}536$ GH200 GPU-hours, and a 16-GH200 ablation run consumes $16 \times 12 = 192$ GH200 GPU-hours. Table 9 summarizes the allocated resources for each run class.

Table 9: Training compute per run. Author-reported allocation and duration.

Run class	GH200 GPUs	Wall hours	GPU-hours
Primary training	128	12	1,536
Ablation training	16	12	192

The total number of generated training videos is

$$\text{Total} = U \times B \times G,$$

where U is the number of rollout updates, B the number of unique task inputs per update, and G the number of generations per task. This count excludes evaluation and diagnostic runs. The reference LTX-2 configuration in Table 8 gives $32 \times 24 = 768$ videos per rollout update; the HY1.5 reference gives $8 \times 16 = 128$. Consequently 25 HY1.5 updates correspond to 3,200 training videos under that configuration. Saved later LTX-2 ablation configurations instead specify $B = 4, G = 24$, or 96 videos per update (9,600 over 100 rollout updates). Multiple inner optimizer updates can reuse the same rollout videos.

The generation counts above follow the stated batch and group sizes for each reference configuration. Resource allocations cover individual primary and ablation training runs.

Configuration variants. Table 8 records the reference settings. Later saved LTX-2 configurations use $\beta_d = 0.02$, four task inputs per update, and additional resolution settings. We report the corresponding generation counts separately above.

A.4 WorldTasks data construction

This section details the WorldTasks data construction introduced in Sec. 4, which supplies image–task–solution pairs without paired human task-execution videos. We construct these annotations from scene images using a VLM, screen the images for usable agent-centered content, and document the task composition and reward prompts below.

A.4.1 Dataset Filtering

The filtering pipeline for the training data in Sec. 4 retains images with enough visual information and actionable scene content for task-conditioned generation. It combines low-level image-quality checks, aesthetic ranking, and VLM-based semantic screening.

We construct the dataset from pre-extracted images. We first remove incomplete or already processed entries, and then apply an image-quality filter. This filter rejects frames that are excessively blurry, underexposed, overexposed, or nearly empty. Concretely, we compute the variance of the Laplacian as a blur indicator, the mean luminance to detect overly dark or bright images, and the fraction of near-black and near-white pixels to remove degenerate frames. In our main setup, we use thresholds of `min_laplacian_var` $= 12.0$, `min_mean_luma` $= 20.0$, `max_mean_luma` $= 235.0$, `max_black_ratio` $= 0.85$, and `max_white_ratio` $= 0.85$.

To further improve visual quality, we rank the surviving frames with an aesthetic score based on a CLIP-based scoring function and keep only the top 90% of samples according to the combined quality score, computed as the arithmetic mean of the component scores. We also apply a vision-language quality screening step that discards frames deemed unsuitable for agent-based video generation.

The VLM is prompted to assess whether an image is appropriate for agent-based video generation based on the prompt given in Fig. 6.

VLM-Based Dataset Quality Filtering Prompt

Look at this image carefully. Is this image suitable as a training sample for an agent-based video generation model?

Consider whether:

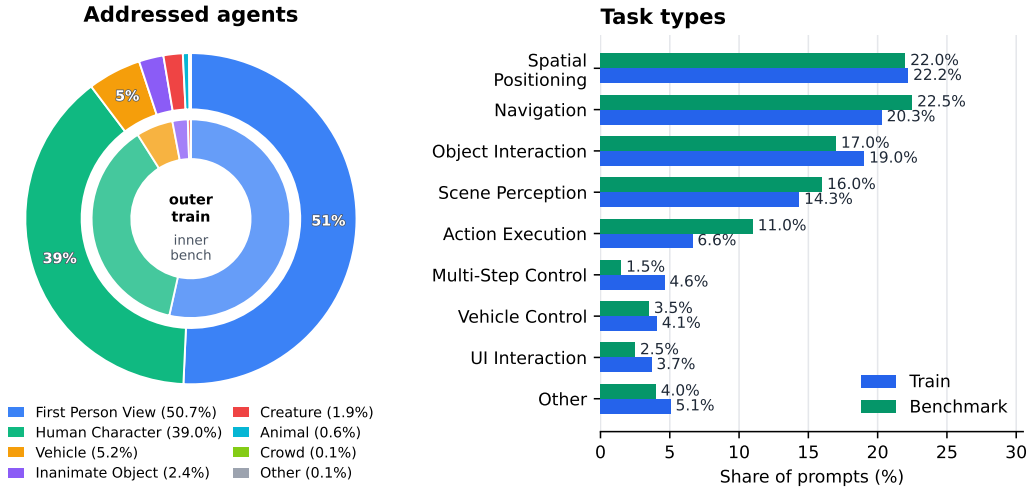
- The scene contains meaningful, diverse visual content (not blank, corrupted, or trivially simple)
- The image quality is acceptable (not severely blurry, overexposed, underexposed, or distorted)
- The image depicts a scene where an agent could plausibly perform tasks

Output Format
Answer with ONLY **Yes** or **No**. Do not include any explanation or additional text.

Figure 6: Prompt used for VLM-based semantic filtering of dataset images.

Samples for which the VLM responds negatively are discarded. This additional semantic filtering step complements the low-level image quality criteria by removing visually valid but uninformative or non-actionable scenes, resulting in a dataset that is both visually and semantically suitable for downstream task-conditioned video generation.

Composition and category mapping. The 146,440-pair inventory and the further filtered 67,440-pair subset are reported separately in Sec. 4. Figure 7 describes the latter composition. The prompt-only benchmark taxonomy has 45 Navigation and 6 Path Clearing examples; the full-results category groups these as Navigation and traversal ($N = 51$). Likewise, 32 Perception and 1 Status Check examples form Perception/state assessment ($N = 33$). Positioning ($N = 44$), Object Interaction ($N = 34$), and Combat Action ($N = 22$) retain their supports. These merged category names are used consistently for the independent breakdown. The remaining categories cover 16 examples.



(a) Addressed-agent categories. **(b)** Task-type distribution.

Figure 7: WorldTasks prompt composition for the training split and WorldTasks-Bench.

A.4.2 REWARD PROMPTS

We present the prompts for the reward models in Sec. 4, used by the training procedure in Sec. 3 and shown in Fig. 8 and Fig. 9. These prompts define yes/no queries for task success and visual-temporal consistency; their log-probability differences supply the continuous training rewards defined in Sec. 4.

Task Success Evaluation Prompt

Instruction: *{instruction}*
Target agent: *{agent_name}*

You are given generated video frames in correct temporal order (frames 0..N-1).
You are judging whether a **target agent** in this temporally ordered video successfully completes an instruction.
If the prompt refers to first-person view, camera perspective, or first-person perspective, interpret that as referring to the camera movement and viewpoint.
Analyze the video carefully and reason strictly from visible evidence. Do not assume intent. Do not infer unseen events. Do not guess.

Evaluation Criteria

1. **Correct Agent Attribution**
   - The required action must be performed by the target agent.
   - If another agent performs the action, the task is **NOT** successful.
2. **Action Progress and Completion**
   - The target agent must clearly complete the instructed action.
   - The instruction must be fully satisfied by the final frame (or earlier with a stable and persistent completed state).
   - For first-person camera view: no other agent should be performing the action.
3. **Realism and Physical Consistency**
   - The outcome must be grounded in objects present in earlier frames.
   - If video quality is too poor to determine completion reliably, the task is **NOT** successful.

