From Video Generation to World Modeling

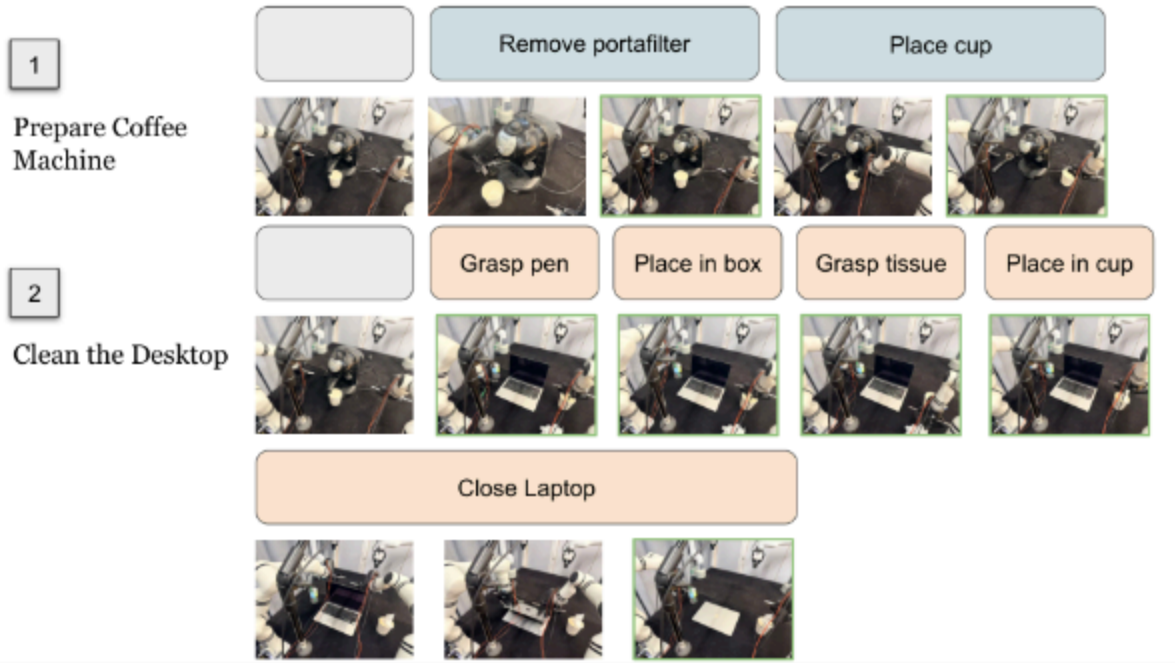
Learning to Predict, Plan, and Act

Harry Yang · HKUST

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Could a generated video help a robot decide?

Real robot tasks, read left to right: preparing coffee and clearing a desk.



A world model predicts how the world changes, including after an action.

Could that prediction help the robot choose what to do next?

Four questions connect the projects

AlignVid: does the video follow the instruction?

PhysHPO: does the event make physical sense?

DMDR: can we generate a good result in fewer steps?

Ongoing robot work: can we learn from the parts that went right?

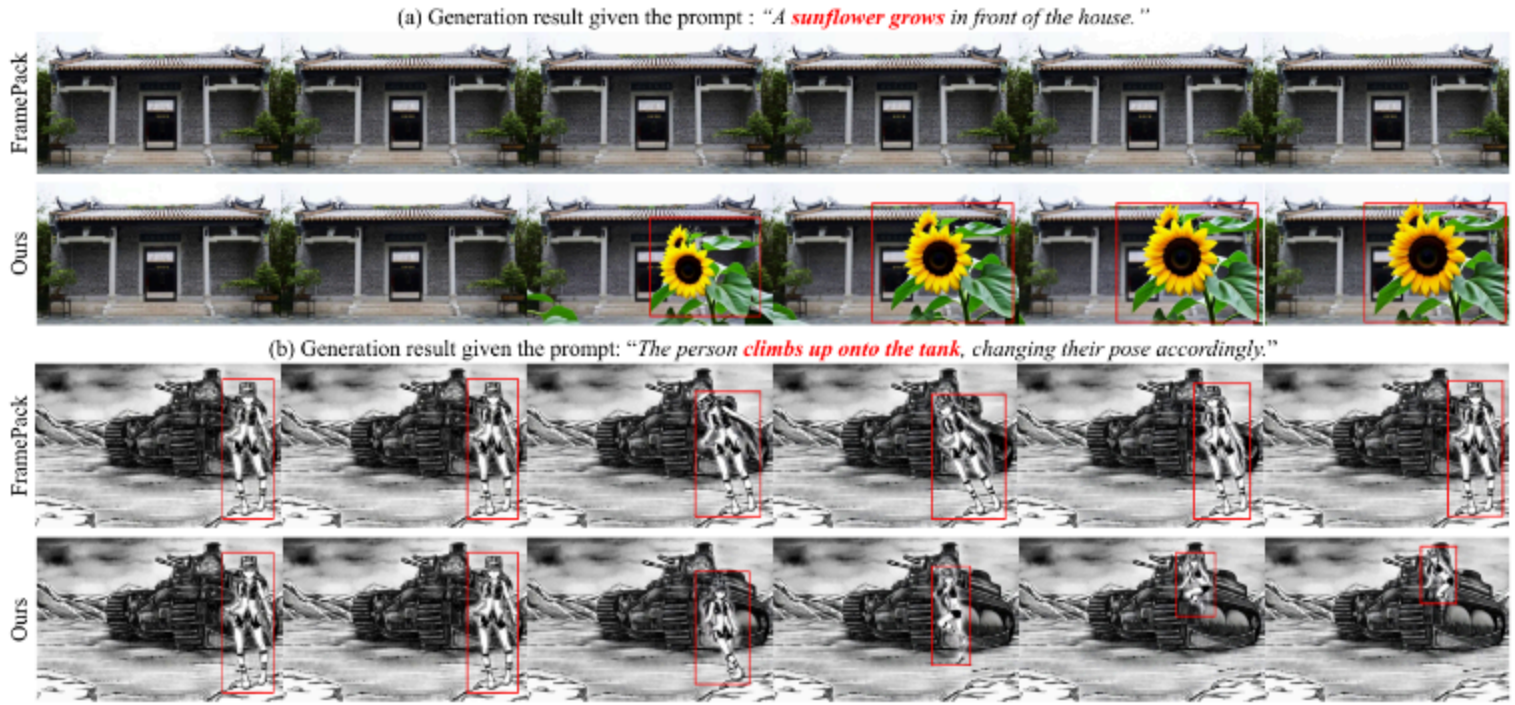
01 AlignVid

An input image tells us where to start. The text tells us what should change.

