Reliability-Aware World Model for Robot Policy Improvement

Anonymous authors
Paper under double-blind review

Abstract

Robot policies can improve by practicing inside a learned world model, but the model is wrong in places, and practicing there teaches the policy behaviors that fail on the real robot. Fixing the model needs new real data, and the usual way to get it is to run the policy on the robot and refit; most of those rollouts, however, land on interactions the model already predicts well, and few reach the places where it is wrong. We present a method in which the world model's own prediction says how far it can be trusted. A reliability score, computed from the self-consistency of the flow model's velocity field and from whether the predicted motion follows the commanded action, is used twice: it weights each imagined policy update, and it selects the interactions at which real data is collected. The world model is a video transformer with a dynamics head and an action- and contact-conditioned value head on a shared backbone, and its action-value gradients drive policy improvement. A flagged interaction is reached on the robot by restoring the logged start and replaying the action prefix; the real outcome refits the model before the next round. On three simulation and three real-robot tasks, two rounds of 20 real rollouts improve success over the initial policy by 15.8–17.1 and 25.0–28.1 points and are ahead of DSRL and a VLAW reproduction on every task at the same rollout budget. Controlled studies show that both uses of the score contribute, and that the collected data corrects the model where it was flagged. Project page.

1 Introduction

World models enable robot policies to learn through imagined interaction: a model predicts future observations from the current context and policy actions, and the policy improves on the generated experience, as in Ctrl-World, RISE, and WMPO (Guo et al., 2025; Yang et al., 2026; Zhu et al., 2025). But policy updates change which actions are attempted and where, and take the model to interactions the available data covers poorly. There it can generate distorted scenes, or plausible outcomes that do not follow the commanded actions. If such outcomes, or the inflated action values computed from them, guide learning, the policy is rewarded for progress that would not occur on the robot, and later optimization favors the overestimated behavior. Sustained improvement therefore requires identifying and correcting the prediction problems encountered during learning.

Recent work addresses this by improving policy and model together. VLAW collects current-policy trajectories and trains the world model on new and historical experience (Guo et al., 2026); WoVR periodically collects rollouts of the updated policy to keep policy and model aligned (Jiang et al., 2026). Such collection follows the policy's visitation distribution, but visitation frequency does not indicate prediction accuracy: frequent interactions may already be modeled well, while an important error may arise only for a particular state–action pair. A limited budget then revisits familiar interactions while problematic predictions wait. Imagined attempts, however, expose predicted consequences and locate suspicious interactions; using them to choose where real data is collected makes the new experience directly relevant to the problems exposed during learning. We score the reliability of every imagined prediction with two signals from the world model itself: the self-consistency of its velocity field, and whether the motion visible in the predicted frames follows the commanded action. The score both weights policy updates and chooses where real feedback is collected. The world model, a shared video transformer with a dynamics head and an action- and contact-conditioned Q-head, predicts future observations and action values, and Q-learning with adjoint matching (QAM) uses the action-value gradients to improve the policy (Li & Levine, 2026). The score down-weights

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.

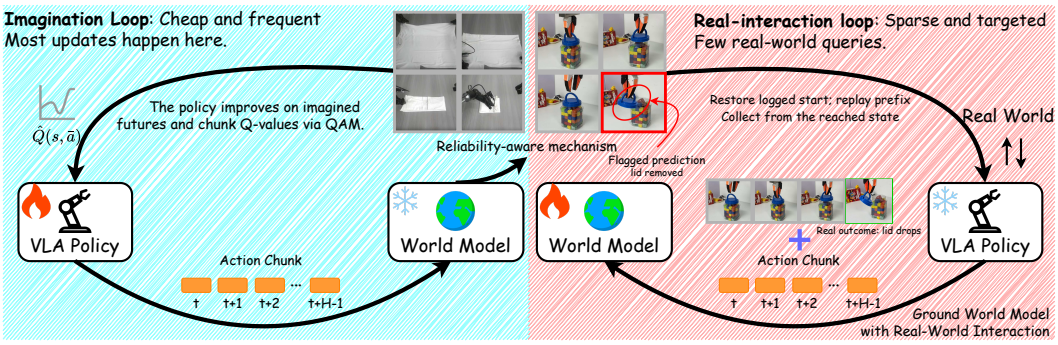


Figure 1: Two connected learning cycles. The reliability score weights policy updates in the left cycle and selects real feedback in the right cycle. Selected interactions are approached by restoring a logged start and replaying its action prefix; actual outcomes update the world model. Flames mark the component being updated; snowflakes mark the frozen component.

unreliable updates; above a threshold it stops the imagined rollout and records the preceding interaction as a collection candidate. For selected candidates the robot restores the logged initial scene, replays the saved action prefix, and collects from the state actually reached; the new outcomes and the historical data update the dynamics and value estimates for the next round (Figure 1).

We instantiate the method with a flow-matching vision–language–action policy on three RoboCasa and three real-robot tasks. Two feedback rounds improve success over the initial policy by 15.8–17.1 points in simulation and 25.0–28.1 on hardware. The contribution is the complete procedure connecting score-weighted, value-guided imagination to executable, targeted model updates; controlled studies attribute the gain to its parts: the two uses of the score, the information in the weights, the correction of the model where it was flagged, and the fidelity and cost of reaching them on hardware.

2 Related work

Policy improvement with world models. Supervised fine-tuning on demonstrations produces capable vision–language–action policies (Kim et al., 2024; Octo Model Team et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025b), but their competence is bounded by demonstration coverage. One direction collects on-robot experience and improves the policy by reinforcement or human-in-the-loop learning (Luo et al., 2024; Wagenmaker et al., 2025; Physical Intelligence et al., 2025a), which is effective but paced by serial physical interaction. Another lets a learned video world model act as the simulator: Ctrl-World, RISE, WMPO, VLA-RFT, and WoVR improve policies in imagination (Guo et al., 2025; Yang et al., 2026; Zhu et al., 2025; Li et al., 2025; Jiang et al., 2026), and VLAW and WoVR co-train the model on rollouts of the improving policy (Guo et al., 2026; Jiang et al., 2026). These methods trust the model wherever the policy explores and refresh it along the policy's visitation distribution; we instead measure the reliability of each prediction and spend the real budget where the model was found unreliable. The policy optimizer, QAM (Li & Levine, 2026), builds on adjoint matching (Domingo-Enrich et al., 2025).

Model uncertainty in model-based reinforcement learning. Estimating where a learned model can be trusted has a long history: ensemble disagreement sets rollout length and data use in MBPO and PETS (Janner et al., 2019; Chua et al., 2018), penalizes returns in MOPO and MOReL (Yu et al., 2020; Kidambi et al., 2020), and directs exploration in Plan2Explore (Sekar et al., 2020), while COPlanner rolls out conservatively but explores optimistically (Wang et al., 2024). These estimates serve compact dynamics models and regulate the policy's own exploration. Our score is computed on generated video, from the flow model's velocity field and the action consequences visible in its predictions, and yields explicit collection requests executed independently of the policy.

