

XPACE: Joint World and Action Modeling from Heterogeneous Experience

World Model Team, XPENG Robotics

Project: <https://xpeng-robotics.github.io/xpace/>

Abstract

A general-purpose robot needs to draw on diverse experience, choose actions, and anticipate how those actions will change the world. We introduce **XPACE**, a unified embodied world model that serves as both a *world action model*, jointly predicting executable robot actions and future video, and a *world simulator*, predicting the visual consequences of prescribed actions. Our key insight is that video prediction can both connect heterogeneous experience to action learning and generate new experience for policy improvement. With a shared video backbone between the policy and simulator, we use action-unlabeled video to learn visual dynamics and action-labeled human and robot demonstrations to jointly learn video and action prediction. Building on this architecture, a coarse-to-fine training curriculum progressively emphasizes robot control while retaining human experience, allowing the policy to learn behaviors beyond those covered by robot demonstrations. Beyond learning from recorded experience, **XPACE** uses its simulator to create additional recovery supervision for the policy. Specifically, we adapt the simulator to its own generated context, synthesize deviation–recovery trajectories around expert demonstrations, and fine-tune the policy on filtered recovery examples. Experiments on XPENG’s IRON humanoid robot show that heterogeneous training improves robustness and enables transfer of human-observed skills to tasks absent from robot demonstrations, while recovery data generated by the model’s own simulator further improves real-world task completion. Together, these results demonstrate how joint world and action modeling connects learning from heterogeneous experience with simulation-driven policy self-improvement.

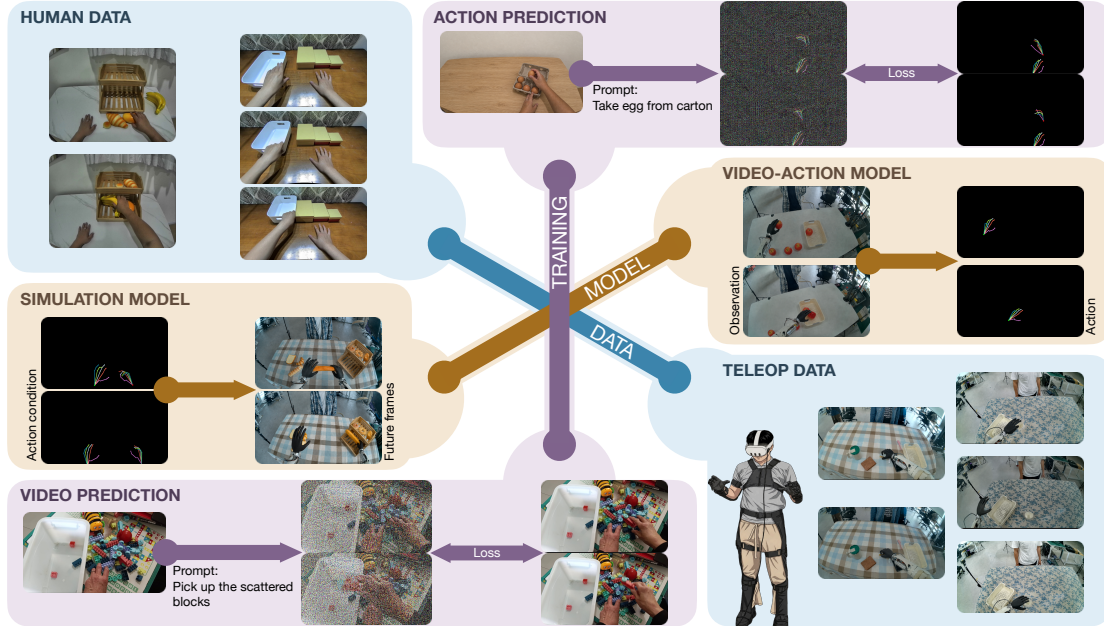


Figure 1: XPACE is a unified embodied world model, serving as both a *world action model*, jointly predicting executable actions and future video from observations, current state, and task instructions, and a *world simulator*, predicting future video from history, prescribed skeleton controls, and camera poses. The model learns jointly from human video and robot demonstrations, with human bridge data supporting cross-embodiment alignment. Recovery trajectories generated by the learned world simulator enable policy self-improvement through DAGger fine-tuning.

1 Introduction

Humans acquire skills by observing others and interacting with the world, and use this experience to anticipate the consequences of their actions and refine their behavior. A general-purpose embodied system should similarly learn from diverse experience, predict and execute actions, and improve its behavior beyond the demonstrations it has observed. Progress in world modeling has brought prediction closer to robot control, from learned environment dynamics for planning and policy learning [1, 2] to visually rich simulation of embodied interactions [3, 4]. More recently, video-action models have connected future prediction and action generation within shared networks [5, 6]. These developments provide important components of a general-purpose embodied system, but joint prediction alone does not establish how to acquire skills from diverse sources and improve them through simulated experience.

Closing this gap requires more than expanding a model’s prediction outputs. Human video offers broad behavioral coverage, but its supervision ranges from vision alone to motion annotations that differ from robot control commands; robot demonstrations provide executable actions but are costly to collect across diverse environments [7, 8]. Simply combining these sources does not explain how their different supervision should contribute to a common policy and simulator. Even with a shared model, training must adapt broad human experience to the target robot without discarding its behavioral coverage. Furthermore, imitation on recorded demonstrations provides limited corrective supervision when execution errors lead the policy into unfamiliar states [9]; predictive simulation is useful for self-improvement only if its outputs can become reliable policy-training examples. These considerations lead to three connected questions: (Q1) *what experience should the model learn from*; (Q2) *how can one model use that experience for both action generation and simulation*; and (Q3) *how should training turn these capabilities into robot control and subsequent self-improvement*?

We introduce **XPACE**, a joint world-action framework that addresses these questions through coordinated data, model, and training designs, as illustrated in Figure 1. It learns from heterogeneous human and robot experience and uses a shared video backbone to support both task-conditioned action generation and action-conditioned simulation. The learned simulator then generates recovery supervision for policy fine-tuning, connecting learning from recorded experience with simulation-driven self-improvement within the same model.

Learning from complementary experience. To address Q1, we combine action-unlabeled egocentric video, kinematically labeled human demonstrations, and robot teleoperation rather than restricting policy learning to robot-collected experience. Action-unlabeled videos supply diverse visual interactions, human motion labels supervise manipulation behaviors, and robot actions provide direct supervision for control on the target embodiment. Task- and appearance-aligned human bridge data further reduce differences in task coverage and visual conditions between human demonstrations and robot deployment. The resulting data pyramid connects broad interaction experience with robot-specific supervision while preserving the distinct information each source provides.

Joint action generation and simulation. To address Q2, we use video prediction as a common learning task across sources and connect it to action prediction through a shared video backbone. In policy mode, observations, the current state, and task instructions condition the prediction of executable actions and future video; in simulation mode, visual history, prescribed skeleton controls, and camera poses condition future-video prediction. An action Transformer reads the backbone’s multi-level visual features, while action-loss gradients update the same backbone used by the simulator. Embodiment-specific action projections accommodate human and robot action spaces while keeping the action Transformer shared. Thus, video-only experience and action-labeled demonstrations can train the same model, with shared visual dynamics supporting both control and simulation.

Progressive training and simulation-driven self-improvement. To address Q3, we first train video prediction and then jointly train action prediction and simulation, applying action losses only where valid supervision is available. A coarse-to-fine curriculum progressively emphasizes bridge and robot data while retaining human experience throughout robot adaptation. Training then extends beyond the recorded demonstrations: we adapt the simulator to self-generated visual context through self-gradient forcing and use it to render prescribed deviation-recovery trajectories around expert demonstrations. We filter the generated trajectories for recovery consistency, combine the retained recoveries with expert continuations, and fine-tune a separate policy copy on the augmented data. This procedure uses the model’s

own simulator to supply corrective experience that is missing from nominal demonstrations, turning its predictive capability into a practical source of policy self-improvement.

Together, these designs contribute a framework that combines heterogeneous supervision, shared action and world modeling, and simulation-generated recovery training. Experiments on XPENG’s IRON humanoid robot show improved policy performance and robustness on the evaluated tasks, including transfer of human-observed skills to tasks absent from robot teleoperation. Video-prediction experiments show that simulator adaptation improves autoregressive prediction while reducing inference cost, supporting recovery-data synthesis. Fine-tuning on the synthetic recoveries further improves real-world task completion, demonstrating that the learned simulator can contribute to improving the model’s own policy rather than serving only as a prediction interface.

2 Related Work

2.1 Data: human experience and robot grounding

Human and robot datasets provide complementary supervision for embodied learning. Ego4D captures diverse everyday activities from a human perspective [10], while Open X-Embodiment pools robot trajectories across embodiments [11] and DROID broadens robot scene and task coverage [8]. Human videos provide visual interaction experience, and tracked human demonstrations additionally supply motion labels; robot trajectories pair observations with executable controls. Combining these sources requires accounting for differences in action representations, camera viewpoints, and task coverage.

Recent work studies how to transfer this human experience to robot control. SWIM learns manipulation affordances from human videos before grounding them in robot interaction [7], while Phantom and Human2Robot explore transferring human demonstrations to robot manipulation [12, 13]. EgoScale combines human pretraining with aligned human–robot adaptation [14]. EgoWAM compares pixel, feature, and motion prediction for transfer under controlled human–robot co-training [15]. Human experience can also train the predictive component: DreamDojo uses continuous latent actions to transfer interaction knowledge from large-scale egocentric video to robot world modeling [16]. In this setting, XPACE jointly learns from action-unlabeled video, kinematically labeled human demonstrations, and robot trajectories, using task- and appearance-aligned bridge data to connect human experience with robot deployment.

Generated experience offers a complementary route to expanding robot data. DreamGen synthesizes robot videos and recovers pseudo-actions with latent-action or inverse-dynamics models [17]; RoboVIP augments manipulation observations through temporally coherent, multi-view generation guided by visual identity prompts [18]. These approaches expand behavioral or visual coverage, whereas our recovery augmentation targets off-trajectory states around recorded expert demonstrations.

2.2 Model: policies, simulators, and shared predictive representations

Robot foundation models connect perception and control through direct action prediction, visual simulation, or video-based planning. RT-1 studies scalable transformer policies, while Octo learns a generalist policy that can adapt across robot observation and action spaces [19, 20]. Diffusion Policy models multimodal action sequences through conditional denoising [21]. Vision–language–action policies such as OpenVLA, π_0 , and Gemini Robotics further connect pretrained visual–language understanding to executable control [22, 23, 24]. Their direct policy interface contrasts with approaches that explicitly expose predicted visual outcomes.

World simulators predict visual outcomes under supplied controls. UniSim learns an interactive simulator from heterogeneous experience [3], and IRASim studies action-conditioned robot video generation [4]. Ctrl-World combines frame-level action conditioning, multi-view prediction, and memory retrieval for extended policy interactions [25]. AnyWorld factorizes action, camera, and embodiment conditions for cross-embodiment generation [26]; SyncWorld instead uses visual calibration episodes to interpret numerical actions in unseen setups without downstream training [27]. These designs address complementary aspects of controllability and transfer across visual and embodiment changes.

Visual planning uses predicted observations as intermediate goals for control. UniPi generates language-conditioned video plans and translates them into actions through an inverse-dynamics model [28], while

SuSIE uses image-editing diffusion to propose subgoals for a goal-conditioned controller [29]. RoboDreamer factorizes video generation for compositional instructions, and ManipDreamer introduces instruction trees and visual guidance [30, 31]. Video Prediction Policy extracts predictive features from a video diffusion model [32], whereas Video Policy combines video and action generation in an end-to-end trainable modular framework [33].

Unified video–action models connect prediction and control within shared networks. UWM uses independent video and action noise levels for policy, dynamics, and video-generation modes [5]; UVA uses shared video–action latents with lightweight decoders and optional video decoding at deployment [6]. VideoVLA jointly forecasts actions and visual outcomes from language and images [34], while DreamZero transfers experience from video-only data through joint video–action modeling [35]. Motus combines understanding, video, and action experts [36]; Genie Envisioner builds policy and simulator modules around a pretrained video model [37]; and τ_0 -WM combines video–action prediction, action-conditioned simulation, and candidate evaluation [38]. LingBot-VA couples autoregressive video–action modeling with asynchronous execution [39], and Riemann-1.0 unifies policy execution and world simulation in a causal autoregressive sequence [40]. Cosmos Policy encodes actions, future images, and values as latent frames within a pretrained video model [41].