Decision Rule
Answer **Yes** only if all of the following are true:

- The target agent performs the required action
- The action is fully completed by the final frame
- The action is physically realistic and consistent

If any condition fails, answer **No**.

Output Format
Return exactly one word: **Yes** or **No**. Do not include any explanation or additional text.

Figure 8: Prompt used for task reward during training.

Visual Quality and Temporal Consistency Prompt

You are given generated video frames in correct temporal order (frames 0..N-1).
You are judging whether this temporally ordered video is successful in terms of visual quality and temporal consistency.
Analyze the video carefully and reason strictly from visible evidence. Do not infer hidden causes. Do not guess missing frames. Do not speculate beyond what is visible.

Evaluation Criteria

1. **Visual Quality**
   - Frames should be clear, coherent, and stable.
   - Severe blur, flicker, distortions, broken rendering, or major artifacts mean the video is **NOT** successful.
2. **Temporal Consistency**
   - Motion and state changes should be smooth and physically coherent over time.
   - No teleportation, popping, identity instability, discontinuous motion, or implausible changes.
3. **Reliability of Evidence**
   - Judge only from visible evidence in the frames.
   - If frames are too unclear to assess reliably, the video is **NOT** successful.
4. **Consistency with Initial Frame**
   - The video must remain consistent in style and quality with the first frame.

Decision Rule
Answer **Yes** only if all of the following are true:

- Visual quality is acceptable overall without severe artifacts
- Temporal consistency is acceptable without severe continuity failures
- Frames are clear enough for reliable judgment

If any condition fails, answer **No**.

Output Format
Return exactly one word: **Yes** or **No**. Do not include any explanation or additional text.

Figure 9: Prompt used for the consistency reward during training.

A.5 WorldTasks-Bench evaluation prompts

For the *WorldTasks-Bench* results in Sec. 5, we present the evaluation prompts used to assess generated videos along three complementary dimensions. The first prompt evaluates whether the instructed task is successfully completed, focusing strictly on end-state correctness. The second prompt verifies correct agent attribution, ensuring that the intended actor performs the specified action. The third prompt measures physical realism and temporal consistency, capturing whether the video exhibits plausible motion and coherent dynamics. Together, these prompts provide a structured and binary evaluation framework that isolates task success, agent correctness, and physical validity.

Task Solved Evaluation Prompt

Instruction: *{eval_prompt}*

You are a strict video benchmark judge.
Decide whether the video successfully solves the instructed task by the end.

Decision Rule

- Return **1** only if the requested action is clearly completed.
- Return **0** if the task is incomplete, incorrect, ambiguous, only partially done, or a different action occurs.

Output Format
Reply with only **0** or **1**.

Figure 10: Prompt used to evaluate whether a generated video successfully completes the instructed task.

Correct Agent Attribution Prompt

Full instruction: *{eval_prompt}*
Expected acting agent: *{agent_name}*
Expected task for that agent: *{agent_task_text}*

You are a strict video benchmark judge.
Decide whether the correct agent is the one performing the instructed action.

Decision Rule

- Return **1** only if the expected agent clearly performs the task.
- Return **0** if another agent performs it, the acting agent is unclear, or the expected agent is absent or incorrect.

Output Format
Reply with only **0** or **1**.

Figure 11: Prompt used to verify that the correct agent performs the instructed action.

Physical Realism and Temporal Consistency Prompt

You are a strict video benchmark judge.
Decide whether the video shows physically realistic execution.

Decision Rule

- Return **1** only if motion, contact, timing, object interactions, and scene dynamics are physically plausible and temporally coherent.
- Return **0** if there is teleportation, impossible motion, severe temporal inconsistency, broken dynamics, identity drift, or obvious non-physical behavior.

Output Format
Reply with only **0** or **1**.

Figure 12: Prompt used to evaluate physical realism and temporal consistency of generated videos.

A.6 DEPLOYMENT LATENCY

To support the inference-latency result in Sec. 5, we test which part of inference is affected by removing the external solution prompt. We compare video-sampling times for the Demonstrator and Executor, then separately time the VLM solution-rewrite call needed by the Demonstrator.

The video-generation measurements use one NVIDIA GH200 GPU on a GH200 node. The LTX-2 evaluation configuration uses batch size one, 512×768 resolution (height $\times$ width), 121 frames at 24 FPS, eight denoising steps, bfloat16 precision, and guidance scale 1.0 (Table 8). A matched timing run measures diffusion sampling separately from solution generation (Tab. 10). The measured sampling latencies are nearly identical; WMSD eliminates the external solution-generation call.

Table 10: Matched diffusion-sampling latency. The external planning call is excluded from these timing columns.

Deployment condition	Mean	Median	p95	External planner
+VLM / Demonstrator	8.90 s	8.29 s	15.75 s	Required
WMSD	8.96 s	8.51 s	15.29 s	None

We additionally time the exact Qwen3.5-27B-FP8 solution-rewrite request on 50 unique image–task pairs after four untimed warm-ups. All requests return nonempty prompts without HTTP errors. The warm-service median is 2.4018 s, the mean is 2.4228 s (sample standard deviation 0.4205 s), the 5th–95th percentile range is 1.8452–3.2297 s, and the minimum/maximum is 1.7947/3.7212 s. The average completion contains 136.86 tokens. Because video generation cannot begin until this sequential call returns, combining the reference 10.1 s LTX-2 8-step generation time with the measured median gives the estimated deployment accounting in Tab. 11. The estimated sums combine the video-generation reference with the measured planner latency; bracketed ranges use the planner's 5th–95th percentiles.

Table 11: Warm-service deployment accounting for LTX-2 8-step generation.

Path	Video	Planner	Estimated sum
Base	10.1 s	0 s	10.1 s
+VLM / Demonstrator	10.1 s	2.402 s	$\approx$ 12.5 s
		$[1.845, 3.230]$	$[11.9, 13.3]$
WMSD	10.1 s	0 s	10.1 s

The solution-generation requests use a separate four-GH200 data-parallel vLLM service with batch size one, temperature 0.5, thinking disabled, and a 512-token budget. Timing excludes server/model startup and compilation. These warm-service measurements capture the planner delay with the VLM already initialized; WMSD avoids this call and any startup required to host the planner.

B FULL RESULTS

This section expands the benchmark comparisons in Sec. 5 to show how evaluation denominators, task categories, and Demonstrator outcomes affect their interpretation. It also details the maze-solving experiments in Sec. 7, which vary the training reward and teacher quality under a common deterministic endpoint verifier.

B.1 BENCHMARK DENOMINATORS AND FROZEN-EXPORT COMPARISON

For the results in Sec. 5, we report the valid-judgment denominators and separate frozen-export counts to distinguish model performance from differences in evaluation accounting. The frozen videos are assessed by Qwen and an independent GPT-5.6 Luna evaluator using separate binary judgments for task completion, agent attribution, and consistency.