Reaching states on a real robot. Autonomous reinforcement learning studies how a robot returns to useful states without human resets (Eysenbach et al., 2018; Sharma et al., 2022). Our collection interface has the opposite structure: it must reach one specific logged state, by scripted perception, placement, and verification followed by action replay, in the perception–planning–verification style

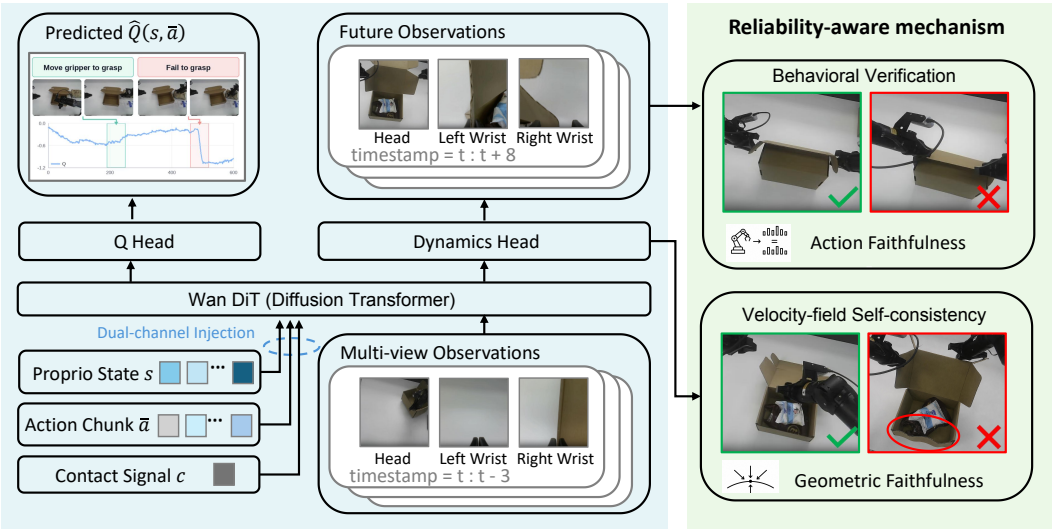


Figure 2: Architecture of the unified world model. The dynamics head predicts the next eight future frames, and the Q-head reads the shared features to predict $\hat{Q}(s, \bar{a})$. The reliability-aware mechanism (right) scores the prediction for geometric and action faithfulness; the small contact head is not drawn.

of programmatic robot control (Liang et al., 2023; Fu et al., 2026) and the reset–execute–verify loop of ENPIRE (Xiao et al., 2026). Unlike those systems, restoration here serves a targeted query, and its fidelity is measured against the imagined state it must reproduce.

3 Method

3.1 Problem statement

We improve a robot policy π_ψ pre-trained by supervised fine-tuning on a dataset $\mathcal{D}_{\text{demo}}$ of successful and failed trajectories; in our experiments π_ψ is the flow-matching vision–language–action policy π_0 (Black et al., 2024). At each step the agent observes a proprioceptive state s and a multi-view observation o, and the policy produces an action chunk $\bar{a} = (a_1, \ldots, a_H)$ of horizon H.

We assume access to (i) $\mathcal{D}_{\text{demo}}$, which trains both the policy and the world model, and (ii) a real environment, a simulator or a physical robot, in which B rollouts can be collected per outer iteration. The objective is to improve π_ψ with most policy updates performed inside imagined rollouts of the world model. The difficulty is that these updates change the actions π_ψ proposes and the states it reaches, taking the model to interactions that $\mathcal{D}_{\text{demo}}$ covers poorly, where its predictions can be unfaithful and an update computed on them rewards progress the real robot would not achieve. The method therefore has to limit the influence of a prediction that cannot be trusted, and direct the real-rollout budget to the interactions where such predictions arise. These two requirements define the two cycles of Figure 1: in the imagined cycle the world model is frozen and the policy is updated on imagined rollouts; in the real cycle the policy collects real rollouts and the world model is updated.

3.2 Unified action-and-contact-conditioned world model

The imagined cycle replaces real execution with the world model, which must tell the policy how a candidate action chunk changes the scene and how far it advances the task. On contact-rich tasks it needs two properties. It must depend on the action, not only on the state: the prediction $\hat{o}'$ and the value $\hat{Q}$ take $(s, \bar{a})$ as input, and the policy update follows $\nabla_{\bar{a}} \hat{Q}$, which a state value $V(s)$ or a model ignoring its action input cannot supply. And it must register contact, because whether a grasp holds or slips often decides the task while changing the image only slightly. Our world model is the 5B-parameter video diffusion transformer (DiT) Wan2.2-TI2V-5B (Wan Team, 2025), trained as a

flow-matching predictor of future observations, with two heads on a shared backbone (Figure 2): a dynamics head predicting future multi-view observations $\hat{o}'$ and a Q-head estimating $\hat{Q}(s, \bar{a})$.

Inputs and conditioning. The model receives multi-view observations o (four history frames), proprioceptive state s, action chunk $\bar{a}$, contact signal c, and a language instruction. The contact signal supplies what images show poorly, whether the gripper holds the object; it is the simulator's finger-contact flag and, on the real robots, the normalized gripper motor current, so no sensor is added. Action and contact enter every DiT block via cross-attention and per-frame AdaLN.

Output heads. The dynamics head outputs a flow-matching velocity field, from which the next H frames are generated and decoded into $\hat{o}'$. The Q-head pools features from the first $N{=}15$ DiT blocks, concatenates a state embedding, and maps them to $\hat{Q}(s, \bar{a})$ with a small MLP; it is differentiable in the action, which supplies $\nabla_{\bar{a}}\hat{Q}$. A small contact head reads the same features and predicts contact at the chunk's last frame, which imagination uses as the next chunk's contact input.

Why the two heads share the encoder. Sharing matters for two reasons. First, the Q-head reads features learned from dense video supervision, which value training from scalar rewards on limited data provides only weakly, so $\hat{Q}$ responds to subtle action and contact effects such as a grasp that fails with little visual change. That is where the value must drop, because the policy gradient is placed where the value drops. Second, sharing makes the predicted video a visible readout of the features $\hat{Q}$ is computed from: when those features miss an interaction, the prediction becomes distorted or ignores the action, whereas the scalar value gives no such sign. Checking the predicted video is therefore a way to decide how much a value-guided update should count.