Other designs change the representation of actions or futures. X-WAM adds spatial prediction and asynchronous video–action denoising [42]; Action Images represents controls as pixel-grounded multiview videos [43]; and FlowWAM uses optical flow as a shared motion representation for policy and world-model modes [44]. Being-H0.7 learns future-informed latent queries without generating future frames at deployment [45]. Fast-WAM retains video co-training but skips test-time future prediction [46], while GigaWorld-Policy uses an action-centered causal design with optional video generation [47]. Together, these works distinguish the representational benefits of future prediction from the cost of explicitly generating futures during control. Within this design space, XPACE shares a causal video backbone between policy and simulation, with an asymmetric action Transformer reading multi-level visual features and skeleton controls and camera poses conditioning simulation.

2.3 Training: self-supervised video learning and supervised action grounding

R3M learns manipulation representations through temporal and video–language objectives [48], while masked visual pretraining learns from egocentric images without action labels [49]. Voltron combines language-conditioned visual reconstruction and visually grounded language generation to learn from human videos and captions [50]. V-JEPA 2 follows large-scale video learning with robot-based action-conditioned prediction and planning [51]. LAPA learns discrete latent actions from frame transitions before adapting with action labels [52]. AdaWorld similarly extracts latent actions for world-model pretraining and subsequent adaptation to new controls [53]. These objectives make action-unlabeled video useful without assuming that it already provides executable robot commands. Our recipe combines video-only pretraining with joint video and action learning, retaining both human and robot action supervision as training increasingly emphasizes the target robot.

Autoregressive simulation additionally requires handling the shift from recorded to generated visual context. Self Forcing trains video diffusion models on generated context [54], while Self Gradient Forcing updates context representations using future-prediction gradients without backpropagating through the full rollout [55]. Persistent Robot World Models addresses multi-step drift through reinforcement-learning post-training on autoregressive rollouts [56]. Our simulator adaptation follows these principles while retaining paired, skeleton- and camera-conditioned flow-matching supervision to learn the visual consequences of prescribed controls (§3.4.1).

2.4 Simulation-based policy improvement and evaluation

Learning from imagined experience predates large video models. Dreamer optimizes behavior through latent imagination [57], and model-based policy optimization uses short model rollouts branched from real data to balance additional experience against model error [58]. Video world models extend this idea to visual manipulation, with different sources of supervision and different ways of updating policies.

Dagger aggregates expert labels on learner-visited states to address imitation-learning distribution shift [9]. WM-Dagger synthesizes and filters recovery trajectories around demonstrations [59], while Hi-WM

collects human corrections during simulated policy rollouts [60]. RISE and World-VLA-Loop also use learned simulation for policy refinement [61, 62]. VLAW iteratively improves a world model with real policy rollouts and uses synthetic rollouts to improve the policy [63]. PlayWorld collects autonomous play interactions to train a simulator for evaluation and model-based reinforcement learning [64]. More recently, Motus2 exposes policy, simulator, and value interfaces through shared weights and couples them for planning and model-based policy improvement [65]. Our focus is to obtain this corrective supervision from a simulator jointly trained with the policy. After self-gradient forcing adaptation, XPACE renders prescribed deviation–recovery trajectories and fine-tunes a separate policy on filtered recoveries and expert data, following DAgger-style recovery augmentation rather than on-policy expert querying.

For evaluation, WorldGym tests whether simulated rollouts predict real-robot policy performance [66]. Our policy-conditioned rollouts support qualitative diagnosis of instruction following and target selection before testing; quantitative evidence for policy self-improvement comes from real-robot trials (§4.3).

3 Method

We present **XPACE**, a unified world and action modeling framework that integrates data, architecture, and training to learn robot control from heterogeneous experience and enable simulation-driven policy self-improvement. To combine the breadth of human experience with the precision of robot demonstrations, we organize these sources into a four-layer data pyramid (§3.1). Learning from this mixture requires an architecture that connects visual prediction with executable actions; we therefore share a video backbone between the simulator and the policy, allowing both to learn from a common representation of visual dynamics (§3.2). Within this architecture, a coarse-to-fine curriculum progressively emphasizes robot supervision while retaining human experience, adapting the shared representation to the deployment embodiment (§3.3). The resulting model supports learning beyond recorded demonstrations: after self-gradient-forcing adaptation, its simulator generates recovery trajectories that provide additional supervision for policy fine-tuning, completing the simulation-driven self-improvement process (§3.4).

3.1 Data

Our data design combines human and robot experience with language conditions and action representations that support learning across sources and embodiments. We organize 5,000 hours of embodied video into four layers: (L1) broad egocentric video for learning visual dynamics, (L2) human video–action data for learning manipulation behaviors, (L3) bridge data for aligning human and robot experience, and (L4) robot teleoperation for learning embodiment-specific control. A separate failure set (F) supplies unsuccessful, low-progress, and recovery trajectories exclusively for simulator training. To associate the recorded behaviors with task instructions, we construct multi-level captions that describe both episode-level objectives and clip-level actions, with caption sampling progressively adapted to policy deployment. We further express human and humanoid robot motion in a shared action schema, allowing demonstrations from both embodiments to supervise action prediction while preserving embodiment-specific mappings.

Data pyramid. As shown in Figure 2, the data pyramid connects broad interaction coverage with robot-executable control through four complementary layers:

- (L1) **In-the-wild egocentric video.** We select interaction-rich, manipulation-relevant clips to provide broad coverage of scenes, objects, and visual dynamics. These clips have no action labels and train the model to predict future video.
- (L2) **Human video–action data.** Egocentric videos paired with hand and wrist poses provide action supervision across diverse manipulation behaviors. The poses use the shared human–robot action schema, enabling the model to learn action prediction from human demonstrations alongside future-video prediction.
- (L3) **Task- and appearance-aligned bridge data.** We construct bridge data to reduce task and visual differences between human demonstrations and robot deployment. *Task-aligned* clips are selected from L1 to match manipulation task families in the robot corpus, concentrating human experience on robot-relevant skills. *Appearance-aligned* clips are collected in environments resembling the robot workspace in camera viewpoint, background, and object arrangement, with-

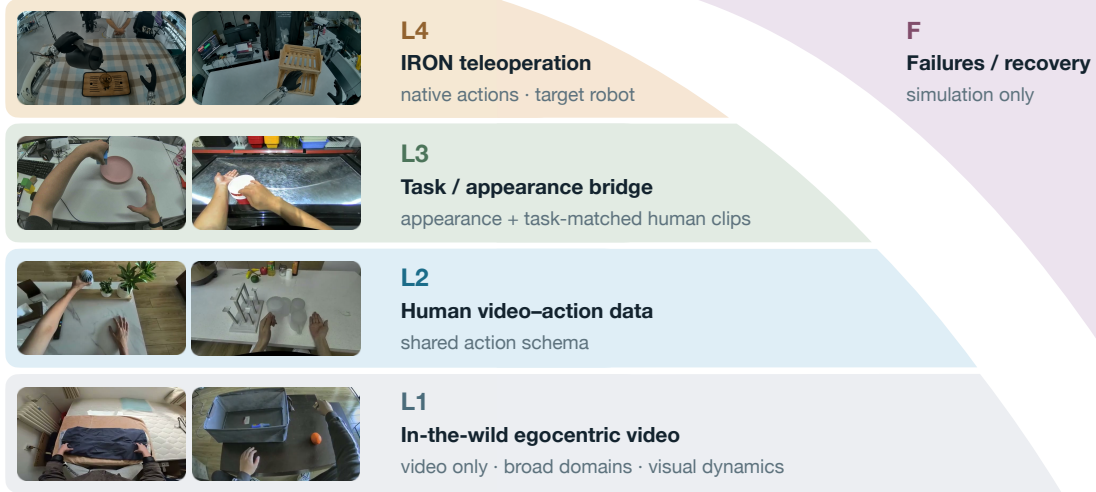


Figure 2: **Data pyramid for XPACE.** The corpus comprises (L1) in-the-wild egocentric video, (L2) human video–action data, (L3) task- and appearance-aligned bridge data, and (L4) IRON teleoperation. Failure and recovery data (F) train simulation only.

out requiring exact task matching. Joint training on these sources encourages cross-embodiment representations through task and visual overlap, without an explicit feature-alignment loss.

(L4) Robot teleoperation. Demonstrations collected on IRON robots, including the deployment embodiment, pair video with native robot actions. They provide direct supervision for executing manipulation tasks on the target hardware.

Failure and recovery data. Training a simulator exclusively on successful demonstrations can bias its predictions toward successful outcomes even when the supplied actions are incorrect. We therefore include failure, low-progress, and recovery trajectories in F to supervise the visual consequences of both successful and unsuccessful motions. These trajectories train video prediction conditioned on recorded skeleton controls and camera poses; their actions are excluded from policy imitation. This supervision supports action-dependent simulation for policy evaluation and recovery-data synthesis.

Multi-level captions. Language provides task instructions for the policy and should accurately describe the actions demonstrated in each training clip. Each training clip is a short temporal segment sampled from a full task episode and may contain only part of the instructed task. We therefore retain the episode-level instruction and supplement it with clip-specific captions and task paraphrases, as shown in Figure 3:

1. **Original task instruction.** The source instruction specifies the overall task objective, preserving the context of the episode.
2. **Dense caption.** A detailed description specifies the scene and action sequence within the sampled clip, using the descriptive language typical of video-model pretraining.
3. **Atomic action caption.** A short imperative command specifies the local action demonstrated in the clip, providing precise action guidance when the original instruction describes a longer task.
4. **Task paraphrases.** LLM-generated rewrites preserve the original task objective while varying its wording, exposing the model to multiple expressions of the same instruction.

These text conditions differ in both temporal scope and linguistic style, allowing training to connect detailed video descriptions with concise task instructions. We sample one condition per example using a stage-dependent mixture (§3.3). Early training emphasizes dense captions to remain close to the pre-trained video model’s text-conditioning distribution; subsequent stages progressively increase original and paraphrased task instructions to match policy deployment. L1 clips without action labels retain descriptive captions throughout training for language-conditioned video prediction.

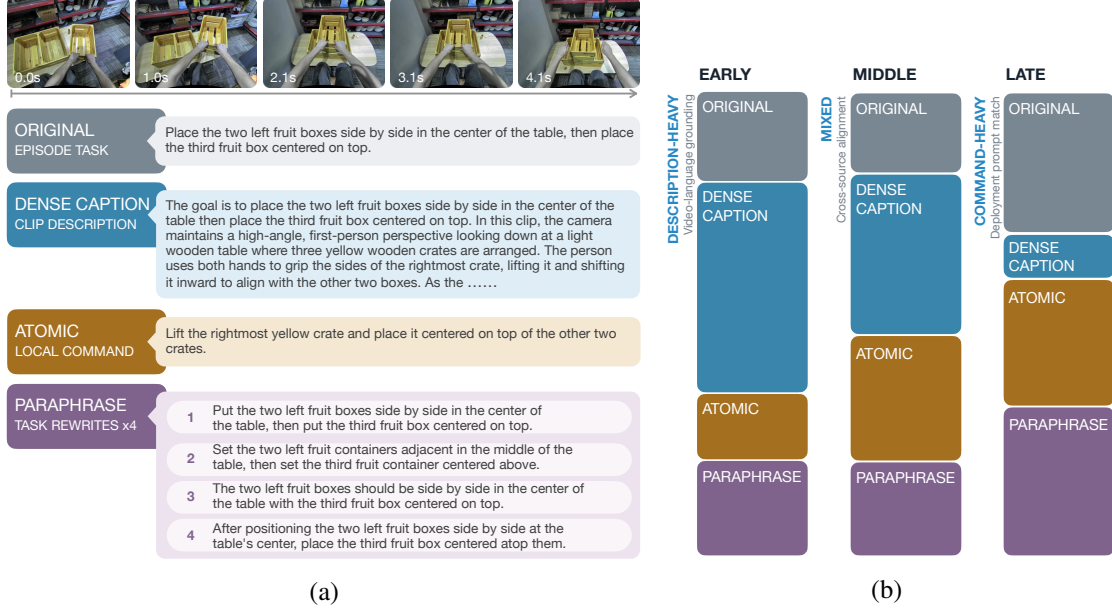


Figure 3: **Multi-level language conditioning and caption sampling.** (a) Each training clip is associated with an original task instruction, a dense description, an atomic action command, and task paraphrases, covering the overall objective, local actions, and alternative wording. (b) Training samples one text condition per example, with the mixture shifting from dense descriptions toward task instructions to match deployment inputs.