Table 12: Benchmark-export counts and secondary mean. Each marginal reports successes/valid judgments. The mean averages the three exact marginal success rates.

Method	Task	Agent	Consistency	Mean
HY1.5	91/196	108/200	156/200	0.595
HY1.5 + WMSD	113/197	126/200	164/198	0.677
LTX-2 (full-step)	63/200	79/200	138/200	0.467
LTX-2 + annotated-video SFT	57/195	77/198	135/198	0.454
LTX-2 + WMSD (full-step)	90/199	100/200	138/199	0.549
Base (8-step)	57/200	77/197	136/196	0.457
Demonstrator (8-step)	99/200	111/194	145/198	0.600
WMSD (8-step)	121/200	134/194	172/195	0.726

On the frozen export, Qwen has task/agent/consistency success counts of 57/74/130 for Base and 123/147/173 for WMSD, all reported against a fixed denominator of 200. These differ from the valid-case benchmark export. Independent counts are 69/101/117 for Base, 107/134/147 for the Demonstrator, and 116/145/154 for WMSD, each out of 200. A secondary mean always averages the three marginal rates and is never assigned a binary numerator.

B.2 Task categories under the independent evaluator

This breakdown expands the task-family comparison in Sec. 5 using the original independent evaluation, grouping benchmark instructions according to the required behavior. Within each category, we compare Base, Demonstrator, and WMSD on the same examples. The updated paired task comparison is reported separately in Appendix B.3.

Figure 13 shows the five named categories (184 tasks). Table 13 also pools the remaining 16 tasks, covering all 200; both use the merged taxonomy in the composition notes.

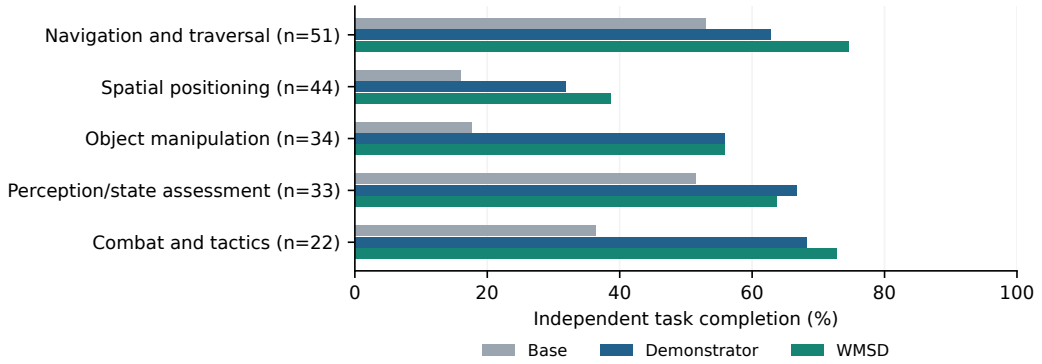


Figure 13: Independent task-completion breakdown. Each category uses the same support for Base, Demonstrator, and WMSD. Labels and counts follow the merged taxonomy.

Table 13: Independent task-completion results by category. Method columns give success counts out of N; differences are percentage points.

Task category	N	Base	Demo.	*WMSD*	Δ Base	Δ Demo.
Navigation and traversal	51	27	32	38	+21.6	+11.8
Spatial positioning	44	7	14	17	+22.7	+6.8
Object manipulation	34	6	19	19	+38.2	+0.0
Perception/state assessment	33	17	22	21	+12.1	−3.0
Combat and tactics	22	8	15	16	+36.4	+4.5
Remaining categories (pooled)	16	4	5	5	+6.3	+0.0

B.3 Retaining Demonstrator successes and rescuing failures

We expand the paired task analysis in Secs. 5 and 6 (Fig. 2b), separating retained Demonstrator successes, rescued failures, and lost successes. The updated task-completion comparison pairs the independent evaluator's labels on 200 matched inputs; the agent and consistency rows retain the original paired evaluation.

WMSD retains 100 of 107 Demonstrator task successes (93.5%) and rescues 16 of 93 Demonstrator failures (17.2%), while losing seven successes (Table 14). Both models fail on the remaining 77 inputs. WMSD therefore completes 116 tasks versus the Demonstrator's 107, an observed gain of nine tasks ($+4.5$ percentage points). The overlap indicates high retention with some additional rescues. A two-sided exact McNemar test on the 23 discordant pairs (16 rescues versus seven losses) gives $p = 0.093$, so the task advantage does not reach the 5% significance threshold. This paired comparison describes how the models' outcomes differ; the matched ablations in Appendix D.2 assess individual training components.

For agent correctness and physical consistency, WMSD retains 104/134 (77.6%) and 121/147 (82.3%) Demonstrator successes, respectively, while rescuing 41 and 33 failures. Their observed totals are 145 versus 134 and 154 versus 147, giving net gains of 11 and seven out of 200.

Table 14: Demonstrator retention and rescue under the independent evaluator. Task completion uses the updated paired evaluation; the other criteria use the original paired evaluation.

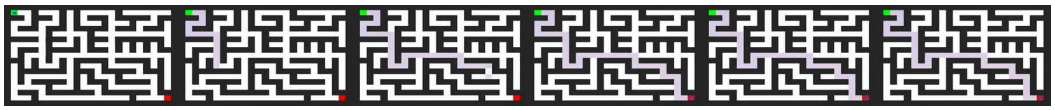
Criterion	Demo. failures	Rescued	Rescue rate	Demo. successes	Retained	Retention rate
Task completion	93	16	17.2%	107	100	93.5%
Correct agent	66	41	62.1%	134	104	77.6%
Physical consistency	53	33	62.3%	147	121	82.3%

B.4 Reward source and Demonstrator quality in Maze

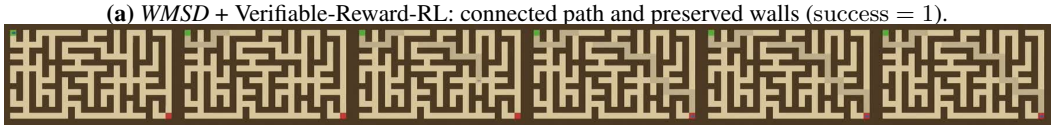
The maze-solving experiments in Sec. 7 test whether WMSD gains persist under non-VLM verification and depend on a teacher that can use privileged solution text; this section gives their setup and controls. Given an initial maze and a task, the video model must generate a trace connecting the start and target while preserving the walls; matched training variants vary the reward source and Demonstrator quality, and all generated outcomes are scored by the same deterministic verifier.

We use the Maze environment and released VideoRLVR recipe (Zhu et al., 2026). The reward-source and teacher-quality variants start from the same Wan2.2-TI2V-5B SFT model and use the same 10,000-example training split, 480×832 resolution, 81 frames, group size 16, SDE-GRPO configuration, and deterministic 100-maze endpoint verifier. The trained-Demonstrator variants use a frozen rank-32 solution-conditioned LoRA; the unadapted-teacher controls omit it. Only teacher conditioning receives the exact route; the Executor and description-blind VLM reward do not.