Training. The dynamics head is trained with the flow-matching objective (Lipman et al., 2023) on $\mathcal{D}_{\text{demo}}$. The Q-head is then trained on eight-step chunks with fitted Q evaluation and clipped double Q while the shared DiT stays frozen, so value training does not alter the dynamics head's predictions; the contact head is trained in the same stage with a binary cross-entropy loss against the logged contact at the end of each chunk. The target for a chunk is its discounted return $\sum_{i<H} \gamma^i r_{t+i}$ plus γ^H times the target-network value of the next logged chunk, cut off at episode end. Because imagined rollouts are short compared with an episode, rewards add a progress potential to the sparse terminal reward, $r_t = \gamma\Phi(s_{t+1}) - \Phi(s_t) + r_{\text{sparse},t}$, where Φ is an action-free progress model initialized from Wan2.2 and fine-tuned on manually labeled trajectories, and r_{sparse} is $+1$ for task success and -1 for failure, judged by task predicates in simulation and by humans on the real robot.

Both heads are thus trained on $\mathcal{D}_{\text{demo}}$ and on the real rollouts collected so far, and their predictions are grounded only on the interactions this data covers. The imagined updates below move the policy away from these interactions, so every imagined prediction has to be checked before it is used.

3.3 Reliability-aware mechanism

When the policy queries an interaction the data does not cover, the dynamics head can fail in two qualitatively different ways. *Geometric failures*: the predicted clean latent leaves the manifold of plausible video, and objects distort, change color, or disappear. *Action-faithfulness failures*: the prediction is geometrically plausible but does not follow the commanded action chunk; the model defaults to a familiar trajectory regardless of the input, as is common for conditional generative models under classifier-free guidance. Both present the policy with outcomes the real robot would not produce. We detect them with two signals computed from the world model's own prediction, so that every imagined chunk can be checked without executing it.

Intrinsic signal: velocity-field self-consistency. The dynamics head is a flow-matching velocity field $v_\theta(x_\sigma, \sigma)$ trained to regress the path velocity $z - x$ along $x_\sigma = (1-\sigma)x + \sigma z$. Re-noising a predicted clean latent $\hat{x}$ with fresh (z, σ) and comparing the field with the path velocity therefore evaluates the model's own training loss at $\hat{x}$, which stays at its training-data level on the manifold and grows off it. At inference, under the same conditioning, we sample K pairs and measure the residual

$$r_{\text{intr}}(\hat{x}) = \frac{1}{K}\sum_{k=1}^{K}\Big\| v_\theta\big((1-\sigma_k)\hat{x} + \sigma_k z_k,\ \sigma_k\big) - (z_k - \hat{x})\Big\|_2^2. \tag{1}$$

which is small for geometrically faithful predictions and grows off the training manifold.

External signal: behavioral verification. The intrinsic signal does not catch action-faithfulness failures, since a low r_{intr} can coexist with an unfaithful action. We therefore check action consistency with two auxiliary networks trained on real trajectories only: a visual probe that maps decoded RGB to an end-effector pose (Lin et al., 2025; Arun et al., 1987; Horn, 1987), and an inverse-dynamics module that maps consecutive states to the action between them. Applying the probe to each predicted frame and the inverse-dynamics module to the resulting state sequence recovers the implied action chunk $\bar{a}_{\text{impl}}(\hat{o}')$, and the external score is its L_1 discrepancy from the commanded chunk:

$$r_{\text{ext}}(\bar{a}, \hat{o}') = \left\| \bar{a}_{\text{impl}}(\hat{o}') - \bar{a} \right\|_1. \tag{2}$$

Combining the two signals. The external signal reads the decoded RGB through the probe, so it is informative only when that RGB is plausible. The two signals therefore cover different failures: r_{intr} establishes whether the prediction is geometrically plausible, and r_{ext} whether it follows the commanded action. A prediction is reliable only when both are low, summarized in one per-state scalar

$$r(s, \bar{a}) = w_{\text{intr}}\, r_{\text{intr}}(\hat{x}) + w_{\text{ext}}\, r_{\text{ext}}(\bar{a}, \hat{o}'), \tag{3}$$

with weights $w_{\text{intr}}, w_{\text{ext}}$ chosen so that the in-distribution baseline $\mathbb{E}_{\mathcal{D}_{\text{demo}}}[r]$ is of order one; since both terms are non-negative, either signal alone raises r.

3.4 Closed-loop targeted improvement

An unreliable prediction raises two problems. For the current policy update it should not be a learning signal. For the world model it marks an interaction whose real outcome is missing: down-weighting leaves the model unchanged, and the same interaction would be mispredicted next round. We therefore use the reliability score twice (Algorithm 1 in Appendix D): to weight each policy update within the imagined cycle and, at its end, to select where the real cycle collects data; the interactions whose updates are suppressed most are those whose real outcomes are collected next.

Imagined update with reliability gating. Each inner-loop step samples logged real states from $\mathcal{D}_{\text{demo}}$ and the real rollouts collected so far and rolls the policy forward in the frozen world model chunk by chunk. The first chunk is conditioned on the logged observation, proprioceptive state, and contact signal; every later chunk on the model's own outputs, with the predicted frames as visual history, the proprioceptive state advanced by the commanded chunk, and the contact head's prediction as contact input. The rollout continues until the score exceeds the threshold τ introduced below or the episode ends, and every imagined chunk is a training sample. The policy is updated with Q-learning with Adjoint Matching (QAM) (Li & Levine, 2026), which follows the critic gradient $\nabla_{\bar{a}}\hat{Q}$ under a reference-flow regularizer, training only lightweight adaptation layers. To keep the policy from exploiting unreliable predictions, each sample's contribution is scaled by an exponential gate

$$w(s, \bar{a}) = \exp\big(-\lambda\, r(s, \bar{a})\big), \tag{4}$$

so that highly unreliable samples contribute negligibly, with λ setting how strongly they are suppressed. The weight multiplies the sample's adjoint-matching loss and is detached from the gradient; the loss is averaged over the batch rather than over the weights, so a batch of unreliable samples contributes little instead of being renormalized, and the regularizer is left unweighted.

Discrete trigger for real rollouts. Each imagined rollout begins at a logged real state s_0 and stops at the first chunk k whose score exceeds τ, because every later chunk would start from an unreliable prediction. The last reliable state $\hat{s}_k$, reached from s_0 by the prefix $\bar{a}_{0:k-1}$, is where the real outcome is missing, and it joins a trigger set

$$\mathcal{T} = \big\{ \hat{s}_k : \; r\big(\hat{s}_k, \bar{a}_{\pi_\psi}(\hat{s}_k)\big) > \tau \big\}, \tag{5}$$

where $\bar{a}_{\pi_\psi}(\hat{s}_k)$ is the chunk the policy currently produces at $\hat{s}_k$; one τ is shared across tasks. Per outer iteration, $\mathcal{T}$ is binned by end-effector position (5 cm cells) and gripper state, and B triggers are drawn uniformly across the non-empty bins, at most two per bin; if fewer exist, all are used and the count reported. Selecting triggers by score rather than visitation spends the budget on interactions the model predicts poorly, including ones on-policy collection reaches only after many rollouts.

