World Action Models: A Survey

Qiuhong Shen , Shihua Zhang , Yue Liao , Qi Li , Zhenxiong Tan , Shizun Wang , Shuicheng Yan , Xinchao Wang *

Posted Date: 18 June 2026

doi: 10.20944/preprints202606.1403.v1

Keywords: World Action Models

 doi:10.20944/preprints202606.1403.v1

Article

World Action Models: A Survey

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan and Xinchao Wang *

National University of Singapore
* Correspondence: xinchao@nus.edu.sg

Abstract

World Action Models (WAMs) are embodied predictive-action models that make a forecast of the future available to action. Recent WAMs repurpose large video generation models, and a parallel line relies on language or vision-language backbones without a video-generation core. This rapid expansion has blurred the boundary among broad world models, video generation models, action-grounded video world models, Vision-Language-Action policies, and WAMs. This survey gives the field a common account. It first clarifies these boundaries, then organizes existing works through two complementary views. The first view asks what each method is required to generate, spanning rendered futures, latent futures, and video-generation-free action reasoning. The second view decomposes each method by predictive substrate, backbone, action coupling, and deployment regime. This anatomy supports a unified discussion of interactability, causality, persistence, physical plausibility, and generalization, followed by data, evaluation, and open challenges. Across these axes, a consistent design pattern emerges: WAMs are not simply video generators with action heads, but predictive-action methods whose design choices trade representational richness against compute, memory, latency, and action-label cost. The field is moving toward methods that generate less of the future while preserving what control requires. The survey homepage is available at https://world-action-models.github.io/.

Keywords: World Action Models

1. Introduction

The long-standing goal of embodied AI is to build agents that perceive, reason, and act in unstructured physical environments. Over the past two years the field has moved toward policies that do more than react to the current observation: they also anticipate how the world may change before choosing what to do. The trend began with Vision-Language-Action (VLA) models, which repurpose pretrained vision-language models to map an observation and an instruction directly to an action. Works such as RT-2 [1], OpenVLA [2], and π_0 [3] showed that the semantic knowledge accumulated during internet-scale pretraining can be grounded into robotic behavior, generalizing across instructions, objects, and embodiments with modest fine-tuning. A standard VLA, however, never models how the environment changes under its own intervention. It learns an observation-to-action mapping with no explicit account of physics, contact, or perspective change. This omission limits the policy's ability to reason about consequences before acting.

A rapidly growing body of work now equips such a policy with an explicit predictive component drawn from world models. Large-scale pretrained video generation models serve as a major source of such components for predicting future dynamics, but they are not the only choice among existing works. Despite their variety, these works fall into three recognizable streams, distinguished by how far the model carries its prediction before it decodes an action. The first stream adopts a video generation backbone all the way to pixels and then decodes action from the rendered future, as in UniPi [4] and the video-plan policies that followed [5,6]. The second keeps the same backbone but halts before pixel decoding, recovering action from intermediate latents, flow fields, or masks, as in VPP [7] and later

 doi:10.20944/preprints202606.1403.v1

latent-space foresight [8,9]. The third forgoes the video generator entirely and predicts in a language-token, audio, or joint-embedding space, as in DUST [10], Audio-WM [11], and latent-dynamics policies over frozen vision features [12]. Despite these differences, the three streams share a contract that a plain VLA does not. Each makes a future substrate action-facing, either by training future and action prediction together or by using a separate action module that consumes the predicted future. We name this family **World Action Models (WAMs)**, following recent usage of this terminology, and make its definition precise in Section 4.

This label has spread faster than the understanding behind it. Works that call themselves World Action Models arrive from the video-generation, robot-learning, and language-model communities, and two of them can agree on the name while sharing almost no implementation details. Two others can avoid the name entirely and yet build the same thing. Underneath the disagreement lies a common interface question: what predicted future is retained for action, and where along the path is the action decoded. Each answer buys predictive richness at a price in compute, memory, and latency inside a control loop, which is why the strongest WAMs tend to dream less of the future while still acting on what they need.