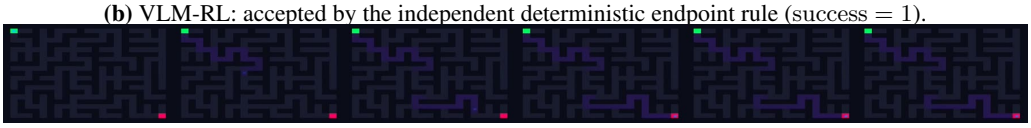
Table 7 reports the main reward-source and teacher-quality results; Table 15 provides the additional teacher controls. Figure 14 illustrates the common deterministic endpoint rule on representative generated rollouts.



(a) *WMSD* + Verifiable-Reward-RL: connected path and preserved walls (success = 1).



(b) VLM-RL: accepted by the independent deterministic endpoint rule (success = 1).



(c) *WMSD* + Verifiable-Reward-RL: walls are preserved, but the detected path is not connected (success = 0).



(d) Verifiable-Reward-RL: a wall-preservation violation causes rejection (success = 0).

Figure 14: Illustrative deterministic Maze verification. Each row shows six uniformly spaced frames from an archived clean training rollout, rescored with the same rule used for test evaluation. Green and red indicate the start and target; the generated trace evolves from left to right. A rollout succeeds only if the trace connects both endpoints and the maze walls remain preserved. The VLM-reward example in (b) is therefore judged by a non-VLM endpoint.

The controls in Tab. 15 test whether the teacher actually uses the privileged route. Solution-conditioned maze generation is out of distribution for the Base SFT teacher before adaptation. To equip it to follow the route, we train a rank-32, solution-conditioned Demonstrator LoRA for five epochs using the standard flow-matching SFT objective. The adapted Demonstrator reaches 0.730 success, and WMSD with rule-based rewards improves further to 0.750 (Table 15). This small adaptation makes the Demonstrator's privileged route useful for training an Executor that receives only the image and task.

Table 15: Maze teacher-quality controls. Rows use the deterministic test verifier. The VLM-trained result is reported from its retained aggregate summary.

Teacher or *WMSD* control	**Success**	**F1**
Base SFT + route text, no trained Demonstrator	0.610	0.750
Five-epoch solution-conditioned Demonstrator LoRA	0.730	0.840
Verifiable-Reward-*WMSD*, unadapted Base-SFT teacher	0.580	0.705
VLM-*WMSD*, unadapted Base-SFT teacher	0.560	0.715
WMSD + Verifiable-Reward-RL, trained Demonstrator	**0.750**	**0.865**
WMSD + VLM-RL, trained Demonstrator	<u>0.740</u>	<u>0.860</u>

Gains with the adapted Demonstrator. With rule-based rewards, WMSD reaches 0.750 success and 0.865 F1, compared with 0.700 and 0.840 for RL only. The retained VLM-reward aggregate also reports higher scores: 0.740 success and 0.860 F1, compared with 0.700 and 0.840 for VLM-RL only. Both reward settings therefore report gains after Demonstrator adaptation.

C Evaluator Validation

These diagnostics support the evaluations in Sec. 5 and the instruction-following analysis in Sec. 6, testing evaluator agreement, temporal sensitivity, and robustness to degraded observations.

C.1 Blinded human evaluation

For the human-preference results in Sec. 5, 18 participants compare WMSD videos with Base or Demonstrator videos for the same tasks, with balanced A/B orientation and hidden method identities.

They assess task completion, correct-agent attribution, physical consistency, and overall preference, with ties and neither-preferred responses allowed.

We report 40 blinded video-pair comparisons involving *WMSD*: 20 against Base and 20 against +VLM/Demonstrator. These are comparison pairs; table N counts individual ratings, including multiple ratings of a pair. Ties and "neither" contribute zero and remain in the denominator of the observed net-preference values. Table 2 reports all four criteria. Included participants completed more than three core-answered trials under the analysis rule.

C.1.1 Detailed and constraint-heavy prompts

This inference-only diagnostic expands the longer-instruction results in Sec. 6, testing tasks with several required actions. We compare Base and WMSD on examples with detailed, constraint-heavy descriptions, giving each model short and detailed prompts with matched images and generation seeds. We assess both overall benchmark scores and the fraction of requested actions visibly completed.

The detailed-prompt diagnostic uses 50 additional examples. The selected descriptions contain 297–478 words (mean 449.3) and at least three action clauses. We match image and generation seed across all four conditions and split each detailed solution into minimal action units for a separate blinded coverage judgment. Tab. 4 shows that *WMSD* retains the ability to use detailed prompts. The prespecified *WMSD* (detailed)−Base(detailed) average-score difference is $+0.053$, and atomic-step coverage differs by $+0.0516$.

WMSD is optimized using short Executor prompts, so its higher average score with that input format is consistent with the training setup. Nevertheless, detailed prompting increases atomic-step coverage by 0.1148. This diagnostic provides a proof of concept for following longer instructions with an Executor trained on short task inputs. Thus task-defining constraints such as object identity, exclusions, ordering, and spatial relations should remain in the Executor prompt even though inferable intermediate execution details can be omitted.

C.1.2 Cross-evaluator and temporal stress tests

These tests probe whether the positive judgments used in Sec. 5 reflect evaluator-specific preferences or insensitivity to action order. We independently rejudge the same frozen videos with GPT-5.6 Luna and compare its labels with Qwen, then apply temporal corruptions and measure how often each evaluator withdraws a previously positive task judgment.

The reward-hacking direction of greatest concern is a Qwen-positive video that the independent GPT-family evaluator rejects. Conditional on a positive Qwen judgment, the independent rejection rate is lower for *WMSD* on every criterion (Tab. 16). The raw number of task disagreements is higher for *WMSD* (28 versus 19) because Qwen marks many more *WMSD* videos successful (123 versus 57). These data do not prove that reward hacking is absent, but they do not indicate a higher conditional Qwen-specific exploitation rate.

Table 16: Independent rejection among Qwen-positive judgments.

Criterion	Base	*WMSD*
Task completion	33.3%	22.8%
Correct agent	25.7%	16.3%
Physical consistency	26.9%	16.8%

We also transform 50 frozen videos per method by reversing frames, shuffling frames, removing the middle, repeating the final frame, or retaining only the first and final frames. Tab. 17 conditions on clean-positive *WMSD* examples and reports the fraction rejected after corruption. Qwen rejects 51.6–67.7% of its previously positive examples. The full corrupted-video averages in Tab. 18 make this evaluator difference explicit. Corruptions do not necessarily invalidate every task: endpoint-based tasks may remain judgeable after middle-frame removal. These rejection rates measure sensitivity to the chosen transformations; judging correctness requires task-specific validity labels.

Table 17: Temporal-corruption rejection rates for clean-positive *WMSD* examples. Qwen and Luna have 31 and 34 clean-positive examples, respectively.

Corruption	Qwen rejects	Luna rejects
Reverse	0.516	0.471
Shuffle	0.613	0.324
Remove middle	0.548	0.118
Repeat final	0.677	0.147
First + final only	0.645	0.265

Table 18: Average scores on temporally corrupted videos. Each cell has 50 valid task, agent, and consistency judgments; values average the three criteria.