ICML 2026 · Liu et al.

“A sunflower grows in front of the house”

Each row is a video over time. Compare the original model with AlignVid below it.



FramePack is the original video model; “Ours” adds AlignVid.

The sunflower appears only after we change how the model uses its inputs.

We noticed that blurring the image could help

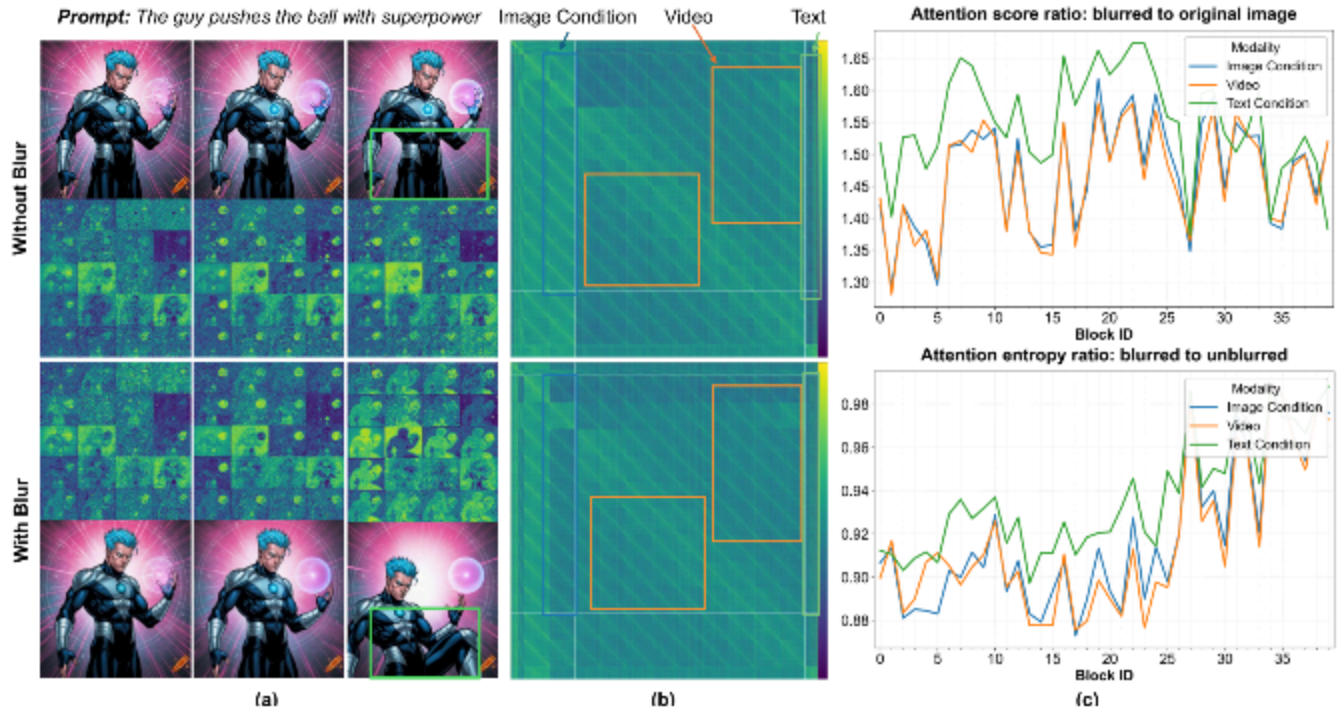
Sometimes, blurring the input image helped the model follow the text.

Attention—the weights used to combine information—became more focused.

Could we get the same benefit without losing image detail?

Blurring the image changes what the model attends to

We change only the input blur, then compare the output and the attention inside the model.



Left: original image above, blurred image below. Middle: attention weights.

Right: blur / original ratios. Entropy below 1 means more focused attention.

We sharpen attention by scaling its scores

Attention Scaling Modulation (ASM): increase the gaps between attention scores.

$$p_j(\alpha) = \frac{e^{\alpha z_j}}{\sum_k e^{\alpha z_k}}$$

$$\frac{dH}{d\alpha} = -\alpha \ \mathrm{Var}_{p(\alpha)}(z) \leq 0$$

z = scores before softmax; p = attention weights; α > 0 = scaling strength.

H measures how spread out the weights are. Larger α makes them less spread out.

This explains the focusing effect. We still need to test whether it helps the video.

We apply the scaling only where it helps

Guidance Scheduling: two switches choose the layers and denoising steps to change.

$$g_{\ell,t} = 1 + b_\ell m_t(\gamma - 1)$$

$$A_{\ell,t} = \text{softmax}\left(\frac{Q(g_{\ell,t}K)^\top}{\sqrt{d}}\right)V$$