Cross-embodiment action schema. Our humanoid robot provides a direct correspondence between human and robot arm and hand motion, facilitating cross-embodiment action alignment. Unlike approaches that transfer human demonstrations to gripper-based robots [12, 13], our shared action representation retains hand articulation alongside wrist motion. We therefore represent human and robot trajectories using a common kinematic schema comprising end-effector poses, hand articulation, and torso/camera poses, with source-specific normalization. Human and robot demonstrations supervise the same action-prediction objective $\mathcal{L}_a$, enabling human motion data to contribute directly to policy learning. To accommodate differences in embodiment kinematics and action dimensions, we use embodiment-specific linear input and output projections with a shared action Transformer, as described in §3.2. This design combines shared action modeling with embodiment-specific mappings for executable control.

3.2 Model Architecture

The heterogeneous data in §3.1 provide both visual dynamics and cross-embodiment action supervision; our architecture connects these signals through a shared predictive representation. As in Figure 4, XPACE uses an asymmetric mixture-of-transformers (MoT) architecture: a Video Transformer predicts future video, and an Action Transformer reads its multi-level visual features to predict executable actions. The video backbone and its output head are shared between policy and simulation modes, while embodiment-specific action projections accommodate the human and robot action spaces. Information flows from video to action in the forward pass, but action-loss gradients propagate into the video backbone during joint training, allowing control supervision to refine the representation used by both modes.

Chunk-wise prediction. Both the video action model and the world simulator are trained with the same chunk-wise video representation and causal attention structure. A frozen video VAE encodes each clip into an initial anchor latent z_0 followed by M chunks $\mathbf{z}^{1:M}$, each containing 4 latent frames (corresponding to 16 video frames). For prediction chunk m , the visual context consists of z_0 and the preceding chunks $\mathbf{z}^{<m}$; this prefix is empty when $m = 1$. We use *chunk-wise causal attention*: attention is bidirectional within each chunk, but access across chunks is restricted to preceding context. The corresponding policy action chunk is $\mathbf{a}^m \in \mathbb{R}^{N_a \times D_a}$, where $N_a = 16$ is the action horizon and D_a is the embodiment-

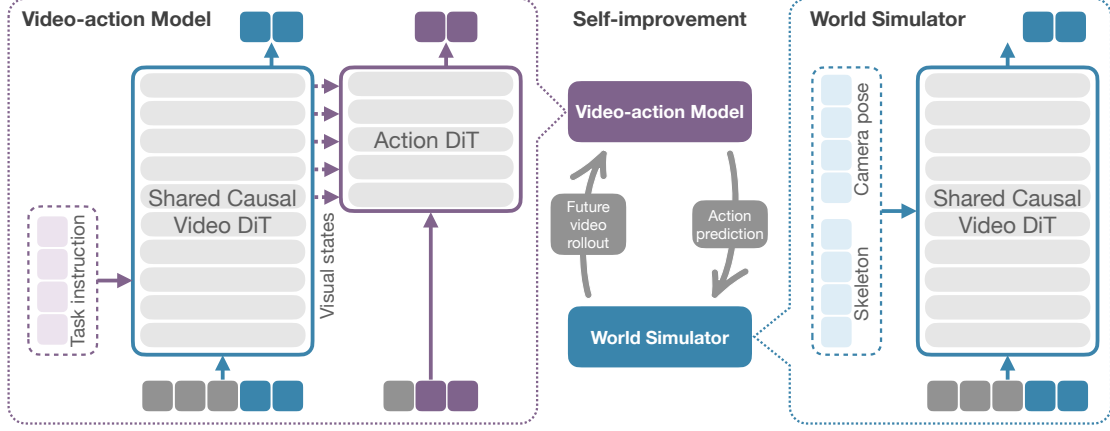


Figure 4: **MoT architecture, operating modes, and self-improvement of XPACE.** The asymmetric mixture-of-transformers (MoT) couples a causal video Transformer with an action Transformer through multi-level video-to-action feature connections. **Left:** In video-action mode, visual history, the current state, and a task instruction condition the joint prediction of future video and executable action chunks. **Right:** In simulation mode, the shared video Transformer predicts future video from visual history, prescribed skeleton controls, and camera poses. **Center:** Simulation connects action sequences to their visual consequences, enabling simulation-driven policy self-improvement. After adaptation with self-gradient forcing, the simulator generates deviation-recovery trajectories that are filtered and aggregated with training data for DAgger-style policy fine-tuning.

specific action dimension; κ^m denotes the aligned 2D action-skeleton sequence. We denote the current state at the start of chunk m by $\mathbf{s}^m$ and the camera-pose sequence aligned with the chunk by $\mathbf{c}_{\text{cam}}^m$. During teacher-forced training, the visual context comes from recorded $\mathbf{z}^{<m}$; SGF replaces the recorded prefix with self-generated chunks $\hat{\mathbf{z}}^{<m}$, as described in §3.4.1.

Two operating modes. Under this shared chunk-wise formulation, the two modes differ in their conditions and active modules:

- **World simulator.** Given visual history, a prescribed 2D action-skeleton sequence κ^m , and camera poses $\mathbf{c}_{\text{cam}}^m$, the video Transformer predicts the visual consequences of the supplied motion:

$$p(\mathbf{z}^m \mid z_0, \mathbf{z}^{<m}, \kappa^m, \mathbf{c}_{\text{cam}}^m). \quad (1)$$

The action Transformer is inactive, and no language instruction is supplied.

- **Video action model.** Given visual history and a language instruction g , the video Transformer predicts a task-conditioned future and supplies its hidden features to the action Transformer, which additionally receives the current state $\mathbf{s}^m$:

$$p(\mathbf{z}^m, \mathbf{a}^m \mid z_0, \mathbf{z}^{<m}, \mathbf{s}^m, g, e), \quad (2)$$

where e selects the embodiment-specific action projections. Prescribed future skeleton and camera-pose controls are withheld, so the policy must infer motions from visual history, the current state, and the instruction.

Skeleton- and camera-conditioned simulation. The simulator receives camera poses as an explicit input alongside the skeleton sequence, rather than using them only to render the skeleton. We incorporate skeleton controls through *token addition* at the Transformer input. The rasterized skeleton sequence is first encoded by the frozen video VAE, then mapped through a separate patch embedding to match the video tokens in spatial and temporal layout and embedding dimension. Each skeleton token is added element-wise to the video token at the corresponding spatial and temporal position, producing a single conditioned token sequence without increasing its length. The skeleton patch embedding is zero-initialized to preserve the pretrained video representation at initialization. We compare token addition with other conditioning strategies in the ablation in §4.2.1. During autoregressive simulation, generated chunks $\hat{\mathbf{z}}^{<m}$ provide the

visual context for predicting $\hat{z}^m$, while skeleton controls κ^m and camera poses $\mathbf{c}_{\text{cam}}^m$ remain externally supplied.

Asymmetric video-to-action policy model. The policy predicts actions from task-conditioned video features through three complementary designs. A learned mode embedding replaces skeleton conditioning on the video tokens, while text cross-attention supplies the task instruction.

- **Asymmetric multi-level action readout.** The 15 action blocks read spatially dense features from video blocks 16–30, respectively, to predict the action chunk $\mathbf{a}^m$. Each action block projects its paired video features into the action hidden dimension, then uses action tokens as queries and the concatenated action and projected video tokens as keys and values. This combines action self-attention with multi-level visual conditioning without spatial pooling. Action tokens do not modify video features in the forward pass, but action-loss gradients propagate through the visual readout into the video backbone.
- **Embodiment-indexed boundary projections.** Each embodiment has its own linear input projection and final output projection for actions. The action Transformer and all other action-branch components are shared across embodiments.
- **Current-state conditioning for the action Transformer.** The action Transformer prepends the normalized current robot state token $\mathbf{s}^m$ to the $N_a = 16$ noisy action tokens and processes them with the existing input projection and shared Transformer. Discarding the state-token output leaves the 16-step action prediction without adding action-head parameters. During training, we randomly omit the current robot state condition for each sample with probability 0.5.

Real-robot deployment. We deploy XPACE for closed-loop control of the IRON robot, using streaming visual encoding and asynchronous action execution.

- **Streaming VAE encoding** processes incoming observations incrementally, producing one latent frame per four new video frames while the robot continues executing queued commands. After 4 video frames, the policy uses the accumulated visual latents, current robot state, and task instruction to generate a new 16-step action plan. This separates the cadence of visual encoding from full policy inference, while observation sampling and command publication remain independent of the inference worker.
- **Overlapping-horizon stitching** connects successive plans without pausing execution for inference. Predicted actions are converted to robot control units and interpolated for command publication at approximately 10 Hz. The publisher preserves already committed commands, discards the overlapping prefix of a new plan according to the remaining queue length, and appends its non-overlapping suffix. A decaying correction smooths the transition at the join, allowing new predictions to enter the running trajectory through publisher-side post-processing.

3.3 Training Paradigm

Before post-training, we train the shared architecture in two stages, as shown in Figure 5. Stage I adapts the video backbone to embodied video; Stage II adds action learning and jointly trains policy and simulation modes, progressively increasing robot supervision while retaining human and video data. The resulting model initializes the simulator adaptation and policy improvement described in §3.4.

Training objectives. We train video and action prediction with flow-matching losses [67], denoted by $\mathcal{L}_v$ and $\mathcal{L}_a$, respectively. Flow times and Gaussian noise are sampled independently for video and action. Simulation updates optimize $\mathcal{L}_v$, while video–action updates optimize $\mathcal{L}_v + \mathcal{L}_a$ with equal loss weights. Video loss is averaged over prediction chunks only; action loss is averaged over valid action-labeled samples and is zero when none are present. Action-unlabeled clips contribute only to video prediction, and F is used exclusively for simulation. Stages I and II use teacher forcing with recorded visual context.

3.3.1 Stage I: Video-only pretraining

Starting from the pretrained video Transformer, we train future-video prediction on egocentric human video and robot teleoperation at near-natural source-mixture ratios. All participating sources are treated

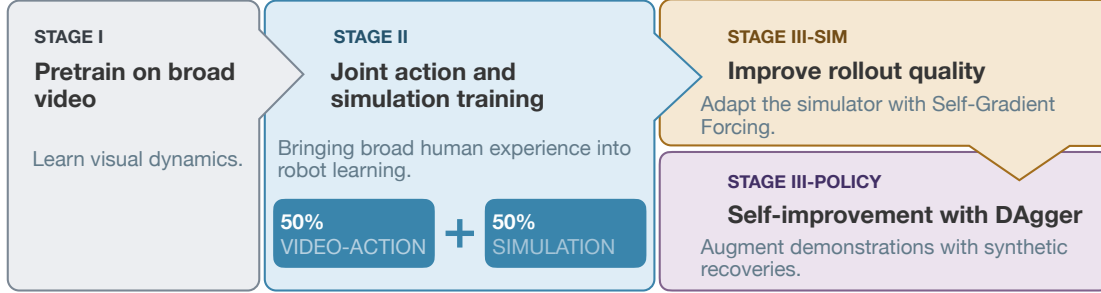


Figure 5: **Training stages and policy self-improvement of XPACE.** Stage I trains video prediction; Stage II adds the action branch and jointly trains policy and simulation modes with a coarse-to-fine data curriculum. Stage III-sim and Stage III-policy initialize separate copies of the Phase II-c checkpoint, updating only simulation and policy, respectively. Stage III-sim adapts the simulator to self-generated context, while the unchanged Phase II-c policy serves as the Before DAgger baseline. The connection from Stage III-sim to Stage III-policy represents synthetic recovery data, not checkpoint initialization; the SGF simulator remains fixed during policy fine-tuning.

as video-only data, with multi-level captions as text conditions and no action loss; F and DAgger data are excluded. This stage lets the backbone learn from broad visual experience before action supervision is introduced. We evaluate its benefit for subsequent policy learning in §4.3.3.

3.3.2 Stage II: Coarse-to-fine joint training

We load the Stage I video weights and initialize the entire action branch, including the action Transformer, visual projections, and embodiment-specific input and output projections. At each update, we sample video-action or simulation mode with equal probability and optimize the corresponding objective. The mode-sampling probabilities and training objectives remain unchanged across all three phases, while the data mixture changes as shown in Figure 6. Each phase continues from the full preceding checkpoint without reinitializing any modules or embodiment-specific projections.