This survey organizes the resulting works so that any new method can be placed and compared, and it does so from two complementary viewpoints. The first is a design-philosophy-level taxonomy, presented in Section 3, that categorizes works by where action is decoded along the inference path and sorts every WAM into exactly one of three families: Render-and-Decode, which carries generation to pixels, Latent-Only, which stops at intermediate representations, and Video-Generation-Free, which is built without the video generator. The second viewpoint, developed in Section 4, is a component-level anatomy. It places each WAM on four axes, the predictive substrate that represents the future available to action, the architectural backbone that produces the prediction, the action coupling that joins prediction to control, and the deployment regime the model is built for. A unified notation, made precise in that section, expresses every WAM as a 4-tuple over these axes. Across both viewpoints we cover the works in our census, listed in Tables 1 and 2, the first holding the video-generation-based families of Render-and-Decode and Latent-Only and the second the Video-Generation-Free family that drops the video backbone. The two viewpoints are complementary, since the philosophy fixes what a model is required to generate while the 4-tuple fixes how it is built.

Organizing the design space is only half of what this survey provides. The other half is a critical account of what these models must do once they are deployed in embodied settings. We examine five properties that embodiment demands, treated in Section 5. A WAM must be interactable, accepting control signals during generation rather than only at the start. It must be causal, never letting the future leak into the action executed now. It must be persistent, holding long-horizon predictions together as the robot acts and replans. It must be physically plausible, predicting futures that the embodiment can actually realize. And it must generalize, keeping the same predictive-action contract useful when tasks, objects, scenes, cameras, and embodiments change. For each property we examine how existing methods meet or miss it, and we track what their solutions cost in compute, memory, data, and the effort to evaluate them.

The remainder of this survey is organized as follows.

- **From World Models to World Action Models (Section 2).** The discussion separates VLAs, broad world models, video generation models, video world models, and WAMs, then defines the action-facing future contract that turns a world model into a WAM.
- **Three Design Philosophies of World Action Models (Section 3).** The philosophy-level taxonomy places every WAM considered here into Render-and-Decode, Latent-Only, or Video-Generation-Free according to what each method is required to generate.
- **What Makes a World Action Model (Section 4).** The formal anatomy introduces the unified notation and the four axes: predictive substrate, architectural backbone, action coupling, and deployment regime. It then records each WAM as a 4-tuple in Tables 1 and 2.

 doi:10.20944/preprints202606.1403.v1

- **Core Properties of World Action Models (Section 5).** The property discussion examines interactability, causality, persistence, physical plausibility, and generalization, together with how existing methods address each requirement.
- **Data and Evaluation (Section 6).** The section organizes the data sources that fuel WAM training, from teleoperation and portable human demonstrations to simulation and internet-scale egocentric video, alongside the evaluation practices the field is settling on.
- **Open Challenges (Section 7).** The final discussion identifies the open problems that the four-axis decomposition brings into view, grouped by where they arise in the WAM stack.

Throughout, one observation persists through the survey. Every design choice in a WAM carries a practical consequence for compute, memory, and latency inside a control loop. The interesting question is rarely how much a WAM costs on its own. It is how that cost trades against the properties embodiment imposes. Viewed over time, the field has moved toward generating less of the future while holding on to what control requires.

2. The Emergence of World Action Models

2.1. Vision-Language-Action Models: Policies from Vision-Language Pretraining

Consider an embodied agent that observes o, receives an optional language instruction l, and must choose an action a. Vision-Language-Action (VLA) models became influential because they showed that pretrained vision-language and language backbones could be grounded into this action channel. In its simplest form, a VLA learns

$$\mathcal{L}_{\mathrm{VLA}}(\theta) = \mathbb{E}_{(o,l,a)}[-\log p_\theta(a \mid o, l)]. \tag{1}$$

RT-2 [1] established the web-to-robot transfer pattern by co-fine-tuning a VLM to output action tokens. OpenVLA [2] made the recipe open and scalable across robot data mixtures, while the π series [3,13,14] moved the action side toward continuous flow matching and broader embodiment coverage. These methods are the closest neighbors of WAMs because they already bind vision, language, and action in one action-producing policy.

The VLA objective also states the boundary. Equation 1 does not require the model to predict what will be observed after the action is executed. A direct VLA can inherit semantic and spatial priors from a pretrained VLM, yet still act from the present observation alone. It becomes a WAM only when a predicted future observation helps produce, choose, or check the action.