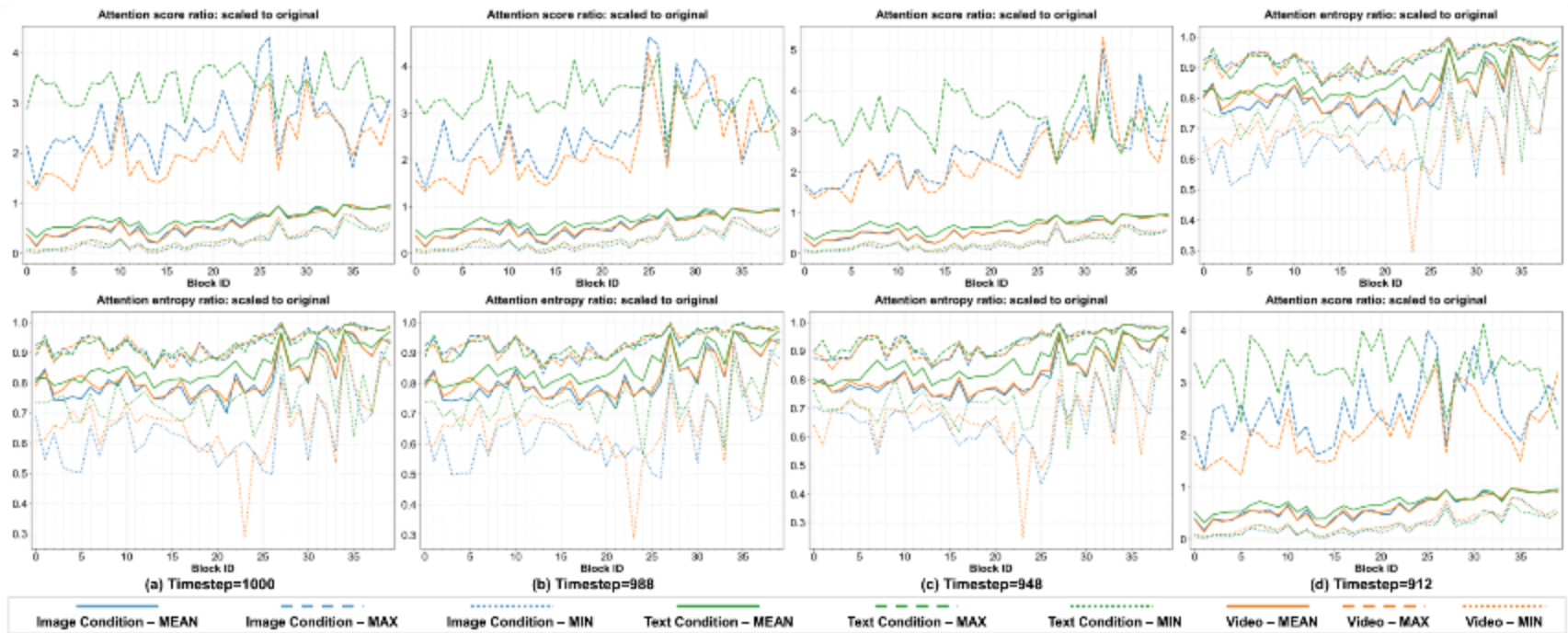
b selects a layer; m selects a step. Both must be on to use gain γ > 1.

Q, K and V are attention queries, keys and values; d is the key dimension.

We favor foreground-focused layers and selected steps. The model is not retrained.

Attention changes differently across layers and steps

Each column is a denoising step; each horizontal axis runs through model layers.



Blue = image, green = text, orange = video. Every ratio compares scaling with no scaling.

Entropy ratios below 1 mean more focused attention. The effect varies, so we use the switches.

Did the model actually make the requested edit?

We built OmitI2V: 367 image-and-text examples with requested visual edits.

For each video, we ask: did it modify, add, or remove the requested content?

We also score visual quality, because following the text can come at a cost.

Wan 2.1 follows the text better, with some loss in quality

OmitI2V score ↑	Wan 2.1	+ AlignVid
Modification	72.35	77.20
Addition	71.75	79.54
Deletion	63.13	69.47
Aesthetic quality	63.12	61.63

On Wan 2.1, adding content improves from 71.75 to 79.54: a gain of 7.79 points.

Higher is better in each row. Aesthetic quality falls by 1.49 points.

The event happens. Does it make physical sense?

THE NEXT QUESTION

The sunflower now appears when we ask for it.

But does the event unfold in a physically reasonable way?

02 PhysHPO

Teach the model which part of a video went wrong

NeurIPS 2025 · Chen et al.

A video can go wrong in different ways

Each pair shows the original model above and PhysHPO below; time runs left to right.

Watch the liquid change, the object hit the liquid, and the apple being peeled.

The training signal should distinguish a wrong state, wrong motion, and missing content.

DPO learns from a preferred and a rejected video

Direct Preference Optimization (DPO): learn to favor one video over another.

$$\Delta_\theta = \log \frac{p_\theta(y_w \mid x)}{p_{\text{ref}}(y_w \mid x)} - \log \frac{p_\theta(y_l \mid x)}{p_{\text{ref}}(y_l \mid x)}$$

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E} \log \sigma(\beta \Delta_\theta)$$

x = prompt; y_w = preferred video; y_l = rejected video; β sets the strength.

Δ compares their likelihood ratios. A larger Δ means a stronger preference.

This is the conceptual form; diffusion training estimates it through denoising.