- **Phase II-a: Broad co-training.** Broad human and action-unlabeled video dominate the mixture, while all action-labeled human and robot sources contribute to action prediction.
- **Phase II-b: Increased robot and bridge supervision.** We increase robot teleoperation and bridge data, retain a reduced video-only stream, and introduce F in simulation updates only.
- **Phase II-c: Target-robot co-training.** Target-robot data dominate, with a small replay mixture of human demonstrations, bridge data, and action-unlabeled video retained rather than switching to robot-only fine-tuning.

Replay maintains human action supervision and video prediction during robot adaptation to limit overfitting and forgetting. We evaluate this through matched robot-only and human-mixed continuations in §4.3.4 and analyze transfer across task categories in §4.3.5. For language-conditioned video-action updates, caption sampling shifts from dense descriptions toward original and paraphrased task instructions by II-c, following Figure 3; L1 clips retain descriptive captions. Simulation is therefore trained throughout Stage II; Stage III-sim adapts it to generated visual context rather than introducing it for the first time.

3.4 Post Training

Post-training aims to improve the policy through world-model-generated corrective supervision. Although Stages I–II establish action generation and trajectory-conditioned simulation, policy behavior cloning primarily covers successful expert demonstrations. It therefore provides limited recovery supervision for off-trajectory states encountered during closed-loop execution. We adopt a DAgger-style approach: synthesize deviation–recovery trajectories around expert demonstrations using the world model, then aggregate the recovery examples with expert data for policy fine-tuning.

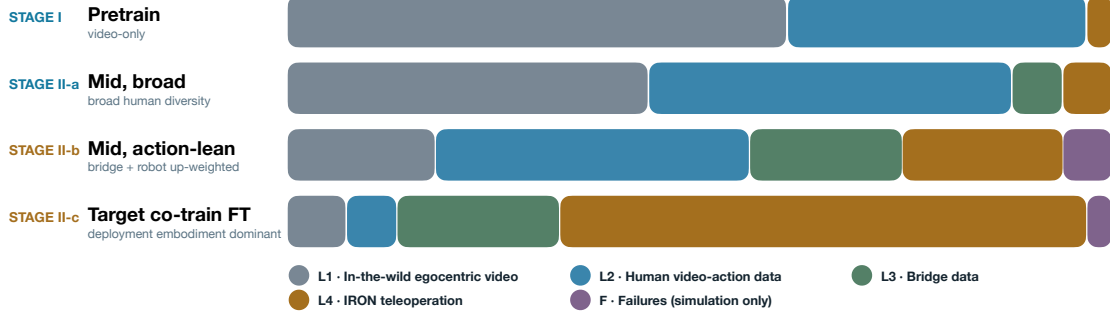


Figure 6: **Source-mixture curriculum through Stages I and II.** Segment widths indicate configured sampling weights; exact values are omitted. Stage I uses the participating sources as video-only data. Across Stage II, broad L1/L2 data are reduced while L3 bridge and L4 robot data gain weight, with target-robot data dominant in II-c. F enters in II-b through simulation updates only. These source proportions are distinct from the fixed 50/50 sampling of video–action and simulation modes.

The effectiveness of this approach depends on the simulator faithfully following prescribed motions under its own generated context. However, teacher-forced training uses clean recorded histories, whereas autoregressive inference conditions on previous predictions, allowing errors to accumulate. We therefore first adapt the simulator with self-gradient forcing in **Stage III-sim**. The adapted world model then synthesizes and filters recovery trajectories for policy updating in **Stage III-policy**. This ordered procedure links simulator adaptation, DAgger data synthesis, and policy refinement in a world-model-driven self-improvement loop.

3.4.1 Stage III-sim: Chunk-wise self-gradient forcing

We adopt self-gradient forcing (SGF) [54, 55] to adapt the simulator from teacher-forced training to self-generated context. SGF initializes a separate copy of the Phase II-c checkpoint and optimizes only the simulation objective, with the action Transformer inactive. The Phase II-c checkpoint is preserved for base-policy evaluation and subsequent policy initialization. We retain the chunk-wise representation defined in §3.2: an observed anchor z_0 followed by M prediction chunks $\mathbf{z}^{1:M}$. For chunk m , the simulator condition is $c^m = (\kappa^m, \mathbf{c}_{\text{cam}}^m)$, comprising the aligned 2D action-skeleton sequence and camera poses; no language condition is used. We combine self-conditioned rollout generation with context-aware flow matching against the corresponding recorded targets.

Self-conditioned rollout generation. Starting from z_0 , we generate chunks autoregressively using an eight-step ODE sampler Ψ_θ^K :

$$\widehat{\mathbf{z}}^m = \Psi_\theta^K(\epsilon_z^m \mid z_0, \widehat{\mathbf{z}}^{<m}, c^m), \quad \epsilon_z^m \sim \mathcal{N}(0, I), \quad K = 8. \quad (3)$$

Here $\widehat{\mathbf{z}}^{<m}$ contains all previously generated chunks and is empty for the first prediction. Only z_0 is taken from the recorded video; subsequent visual context is generated by the simulator, while skeleton controls and camera poses remain externally supplied. The complete rollout is executed without gradient tracking, using the same autoregressive context and sampling budget as SGF inference.

Context-aware flow-matching optimization. We detach the generated prefix and re-encode it with gradients so that training updates the context learner without backpropagating through the rollout sampler. For ground-truth chunk $\mathbf{z}^m$, we form $\mathbf{z}_{u_z}^m = (1 - u_z)\epsilon_z + u_z\mathbf{z}^m$, with $\epsilon_z \sim \mathcal{N}(0, I)$ and target velocity $v^{z, m*} = \mathbf{z}^m - \epsilon_z$. The objective is

$$\mathcal{L}_{\text{III-sim}} = \frac{1}{M} \sum_{m=1}^M w_m \mathbb{E}_{u_z, \epsilon_z} \left[\|v_\theta^z(\mathbf{z}_{u_z}^m, u_z, c^m \mid z_0, \text{sg}(\widehat{\mathbf{z}}^{<m})) - v^{z, m*}\|_2^2 \right], \quad (4)$$

where w_m weights each chunk and sg stops gradients through generated latent values. Rather than adopting a distribution-matching objective such as DMD [68], we retain ℓ_2 loss: trajectory-conditioned simulation must reproduce the consequences of the supplied controls, not merely generate plausible continuations. Distribution matching alone does not enforce this sample-level correspondence. This adaptation

prepares the simulator for the deviation–recovery synthesis used in Stage III-policy. Section 4.2.2 evaluates its effect on prediction fidelity and inference cost over short- and long-horizon rollouts.

3.4.2 Stage III-policy: Improve from aggregated corrections

Self-gradient forcing turns the simulation slice into a stable video predictor. We next use this improved world model to expand the policy training distribution beyond the successful expert trajectories observed in L4. Following the DAgger principle and world-model-based recovery synthesis [9, 59], we construct off-trajectory states around expert demonstrations, synthesize their visual consequences with the simulation slice, and retain only trajectories that return consistently to the expert manifold.

Deviation–recovery trajectory synthesis. Given an expert trajectory of length T , we choose a pivot frame t_p and identify the arm with the largest cumulative end-effector displacement as the arm to perturb. We displace its end-effector position along a sampled unit direction $v_d \in \mathbb{R}^3$, using step magnitude μ and deviation horizon $k \in \{8, 16\}$. The augmented position at synthesized step j is

$$p_{t_p+j}^{\text{aug}} = p_{t_p} + \mu d(j) v_d, \quad d(j) = \begin{cases} j, & 1 \leq j \leq k, \\ 2k - j, & k < j \leq 2k. \end{cases} \quad (5)$$

The first k frames move the selected end effector away from the expert trajectory, reaching the deviation endpoint at $t_p + k$; the next k frames return it to the pivot pose at $t_p + 2k$. We retain directions whose angle to the subsequent expert motion lies between 90° and 150° , avoiding both negligible perturbations and directly contradictory recovery supervision. The other arm, rotations, camera state, and remaining channels are held at their pivot values during the synthesized segment.

World-model rendering and expert-tail composition. We render a 2D skeleton sequence from the augmented state trajectory and use the real video up to the pivot as visual context. Conditioned on this skeleton sequence and the corresponding camera poses, held at their pivot values, the Stage III-sim checkpoint generates the visual deviation and recovery. We use the world model only for this newly inserted $2k$ -frame segment: the prefix before the pivot and the task continuation after recovery are taken from the original expert video. The resulting clip has the source layout:

$$\underbrace{o_{0:t_p}}_{\text{GT prefix}} \parallel \underbrace{\hat{o}_{t_p+1:t_p+2k}}_{\text{WM deviation+recovery}} \parallel \underbrace{o_{t_p+1:T}}_{\text{GT expert tail}}. \quad (6)$$

This composition prevents long-horizon world-model drift from corrupting the task continuation while preserving a visually grounded transition from an off-trajectory state back to successful behavior. The policy-training crop starts at the deviation endpoint $t_p + k$, so each example contains the difficult state, the complete recovery, and the subsequent expert task progress.

Consistency-guided filtering. World-model rollouts may contain weak perturbations, incomplete recovery, or visual discontinuities. We therefore compare the deviation endpoint, recovery endpoint, and expert pivot using DINOv3 feature similarity, PSNR, and SSIM¹. Let $\mathcal{Q} = \{q_{\text{DINO}}, q_{\text{PSNR}}, q_{\text{SSIM}}\}$ denote the three quality metrics. For every $q \in \mathcal{Q}$, an eligible rollout must satisfy:

$$\begin{cases} \Delta_q \triangleq q(\hat{o}_{t_p+2k}, o_{t_p}) - q(\hat{o}_{t_p+k}, o_{t_p}) > 0, \end{cases} \quad (7a)$$

$$\begin{cases} R_q \triangleq q(\hat{o}_{t_p+2k}, o_{t_p}) \geq \tau_q. \end{cases} \quad (7b)$$

Equation (7a) is a relative recovery criterion: the recovery endpoint must be closer to the expert pivot than the deviation endpoint. Equation (7b) is an absolute consistency criterion, where τ_q is the acceptance threshold for metric q ; it ensures that the recovery endpoint is sufficiently similar to the expert pivot to support a smooth transition into the expert tail. Among eligible candidates, we convert the DINOv3, depth-weighted PSNR, and depth-weighted SSIM margins into within-pool percentile ranks and average them into a composite quality score. We retain the highest-scoring candidates and construct a task-balanced mixture across the $k = 8$ and $k = 16$ recovery horizons. The resulting recovery-tail clips constitute the DAgger dataset D_{dagger} .

¹We compute semantic similarity using a frozen DINOv3-Large visual encoder [69]. For PSNR and SSIM, inverse-depth weighting emphasizes nearby robot–object interaction regions while down-weighting the background.

Dagger aggregation and policy update. Stage III-policy initializes a separate policy copy directly from Phase II-c, not from the SGF-adapted checkpoint. Only this policy copy is fine-tuned; the SGF simulator remains fixed. We retain the Phase II-c data recipe (§3.3.2) and mix in D_{dagger} for policy fine-tuning. DAgger recovery examples account for 8% of the training mixture, with the remaining data drawn from the Phase II-c sources. This retains the original robot supervision and replay data while adding corrective examples. Training on this mixture preserves nominal expert behavior while exposing the policy to off-distribution states paired with successful recovery trajectories, thereby improving policy robustness under closed-loop execution.

4 Experiments

4.1 Experimental Setup

Setup. We evaluate video prediction quality and real-robot policy performance, and examine the effects of training data and simulation-generated recovery supervision on policy learning. The base XPACE checkpoint serves as both a video simulator and a robot policy; the self-improvement experiment evaluates a subsequent, separately fine-tuned policy. Simulation predicts future video under prescribed skeleton controls and camera poses. Policy evaluation combines action prediction on recorded trajectories with closed-loop execution on the deployment robot, separating offline prediction quality from physical task completion. Simulation uses 480×832 images (height $\times$ width). External baselines assess overall policy performance, while internal ablations examine conditioning design, simulator adaptation, and the policy-training data recipe.

Evaluation benchmarks. Simulation and offline policy evaluation use the same evaluation dataset, comprising 19 subsets collected separately from the training data. Each episode belongs to one subset, and none of the source datasets is used for training. The in-distribution (ID) subsets contain pick-and-place interactions representative of robot training data. The out-of-distribution (OOD) subsets evaluate changes in visual conditions, including scene appearance, object poses, and distractors, as well as variation in objects, language instructions, spatial arrangements, task composition, and action primitives. OOD therefore includes both familiar behaviors under changed conditions and behaviors absent from robot demonstrations; some of the latter are covered by human training data. The subsets retain their original collection-based grouping rather than being repartitioned into uniformly task- or scene-disjoint splits. We report simulation results separately on ID and OOD subsets, and aggregate offline policy results across all 19 subsets under the label *Benchmark*.