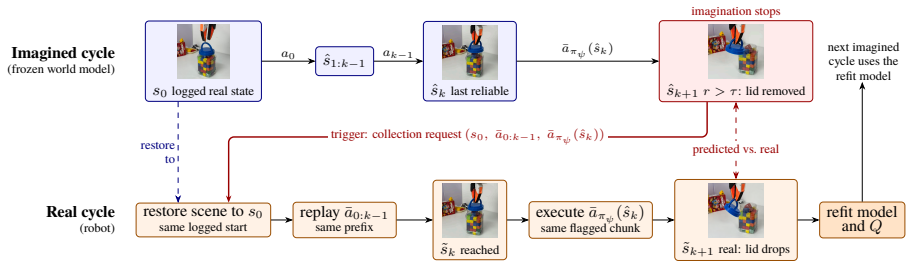


Figure 3: Reaching a trigger on hardware (frames from one OpenJar trigger). The imagined cycle (top) starts from a logged real state s_0 and stops at $\hat{s}_{k+1}$, where the model predicts the lid coming off; the flag issues a collection request with s_0, the prefix $\bar{a}_{0:k-1}$, and the flagged chunk. The real cycle (bottom) restores the same start, replays the same prefix to reach $\tilde{s}_k$, and executes the same chunk; the lid actually slips and drops. Predicted and reached states at chunk k look identical, so one frame serves for both. The difference at $k+1$ is what the refit corrects.

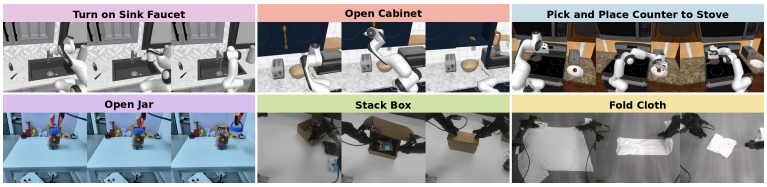


Figure 4: The six tasks. Simulation: turn the sink faucet to a target angle, open the cabinet, and move an object from counter to stove. Hardware: rotate and detach the jar lid (OpenJar); place two bagged items in the box and close it (StackBox); fold a spread-out garment into a neat block (FoldCloth).

Reaching a trigger and updating the world model. A trigger is an imagined state; to obtain its real outcome, the robot must reach the corresponding physical interaction, which it does by reconstructing the history of the imagined rollout (Figure 3). The simulator resets to s_0 exactly. On the physical robot, a scripted RGB-D routine re-estimates each object's pose and moves it back to its logged pose, verifying the result and retrying on failure; FoldCloth is restored to its logged coarse configuration (Appendix C). Replaying the logged prefix $\bar{a}_{0:k-1}$ then re-establishes contact-rich and in-hand configurations. From the state actually reached the robot first executes the flagged chunk $\bar{a}_{\pi_\psi}(\hat{s}_k)$, so that the real outcome answers the action the model was flagged on, and then continues under π_ψ. The recorded trajectory joins the training set, and the world model and Q-head are refit together with a replay batch drawn uniformly over the state space of the original data.

The refit model is frozen for the next imagined cycle, which now has real outcomes where the previous one stopped. The two cycles alternate for a fixed number of outer iterations (two in our experiments).

4 Experiments

4.1 Setup

We evaluate on three RoboCasa simulation tasks (Nasiriany et al., 2024) and three physical tasks on two robot platforms (Figure 4; Appendix A), with π_0 as the policy (Black et al., 2024). Each task starts from 150 logged trajectories (100 successful, 50 failed) that train the policy, world model, and progress model. *Imagined* is the policy after imagined improvement, before any new real data; *Online 1* and *Online 2* follow one and two feedback rounds of $B = 20$ real rollouts each. Simulation success is measured over 170 episodes per cell and real-robot success over 32 trials.

We compare against two ways of spending the same 40 real rollouts. DSRL (Wagenmaker et al., 2025) uses them to train the policy directly on the robot; VLAW†, our reproduction of VLAW on the Ctrl-World codebase (Guo et al., 2025; 2026), uses them to co-train the world model. Both collect current-policy rollouts from logged starting states, as WoVR does to refresh its model (Jiang et al., 2026). A *random-candidate* control isolates our selection rule: it collects the same number of rollouts with the same restoration and replay, but picks its B targets uniformly at random from the states the imagined rollouts passed through, without looking at the score.

4.2 Policy improvement

Table 1 shows the same picture on all six tasks. Imagined improvement alone raises success over π_0 without new interaction, and each feedback round raises it further. After two rounds the policy succeeds 15.8–17.1 points more often than π_0 in simulation and 25–28 points more often on the real robots, and it beats the better of DSRL and VLAW† on every task.

Table 1: Success rates (%) over 170 simulation episodes or 32 real trials per cell. DSRL and VLAW† receive 40 rollouts per task; our three stages receive 0, 20, and 40.

Task	π_0	DSRL	VLAW†	Imagined	Online 1	Online 2
SinkFaucet	27.6	32.4	39.4	31.8	34.7	**43.5**
OpenCabinet	21.8	34.1	28.8	23.5	27.1	**37.6**
PickPlace	25.3	25.9	30.6	28.8	32.4	**42.4**
OpenJar	31.3	50.0	53.1	40.6	53.1	**59.4**
StackBox	6.3	18.8	21.9	12.5	25.0	**34.4**
FoldCloth	9.4	18.8	31.3	18.8	28.1	**34.4**

Thirty-two real trials give a 95% confidence interval of roughly ± 16 points, and 170 simulation episodes roughly ± 7 points; the PickPlace margin over the stronger baseline is 11.8 points (95% interval [1.6, 21.9]). The table establishes the ordering $\pi_0 <$ Imagined $<$ Online 1 $<$ Online 2 on every task, and Online 2 ahead of both baselines on every task. Training-seed variation is small: three seeds of the full PickPlace loop (Table 6) give per-cell standard deviations of 0.9–2.2 points.

4.3 An action-responsive value

Section 3.2 argued that the value must drop at the action causing a failure, where the policy gradient lands, and that contact input and the shared encoder enable this. The critic does this on real rollouts: on successful StackBox and FoldCloth rollouts the value rises one step per subtask, not only at termination, and on failed rollouts it drops as soon as the robot fails to establish stable contact, well before the task ends (Appendix Figure 7). Figure 5 shows a grasp that fails at frame 67 with little visible change: the action-conditioned value drops at that frame; a vision-only value drops 15–20 frames later, once the failure is visible.

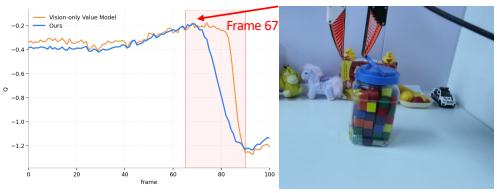


Figure 5: Action-conditioned versus vision-only value on a grasp-failure example. At the marked frame 67, the grasp fails with little visual change; the action-conditioned estimate falls earlier than the vision-only one.