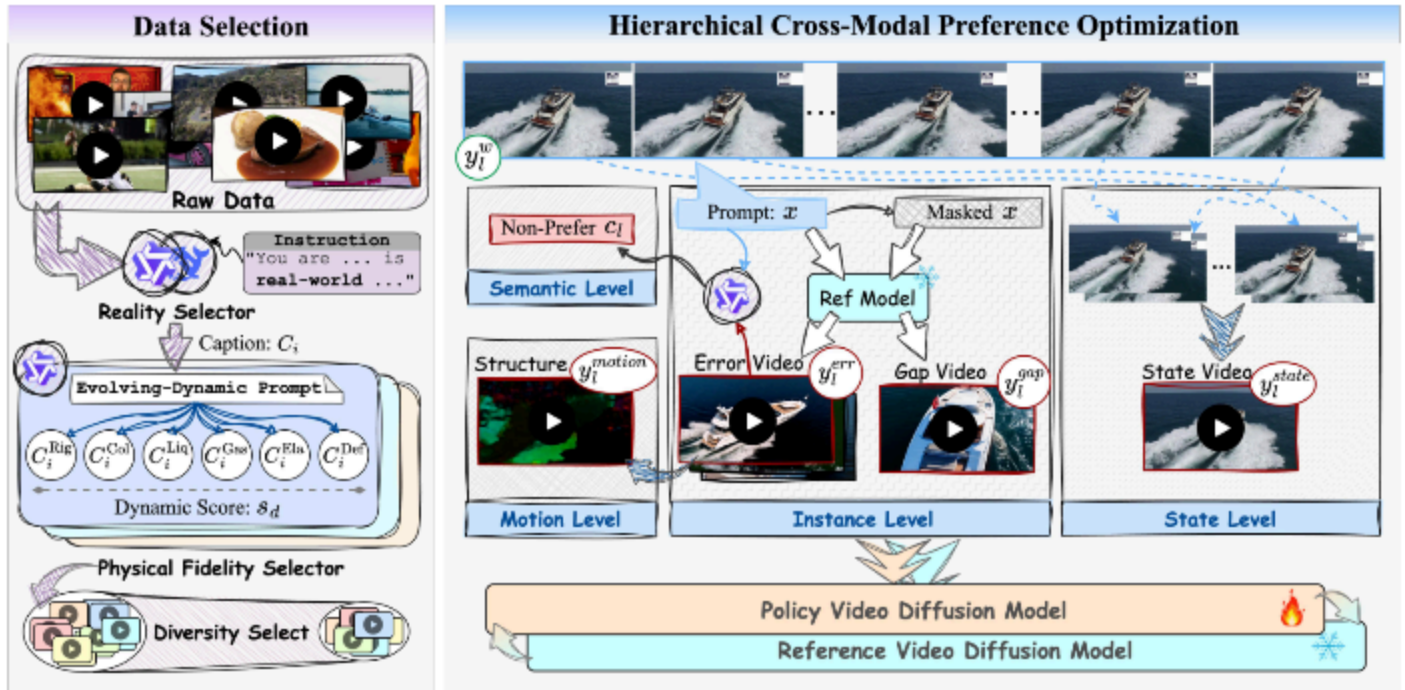
We compare the videos in four ways

Level	What we compare
Instance	Whole clips with errors or missing content
State	First and last frames
Motion	How objects move
Semantic	Captions with changed event details

PhysHPO adds specific comparisons to ordinary whole-video preference training.

We use these comparisons to train one generator

Left: choose videos. Right: create comparisons. Bottom: train the generator.



Use a real clip as the preferred example, then build alternatives with specific errors.

All four signals train the policy model. The reference model stays fixed.

Each loss term helps catch a different mistake

One objective combines whole-video, endpoint, motion, and caption comparisons.

$$\mathcal{L} = \mathcal{L}_{\text{Instance}} + \lambda \mathcal{L}_{\text{State}} + \rho \mathcal{L}_{\text{Motion}} + \mu \mathcal{L}_{\text{Semantic}}$$

λ, ρ and μ set how much each extra comparison contributes.

Matching the first and last frames still allows impossible motion in between.

Motion checks the path between frames; captions check the event against the text.

We select videos with useful physical examples

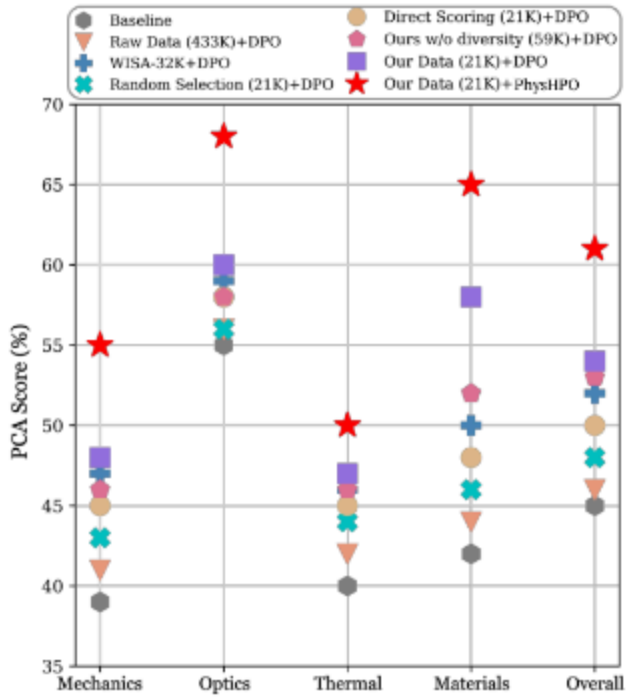
We select about 21,000 videos with realistic behavior and varied events.

They come from a pool of 433,000 videos; more data is not always more useful.

For a fair test, ordinary DPO and PhysHPO use the same selected videos.

Choosing varied examples helps more than adding similar ones

Vertical axis: physical commonsense score (%). Each group tests a type of physical behavior.



Purple squares: the varied 21K set beats the 59K set shown by pink markers.

Then compare red stars: PhysHPO improves further on the same 21K videos.

PhysHPO improves on DPO using the same data

CogVideoX-5B	VideoPhy ↑	PhyGen ↑	VBench ↑
Base	39.6	0.45	81.9
Vanilla DPO	41.3	0.54	82.4
PhysHPO	45.9	0.61	82.8

VideoPhy and PhyGen test physical plausibility; VBench tests general video quality.

Higher is better. DPO and PhysHPO use the same selected data and base generator.

The score drops when we remove the finer comparisons

Training comparisons	VideoPhy ↑
Full hierarchy	45.9
Without semantic level	45.1
No semantic or motion terms	43.6
Whole-clip comparisons only	41.9
Same-data vanilla DPO	41.3

VideoPhy measures physical plausibility; higher is better.

The first four rows remove terms in sequence. The last row is ordinary DPO.

The robot also needs the result in time

THE NEXT QUESTION

A useful prediction also has to arrive before the robot makes its decision.

The next study asks how to generate faster while keeping output quality.

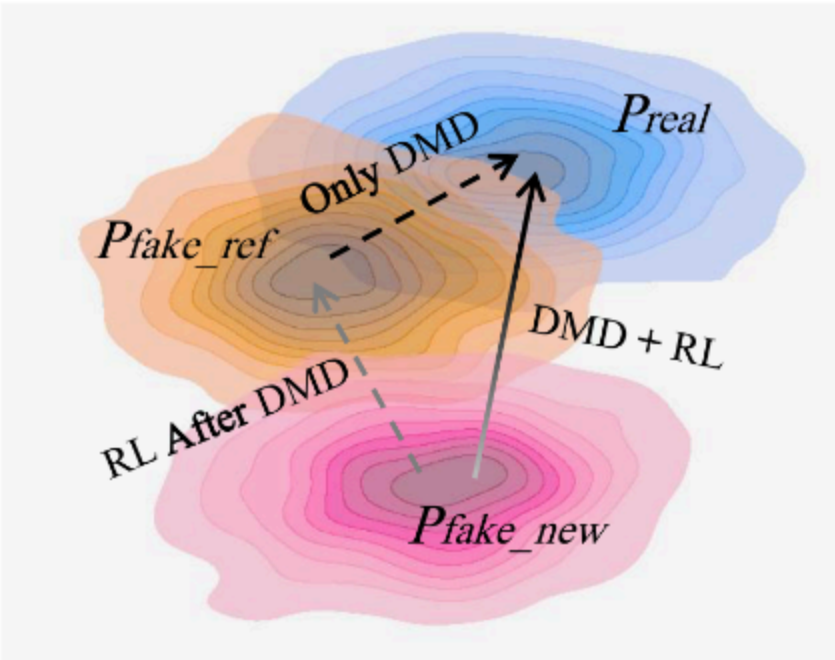
03 DMDR

Make image generation faster while keeping the outputs we prefer

ECCV 2026 · Jiang et al.