	Qwen		Luna	
Corruption	**Base**	***WMSD***	**Base**	***WMSD***
Reverse	0.427	0.573	0.413	0.527
Shuffle	0.333	0.353	0.387	0.573
Remove middle	0.400	0.427	0.360	0.740
Repeat final	0.327	0.327	0.327	0.713
First + final only	0.273	0.280	0.347	0.587

C.1.3 Robustness to degraded initial observations

This diagnostic tests whether the gains in Sec. 5 survive the imperfect observations targeted by the dataset filters in Sec. 4. We generate Base and WMSD videos from matched inputs under clean images, blur, JPEG compression, darkening, and crop/resize, and score the outputs using the WorldTasks criteria.

We evaluate 40 matched examples under clean input and four low-level degradations. Tab. 19 reports average WorldTasks scores. *WMSD* leads Base in every condition, although blur causes the largest absolute degradation. This diagnostic addresses principal filters used in dataset construction, but does not cover every in-the-wild failure mode.

Table 19: Robustness to degraded initial observations. Each condition contains 40 matched examples.

Input condition	Base	*WMSD*	Gain
Clean	0.492	0.708	+0.216
Blur	0.433	0.583	+0.150
JPEG compression	0.442	0.700	+0.258
Darkening	0.475	0.742	+0.267
Crop and resize	0.433	0.725	+0.292

D Additional Experiments

We examine alternative distillation objectives and choices of rewards and teacher updates. These studies provide context for the training design in Sec. 3 and the component analysis in Sec. 6.

D.1 Distribution-matching alternative

This exploratory alternative to the local matching objective in Sec. 3 asks whether privileged solution conditioning can be transferred by matching output distributions. It replaces local velocity-field matching with a DMD-style objective that compares noisy-sample scores from the description-conditioned teacher and an estimate of the student distribution.

We explored a DMD-based alternative inspired by Jiang et al. (2025): a task-conditioned generator is trained against a description-conditioned teacher through noisy-sample score differences, using an auxiliary student-distribution score estimate as in DMD (Yin et al., 2024). The target is an approximate distributional KL, rather than the local field discrepancy in Eq. 2. The explored implementation

diverged and was not used in WMSD. This is a negative implementation observation, not evidence that DMD is generally unsuitable for privileged-conditioning transfer.

D.2 Matched reward and anchor ablations

To isolate the contributions of pretrained-model regularization, privileged execution conditioning, and the detached distillation reward, we compare four configurations on the separate 200-example ablation set (Fig. 15). We hold the backbone, training data, external reward components and weights, rollout budget, and optimizer fixed. All reference models are frozen, and all anchored variants use the same regression coefficient and state-sampling procedure on Executor-generated videos. The Executor receives only the image and task in every configuration.

Table 20: Matched and exploratory configurations. The first four rows hold task, consistency, and PickScore rewards fixed; anchored variants use frozen references on Executor-generated states. The last two rows use task/PickScore feedback and update the Demonstrator, with different state sources and update schedules.

Variant	Anchor conditioning	Distill. reward	Teacher
RL alone	None	No	—
RL + task-conditioned anchor	Image + task	No	Frozen
RL + description-conditioned anchor	Image + description	No	Frozen
Full WMSD	Image + description	Yes	Frozen
Off-Policy + Dem RL	Image + description	No	Updated
Alternating + RL	Image + description	Yes	Updated

What each comparison isolates. Relative to RL alone, the task-conditioned anchor tests regularization toward a pretrained generator without privileged information; replacing its task conditioning with the execution description isolates the value of that information under the same regression objective (Eq. 2). Full WMSD adds only the detached distillation reward (Eq. 3), testing whether rewarding the mismatch helps beyond the unchanged description-conditioned anchor and external rewards.

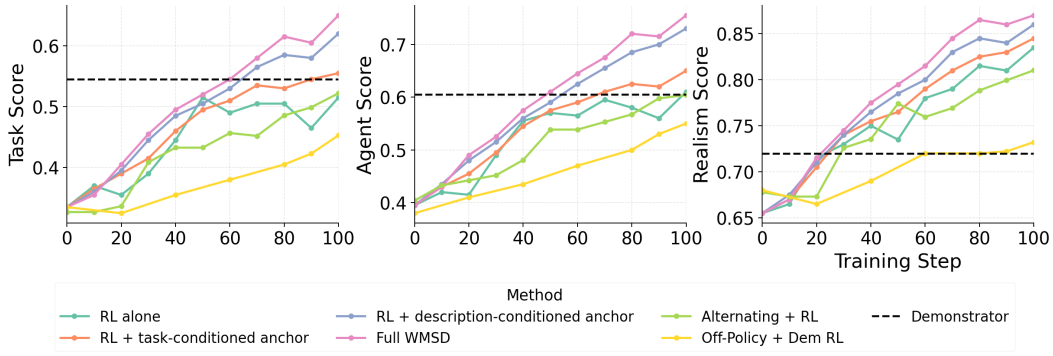


Figure 15: Matched reward and anchor comparison. Task, agent, and realism (physical/temporal consistency) scores on the separate 200-example ablation set. The four principal curves successively test an anchor, privileged description conditioning, and the detached distillation reward with the remaining settings fixed. The dashed line marks the Demonstrator. Alternating + RL and Off-Policy + Dem RL are exploratory references with different teacher-update schedules and reward compositions.

Results. At training step 100, task scores are approximately 0.52 for RL alone, 0.56 with the task-conditioned anchor, 0.62 with the description-conditioned anchor, and 0.65 for full WMSD. Agent and realism scores follow the same ordering. The task-conditioned control shows that generic pretrained-model regularization accounts for part of the improvement; the further gain from description conditioning supports the value of privileged execution information. Full WMSD improves over the otherwise identical description-conditioned-anchor control on all three metrics, supporting an additional benefit from using the mismatch as a detached reward as well as a regression anchor. The incremental reward gain is smaller than the gain from changing the anchor's conditioning.

Relation to the exploratory configurations. The additional Alternating + RL and Off-Policy + Dem RL curves provide context for alternative training schedules. The former interleaves Executor and Demonstrator updates; the latter trains the Demonstrator with task/PickScore rewards and fits the Executor on Demonstrator-generated states. These variants change teacher updates and reward composition and therefore do not isolate components of the final method. The earlier exploratory *On-policy + RL* setting used task/PickScore feedback and an anchor; the new description-conditioned-anchor control instead retains all three external rewards of full WMSD. The distillation-only comparison in Fig. 3 remains separate and uses no external rewards.

Synthetic teacher supervision without RL. The existing *Off-policy* experiment in Fig. 3 is a close comparison to synthetic teacher supervision: it trains the task-conditioned Executor on states from description-conditioned Demonstrator videos, using the frozen Demonstrator's velocity predictions as regression targets, without RL or matching on Executor-generated states. We therefore use this existing experiment to assess learning from teacher-generated videos without the on-policy machinery. Its scores improve initially but level off while variants using Executor-generated states continue improving. This supports matching on the Executor's own state distribution in the tested setting. Off-policy field matching is an analogue of synthetic-video SFT; it does not separately test standard supervised fine-tuning on success-filtered Demonstrator videos.