Table 2 measures this on PickPlace. At each of 100 held-out contexts, four candidate chunks are executed in the simulator under a fixed continuation policy, giving their actual returns; ranking accuracy is how often $\hat{Q}$ orders a candidate pair as their returns do, and return error is the normalized RMSE of $\hat{Q}$ on 300 held-out transitions with executed actions. A vision-only value gives every candidate at a context the same value, so its ranking is chance by construction and is omitted. The full critic ranks 96% of pairs correctly. Removing the contact input costs most, dropping accuracy to 78%: the pairs it gets wrong are grasps that fail with little visual change, exactly what contact reveals. The same head on a separate encoder, with contact, keeps ranking at 88% but raises the return error from 0.08 to 0.11, the price of no longer reading features the dynamics head learned from video.

Table 2: Critic controls on PickPlace: ranking accuracy over four candidate chunks at 100 held-out contexts, and Q return error on 300 held-out transitions (shared data and training).

Encoder	Contact	Ranking acc. (%)	NRMSE
Shared	no	78	0.15
Separate	yes	88	0.11
Shared	yes	**96**	**0.08**

4.4 The reliability score and its two uses

Detecting unfaithful predictions. Table 3 uses three small held-out sets of predictions whose true outcome is known, labeled by comparing each prediction with the true future under the same action. Both signals flag every geometric failure and raise at most one false alarm on in-distribution predictions; they differ on action-unfaithful predictions, where the intrinsic signal catches 5 of 20 and the external signal 17 of 20. On a larger set of 200 predictions (50 geometric, 50 action-unfaithful, 100 in-distribution; two annotators, 91 failures), the combined score reaches precision 0.96 and recall

0.98 with one false alarm among the 100 in-distribution predictions, and scaling τ, the weights, or K by a factor of two either way changes at most 4 of the 200 decisions. Running the whole loop with one signal doing both jobs (Table 4) loses about 10 points at Online 2 on each of PickPlace, SinkFaucet, and OpenJar: the failures one signal misses are ones the policy would otherwise exploit or the collection would otherwise skip.

Table 5 replaces these hand-constructed sets with an independent label: on PickPlace we execute the same action in the simulator for 600 held-out predicted chunks and call a prediction wrong when its outcome (contact, object displacement, gripper state) disagrees with what actually happened, which gives 150 errors. The combined score separates them from correct predictions with AUROC 0.97; each signal alone is worse (0.92 and 0.82), so the two catch different failures. Two generic uncertainty estimates are worse still, an ensemble of prediction heads at 0.83 and distance to the training data at 0.65: most failures occur close to the training data, where the model renders the wrong action, not far from it. Among the 70% lowest-scoring predictions, which the gate keeps, 1.5% are wrong under the combined score, against 6.5% for the ensemble and 13.5% for data distance.

Table 3: Detections on held-out predictions. Each entry is flagged/total.

Prediction set	r_{intr}	r_{ext}
In-distribution	1/20	1/20
Geometric failure	19/19	19/19
Action-unfaithful	5/20	17/20

Table 4: Using one signal for both gating and collection. Entries are Imagined/Online 1/Online 2 success (%).

Score	PickPlace	SinkFaucet	OpenJar
r_{intr}	24.7/27.6/30.6	24.7/28.8/32.9	37.5/40.6/50.0
r_{ext}	27.0/28.8/32.4	23.5/30.0/30.0	28.1/37.5/46.9
Both	**28.8/32.4/42.4**	**31.8/34.7/43.5**	**40.6/53.1/59.4**

Separating the two uses. Table 6 separates the two jobs on PickPlace: gating on or off, crossed with targets chosen by score or at random, each with three training seeds (seed 1 of the full method is the run in Table 1), everything else held fixed. Both jobs help, and their effects add: gating gains about 9 points whether collection is random or by score, and score-selected collection about 7 points whether gating is on or off. Every gain is positive in all three seeds, and the two effects do not interact. Gating protects the current update from bad predictions; score-selected collection puts real data where the next model needs it. Gating also works before any real data is collected: over a 16-fold range of λ, Imagined success moves by less than one point (28.2, 28.8, 28.2), while removing the gate drops it to 22.4.

Table 5: Detection on 600 held-out PickPlace chunks (150 errors, labeled by simulation). Retained error: error rate among the 70% lowest-score predictions.

Score	AUROC	Retained error (%)
Distance to training data	0.65	13.5
Prediction-head ensemble	0.83	6.5
r_{intr}	0.92	4.0
r_{ext}	0.82	5.5
Combined r	**0.97**	**1.5**

Table 6: Factorial on PickPlace: Online-2 success (%) over three seeds.

Gate	Collection	Online-2 (%)
None	Random	25.7 ± 0.9
None	Score	32.5 ± 1.5
Score	Random	34.3 ± 0.9
Score	Score	$\mathbf{41.6 \pm 1.4}$
Shuffled	Score	29.0 ± 2.2
Constant mean	Score	32.2 ± 1.5

Weight information versus weight magnitude. Two controls separate weight information from weight magnitude, keeping score-selected collection and changing only weights. The batch-mean weight for every sample gives 32.2, as with no gating (32.5); shuffling weights within a batch gives 29.0, worse than none. The gate's gain comes from which samples it down-weights, not how much.

4.5 Model correction by targeted feedback

Collecting at flagged interactions improves the policy (Section 4.4). With 500 imagined rollouts per task, the fraction triggering a real rollout roughly halves from Online 1 to Online 2 on all three physical tasks (OpenJar 13.2 to 5.6%, StackBox 18.4 to 8.2%, FoldCloth 23.4 to 12.6%), but a trigger rate cannot separate a corrected model from a policy avoiding flagged regions or a lenient detector.

Table 7 therefore measures prediction error against actual outcomes on the 300 held-out PickPlace transitions of Section 4.3, frozen before collection and split by the initial model's score into 150

suspicious ($r > \tau$) and 150 ordinary transitions. The initial model is compared with three refits from seed 1 of Table 6: on 40 random-candidate rollouts and on 40 score-selected rollouts, both with old-data replay, and on the same 40 score-selected rollouts without replay. Targeted data fixes the flagged region: outcome error on suspicious transitions falls from 30.7% to 8.0%, whereas random collection only brings it to 22.7%; error on ordinary transitions also falls, from 12.0% to 5.3%, and the Q return error drops from 0.27 to 0.08. The policy improved on each refit reaches that condition's success in Table 6. Without replay the suspicious region is fixed just as well (6.0%), but the ordinary region returns to its initial error (12.7%), the Q error doubles (0.17), and policy success falls to 36.5%: the replay of Section 3.4 keeps the correction from being paid for with forgetting.