Real-robot evaluation. We evaluate policies on the IRON-R01-1.11 humanoid, measuring task-progress scores and success rates on selected tasks with 20 trials per task per method under matched resets. We first present simulation fidelity and rollout efficiency (§4.2), followed by real-robot policy performance, human-to-robot skill transfer, training-data ablations, generalization analysis, and self-improvement (§4.3).

Evaluation questions. The experiments address six questions: (Q1) Can the simulator predict controllable futures and remain stable under autoregressive rollouts? (Q2) How well does the complete policy perform on real-robot ID, task-level OOD, and distractor-robustness tests? (Q3) Can manipulation skills learned from human data transfer to the robot on tasks absent from robot demonstrations? (Q4) What do video pretraining and continued human supervision contribute under matched adaptation settings? (Q5) How do these benefits vary with human and robot behavioral coverage? (Q6) Can simulation improve policy development through recovery supervision and qualitative inspection of policy-conditioned outcomes?

4.2 Simulation

We evaluate video prediction under prescribed skeleton controls and camera poses on the ID and OOD subsets of the evaluation dataset (§4.1). We first compare skeleton-conditioning strategies, then assess whether self-gradient forcing (SGF) improves autoregressive prediction for subsequent policy evaluation and recovery-data synthesis. We report PSNR and SSIM for image fidelity, Optical Flow Similarity

Table 1: **Skeleton-conditioning ablation.** Evaluation on 97-frame videos with 50 denoising steps. Bold marks the best quality metric within each split. Latency is in seconds and memory in GB; measurements use four significant figures.

Injection	In-distribution						Out-of-distribution					
	PSNR $\uparrow$	SSIM $\uparrow$	OFS $\uparrow$	DINO $\uparrow$	Latency $\downarrow$	Mem. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	OFS $\uparrow$	DINO $\uparrow$	Latency $\downarrow$	Mem. $\downarrow$
AdaLN	17.12	0.5818	0.3864	0.9770	61.25	41.41	16.88	0.5818	0.4500	0.9678	59.68	41.41
Cross-attention	16.29	0.5631	0.3308	0.9764	59.60	41.27	16.31	0.5703	0.3723	0.9688	59.50	41.27
Channel concat.	17.62	0.5902	0.3760	0.9732	57.76	41.27	17.47	0.5865	0.4565	0.9646	57.13	41.27
Token addition	18.07	0.6030	0.3854	0.9809	57.68	41.27	17.99	0.6109	0.4632	0.9765	56.96	41.27

(OFS)² for motion consistency, and DINO feature similarity³ for visual feature agreement. Higher values indicate better quality for all four metrics; latency and memory measure inference cost.

4.2.1 Short-horizon prediction: skeleton conditioning

Setup. All four variants are initialized from Wan2.2-TI2V-5B [70] and trained on L4+F with teacher forcing. They receive skeleton control latents aligned with the RGB video latents and are evaluated on 97-frame videos with 50 ODE sampling steps (Table 1).

Conditioning strategies. **Channel concatenation** joins RGB and skeleton latents before patch embedding; **token addition** embeds them separately and adds aligned tokens at the input. **AdaLN** uses skeleton tokens to modulate self-attention and feed-forward features in each DiT block, while **cross-attention** lets video tokens attend to skeleton tokens within the same latent frame in each block.

Results. Token addition achieves the highest PSNR, SSIM, and DINO scores on both splits and the highest OOD OFS, with ID OFS slightly below AdaLN. It improves PSNR over AdaLN by 0.95 dB on ID and 1.11 dB on OOD without increasing inference cost, supporting its use as the default.

4.2.2 Long-horizon rollout: self-gradient forcing

Setup. We further train the token-addition baseline with SGF on L4+F, following Stage III-sim (§3.4.1); this model is denoted **Baseline + SGF**. Both models generate video autoregressively from an initial observed frame under prescribed controls. The baseline uses 50 ODE sampling steps per chunk at inference, while the SGF-trained model uses 8 to match the sampling schedule used during SGF training. Evaluation covers 97-frame videos (100 ID and 52 OOD) and 193-frame videos (22 ID and 11 OOD), corresponding to six and twelve prediction chunks, respectively (Table 2).

Results. The SGF-trained baseline improves all four quality metrics on both splits at both horizons. At 193 frames, PSNR increases by 0.89 dB on ID and 1.01 dB on OOD, with $3.00\times$ and $3.30\times$ speedups and essentially unchanged memory use. The consistent gains in image fidelity and motion consistency across ID and OOD videos demonstrate the effectiveness of SGF for autoregressive prediction.

Simulator used for recovery-data synthesis. For recovery-data synthesis, we apply Stage III-sim SGF fine-tuning to the full XPACE checkpoint after Phase II-c (§3.3.2), rather than to the L4+F ablation baseline. Table 3 reports this final simulator’s prediction quality, averaged over the same evaluation videos at both horizons. The simulator achieves excellent prediction quality on both ID and OOD videos, supporting its use for recovery-data synthesis. We then use this simulator to generate the DAgger data for the real-robot self-improvement experiment in §4.3.6.

²OFS measures the average cosine similarity between optical flow fields computed using OpenCV Farneback from predicted and ground-truth videos.

³DINO feature similarity is computed using a DINOv3-large encoder.

Table 2: **Effect of self-gradient forcing on video prediction.** Baseline + SGF is the Token addition baseline further trained with SGF. The baseline uses 50 ODE sampling steps per chunk; the SGF-trained baseline uses 8. Bold marks the best quality metric within each horizon and split. Latency is in seconds and memory in GB; measurements use four significant figures.

Method	In-distribution						Out-of-distribution					
	PSNR $\uparrow$	SSIM $\uparrow$	OFS $\uparrow$	DINO $\uparrow$	Latency $\downarrow$	Mem. $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	OFS $\uparrow$	DINO $\uparrow$	Latency $\downarrow$	Mem. $\downarrow$
<i>Short-horizon (97 frames)</i>												
Baseline	18.07	0.6030	0.3854	0.9809	57.68	41.27	17.99	0.6109	0.4632	0.9765	56.96	41.27
Baseline + SGF	19.17	0.6394	0.5696	0.9848	20.95	41.27	18.72	0.6347	0.5467	0.9799	20.00	41.27
<i>Long-horizon (193 frames)</i>												
Baseline	16.94	0.5651	0.4035	0.9782	119.7	44.96	17.74	0.6009	0.4878	0.9755	120.2	44.96
Baseline + SGF	17.83	0.5976	0.4797	0.9825	39.86	44.94	18.75	0.6350	0.5563	0.9797	36.44	44.96

4.3 Policy

Evaluation suite. Our primary policy benchmark consists of three real-robot manipulation tasks executed on the deployment humanoid: banana pick-and-place across five fixed object positions, pouring water from a bottle, and bowl stacking. Each method is evaluated for 20 trials per task under matched resets. Tasks 1 and 2 are in-distribution (ID), whereas Task 3 evaluates task-level out-of-distribution (OOD) generalization relative to L4 robot demonstrations. Bowl stacking is absent from L4 but present in L2 human data and bridge data. Task 3 therefore tests transfer of human-observed skills to robot execution, not a task unseen across all training sources. Task 4, introduced separately below, adds distractor objects and is treated as a robustness experiment rather than part of the primary three-task benchmark. Because a binary outcome does not distinguish early failure from partial completion, we report two metrics per task: **success rate** (SR), and a dense **task-progress score** that awards partial credit at fixed milestones (e.g., reach, grasp, transport, place; per-task rubrics in §4.3.1). We compare against two external baselines under the same rollout protocol, embodiment, and trial budget: **GR00T** [71], a VLA policy, and **DreamZero** [35], a world-model-based policy. All methods are trained on the same robot teleoperation corpus and the same action-supervised human data; the video-only adaptation in Stage I is used by XPACE alone.

Table 3: **Final simulator (Phase II-c + SGF).**

Frames	PSNR $\uparrow$	SSIM $\uparrow$	OFS $\uparrow$	DINO $\uparrow$
<i>In-distribution</i>				
97	19.73	0.6612	0.6007	0.9853
193	18.11	0.6086	0.5170	0.9834
<i>Out-of-distribution</i>				
97	18.44	0.6274	0.5642	0.9758
193	18.05	0.6142	0.5552	0.9705

Comparison and attribution protocol. The external comparison varies the complete model and training recipe and therefore measures end-to-end performance. Attribution instead uses the internal controls below. We first compare robot-only and full-data training in closed loop, then examine Stage I exposure under identical Stage II fine-tuning and human co-training under matched initialization and budget. Coverage-stratified analysis identifies where the human-data gains occur. Finally, we evaluate policy refinement using recovery data from the fixed simulator. This order separates overall performance, training-recipe contributions, the distribution of transfer gains, and self-improvement.

4.3.1 Real-robot performance and robustness

Tasks and progress rubrics. Milestones and success criteria were fixed before evaluation. **Task 1 — banana pick-and-place (ID; position robustness):** “pick up the banana and place it in the basket.” The banana is placed at five predefined workspace locations, with four trials per location. Every method uses the same positions and resets for a fair comparison, so the task measures position robustness under controlled initial conditions rather than robustness to randomly sampled positions. Credit: 0.25 hand touches the banana, 0.5 banana lifted, 0.75 banana touches the basket, 1.0 banana placed inside (success). **Task 2 — pour water (ID):** “pour the water out of the bottle.” Credit in three equal milestones: hand

touches the bottle, bottle lifted, water poured (bottle rotated past 45°; success). This task stresses sustained wrist rotation with an in-hand object. **Task 3 — stack bowls (task OOD):** “stack the bowls together.” Credit: 0.25 hand touches a bowl, 0.5 bowl lifted, 0.75 bowl transported to contact the other bowl, 1.0 bowls stacked (success). This task stresses rim alignment and precise release. The progress score is the per-trial credit averaged over the 20 trials; SR counts only full completions.

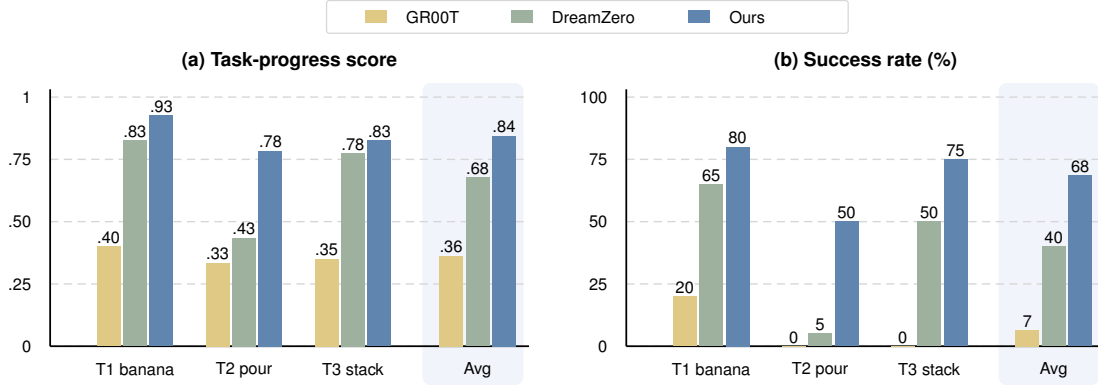


Figure 7: Real-robot policy evaluation (measured): (a) task-progress score and (b) success rate on the three-task suite (ID: Tasks 1–2; task-level OOD: Task 3), 20 trials per task per method. XPACE leads on every task under both metrics. Progress credits intermediate milestones, while success requires completing the entire manipulation.

Results. XPACE leads on every task under both metrics (Figure 7): 68.3% average SR vs. 40.0% for DreamZero and 6.7% for GR00T, with average progress 0.84 vs. 0.68 and 0.36. The comparison measures the complete model and training recipe: supervised corpora are matched, but XPACE additionally receives Stage I video adaptation. It should not be read as an architecture-only comparison.

The task breakdown is more informative than the average alone. Pouring has the lowest success rate for every method and the largest success-rate gap between XPACE and DreamZero (10/20 versus 1/20). Unlike transporting an object to a receptacle, it requires sustained rotation while maintaining an in-hand grasp. Progress also rises from 0.43 to 0.78, indicating an improvement beyond binary completion alone. Nevertheless, 0.78 progress alongside 50% success shows that reliable completion remains a challenge even for the full model. Aggregate scores distinguish partial progress from completion, but do not by themselves identify which motion caused a failure.