2.2. World Models: Predictive Dynamics Beyond Video

A world model asks the complementary question. Rather than predicting the action directly, it predicts what future observation o' should follow from the current observation and an intervention. A recent robot-learning survey frames world models as predictive representations of how environments evolve under actions, with uses in policy learning, planning, simulation, evaluation, and data generation [15]. For the present survey, the minimal form is

$$\mathcal{L}_{\mathrm{WM}}(\theta) = \mathbb{E}_{(o,l,a,o')}[-\log p_\theta(o' \mid o, a, l)], \tag{2}$$

where o' is written as an observation for clarity. The symbol is deliberately broad in this section. Depending on the method, it may be rendered pixels, a hidden feature, a geometric state, an affordance map, an audio cue, a symbolic state, or a token-level description. Section 4 later separates these cases. In latent model-based reinforcement learning, PlaNet [16], DreamerV3 [17], TransDreamer [18], and Dreamer 4 [19] instantiate this idea with compact dynamics states used for imagination and planning. V-JEPA [20] and V-JEPA 2 [21] show the same predictive idea in feature space, while iVideoGPT [22], RoboDreamer [23], EnerVerse [24], and InteractiveWorldSimulator [25] express it through token or video prediction.

 doi:10.20944/preprints202606.1403.v1

Video generation is one visible instance of this broader idea, not its definition. A video generation model learns

$$\mathcal{L}_{\text{VGM}}(\theta) = \mathbb{E}_{(r,o')}\big[-\log p_\theta(o' \mid r)\big], \tag{3}$$

where r is a prompt, image, observation, or other conditioning signal. When the conditioning includes the agent's action and the output is a future visual observation, the same family becomes a video world model:

$$\mathcal{L}_{\text{VWM}}(\theta) = \mathbb{E}_{(o,l,a,o')}\big[-\log p_\theta(o' \mid o,a,l)\big]. \tag{4}$$

Internet-scale video generators such as Wan, CogVideoX, Cosmos, NOVA, Sora, Latte, AnimateDiff, and VideoPoet [26–34] made this branch newly influential because they supplied transferable visual dynamics priors. The rise of WAMs follows from this broader world-model success. Video world models gave the field a high-capacity source of predictive structure, while feature, geometric, audio, symbolic, and token predictors showed that the useful future need not always be rendered video.

2.3. World Action Models: Making the Future Action-Facing

A World Action Model begins when the predicted future observation becomes part of how the action is obtained. The historically early form is a cascade:

$$p_\Theta(o', a \mid o, l) = p_\theta(o' \mid o, l)\, q_\psi(a \mid o, o', l). \tag{5}$$

The action module q_ψ may be an inverse-dynamics model, a pose tracker, a trajectory optimizer, a planner, or a separately trained policy. Presenting this cascaded form first is historically defensible. Early WAMs such as UniPi [4], VLP [5], and AVDC [6] generated or scored future visual trajectories before recovering executable actions with a separate module. Later cascaded WAMs keep the same logic while changing what o' looks like, as in video-to-trajectory, flow-to-policy, trace-to-control, point-track-to-policy, or future-audio-to-policy pipelines [11,35–42]. These works remain WAMs because the predicted future is used to obtain the action, even when the predictor and actor are trained separately.

A second form proposes an action first and then predicts its consequence:

$$p_\Theta(o', a \mid o, l) = q_\psi(a \mid o, l)\, p_\theta(o' \mid o, a, l). \tag{6}$$

This factorization covers candidate-action scoring and planning, where a proposed action is evaluated through its predicted consequence. It is still a WAM when the predicted consequence decides which action is executed [43–47].

The same idea can also live inside one trained model:

$$\mathcal{L}_{\text{WAM}}(\theta) = \mathbb{E}_{(o,l,o',a)}\big[-\log p_\theta(o', a \mid o, l)\big]. \tag{7}$$

Here one backbone or tightly coupled set of experts predicts the future observation and the action together. GR-1 and GR-2 [48,49] helped establish this shared video-action direction with autoregressive image and action tokens, while PAD [50], UWM [51], WorldVLA [52], DreamZero [53], AIM [54], and newer joint designs [55–57] instantiate the same idea through diffusion, autoregression, or hybrid backbones. Feature-tap methods such as Fast-WAM [58] show that the future branch need not be fully rendered at inference as long as the action path still uses a representation shaped by future prediction.