A fast student learns from both a teacher and rewards

DMD = Distribution Matching Distillation. Clouds show output distributions; arrows show training.



Distillation teaches a fast student to copy a slower teacher. DMD matches their distributions.

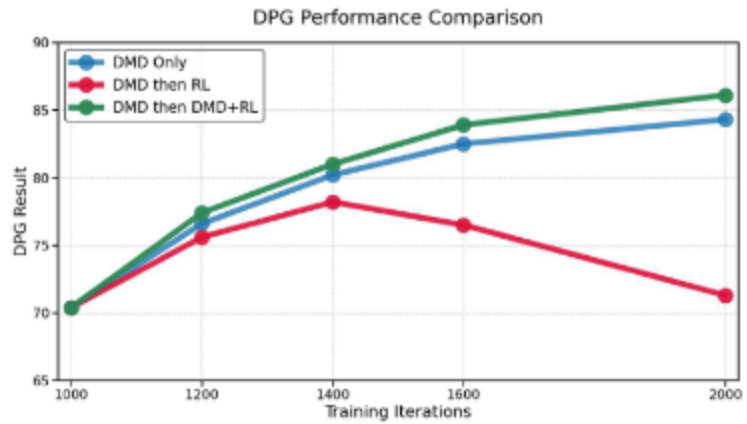
DMDR keeps this teacher feedback during reinforcement learning (RL) from rewards.

A higher training reward can hide worse results

DPG = Dense Prompt Graph Benchmark: how well images follow detailed prompts.

(a) Training Reward Score Comparison

(b) Test-set Reward Score Comparison



(c) DPG Score Comparison

Left: training reward. Middle: test reward. Right: prompt following (DPG). Higher is better.

Red uses reward alone; green keeps teacher feedback. Red gains reward but loses prompt following.

Distribution Matching Distillation (DMD)

Generate an image, add noise, and compare the directions suggested by teacher and student.

$$\nabla_\theta \mathcal{L}_{\text{DMD}} \approx -\mathbb{E}\left[w(t)(s_T(x_t, t) - s_\theta(x_t, t)) \frac{\partial G_\theta(z)}{\partial \theta} \right]$$

A score is a gradient of log density: a direction toward more likely noisy samples.

sT is the teacher score; sθ estimates the score of current student outputs.

Their difference guides the generator update. A separate model learns sθ.

Two changes help when the student is still learning

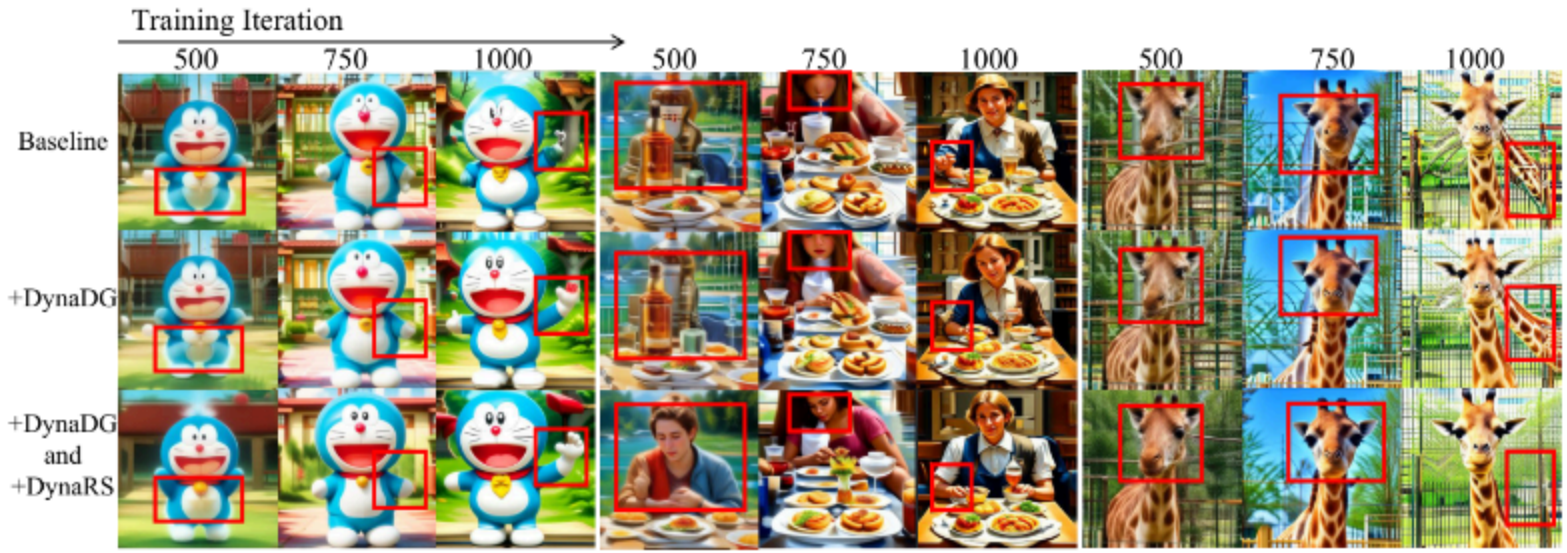
At first, student images are far from the teacher's. Comparing them is difficult.

Dynamic Distribution Guidance (DynaDG) temporarily brings the teacher score closer.