On banana pick-and-place, XPACE succeeds in 16/20 trials with successes at all five tested positions; on task-level OOD bowl stacking, it succeeds in 15/20. These results motivate a more targeted question: does the broader training corpus help when the robot encounters spatial variation or distracting objects, beyond the advantage of the complete recipe over external baselines?

Human experience improves policy robustness. Broader human and video experience is intended to support not only skill transfer, but also robust execution when spatial arrangements and surrounding objects vary. We test this in closed loop by comparing the **robot-only variant**, trained without human data or unsupervised video, with the **full-data model**. Task 1 evaluates banana pick-and-place across five

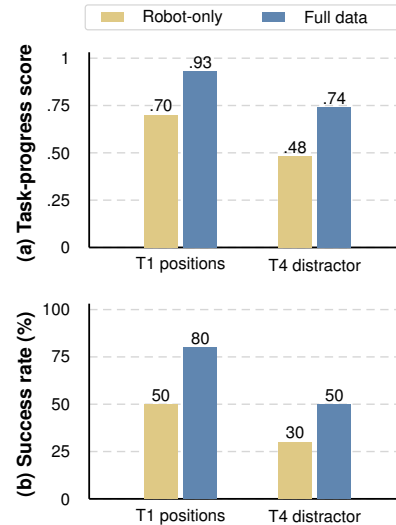


Figure 8: **Closed-loop human-data ablation.** (a) Task-progress score and (b) success rate for the robot-only and full-data models on position and distractor robustness, with 20 trials per task.

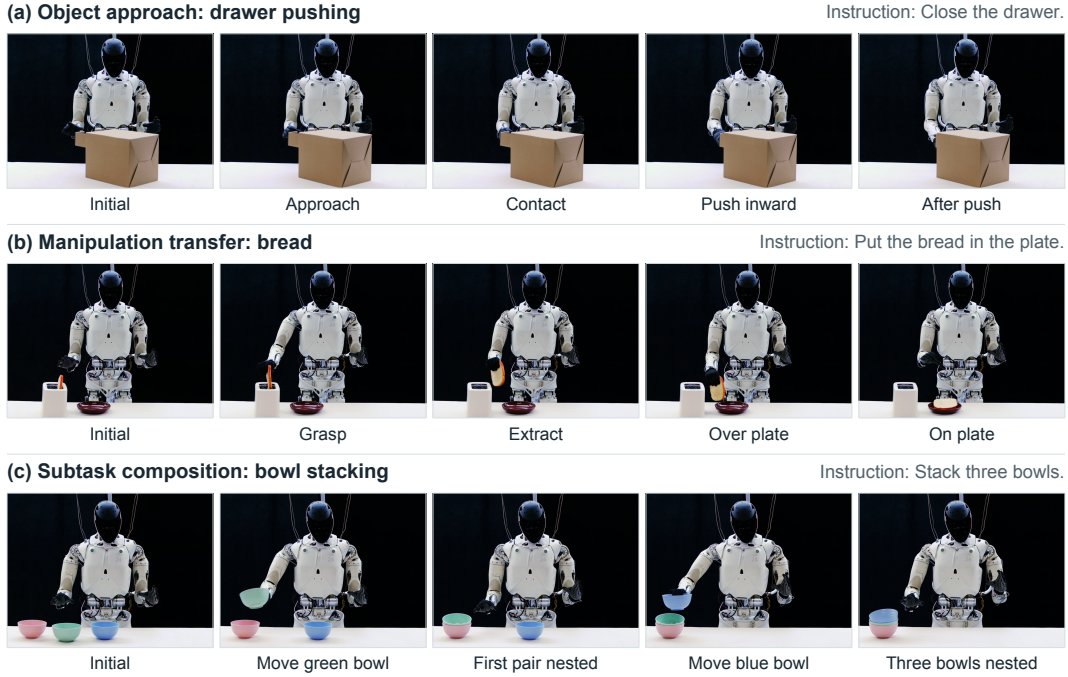


Figure 9: Real-robot transfer from human experience (qualitative). Each row shows five keyframes of XPACE following the displayed instruction. (a) Drawer pushing illustrates approach and contact with an object absent from robot training data. (b) Bread transfer adapts manipulation of an object present in both human and robot data to a toaster-to-plate task absent from robot demonstrations. (c) Three-bowl stacking composes two successive placements under a single task instruction. These examples illustrate a progression from object approach to manipulation transfer and sequential subtask composition.

fixed positions. A separate distractor test (Task 4) asks the robot to “pick up the block and put it in the box” with a cucumber, a flower, and a toy duck—objects absent from the robot training corpus—placed in the scene. Each model receives 20 trials per task, scored using the same four-milestone pick-and-place rubric.

The full-data model improves both spatial and distractor robustness (Figure 8). Across the five positions, success rises from 50% to 80% and progress from 0.70 to 0.93. With unseen distractors, success rises from 30% to 50% and progress from 0.48 to 0.74. The gains therefore extend beyond intermediate milestones to more completed trials, including in an in-distribution task family. These results support the value of broader experience for executing familiar manipulation skills under spatial and visual variation, not only for transferring behaviors absent from robot demonstrations.

This comparison evaluates human action supervision and unsupervised video learning as a combined data recipe. The following controlled studies (§4.3.3 and §4.3.4) examine video pretraining and continued human supervision separately through offline action prediction.

4.3.2 Human-to-robot skill transfer

Qualitative transfer. We examine three real-robot tasks that probe progressively richer forms of transfer from human experience: approaching an object absent from robot training, manipulating a familiar object with different source and target receptacles, and composing successive manipulations under a single task instruction. Figure 9 shows representative executions at these three levels.

Assessment criteria. Evaluation follows task-specific milestones and visible outcomes. For drawer pushing, we assess target-directed approach, contact, and inward displacement. Bread transfer requires

extracting the slice from the toaster and releasing it onto the plate. Bowl nesting requires completing both placements in sequence, with all three bowls remaining nested after the final release.

Results. Drawers are entirely absent from the robot training corpus. Nevertheless, XPACE approaches the drawer, establishes contact, and pushes it inward, illustrating transfer of approach-and-contact behavior from human experience to an unseen object. Bread appears in both human and robot data, but the toaster-to-plate task is absent from robot demonstrations. Successful extraction and placement extend transfer from object approach to familiar-object manipulation under a changed source–target configuration. Finally, under a single bowl-nesting instruction, the robot places the green bowl inside the pink bowl and then places the blue bowl into the resulting pair. The second placement builds on the state produced by the first, demonstrating sequential subtask composition within one task. Together, these examples progress from approaching an unseen object, through transferring object manipulation across receptacle configurations, to composing multiple manipulation steps toward a single goal.

4.3.3 Contribution of video pretraining

Setup. The closed-loop comparison above establishes the benefit of the full data recipe. We next examine its video-only pretraining component: does Stage I pretraining improve subsequent policy learning, and how does the benefit change with cumulative video exposure? We take pretrained checkpoints along a single Stage I trajectory corresponding to fractions $r \in \{0, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$ of the Stage I video corpus consumed ($r=0$ retains the pretrained video-DiT initialization but receives no Stage I adaptation), then run an *identical* supervised action fine-tuning recipe from each, with the action head freshly initialized from the same seed in every arm so the backbone is the only variable. We report teacher-forced action evaluation loss (MSE in the normalized action space, using recorded context for each chunk) on the shared evaluation dataset described in §4.1. Every arm is evaluated on the same episodes, with results aggregated across the suite’s mixed distributional coverage.

Results and implications. Stage I improves policy learning under the fixed adaptation budget: every nonzero exposure reduces action loss on the evaluation dataset, as shown in Figure 10. The first eighth of exposure already captures approximately two-thirds of the full-exposure benchmark reduction (8.1% versus 12.5%). The reduction reaches 9.0% at both quarter and half exposure, and 12.5% with full Stage I exposure. Thus, modest exposure provides a substantial early benefit, while full exposure gives the best aggregate result in this sweep.

The benefit appears after every backbone receives the same action-training recipe. Video prediction therefore provides useful preparation for control even without action labels, supporting the separation of broad predictive learning from action grounding in §3.3. Robot supervision need not supply all of the experience that makes action prediction effective. This comparison measures the value of additional Stage I training: corpus exposure and optimization compute increase together.

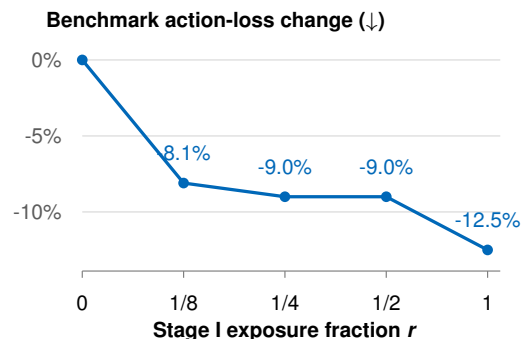


Figure 10: Stage I exposure and benchmark action loss relative to $r=0$, after matched fine-tuning (tail-window average).

4.3.4 Human supervision during robot adaptation

Should human supervision end when robot adaptation begins? Video-only adaptation tests whether visual experience benefits the policy backbone (§4.3.3). Here we ask whether human action supervision should serve only as an initialization or remain available while the model learns robot execution. We compare three recipes on the held-out action benchmark: a robot-only baseline; human–robot mid-training followed by robot-only fine-tuning (mid-train→FT); human–robot *co-training* kept through the final stage, including the bridge-mined subsets. The two human-data recipes share one mid-trained initialization, learning rate, and budget, so the mid-train→FT vs. co-train comparison isolates a single variable:

whether human data stays in the mixture during the final stage. Table 4 reports changes relative to the robot-only baseline, averaged over nine late-training checkpoints.

Table 4: Supervised human data: held-out action-loss change vs. a robot-only baseline (mean over nine checkpoints; across-checkpoint s.d. in percentage points in parentheses). The checkpoint spread describes variation within a training run, not uncertainty across independent seeds. Co-training gives the larger mean reduction under a matched initialization.

Recipe	Benchmark Δ action loss↓
Robot-only	0 (ref.)
Human mid-training then robot-only FT	−8.5% (s.d. 2.7)
Human–robot co-training	−14.0% (s.d. 2.8)

Human experience as a regularizer during robot adaptation. With the same mid-trained initialization and final-stage budget, co-training reduces mean benchmark action loss by 14.0% relative to robot-only, compared with 8.5% for human mid-training followed by robot-only fine-tuning (Table 4). Because the two human-data arms begin from the same checkpoint, their gap shows that the benefit of human experience is not exhausted by mid-training. Its continued presence in the adaptation mixture matters for the final action predictor.

A plausible explanation is that continued human supervision constrains how the shared backbone specializes. Robot-only updates reward performance on a narrower set of scenes and behaviors, but do not require the model to retain capabilities supported only by earlier human experience. This creates a risk of overfitting to robot-specific correlations and forgetting previously learned behaviors. Human co-training keeps those behaviors under active supervision: human action targets continue to train motion prediction, while paired video targets encourage the backbone to retain visual-dynamics knowledge beyond the robot corpus. In this interpretation, human data acts as rehearsal during adaptation, allowing robot execution to improve without leaving broader visuomotor capabilities unconstrained.

This motivates the Phase II-c design: retain human supervision to preserve transferable features and continue learning beyond the robot corpus while specializing to deployment. We next examine whether the benefit is confined to behaviors missing from robot training or also improves those the robot has already demonstrated.

4.3.5 Cross-embodiment generalization

Coverage-stratified analysis. The preceding comparison establishes the aggregate benefit of human supervision. Here we examine how that benefit is distributed across familiar robot behaviors and behaviors absent from robot demonstrations. We group the 19 curated benchmark subsets into eight task-semantic families, then assign coverage strata along two axes: whether the corresponding action family is *covered by robot teleoperation*, and whether it is *densely covered by the human corpus*. These semantic families and coverage strata are secondary analysis groupings, not the benchmark’s original dataset partitions. Figure 11 reports the reduction in action evaluation loss for the two human-data recipes of §4.3.4 relative to a robot-only baseline, averaged across nine late-training checkpoints. Stratification reveals two complementary contributions: strengthening behaviors covered by teleoperation and extending supervision to behaviors that robot training lacks.