The WAM definition is therefore direct and is summarized in Figure 1. A VLA models $p(a \mid o, l)$ without needing an explicit future. A world model models $p(o' \mid o, a, l)$ or $p(o' \mid o, l)$ without necessarily choosing the action. A WAM links the two: its predicted future observation helps produce the action, score the action, or train the action path inside one model. This definition uses o' as a Section 2 notational simplification. Section 4 instantiates that simplification as different predictive substrates and expands the equations into horizon-level notation. A direct VLA with an auxiliary future loss, a simulator used only as an RL environment, or a future head discarded before action use does not satisfy the definition.

 doi:10.20944/preprints202606.1403.v1

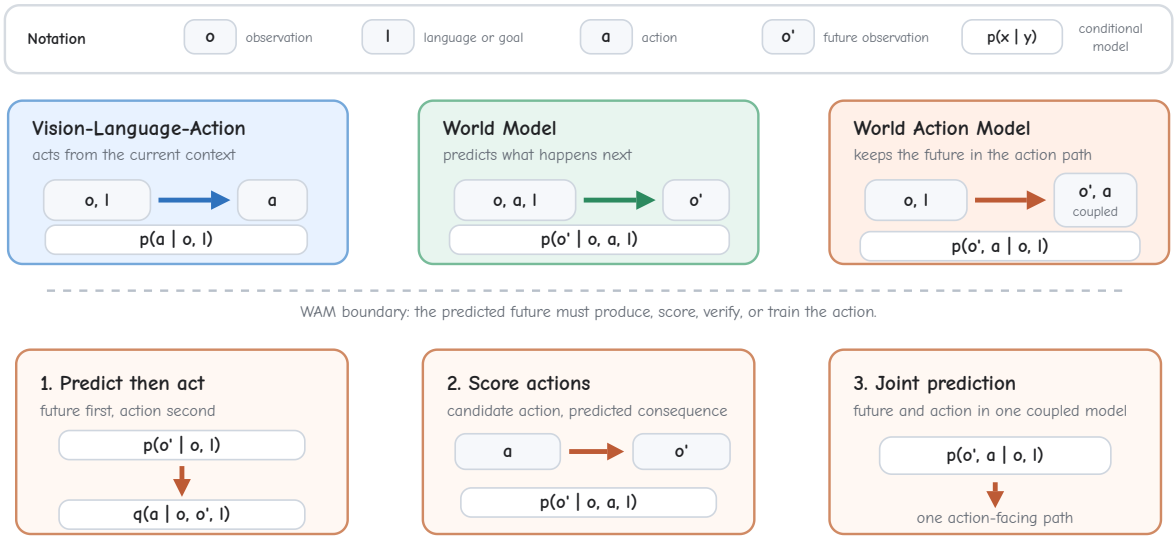

Figure 1. Definition of a World Action Model. A direct VLA predicts action from the present context, and a world model predicts a future observation. A WAM requires that future to stay in the action path, either through predict-then-act cascades, action-scoring rollouts, or joint future-action prediction.

3. Three Design Philosophies of World Action Models

Existing world action models share the WAM definition introduced in Section 2. They differ, however, in where the future predictor meets the action module. Some methods train a shared video-action backbone, while others keep a world model and a separate policy, tracker, inverse-dynamics module, or optimizer in a cascade. The training pipeline alone therefore does not expose the main design split. The decisive distinction is visible either in the inference forward pass or in how action supervision is wired during training: action prediction may pass through a rendered pixel future, stop inside a video backbone before rendering, or avoid the video-generation path altogether.

The resulting taxonomy has three mutually exclusive philosophies. **Render-and-Decode** runs the video-generation backbone all the way to pixel output before action is obtained. The action module may be trained jointly with the generator, trained separately, or supplied by tracking and trajectory optimization. **Latent-Only** keeps the video-world-model lineage but stops the inference path before pixel decoding. In many cases this means a video-generation trunk whose VAE decoder is bypassed, while in others it means a video-trained or video-derived latent predictor whose action-facing future has no decoded-pixel stage. Its action signal comes from intermediate latents, partially denoised features, flow fields, semantic masks, or value maps. **Video-Generation-Free** removes the video-generation backbone from the predictive path. The predictive component instead produces a future representation in the embedding or token space of a large language model, a vision-language model, a joint-embedding-predictive encoder, a deterministic regressor over a frozen vision foundation model, or a non-video diffusion or hybrid backbone over a compact non-pixel substrate. The taxonomy therefore tracks the requirement imposed during inference or action-supervision, rather than the training data, backbone family, or action interface alone.