Other exploratory variants. We compared a per-step distillation reward with the trajectory-level distillation reward in Eq. 3 and found only minor differences in final performance. The reported comparison used a video-level aggregate reward; in AWM this is estimated on re-noised generated latents. We also investigated sharing weights between the Executor and Demonstrator. Across hyperparameter settings and EMA schedules, using the Executor weights as the Demonstrator led to unstable training, so all main results use a fixed Demonstrator.

D.3 Alternating Training Algorithm

The procedure in Alg. 2 extends the training method in Sec. 3 with an exploratory variant that interleaves Executor optimization and optional Demonstrator updates. Executor rounds combine task and distillation rewards with Demonstrator anchoring, while optional teacher rounds use task rewards and a base-model anchor; disabling alternation recovers the fixed-Demonstrator schedule.

Algorithm 2 GRPO/AWM with Demonstrator Anchoring

Require: pretrained base model f_0 with velocity field v_0
Require: Executor sampler $p_\theta(\cdot \mid c_{\mathrm{E}})$ and velocity field v_θ
Require: Demonstrator sampler $p_\phi(\cdot \mid c_{\mathrm{D}})$ and velocity field v_ϕ; fixed for the main *WMSD* setting
Require: VLM-generated condition pairs $\mathcal{S} = \{(c_{\mathrm{E}}, c_{\mathrm{D}})\}$
Require: reward weights $\lambda_{\mathrm{task}}, \lambda_{\mathrm{distill}}$
Require: Executor anchor coefficient β_d and optional Demonstrator anchor coefficient β_ϕ
Require: optional alternation period N; set $N = 0$ to keep the Demonstrator fixed
for iteration $e = 1, 2, \ldots$ **do**
  sample $(c_{\mathrm{E}}, c_{\mathrm{D}}) \sim \mathcal{S}$
  if $N > 0$ **and** $e \bmod N = 0$ **then**
    Optional Demonstrator round
    sample rollout group $\{\tau_i\}_{i=1}^{G} \sim p_\phi(\cdot \mid c_{\mathrm{D}})$
    compute task rewards $r_{\mathrm{task}}(\tau_i; \mathcal{I}, \mathcal{T})$
    compute group-relative RL loss $\mathcal{L}_{\mathrm{RL}}(\phi)$ from the task rewards
    compute base-model anchor loss $\mathcal{L}_{\mathrm{base}}$ against f_0
    update ϕ by minimizing $\mathcal{L}_{\mathrm{RL}}(\phi) + \beta_\phi \mathcal{L}_{\mathrm{base}}$
  else
    Executor round
    sample rollout group $\{\tau_i\}_{i=1}^{G} \sim p_\theta(\cdot \mid c_{\mathrm{E}})$
    compute task rewards $r_{\mathrm{task}}(\tau_i; \mathcal{I}, \mathcal{T})$
    compute distillation rewards $r_{\mathrm{distill}}(\tau_i)$ using Eq. 3
    set $R_i = \lambda_{\mathrm{task}} r_{\mathrm{task}}(\tau_i) + \lambda_{\mathrm{distill}} r_{\mathrm{distill}}(\tau_i)$
    compute group-relative RL loss $\mathcal{L}_{\mathrm{RL}}(\theta)$ from $\{R_i\}_{i=1}^{G}$
    compute Demonstrator anchor loss $\mathcal{L}_{\mathrm{anchor}}$ using Eq. 3
    update θ by minimizing $\mathcal{L}_{\mathrm{RL}}(\theta) + \beta_d \mathcal{L}_{\mathrm{anchor}}$
  end if
end for
Evaluation: use $p_\theta(\cdot \mid c_{\mathrm{E}})$

E Supporting Theory

The mathematical results supporting Sec. 3 comprise a stochastic score-function identity and a deterministic flow bound with different assumptions. Neither is a task-success guarantee. The practical normalized AWM surrogate is described separately from the exact stochastic identity.

E.1 Teacher-state versus Executor-state matching

To formalize the matching alternatives in Secs. 3 and 6, the off-policy objective uses teacher states:

$$\mathcal{L}_{\mathrm{off}} = \mathbb{E}_{(x_t,t)\sim p_{\theta'}(\cdot|c_{\mathrm{D}})} \ell_\theta(x_t, t). \tag{8}$$

The idealized on-policy cost uses actual Executor flow-path states; AWM instead uses re-noised terminal samples. No bound transferring mismatch between these state distributions is established here. Teacher agreement alone does not establish physical correctness.

E.2 Practical flow-model optimization

This section details optimizer choices for Sec. 3: $\mathcal{L}_{\mathrm{RL}}$ in Eq. 4 determines how detached advantages train the Executor, while the Demonstrator anchor is added separately. Our LTX-2 experiments use AWM, and HY1.5 uses FlowGRPO (Table 8).

AWM score and weighting. AWM (Xue et al., 2025) operates on re-noised generated terminal latents. We describe the implemented AWM-demo surrogate using normalized scheduler time u and a noise-directed prediction f_θ, whose regression target is $\epsilon - x_1$; Sec. 3 uses the opposite, noise-to-data time convention. For generated video i and regression sample m, draw independent $\epsilon_{im} \sim \mathcal{N}(0, I)$ and form $z_{im} = (1 - \sigma(u_{im}))x_{1,i} + \sigma(u_{im})\epsilon_{im}$ on unconditioned latent coordinates. Conditioning-image coordinates stay fixed. Here $\sigma(u)$ is the scheduler's noise level, interpolated

between its stored timestep–sigma pairs when needed. The reference LTX-2 setup samples $M = 4$ regression times by stratifying the scheduler-index range, including its initial noisy index; the selected times are shared within a minibatch.

Let q_{id} be one on unconditioned latent coordinates and zero on conditioning coordinates. The element-averaged regression error and generalized-Huber score are

$$e_{\theta,im} = \frac{\sum_d q_{id}\big[f_\theta(z_{im}, u_{im} \mid c_{\mathrm{E},i})_d - (\epsilon_{im} - x_{1,i})_d\big]^2}{\max(\sum_d q_{id}, 1)}, \tag{9}$$

$$s_{\theta,im} = -\frac{u_{im}}{p}\big[(e_{\theta,im} + \delta)^p - \delta^p\big], \qquad p = \tfrac{1}{4}, \quad \delta = 10^{-10}. \tag{10}$$

Thus `ghuber` transforms the mean squared error, rather than averaging a coordinate-wise Huber loss. The implementation accumulates the error and transformation in float64 and returns the score in float32. This score is a regression surrogate, not an exact ODE trajectory log density.

Ratios, clipping, and stored scores. With detached video advantages A_i from Eq. 7, the reference configuration uses

$$\widetilde{A}_i = \mathrm{sg}[\mathrm{clip}(A_i, -5, 5)/5], \qquad \rho_{im}(\theta) = \exp(s_{\theta,im} - s_{\mathrm{old},im}), \tag{11}$$

$$\mathcal{L}_{\mathrm{AWM}} = -\mathbb{E}_{i,m}\Big[\min\Big\{\rho_{im}\widetilde{A}_i, \mathrm{clip}(\rho_{im}, 0, 2)\widetilde{A}_i\Big\}\Big]. \tag{12}$$

The ratio interval $[0, 2]$ comes from the configured clip offsets $[-1, 1]$ in Table 8. The `exp_first` setting exponentiates each timestep's score difference and applies the clipped objective before averaging over video–timestep pairs; it does not exponentiate an averaged score difference.