Human supervision strengthens and extends robot capabilities. Human experience benefits both sides of the coverage spectrum. Co-training reduces action loss by 10.0% on robot-dense, human-dense behaviors and by 22.7% on robot-absent, human-dense behaviors (Figure 11). Shared task coverage is therefore useful, not redundant, while broader human coverage enables transfer beyond the behaviors demonstrated by the robot. The two human-data recipes show the same ordering across strata, with smaller gains under sparse human coverage and no comparable benefit when both sources lack the behavior. The central finding is not that human data should replace robot demonstrations, but that robot demonstrations need not define the full range of experience available for learning control.

Implications for the data curriculum. Together with the matched adaptation comparison, these results support retaining human supervision in Phase II-c for both robot-covered and robot-absent behaviors.

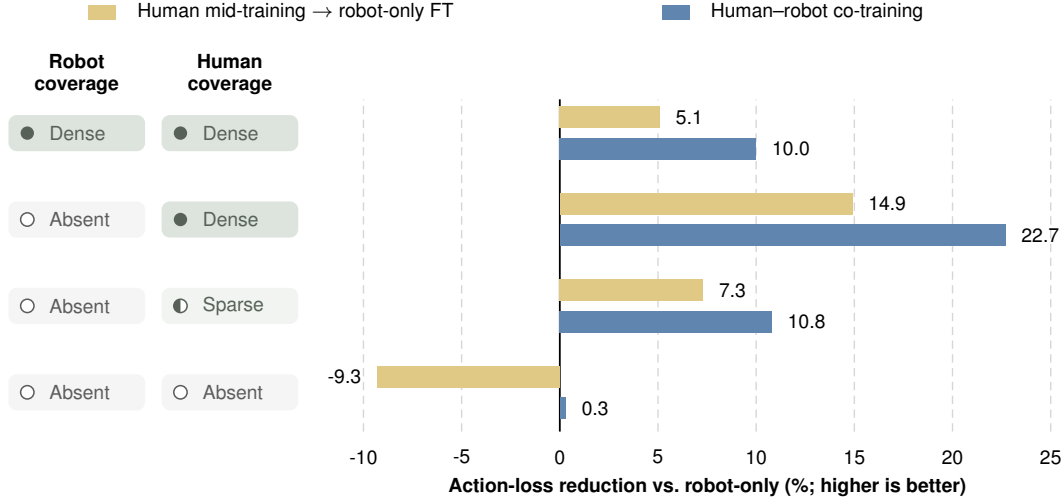


Figure 11: **Human-to-robot transfer across training-data coverage.** Relative reduction in held-out action loss against robot-only, averaged over nine late-training checkpoints. Positive values indicate lower loss; negative values indicate higher loss. Coverage refers to the corresponding task family in each training source.

They are consistent with the complementary data roles in §3.1, but do not isolate the contribution of bridge data or establish a particular feature-alignment mechanism. The next experiment examines a different source of additional policy supervision: recovery examples synthesized by the simulator rather than drawn from recorded human experience.

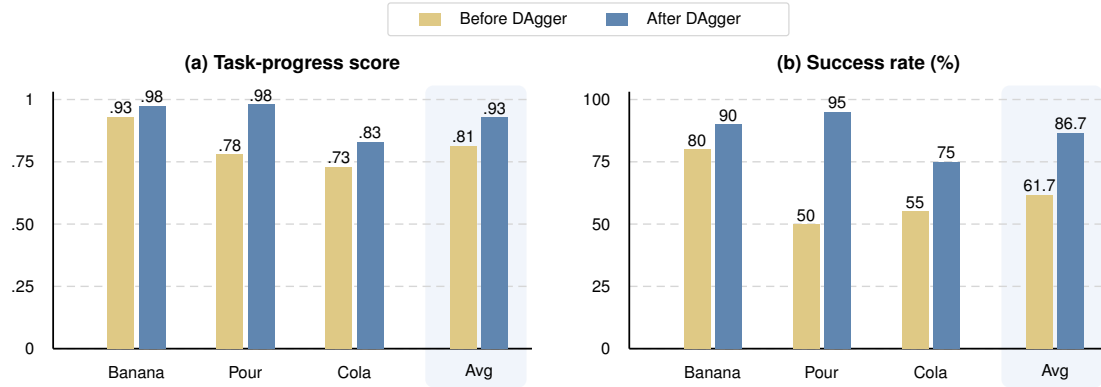


Figure 12: **Policy self-improvement with world-model-generated recovery data.** Before DAgger is the Phase II-c checkpoint used in the main experiments; After DAgger is its policy-fine-tuned copy. We compare banana pick-and-place, water pouring, and cola handover using (a) normalized task-progress score and (b) success rate, with 20 trials per task per method. Before DAgger results for banana pick-and-place and water pouring reuse the main-experiment trials, with identical task definitions and scoring. The DAgger-fine-tuned policy uses the Phase II-c data recipe with an 8% mixture of filtered recovery trajectories synthesized by the SGF-adapted Phase II-c simulator (Table 3). Shaded columns show unweighted task averages.

4.3.6 Policy self-improvement

Joint world and action modeling enables policy self-improvement: the world model synthesizes recovery experience that feeds back into policy training. We evaluate this simulation-to-policy feedback through

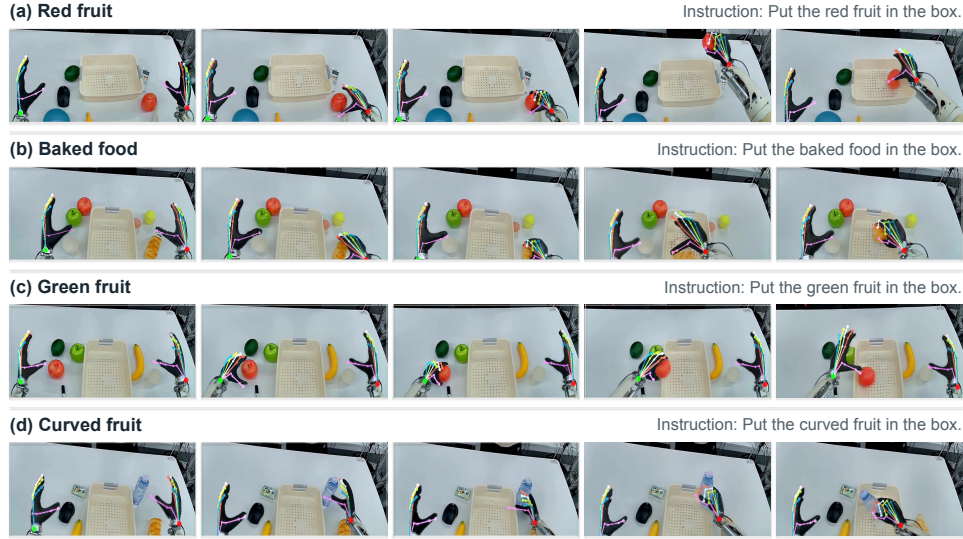


Figure 13: **Policy-simulation closed-loop evaluation.** Policy-predicted actions drive the simulator through skeleton controls and corresponding camera poses, and generated frames are fed back to the policy as subsequent observations. Each row shows the instruction and simulated frames with skeleton overlays. The first two rows show correct target selection; the last two show a red fruit selected instead of a green fruit, and a bottle instead of a curved fruit.

real-robot performance after DAGger fine-tuning. Policy-conditioned simulation further supports this process by exposing behavioral errors for qualitative diagnosis.

Simulation-driven policy improvement. We compare **Before DAGger**, the Phase II-c checkpoint used in the main experiments, with **After DAGger**, a separately fine-tuned policy initialized from that checkpoint. Only the policy copy is fine-tuned, using a mixture of 8% synthesized recovery examples and 92% data sampled according to the Phase II-c recipe; the SGF simulator remains fixed. This fine-tuning stage completes in less than one day. The SGF-adapted Phase II-c simulator (Table 3) generates deviation-recovery trajectories; consistency-filtered recovery clips are joined with recorded expert tails as supervision, following §3.4.2. Real-robot evaluation covers banana pick-and-place, water pouring, and cola handover, reporting task-progress scores, success rates, and their unweighted task averages, as shown in Figure 12.

Tasks and evaluation protocol. Banana pick-and-place and water pouring use exactly the tasks, scoring rules, and initial conditions defined in §4.3.1; banana success requires placement inside the basket, not grasping alone. Their Before DAGger results reuse the same evaluation trials reported in Figure 7, rather than a new evaluation. The comparison follows the main protocol of 20 trials per task per method, with matched initial conditions and resets before and after fine-tuning. For **cola handover**, the robot hands the cola to a person standing in front of it. The task awards one point for each of three cumulative milestones: the hand touches the cola, the cola is lifted, and the cola is handed to the person. Completing the handover constitutes success and earns the maximum of three points. We divide each trial’s raw score by three and average over the 20 trials to obtain normalized task progress; success rate is the fraction of trials completing all three milestones.

Results. DAGger fine-tuning consistently improves task-progress scores and success rates across all three tasks. The mean task-progress score increases from 0.81 to 0.93, while the mean success rate increases from 61.7% to 86.7%, an absolute improvement of 25 percentage points. The largest improvement occurs in water pouring, where task progress increases from 0.78 to 0.98 and success rate from 50% to 95%. Success rates also increase for banana pick-and-place (80% to 90%) and cola handover (55% to 75%), indicating that the benefit is not restricted to a single task.

The concurrent improvements in task progress and success rate indicate more complete task execution, rather than gains in intermediate milestones alone. These real-robot results provide empirical support for simulation-driven policy self-improvement through fine-tuning with world-model-generated recovery supervision. The reported comparison is before versus after fine-tuning; it does not separate recovery-data effects from the additional optimization budget.

Policy–simulation closed-loop diagnosis. We inspect policy behavior through closed-loop interaction with the simulator. Given an observation, the current state, and a language instruction, the policy model predicts actions, which supply skeleton controls and corresponding camera poses for the simulator to generate a predicted video. The generated frames then serve as observations for the policy model to predict the next actions, forming a repeated policy–simulation closed-loop evaluation. This procedure supports qualitative assessment of instruction following, target selection, and visible task progress without requiring physical robot execution for each rollout.

Figure 13 presents four object-placement examples. In the first two, the policy selects the requested red fruit and baked food, respectively. The remaining examples reveal incorrect target selection: a red fruit instead of the requested green fruit, and a bottle instead of the requested curved fruit. These examples demonstrate that simulated rollouts can identify instruction-following errors even when the generated motions appear plausible. By reducing the need for robot operation and physical scene resets, simulation provides a lower-cost setting for qualitative policy evaluation and failure analysis before real-robot testing.

5 Conclusion

We introduced XPACE, a unified embodied world model that serves as both a world action model and a world simulator through a shared predictive video backbone. Our results highlight the complementary roles of human experience and robot supervision: human data broaden the interaction knowledge available for learning, while robot demonstrations ground that knowledge in executable control. Retaining human experience during robot adaptation supports robustness and skill transfer, suggesting that robot demonstrations need not define the full range of behaviors a policy can learn.

Beyond learning from recorded demonstrations, the learned world simulator provides recovery experience for policy improvement and a setting for inspecting policy behavior. This connects prediction and control not only through shared representations, but also through the use of generated experience to improve subsequent action learning.

Looking ahead, an embodied world model could help a robot decide not only how to act, but also what experience to seek next. Human observations could suggest unfamiliar skills, imagined rollouts could expose gaps in the policy, and targeted physical interaction could test these predictions and refine both the policy and the simulator. This would extend the role of simulation from generating recovery examples to guiding exploration, comparing candidate actions, and constructing progressively more demanding learning curricula. Realizing this vision requires moving beyond the demonstration-centered recovery setting studied here: models must represent uncertainty, remain reliable over longer and contact-rich interactions, and incorporate real-world feedback without forgetting previously acquired skills. The longer-term opportunity is a robot whose experience is not merely a fixed training resource, but something it can actively expand—learning from people, testing possibilities in imagination, and grounding what it learns in the physical world.

Contributors

Jiacheng Wei^{*§}, Jerry Bai^{*§}, Xiaoyu Yue^{*}, Zidong Wang^{*}, Xiaoyang Guo^{*†}, Cheng Chen, Fanqi Pu, Fan Wu, Zhixu Yue, Yizhuo Li, Feng Qiu, Bo Liu, Yuying Ge, Hui Zhou, Chenyi Chen, Yixiao Ge[‡]

^{*}Core contribution. [§]Equal contribution. [†]Project lead. [‡]Supervision. All authors are with XPENG Robotics.