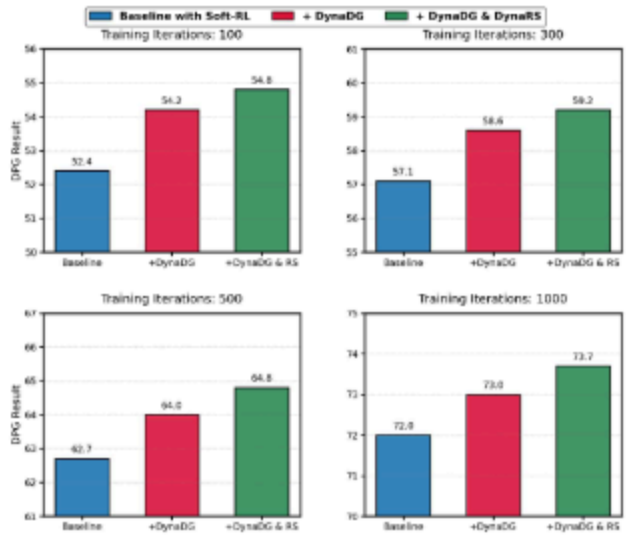
Dynamic Renoise Sampling (DynaRS) adds more noise at first, making comparison easier.

Objects take shape earlier with these training changes

Columns show later training checkpoints; rows add the two changes from the previous slide.



(a) Visual Comparison



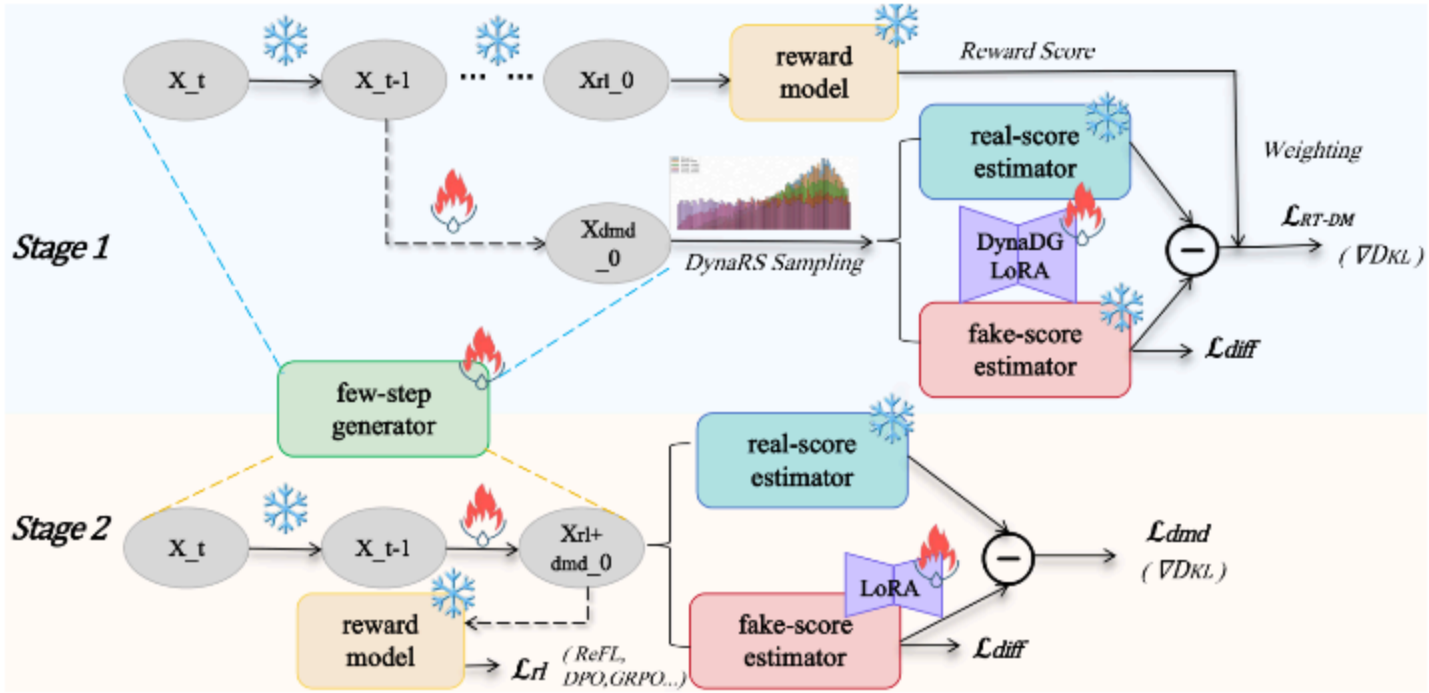
(b) DPG Score Comparison

Compare the highlighted objects at the same checkpoint: their shapes resolve earlier.

Right: DPG checks prompt following. Higher bars support the visual comparison.

Two stages: learn from the teacher, then refine with rewards

Top = stage 1; bottom = stage 2. Snowflakes mark fixed models; flames mark trainable parts.



First: reward decides how strongly to learn from each teacher–student comparison.

Then: optimize reward directly too. Keep DMD active so the teacher still guides training.

The final objective keeps both sources of feedback

Reward favors outputs we like; distribution matching keeps the teacher involved.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{DMD}} + \lambda_{\text{RL}}\mathcal{L}_{\text{RL}}$$

Reward favors preferred outputs; DMD keeps the student close to the teacher.

λRL controls the balance between them.

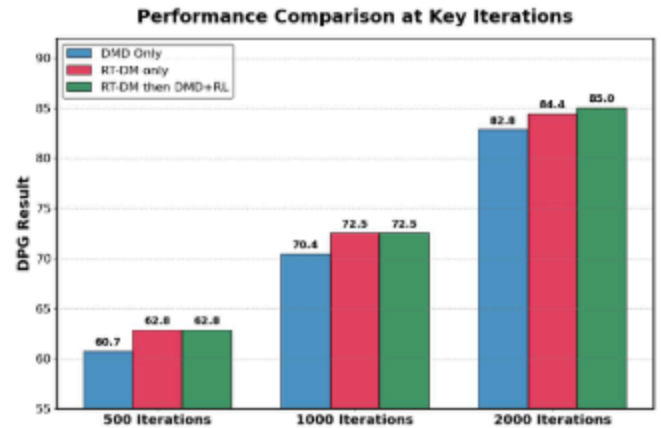
A better training objective still needs an independent check of output quality.