Before each inner optimization epoch, the code samples and caches regression times, noises, and detached old scores for all minibatches, using the Executor weights before that epoch's updates. The reference setup has one inner epoch and uses current, non-EMA weights for this pass. Current scores reuse the cached times and noises. In the on-policy implementation, the first a minibatches instead set $s_{\mathrm{old},im} = \mathrm{sg}[s_{\theta,im}]$, where a is the runtime gradient-accumulation count after multiplication by M; their forward ratios equal one while gradients still pass through the current scores. Later minibatches retain the cached old scores throughout the epoch.

The Demonstrator is evaluated with its execution description at the same noisy latent and timestep. Its mismatch supplies a detached video reward and the separate, untransformed regression anchor in Eq. 4; the anchor is not part of s_θ or the ratio. Gradients of the combined objective are clipped to norm 1 before an optimizer step in the reference configuration.

FlowGRPO (Liu et al., 2025) instead introduces stochastic transitions into the flow sampler and optimizes clipped transition-likelihood ratios. In a finite-step stochastic model, $p_\theta(\tau) = p_0(x_0)\prod_k p_\theta(x_{k+1} \mid x_k, c)$ admits the score-function decomposition when the stated differentiability and support conditions hold. This justifies the identity below in that setting; it does not establish exact equivalence to normalized AWM. The final method freezes the Demonstrator, whereas the separate exploratory algorithm permits explicitly scheduled teacher updates.

NFT. Diffusion Negative-aware FineTuning (DiffusionNFT) (Zheng et al., 2025) optimizes a forward-process flow-matching objective that contrasts high- and low-reward generations. It uses re-noised terminal samples without requiring reverse-trajectory likelihoods. Its reward-conditioned objective is a possible alternative for $\mathcal{L}_{\mathrm{RL}}$ in Eq. 4, with the same fixed-state anchor; we report no NFT experiments.

E.3 Method Derivations and Proofs

In this section we provide the derivations and proofs from Sec. 3.

Gradient Decomposition Assume a finite-step stochastic sampler with densities relative to a common dominating measure, parameter-independent support, and conditions permitting differentiation under the integral. The derivative of C_θ below is partial in θ with the sampled path held fixed. These assumptions justify Eq. 5; they do not assign a nonsingular path density to a deterministic ODE sampler.

$$
\begin{aligned}
\mathcal{L}_{\text{on}} &= \int p_\theta(\tau \mid c_{\text{E}})\, C_\theta(\tau)\, d\tau, \\
\nabla_\theta \mathcal{L}_{\text{on}} &= \int \nabla_\theta \big[p_\theta(\tau \mid c_{\text{E}})\, C_\theta(\tau) \big] d\tau \\
&= \int \Big[C_\theta(\tau) \nabla_\theta p_\theta(\tau \mid c_{\text{E}}) + p_\theta(\tau \mid c_{\text{E}}) \partial_\theta C_\theta(\tau) \Big] d\tau \quad \text{(product rule)} \\
&= \int p_\theta(\tau \mid c_{\text{E}}) C_\theta(\tau) \nabla_\theta \log p_\theta(\tau \mid c_{\text{E}})\, d\tau \quad \text{(score function trick)} \\
&\quad + \int p_\theta(\tau \mid c_{\text{E}}) \partial_\theta C_\theta(\tau)\, d\tau \\
&= \mathbb{E}_{\tau \sim p_\theta} \big[C_\theta(\tau) \nabla_\theta \log p_\theta(\tau \mid c_{\text{E}}) \big] + \mathbb{E}_{\tau \sim p_\theta} \big[\partial_\theta C_\theta(\tau) \big] \quad \text{(rewrite as expectations).}
\end{aligned}
$$

Proposition 1 (Informal on-policy control)**.** *Fix a paired condition $(c_{\text{E}}, c_{\text{D}})$. Assume both flows exist on $[0,1]$ and have finite second moments. Use the normalized Euclidean norm $\|x\|_d = \|x\|_2/\sqrt{d}$ in both ℓ_θ and W_2, consistent with averaging over d latent elements. Assume the teacher velocity field is L-Lipschitz in x and that the student and teacher flows are initialized from the same base noise $x_0 \sim p_0$. If the student matches the teacher's velocity field on its own trajectories, namely*

$$
\mathbb{E}_{x_0 \sim p_0} \left[\int_0^1 \ell_\theta(x_t^\theta, t; c_{\text{E}}, c_{\text{D}})\, dt \right] \le \varepsilon^2,
$$

then the terminal distributions induced by the student and teacher are close. In particular, under the natural coupling given by the shared initial noise,

$$
W_2 \left(p_\theta(x_1 \mid c_{\text{E}}), p_{\theta'}(x_1 \mid c_{\text{D}}) \right) \le e^L \varepsilon, \tag{13}
$$

where $\varepsilon \ge 0$.

Proof. Couple the two flows with the same initial noise, and let $\Delta_t = x_t^\theta - x_t^{\theta'}$ and $e_t = \|v_\theta(x_t^\theta, t \mid c_{\text{E}}) - v_{\theta'}(x_t^\theta, t \mid c_{\text{D}})\|$. Then $\Delta_0 = 0$. Adding and subtracting the teacher velocity at the student state and using the teacher's L-Lipschitz property gives

$$
\frac{d}{dt} \|\Delta_t\| \le e_t + L \|\Delta_t\|.
$$

Grönwall's inequality followed by Cauchy–Schwarz yields

$$
\mathbb{E}\|\Delta_1\|^2 \le e^{2L}\, \mathbb{E} \left(\int_0^1 e_t\, dt \right)^2 \le e^{2L}\, \mathbb{E} \int_0^1 e_t^2\, dt \le e^{2L} \varepsilon^2.
$$

The shared initialization is a valid coupling of the terminal laws. By the definition of W_2,

$$
W_2(p_\theta(x_1 \mid c_{\text{E}}), p_{\theta'}(x_1 \mid c_{\text{D}})) \le \left(\mathbb{E}\|\Delta_1\|^2 \right)^{1/2} \le e^L \varepsilon.
$$

□

F Qualitative Examples

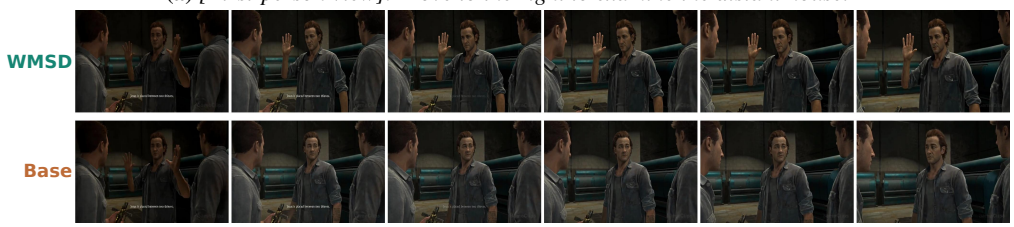
We complement the evaluations in Secs. 5–7 with generated rollout comparisons and illustrate the synthetic supervision in Sec. 4 with task–solution annotations. The examples make the visible behaviors and the content of the privileged text directly inspectable.