Acknowledgement Yuguo Gan, Jianping Li, Yufan Ren, Kunpeng Song, Wenqian Sun, Huimin Pan, Siyang Wang, Xiwen Zhang, Xiaoji Zheng, Ruochong Zheng, Yingji Zhong

References

- [1] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. URL <https://arxiv.org/abs/1803.10122>.
- [2] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. DayDreamer: World models for physical robot learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR, 2023. URL <https://proceedings.mlr.press/v205/wu23c.html>.
- [3] Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning Interactive Real-World Simulators. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c4d66eae503694424123b93ac0fbaf17-Abstract-Conference.html.
- [4] Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. IRASim: Learning interactive real-robot action simulators. *arXiv preprint arXiv:2406.14540*, 2024. URL <https://arxiv.org/abs/2406.14540>.
- [5] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.015. URL <https://www.roboticsproceedings.org/rss21/p015.html>.
- [6] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.074. URL <https://www.roboticsproceedings.org/rss21/p074.html>.
- [7] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.012. URL <https://roboticsproceedings.org/rss19/p012.html>.
- [8] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.120. URL <https://www.roboticsproceedings.org/rss20/p120.html>.
- [9] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, 2011. URL <https://proceedings.mlr.press/v15/ross11a.html>.
- [10] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, et al. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Grauman_Ego4D_Around_the_World_in_3000_Hours_of_Egocentric_Video_CVPR_2022_paper.html.
- [11] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *IEEE International Conference on Robotics and Automation*, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL <https://arxiv.org/abs/2310.08864>.
- [12] Marion Lepert, Jiaying Fang, and Jeannette Bohg. Phantom: Training Robots Without Robots Using Only Human Videos. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 4545–4565. PMLR, 2025. URL <https://proceedings.mlr.press/v305/lepert25a.html>.

-
- [13] Sicheng Xie, Haidong Cao, Zejia Weng, Zhen Xing, Haoran Chen, Shiwei Shen, Jiaqi Leng, Zuxuan Wu, and Yu-Gang Jiang. Human2Robot: Learning Robot Actions from Paired Human-Robot Videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(13):11078–11086, 2026. doi: 10.1609/aaai.v40i13.38086.
- [14] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. EgoScale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026. URL <https://arxiv.org/abs/2602.16710>.
- [15] Baoyu Li, Xinchun Yin, Mengying Lin, Yixin Zhang, and Danfei Xu. EgoWAM: World action models beyond pixels with in-the-wild egocentric human data. *arXiv preprint arXiv:2607.08436*, 2026. URL <https://arxiv.org/abs/2607.08436>.
- [16] Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. DreamDojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026. URL <https://arxiv.org/abs/2602.06949>.
- [17] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, et al. DreamGen: Unlocking generalization in robot learning through neural trajectories. *arXiv preprint arXiv:2505.12705v1*, 2025. URL <https://arxiv.org/abs/2505.12705v1>.
- [18] Boyang Wang, Haoran Zhang, Shujie Zhang, Jinkun Hao, Mingda Jia, Qi Lv, Yucheng Mao, Zhaoyang Lyu, Jia Zeng, Xudong Xu, and Jiangmiao Pang. RoboVIP: Multi-view video generation with visual identity prompting augments robot manipulation. *arXiv preprint arXiv:2601.05241*, 2026. URL <https://arxiv.org/abs/2601.05241>.
- [19] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL <https://arxiv.org/abs/2212.06817>.
- [20] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. URL <https://arxiv.org/abs/2405.12213>.
- [21] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. URL <https://arxiv.org/abs/2303.04137>.
- [22] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. 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 *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713, 2025. URL <https://proceedings.mlr.press/v270/kim25c.html>.
- [23] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolò Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0 : A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010. URL <https://www.roboticsproceedings.org/rss21/p010.html>.
- [24] Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world. *arXiv preprint arXiv:2503.20020*, 2025. URL <https://arxiv.org/abs/2503.20020>.
- [25] 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. URL <https://arxiv.org/abs/2510.10125>.

-
- [26] Cheng Chen, Jerry Bai, Jiacheng Wei, Boyu Chen, Xiaoji Zheng, Fan Wu, Minghao Yang, Tianrun Chen, Ruibo Li, Xiaoyu Yue, Xiaoyang Guo, Yixiao Ge, Guosheng Lin, and Fayao Liu. AnyWorld: Factorized egocentric world models for cross-embodiment generalization. *arXiv preprint arXiv:2608.29242*, 2026. URL <https://arxiv.org/abs/2608.29242>.
- [27] Yuncong Yang, Zhengtao Han, Furkan Ozyurt, Zeyuan Yang, Han Yang, Junyi Cao, Haoyu Zhen, Yilun Du, and Chuang Gan. SyncWorld: Visual calibration enables world models as zero-shot simulators. *arXiv preprint arXiv:2609.09155*, 2026. URL <https://arxiv.org/abs/2609.09155>.
- [28] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1d5b9233ad716a43be5c0d3023cb82d0-Paper-Conference.pdf.
- [29] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023. URL <https://arxiv.org/abs/2310.10639>.
- [30] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. RoboDreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024. URL <https://arxiv.org/abs/2404.12377>.
- [31] Ying Li, Xiaobao Wei, Xiaowei Chi, Yuming Li, Zhongyu Zhao, Hao Wang, Ningning Ma, Ming Lu, and Shanghang Zhang. ManipDreamer: Boosting robotic manipulation world model with action tree and visual guidance. *arXiv preprint arXiv:2504.16464*, 2025. URL <https://arxiv.org/abs/2504.16464>.
- [32] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024. URL <https://arxiv.org/abs/2412.14803>.
- [33] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025. URL <https://arxiv.org/abs/2508.00795>.
- [34] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025. URL <https://arxiv.org/abs/2512.06963>.
- [35] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL <https://arxiv.org/abs/2602.15922>.
- [36] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 35101–35113, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Bi_Motus_A_Unified_Latent_Action_World_Model_CVPR_2026_paper.html.
- [37] Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Jingbin Cai, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng Yan, Maoqing Yao, and Guanghui Ren. Genie envisioner: A unified world foundation platform for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL <https://arxiv.org/abs/2508.05635>.
- [38] Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0 -WM: A unified video-action world model for robotic manipulation. *arXiv preprint arXiv:2606.01027*, 2026. URL <https://arxiv.org/abs/2606.01027>.

-
- [39] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. URL <https://arxiv.org/abs/2601.21998>.
- [40] Haofeng Sun, Jiangbo Pei, Fei Kang, Zexiang Liu, Yaokun Li, Boyi Jiang, Hua Xue, Cindy Zhou, Wei Li, Yichen Wei, Mengyin An, Fanliang Zhao, Biao Jiang, Zile Wang, Yang Liu, and Yangguang Li. Rieman-1.0: An embodied world action model for physical AI. *arXiv preprint arXiv:2608.27033*, 2026. URL <https://arxiv.org/abs/2608.27033>.
- [41] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. URL <https://arxiv.org/abs/2601.16163>.
- [42] Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026. URL <https://arxiv.org/abs/2604.26694>.
- [43] Haoyu Zhen, Zixian Gao, Qiao Sun, Yilin Zhao, Yuncong Yang, Yilun Du, Pengsheng Guo, Tsun-Hsuan Wang, Yi-Ling Qiao, and Chuang Gan. Action images: End-to-end policy learning via multiview video generation. *arXiv preprint arXiv:2604.06168*, 2026. URL <https://arxiv.org/abs/2604.06168>.
- [44] Yixiang Chen, Peiyan Li, Yuan Xu, Qisen Ma, Jiabing Yang, Kai Wang, Jianhua Yang, Dong An, et al. FlowWAM: Optical flow as a unified action representation for world action models. *arXiv preprint arXiv:2607.13017*, 2026. URL <https://arxiv.org/abs/2607.13017>.
- [45] Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-H0.7: A latent world-action model from egocentric videos. *arXiv preprint arXiv:2605.00078*, 2026. URL <https://arxiv.org/abs/2605.00078>.
- [46] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL <https://arxiv.org/abs/2603.16666>.
- [47] Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, et al. GigaWorld-Policy: An efficient action-centered world-action model. *arXiv preprint arXiv:2603.17240*, 2026. URL <https://arxiv.org/abs/2603.17240>.
- [48] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 892–909. PMLR, 2023. URL <https://proceedings.mlr.press/v205/nair23a.html>.
- [49] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 416–426. PMLR, 2023. URL <https://proceedings.mlr.press/v205/radosavovic23a.html>.
- [50] Siddharth Karamcheti, Suraj Nair, Annie S. Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.032. URL <https://roboticsproceedings.org/rss19/p032.html>.
- [51] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zhohus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khali-dov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL <https://arxiv.org/abs/2506.09985>.

-
- [52] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025. URL <https://arxiv.org/abs/2410.11758>.
- [53] Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 18744–18771. PMLR, 2025. URL <https://proceedings.mlr.press/v267/gao25u.html>.
- [54] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self Forcing: Bridging the Train-Test Gap in Autoregressive Video Diffusion. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f4823f831af67a3ef15e41a85434422a-Abstract-Conference.html.
- [55] Junhao Zhuang, Shiyi Zhang, Yuxuan Bian, Yaowei Li, Yawen Luo, Yijun Liu, Weiyang Jin, Songchun Zhang, Xianglong He, Xuying Zhang, Haoran Li, Haoyang Huang, Zeyue Xue, and Nan Duan. Self gradient forcing: Native long video extrapolation. *arXiv preprint arXiv:2607.20368*, 2026. URL <https://arxiv.org/abs/2607.20368>.
- [56] Jai Bardhan, Patrik Drozdik, Josef Sivic, and Vladimir Petrik. Persistent robot world models: Stabilizing multi-step rollouts via reinforcement learning. *arXiv preprint arXiv:2603.25685*, 2026. URL <https://arxiv.org/abs/2603.25685>.
- [57] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. URL <https://arxiv.org/abs/1912.01603>.
- [58] 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. URL <https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html>.
- [59] Anlan Yu, Zaishu Chen, Peili Song, Zhiqing Hong, Haotian Wang, Desheng Zhang, Tian He, Yi Ding, and Daqing Zhang. WM-Dagger: Enabling efficient data aggregation for imitation learning with world models. *arXiv preprint arXiv:2604.11351*, 2026. URL <https://arxiv.org/abs/2604.11351>.
- [60] Yaxuan Li, Zhongyi Zhou, Yefei Chen, Yanjiang Guo, Jiaming Liu, Shanghang Zhang, Jianyu Chen, and Yichen Zhu. Hi-WM: Human-in-the-world-model for scalable robot post-training. *arXiv preprint arXiv:2604.21741*, 2026. URL <https://arxiv.org/abs/2604.21741>.
- [61] 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. URL <https://arxiv.org/abs/2602.11075>.
- [62] Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-VLA-Loop: Closed-loop learning of video world model and VLA policy. *arXiv preprint arXiv:2602.06508*, 2026. URL <https://arxiv.org/abs/2602.06508>.
- [63] 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. URL <https://arxiv.org/abs/2602.12063>.
- [64] Tenny Yin, Zhiting Mei, Zhonghe Zheng, Miyu Yamane, David Wang, Jade Sceats, Samuel M. Bateman, Lihan Zha, Apurva Badithela, Ola Shorinwa, and Anirudha Majumdar. PlayWorld: Learning robot world models from autonomous play. *arXiv preprint arXiv:2603.09030*, 2026. URL <https://arxiv.org/abs/2603.09030>.

-
- [65] Hongzhe Bi, Zihao Zhou, Yihang Tang, Jingrui Pang, Shuhe Huang, Haitian Liu, Runqing Wang, Shuai Huang, Yichen Wang, Yiming Cheng, Ruowen Zhao, Zhenghua Li, Hengkai Tan, Xiaolong Liu, Jinhui Wan, Jiabao Liu, Min Zhao, Fan Bao, and Jun Zhu. Motus2: A self-evolving general world model for dexterous manipulation. *arXiv preprint arXiv:2608.30237*, 2026. URL <https://arxiv.org/abs/2608.30237>.
- [66] Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. WorldGym: World model as an environment for policy evaluation. *arXiv preprint arXiv:2506.00613*, 2025. URL <https://arxiv.org/abs/2506.00613v3>.
- [67] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=PqvMRDCJT9t>.
- [68] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6613–6623. IEEE, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yin_One-step_Diffusion_with_Distribution_Matching_Distillation_CVPR_2024_paper.html.
- [69] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025. URL <https://arxiv.org/abs/2508.10104>.
- [70] Team Wan, Ang Wang, Baole Ai, Bin Wen, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL <https://arxiv.org/abs/2503.20314>.
- [71] Johan Bjorck et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL <https://arxiv.org/abs/2503.14734>.