The second stage helps after early progress levels off

Blue = DMD alone; red = reward-weighted DMD; green adds direct reward learning in stage 2.

(a) Training Reward Score Comparison

(b) Test-set Reward Score Comparison



(c) DPG Score Comparison

Training runs left to right. The middle plot uses test reward; the right uses prompt following.

Green improves after the stage switch; keeping reward only as a weight levels off.

DMDR scores higher than DMD2 at the same one-step budget

SDXL model	Steps	CLIP ↑	HP Score ↑
Base	50	34.76	27.15
DMD2	1	34.30	26.60
DMDR	1	35.62	32.01

Steps = generator evaluations. CLIP checks text–image match; HP scores human preference.

Compare the two one-step rows. These step counts do not measure total system latency.

Can a fast model also learn to act?

THE NEXT QUESTION

We have seen how rewards can improve a fast image generator.

Can we use a similar idea to generate useful robot actions?

04 Drifting Policies

Generate actions quickly and learn from the parts the robot got right

Ongoing work · preliminary results

From images and an instruction to a short action sequence

A vision-language-action model (VLA) uses what it sees and what we ask to choose actions.

$$o = (I, \ell, s), \qquad a = (a_1, \ldots, a_T)$$
$$a = f_\theta(\varepsilon, h(o)), \qquad \varepsilon \sim \mathcal{N}(0, I)$$

The action expert is the part that turns these features into robot commands.

An action chunk is a short sequence of commands, generated in one forward pass.

o = images, text and state; a = action chunk; ε = random noise for varied candidates.

Teach the action generator which examples to move toward

A drifting field is a direction: toward successful actions and away from negative targets.

$$V(x) = V^{+}(x; y^{+}) - V^{-}(x; y^{-})$$

$$\mathcal{L} = \frac{1}{G} \sum_{i} \| x_i - \mathrm{sg}[x_i + V(x_i)] \|^2$$

x = generated action; V = movement direction; G = number of candidates.

sg means stop-gradient: hold the target x + V fixed while updating the generator.

Positive targets attract; negative targets or other generated candidates provide repulsion.

Give credit for the parts of a task the robot completed

For example: picking up the cup is useful even if preparing the coffee later fails.

$$\widehat{A}(s) = z_s - \bar{r}_k, \qquad \bar{r}_k = \frac{1}{|S_k|} \sum_{s \in S_k} z_s$$

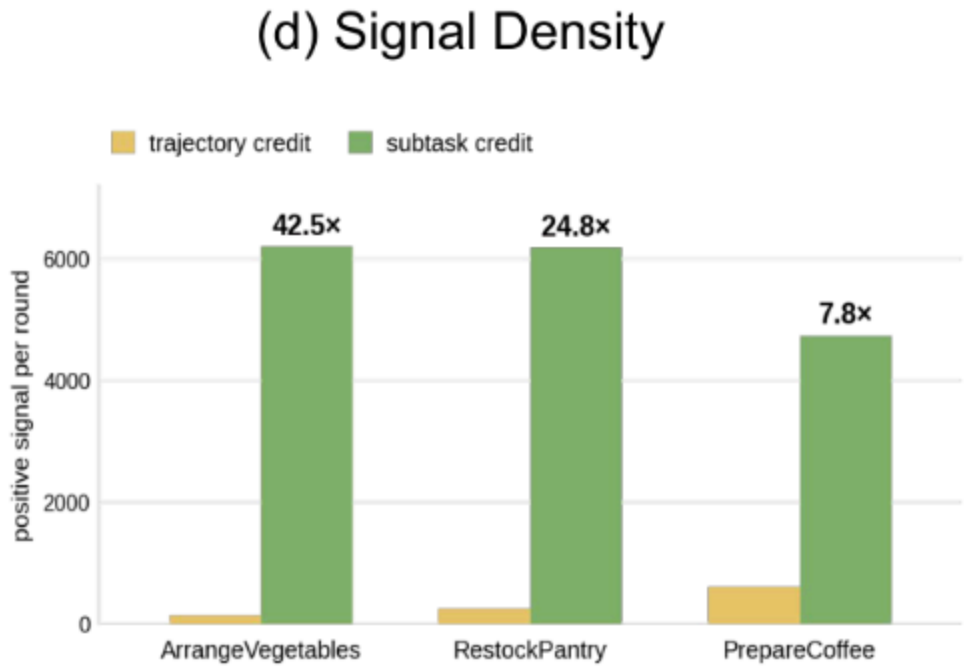
A verifier checks task-state rules and requires success to persist.

Each segment gets credit relative to that subtask's average success rate.

z = subtask success; r̄ = its average success rate; Â = credit for the segment.

Partial success gives us more feedback to learn from

Vertical axis: positive training signals per round. Each pair uses the same robot attempts.