F.1 Task-conditioned rollout comparisons

These WMSD and Base frame sequences illustrate the task-solving comparisons in Secs. 5 and 7 using the same task instructions across both video backbones. Additional examples illustrate the consistency-reward comparison in Sec. 6, showing the artifacts that this reward targets.



(a) *[First-person view]: Move to the right to examine the distant house.*



(b) *[Man in center]: Lower one hand slowly.*



(c) *[First-person view]: Walk toward the building entrance on the right.*



(d) *[First-person view]: Turn right to face the welcome sign on the easel.*

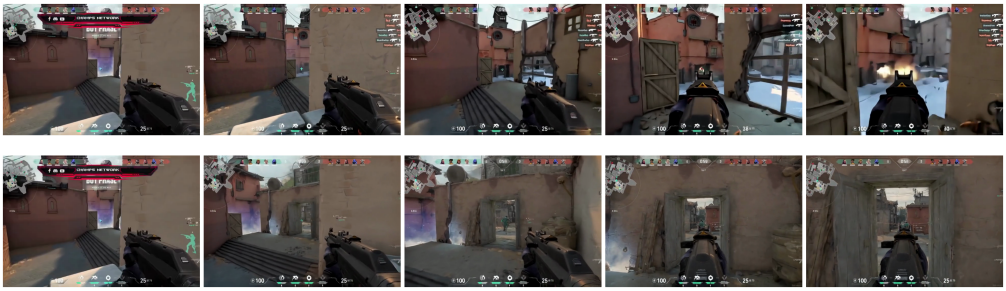


(e) *[First-person view]: Use the rope to swing toward the nearest visible shipwreck's hull.*

Figure 16: Qualitative comparisons between *WMSD* and the base model across LTX-2 and HunyuanVideo-1.5. Each subfigure shows six uniformly sampled frames from the generated videos.



(a) Prompt: "*[Lara Croft]: Turn left and position yourself directly in front of the central arch.*"



(b) Prompt: "*[First-person view]: Aim the weapon at the doorway entrance.*"

Figure 17: Two examples: the first row uses the consistency reward, while the second row does not. The second row shows that the model generates (a) the arch + Lara Croft and (b) the doorway illustrating implausible appearances under task-only feedback.

F.2 WorldTasks task and solution prompts

To illustrate the structure and noise of the synthetic supervision in Sec. 4, we show four dataset samples below. Each example contains the first frame together with the first two task prompts and their corresponding descriptive solution prompts. Examples are shown in Fig. 18 and Fig. 19.



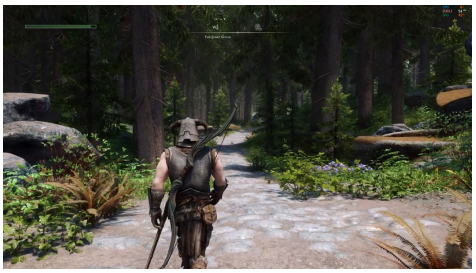
Sample ID: `100_4`

1. **Task 1:** [Man in blue shirt]: Step onto the yellow lane marking and stop exactly at the white arrow's tip.

   Description 1: The man in the blue shirt begins walking forward along the center of the road, his feet deliberately stepping onto the double yellow lane marking, and continues moving straight ahead until he reaches the tip of the white directional arrow painted on the asphalt, where he halts and stands still.

2. **Task 2:** [Person in blue shirt]: Move forward to the nearest building.

   Description 2: The person in the blue shirt begins walking forward along the center of the road, maintaining a steady pace toward the building on the left side of the street, their body oriented directly ahead as they cross the yellow double lines; after a few steps, they continue moving forward until they reach the sidewalk adjacent to the building, then they halt beside the American flag mounted on the building's facade, coming to a complete stop with their feet planted on the pavement.



Sample ID: `145_4`

1. **Task 1:** [Character with horned helmet]: Use the bow to aim at the tree trunk directly ahead.

   Description 1: The character with the horned helmet slowly turns their upper body toward the tree trunk directly ahead, simultaneously drawing the bowstring back with their right hand while keeping their left hand steady on the bow's grip, their gaze fixed on the target as the bowstring tenses and the arrow nocks align with the trunk.

2. **Task 2:** [Character with horned helmet]: Move to the largest boulder and stop beside its left edge.

   Description 2: The character with the horned helmet begins walking forward along the stone path, their body oriented toward the largest boulder visible to the left, and after a few steps, they decelerate, shifting their weight slightly as they turn their head to the left to align their gaze with the boulder's edge, then halt precisely beside its left side, their right hand resting on their hip while their left hand remains near the hilt of their weapon.

Figure 18: Two representative samples from *WorldTasks*. Each sample includes an initial frame, task prompts, and corresponding descriptive solutions.



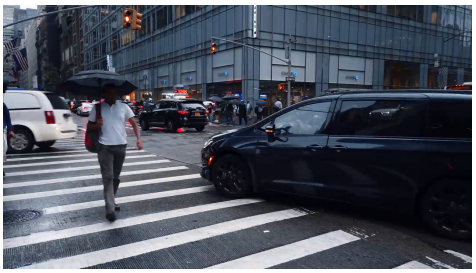
Sample ID: `6888_1`

1. **Task 1:** [Driver in racing suit]: Press the red button on the steering wheel's right side.

   Description 1: The driver's right hand, clad in a black racing glove, moves slightly forward and inward, pressing the red button located on the right side of the steering wheel, while the left hand remains steady on the left side of the wheel, and the vehicle continues forward along the track with the dashboard displaying 186 MPH and an overtaking indicator active.

2. **Task 2:** [First-person view]: Align the car's front bumper with the white track curb ahead.

   Description 2: The driver's hands grip the steering wheel firmly, thumbs pressing the paddle shifters while the left hand subtly adjusts its position to maintain control; simultaneously, the right hand makes a slight inward rotation of the wheel to initiate a gentle steering correction toward the white track curb ahead, and the car's front bumper begins to approach the curb as the vehicle decelerates slightly, aligning its front edge with the curb's edge while the dashboard display updates to reflect the new position and speed.



Sample ID: `7637_6`

1. **Task 1:** [Man holding black umbrella]: Step off the crosswalk and put down the umbrella parallel to the crosswalk lines.

   Description 1: The man holding the black umbrella continues walking forward, stepping off the crosswalk onto the sidewalk, then lowers his arm and extends his hand toward the curb, releasing the umbrella to rest against the sidewalk edge.

2. **Task 2:** [Black minivan]: Align its front bumper with the white pedestrian lane marking.

   Description 2: The black minivan advances forward while maintaining its current trajectory, its front bumper gradually moving closer to the white pedestrian lane marking on the asphalt, adjusting its position as it proceeds along the crosswalk.

Figure 19: Two representative samples from *WorldTasks*. Each sample includes an initial frame, task prompts, and corresponding descriptive solutions.