Table 7: Model correction on PickPlace over 300 held-out transitions (150 suspicious, 150 ordinary); outcome error is the fraction of predictions the simulator contradicts.

	Outcome error (%)			
Model update	Suspicious	Ordinary	Q NRMSE	Policy success (%)
No new data	30.7	12.0	0.27	28.8
Random candidate, +40	22.7	7.3	0.18	34.1
Score-selected, +40	**8.0**	**5.3**	**0.08**	**42.4**
Score-selected, no old replay	6.0	12.7	0.17	36.5

4.6 Physical realizability and acquisition cost

The real cycle works only if the robot can reach the flagged interaction. Table 8 measures the two steps of Section 3.4 on 20 triggers per task. Restoration puts rigid objects within 1.2–1.8 cm of their logged positions and the cloth in its logged coarse configuration every time; after replay, the end-effector is within 6.5–10.4 mm and the object within 1.4–1.8 cm of the imagined state. Errors of this size do not break the loop: on PickPlace, adding object offsets of 1, 2, and 5 cm to exact simulator resets lowers Online-2 success from 42.4% to 41.7, 39.3, and 30.7%, so the measured 1–2 cm sits where the method still beats VLAW† (39.3 versus 30.6).

Table 8: Restoration and replay on 20 triggers per task: initial-scene error after restoration, replay error at the flagged interaction; cloth configuration is a coarse category.

	Initial scene		After replay		
Task	Centroid / class	IoU	EE (mm)	Object / class	Success
OpenJar	1.2 cm	0.93	6.5	1.4 cm	19/20
StackBox	1.8 cm	0.91	10.4	1.8 cm	18/20
FoldCloth	100%	0.86	8.6	100%	19/20

Over five replays of each of these 20 triggers, imagined and reached states agree on whether the object is held in 98%, 92%, and 95% of replays on OpenJar, StackBox, and FoldCloth, with object yaw errors of 1.5° and 2.0° and 2.4 cm cloth keypoint error (Appendix C): replay re-establishes the interaction, not only position.

Table 1's budget counts rollouts, not robot time. Per rollout, restoration takes 33–86 s and replay 12–22 s, against 11–34 s for the collection itself, depending on the task; no one touches the scene.

5 Conclusion

A policy that practices inside a learned world model improves only where the model is right, and the model is corrected only where real data arrives. This paper ties the two together with one reliability score computed from the model's own prediction: where a prediction is unreliable, the score down-weights the imagined update and sends the robot there to collect the real outcome. On six tasks this loop improves the initial policy by 15.8–17.1 points in simulation and 25–28 on hardware and is ahead of on-policy collection on every task at the same rollout budget; the controlled studies show that both uses of the score contribute, that the gate acts through which samples it down-weights, and that the requested data corrects the model where flagged without forgetting elsewhere.

AI USE STATEMENT

Large language models were used as writing assistants, to polish the authors' manuscript and to help write the scripts that regenerate figures and tables from recorded results. They were not used to design the method, run experiments, or produce or analyze data. All technical content, results, and conclusions were written and checked by the authors, who take full responsibility for them.

REPRODUCIBILITY STATEMENT

Appendix A lists the task definitions and success criteria, hardware, and model and optimization settings. Appendix B gives the protocol of each controlled study in Sections 4.3 to 4.5, including how validation sets were constructed and labeled, and the per-seed counts behind Table 6. Appendix C describes the restoration script and the quantities it logs for every collection request, and Appendix D gives the full algorithm. Videos of the simulation and real-robot tasks are on the anonymous project page; code will be released there.

REFERENCES

K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 2018.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations*, 2025.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*, 2025.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. VLAW: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026.

Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, Yu Wang, Haoran Li, Chao Yu, and Dongbin Zhao. WoVR: World models as reliable simulators for post-training VLA policies with RL. *arXiv preprint arXiv:2602.13977*, 2026.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, 2024.

Hengtao Li, Pengxiang Ding, Runze Suo, Yihao Wang, Zirui Ge, Dongyuan Zang, Kexian Yu, Mingyang Sun, Hongyin Zhang, Donglin Wang, and Weihua Su. VLA-RFT: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. *arXiv preprint arXiv:2510.00406*, 2025.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *International Conference on Learning Representations*, 2026.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation*, 2023. URL https://code-as-policies.github.io/.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *arXiv preprint arXiv:2410.21845*, 2024.

Jindi Lv, Hao Li, Jie Li, Fankun Kong, Yang Wang, Pengfei Yi, Yifei Nie, Xiaofeng Wang, Zheng Zhu, Chaojun Ni, Qiuping Deng, Hengtao Li, Jiancheng Lv, and Guan Huang. ViVa: A video-generative value model for robot reinforcement learning. *arXiv preprint arXiv:2604.08168*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, 2020.

Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. In *International Conference on Learning Representations*, 2022.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Wan Team. Wan2.2: Open and advanced large-scale video generative models. Official model repository, 2025. URL https://github.com/Wan-Video/Wan2.2.

Xiyao Wang, Ruijie Zheng, Yanchao Sun, Ruonan Jia, Wichayaporn Wongkamjan, Huazhe Xu, and Furong Huang. COPlanner: Plan to roll out conservatively but to explore optimistically for model-based RL. In *International Conference on Learning Representations*, 2024.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, Ping Luo, Xiangyu Yue, and Hongyang Li. RISE: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World model-based policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.09515*, 2025.

Appendix: Supplementary Material

A Task and implementation details

Simulation uses RoboCasa with the PandaOmron embodiment, a 16-dimensional proprioceptive input plus one contact scalar, 12-dimensional actions, and three 256×256 RGB views at 20 FPS. TurnOnSinkFaucet succeeds when the faucet angle exceeds 0.4 rad. OpenCabinet requires normalized openness above 0.9. PickPlaceCounterToStove requires the apple centroid within 7 cm of the target pan and the end effector at least 25 cm from the object. The separation condition verifies release.

OpenJar uses a UR5e arm, a WSG 32-068 gripper, and a Femto Bolt head camera at 10 Hz. StackBox and FoldCloth use dual PiPER arms and three RealSense D435i cameras (head and two wrists) at 15 Hz. OpenJar requires rotating and detaching the lid; StackBox requires placing two bagged items into the box and closing the box; FoldCloth requires folding a spread-out garment into a neat block. Success on the real robot is judged by a human against these criteria.

Table 9: Model and optimization settings.