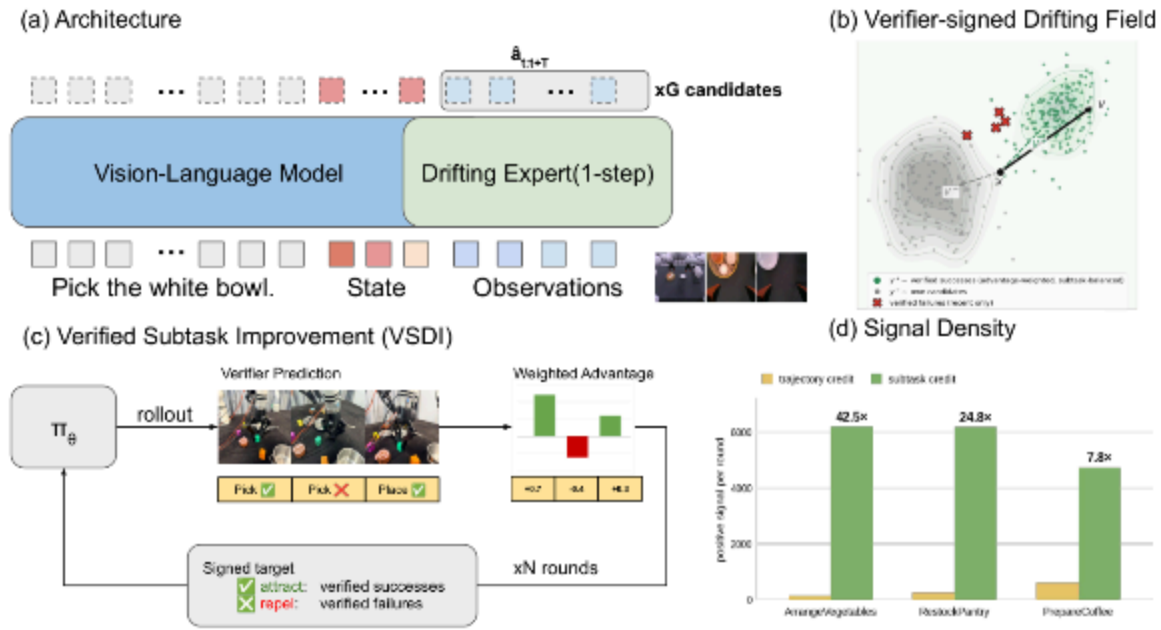


Yellow counts only whole-task success; green also credits completed subtasks.

For arranging vegetables, subtask credit gives 42.5 times as many positive signals.

Try tasks, check completed subtasks, then train again

Verified Subtask Improvement (VSDI) is this learning loop; panel (c) shows the steps.



Top: generate action candidates. Bottom left: check results and build new training targets.

Keep successful examples, use failures as negatives, and retain demonstrations.

Train from the base checkpoint, then compare with the current policy before accepting it.

Familiar task combinations improve; new ones are harder

Task split	π0.5	GR00T N1.5	Ours
Single tasks	39.6	43.0	45.2
Familiar combinations	7.1	9.6	16.7
New combinations	1.2	4.4	2.1

RoboCasa365 robot tasks: success (%) on single tasks and familiar or new combinations.

Preliminary. Baseline results come from different training setups; compare with care.

Both the direct action generator and subtask credit help

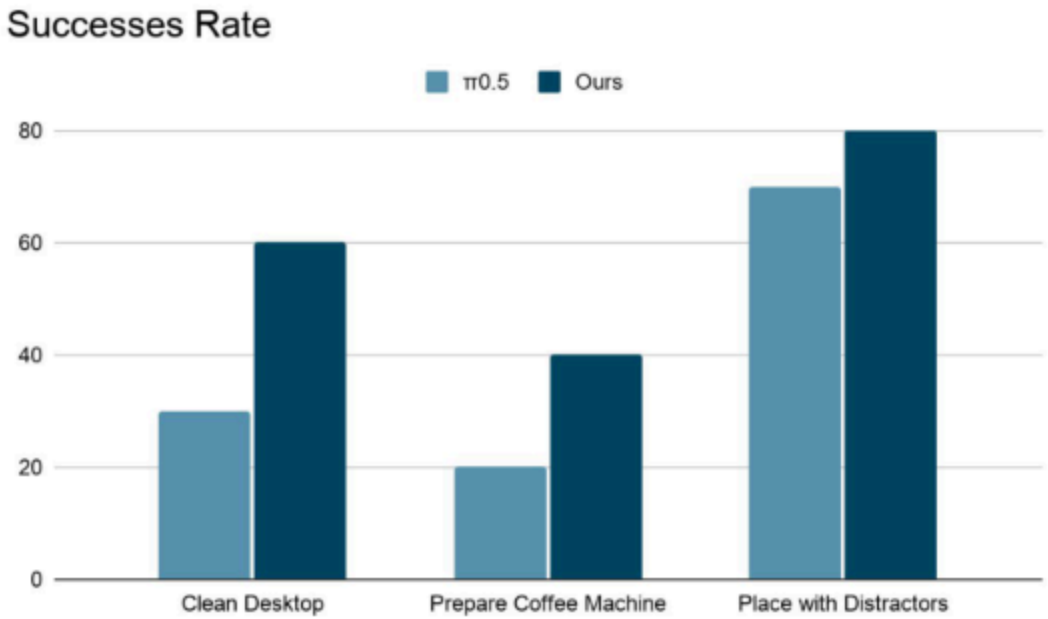
Training method	Success ↑
Original action expert, imitation only	6.5
Drifting expert, imitation only	10.8
Credit for the whole task only	11.4
Full method (VSDI)	16.7

Success (%) on familiar task combinations, using our own training setup.

The full method combines drifting actions and verified subtask credit. Preliminary results.

Early real-robot results improve on three tasks

Success (%) on three real-robot tasks. Light blue = π0.5 baseline; dark blue = our method.



Ten trials per task: successes rise from 3 to 6, 2 to 4, and 7 to 8.

These are early results. More trials and more varied tasks are needed.

The missing step: test whether predictions help decisions

The model must predict what changes when the robot chooses a different action.

It must track objects over time and represent what it is uncertain about.

Keep the planner fixed, change the prediction model, and measure task success.

Back to the robot making coffee

It needs the instruction to change what the model predicts.

It needs plausible consequences, generated in time to choose an action.

Then it needs feedback from what actually happened.

Backup · Why attention entropy goes down

$$\frac{d}{d\alpha}\left(\log Z - \alpha \mathbb{E}_p[z]\right) = \mathbb{E}_p[z] - \mathbb{E}_p[z] - \alpha \operatorname{Var}_p(z)$$

H = log Z − α E[z]; dE[z]/dα = Var(z).

This result holds for one attention row with fixed scores.

Backup · What each comparison tests

AlignVid: the same model with and without attention scaling.

PhysHPO: the same data with DPO or the four-level objective.

DMDR: the same number of steps; deployment also needs runtime tests.

Backup · These robot results use different training setups

LIBERO-Plus: 85.2 zero-shot; 90.8 after supervised fine-tuning.

The gap is not an improvement from reinforcement learning.

The real-robot study has only ten trials per task.

Papers and ongoing work

AlignVid · ICML 2026 · arXiv:2512.01334v2

PhysHPO · NeurIPS 2025 · arXiv:2508.10858

DMDR · ECCV 2026 · arXiv:2511.13649v5

Drifting Policies for VLA · ongoing manuscript