Setting	Value
Backbone / precision	Wan2.2-TI2V-5B, Wan VAE, bfloat16
Temporal layout	4 history + 8 future RGB frames; 1 history + 2 future latent frames
Chunk size	$H = 8$
Dynamics initialization	Batch 2; 100k steps; AdamW; lr $4 \cdot 10^{-5}$; decay .01; warmup 500; gradient clip 1.0
Q fitting	Batch 8; 100k steps; lr $4 \cdot 10^{-5}$; discount .99; Polyak .005; DiT frozen
Policy improvement	Imagined rollouts until $r > \tau$ or episode end; 8 flow steps; 4 adjoint steps; imagination weight .25; adapters only
Simulation policy training	5k steps; lr 10^{-6}
Added data	20 trajectories/task/round; two rounds
Reliability score	$K = 4$ noise pairs with $\sigma_k \sim \mathcal{U}[0.2, 0.8]$; w_{intr} and w_{ext} scale each term to an in-distribution mean of 0.5; τ is the 95th percentile of r on held-out in-distribution predictions; $\lambda = 1$
Q-head	Pools the conditioning-frame tokens after the first 15 of the 30 DiT blocks, concatenates state and action-chunk embeddings, two three-layer MLPs whose minimum is used; during policy improvement the world model and critic are frozen and $\nabla_{\tilde{a}} \hat{Q}$ flows through the action input only, not through frame generation
Progress model	ViVa value model (Lv et al., 2026) initialized from Wan2.2-TI2V-5B, taking multi-view RGB, proprioception, and contact; one model for the three simulation tasks and one per real task
Reward	$\gamma\Phi(s_{t+1}) - \Phi(s_t)$ with $\Phi = -C$ (the model's remaining-cost output), plus $+1$ at success and -1 at failure
Policy adaptation	LoRA adapters (rank 32) on the action expert
Probe / inverse dynamics	Depth Anything 3 with gripper segmentation and Arun/Horn alignment; three-layer MLP from (s_t, s_{t+1}) to a_t, trained on the demonstration trajectories
Detection baselines	Five prediction heads on the frozen features; nearest-neighbour cosine distance in the same feature space
Compute	8 A100 or A800 GPUs
Frameworks	PyTorch/Accelerate for model; OpenPI/JAX for policy

Progress labels. The progress model predicts a remaining cost C. On successful trajectories the label decreases linearly from 1 to 0; on failed trajectories an annotator marks the failure onset, before which the label decreases as on a success and after which it rises sharply and then settles at a terminal cost above the success level. The shaping potential is $\Phi = -C$.

B Protocols of the controlled studies

Baselines. VLAW† is our reproduction of VLAW on the Ctrl-World codebase with the same Wan2.2-TI2V-5B backbone, the same 150 logged trajectories, and the same π_0 initialization as our method; it spends the 40 additional rollouts on co-training the world model and the policy with the codebase's default objectives and schedule, without any reliability score. DSRL uses the public implementation with the same π_0 as the base policy and spends the same 40 rollouts on its latent-space policy updates. Both baselines and our method are evaluated on the same episodes.

Factorial and matched-weight controls. PickPlace, three training seeds; seed 1 of the full method is the run of Table 1. Six conditions: gating on/off crossed with random-candidate versus score-selected collection, plus shuffled and constant-mean weights under score-selected collection. All conditions share the initial checkpoints, two rounds of 20 rollouts, update and refit steps, the stopping rule, and restoration/replay; after replay every condition executes the flagged chunk first. In particular, imagined rollouts stop at $r > \tau$ in every condition, so the gate conditions differ only in the per-sample weights of Equation 4. Each cell is evaluated on 170 episodes at Online 2. Effects are computed within seed and reported as means; per-seed counts are in Table 10.

Table 10: Per-seed results on PickPlace: Online-2 successes out of 170 episodes (success rate in parentheses). Seed 1 of the full method is the run of Table 1.

Update weight	Collection	Seed 1	Seed 2	Seed 3
None	Random candidate	45 (26.5)	44 (25.9)	42 (24.7)
None	Score-selected	53 (31.2)	55 (32.4)	58 (34.1)
Score weight	Random candidate	58 (34.1)	60 (35.3)	57 (33.5)
Score weight	Score-selected	72 (42.4)	72 (42.4)	68 (40.0)
Shuffled weights	Score-selected	51 (30.0)	52 (30.6)	45 (26.5)
Constant mean weight	Score-selected	55 (32.4)	52 (30.6)	57 (33.5)

Independent score validation. PickPlace. Held-out episodes are split by whole trajectory so that no chunk neighbors a training chunk. Each held-out context yields one predicted chunk under the policy's action and one simulator execution of the same action; a prediction is labeled an error when its task-relevant outcome (grasp contact, object displacement, gripper state), read from the predicted frames with the same probe and segmentation used elsewhere, disagrees with the simulator outcome, with thresholds set on in-distribution samples. On the 200-prediction annotated set the combined score gives 89 true positives, 2 misses, 4 false alarms, and 105 true negatives. Scores are the loop's own r_{intr}, r_{ext}, and r; the baselines are nearest-neighbor distance to the training set in the frozen shared feature space and the disagreement of a small ensemble of prediction heads trained on those features. AUROC and the 70% retained-error point are computed on the same samples.

Model correction with held-out outcomes. PickPlace. Before any collection, a validation set of 300 transitions with executed actions is frozen and split by the initial model's score into 150 suspicious ($r > \tau$) and 150 ordinary transitions; identities and labels then stay fixed. These episodes are excluded from collection, retraining, score calibration, and progress-model fitting. Each condition collects 40 rollouts with the same policy and the same restoration and replay access, and the no-new-data control receives the same optimization budget on historical data, which separates the effect of new observations from that of additional refitting. Outcome error is read with the labeling rule of the score-validation protocol above; return NRMSE is normalized by the training-set return standard deviation; the no-old-replay control refits on the 40 new rollouts only. Changes are bootstrapped at the episode level.

Action ranking and return error. PickPlace. At 100 held-out contexts, four feasible candidate chunks are each executed in the simulator followed by the fixed continuation policy; their returns label pairwise preferences, with ties within a prespecified tolerance excluded. Ranking accuracy is the fraction of pairs ordered correctly by $\hat{Q}$. Return NRMSE reuses the 300 transitions and actual returns of the model-correction protocol, normalized by the training-set return standard deviation. The three critics share data and training schedule at the Online-2 stage: the full critic is the score-selected refit of Table 7; the others are the same critic trained without the contact input, and the same

head trained with contact on a separate encoder, a second Wan2.2-TI2V-5B DiT initialized from the same pretrained weights but not fine-tuned for dynamics, with the same blocks pooled.

Weighted policy objective. With v_θ the policy's flow, v_{ref} the frozen reference flow, and u_i the adjoint-matching target built from $\nabla_{\bar{a}}\hat{Q}$ at sample i, one update minimizes

$$\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B} \text{sg}\big[w(s_i,\bar{a}_i)\big]\,\big\|v_\theta(x_i,t_i) - u_i\big\|_2^2 + \beta\,\frac{1}{B}\sum_{i=1}^{B}\big\|v_\theta(x_i,t_i) - v_{\text{ref}}(x_i,t_i)\big\|_2^2,$$

where sg stops the gradient through the weight and β is the reference-flow coefficient; the weight multiplies only the adjoint-matching term, and the average is over the batch size B.

Statistics. Simulation cells report successes out of 170 episodes and real cells out of 32 trials; Wilson intervals describe success under a fixed policy. Factorial effects are paired within seed and reported as means, with the interaction as the difference of differences. Three seeds bound training variability only coarsely; comparisons closer than the reported spread would need more repeats.

C Physical collection: correspondence and cost

Table 8 measures restoration at the logged initial scene and replay at the flagged interaction on 20 triggers per task; a trigger counts as successful when restoration passes verification and the replayed prefix reaches the flagged interaction, retries included. Table 11 adds task-relevant correspondence at the reached interaction, computed over five replays of each of the same 20 triggers (100 per task): the fraction of replays in which the contact mode (object held or not) agrees between the imagined and reached states, the object yaw error for rigid objects, and the keypoint error for cloth. Table 12 reports the robot time of each stage per completed rollout, including retries, and the hands-on human time per rollout. Restoration and replay are extra robot motion that on-policy collection does not need; they are executed autonomously, so the resource they consume is robot time rather than human time.

Table 11: Correspondence at the reached interaction on the 20 triggers per task of Table 8. Contact-mode agreement is the fraction of the five replays per trigger in which the imagined and reached states agree on whether the object is held.

Task	Contact-mode agreement (%)	Object yaw error	Cloth keypoint error
OpenJar	98	1.5°	—
StackBox	92	2.0°	—
FoldCloth	95	—	2.4 cm

Table 12: Robot and human time per completed targeted rollout, averaged over the 20 triggers per task. Each stage includes its sensing, verification, and retries; hands-on is direct human manipulation of the scene or robot.

Task	Reset (s)	Replay (s)	Collect (s)	Total (s)	Hands-on (s)
OpenJar	33	12	11	56	0
StackBox	42	15	18	75	0
FoldCloth	86	22	34	142	0

Contact interface. On the real robots the contact label used to train the contact head is the normalized gripper motor current thresholded at the level calibrated on the demonstrations to separate free motion from holding; in imagination the contact head's probability is thresholded at 0.5 before it is fed back as the next chunk's contact input.

Restoration script. The routine segments each object, back-projects its mask, estimates the rigid-object centroid and yaw with RANSAC/OBB, and picks-and-places or reorients the object to its logged centroid and yaw. Gripper torque verifies each grasp; the post-placement pose is re-estimated and the step retried on failure before the arm returns to its logged joints. Restoration counts as successful when every rigid object is within a task-specific tolerance of its logged pose, set between 2

and 5 cm and between 3° and 8° across tasks, with at most three retries per object; a request whose restoration still fails is skipped, and the next trigger of the same bin takes its place, so the per-round budget of B executed rollouts is unchanged. FoldCloth is restored to its logged coarse configuration and placement. The routine follows the perception, planning, and verify–retry structure of programmatic robot control (Liang et al., 2023; Fu et al., 2026). The released configuration lists object identifiers, segmentation model, camera calibration, placement tolerances, verification criteria, and retry limits; each request logs the measured initial pose, the final verified pose, the replayed actions, and the reached observation.

D Algorithm and visual evidence

The pseudocode below complements the two-cycle overview in Figure 1. Figure 6 shows the two hardware platforms and Figure 7 the value curves along successful and failed rollouts.

Checked imagination and targeted correction

Input: initial real data $\mathcal{D}_0$, policy π_0, threshold τ, budget B.
1. Fit dynamics on $\mathcal{D}_0$; freeze DiT and fit the Q-head (clipped double Q).
2. For each feedback round, initialize the candidate buffer $\mathcal{T}$.
3. Sample logged starting contexts; keep the model and critic fixed.
4. Generate a policy chunk, predicted video, end-of-chunk contact, Q, and both check scores.
5. Weight the QAM contribution by $e^{-\lambda r}$; update policy adapters.
6. If $r > \tau$, record start/prefix/flagged interaction in $\mathcal{T}$ and stop.
7. Otherwise advance the imagined context and repeat from step 4 until the episode ends.
8. Select up to B candidates by spatial subsampling within $\mathcal{T}$.
9. Restore each logged start, replay its prefix, collect actual outcomes.
10. Add real data to $\mathcal{D}$; update dynamics and then the critic.

Output: improved policy and updated dynamics–value model.

Algorithm 1: Parameter updates and data flow. The weight in step 5 multiplies each sample's adjoint-matching loss and is not differentiated (Section 3.4); if fewer than B eligible candidates exist, all are used and the count is reported.



(a) UR5 Arm Platform (b) Dual-arm Piper Platform

Figure 6: Hardware platforms. Left: the UR5e platform for OpenJar. Right: the dual-PiPER platform for StackBox and FoldCloth. Camera and control settings are specified in Appendix A.

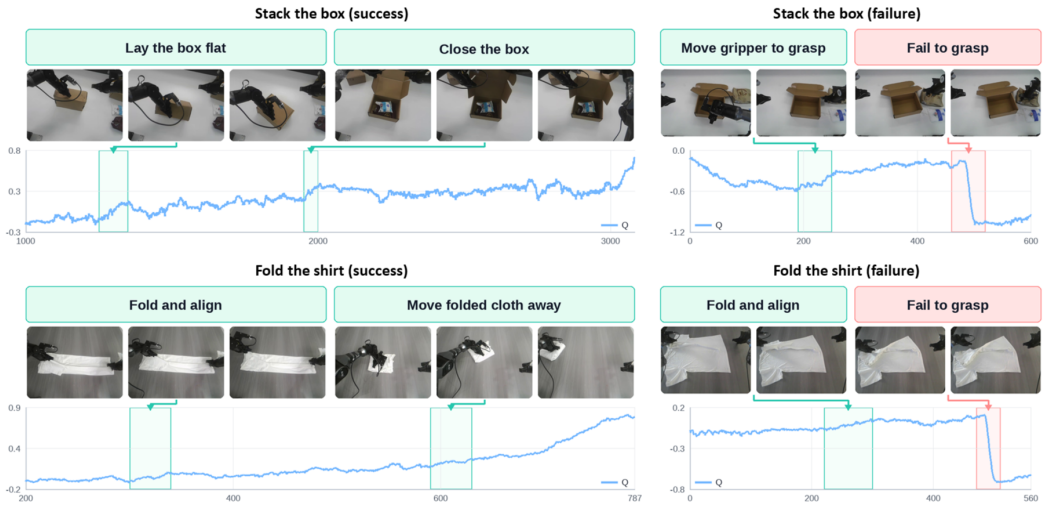


Figure 7: Q values along successful and failed rollouts. The top row shows StackBox and the bottom row FoldCloth. Green annotations mark subtask progress and red annotations mark grasp failures.