This split is separable from the cascaded-versus-joint distinction introduced in Section 2.3. Cascaded-versus-joint asks how prediction and action are arranged inside the model. The philosophy-level taxonomy asks where the action prediction is grounded along the inference path: at the rendered pixel output for Render-and-Decode, at an intermediate latent or feature for Latent-Only, or outside the video-generation path for Video-Generation-Free. A Latent-Only WAM can be either cascaded or joint, and the same holds for a Render-and-Decode WAM. Figure 2 sketches the inference data flow under each philosophy.

Figure 3 adds the chronological dimension that the static taxonomy omits. Early WAMs largely adopt the direct path from video generation to control: generate a visual future, then decode action from it. Later Latent-Only methods preserve a video-derived prior but move the action signal earlier

 doi:10.20944/preprints202606.1403.v1

in the inference path, before pixel decoding. Video-Generation-Free methods appear as a more recent alternative, replacing the video generator with predictive supervision in the embedding or token space of LLM, VLM, JEPA, or non-video diffusion or hybrid backbones. Over the same period, the application domain has also broadened. Recent entries now cover tabletop manipulation, dexterous hand control, contact-rich tactile interaction, autonomous driving, and aerial manipulation. They instantiate the same three philosophies under substantially different latency requirements and observation regimes. With this chronological view in place, the remainder of this section explains how each philosophy instantiates this design choice across the WAM census. Section 4 then gives the component-level anatomy behind these choices through substrate, backbone family, action coupling, and deployment regime.

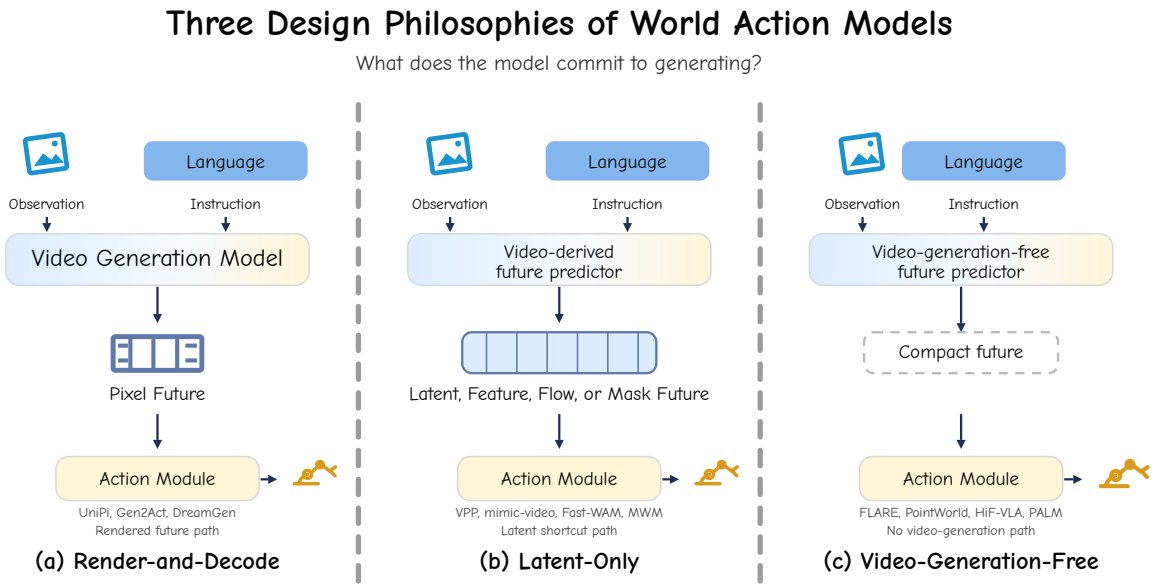

Figure 2. Three design philosophies of World Action Models. The columns separate the last future representation required before action is decoded: a rendered pixel future, an intermediate video-derived latent or feature, or a non-video-generation representation. The categorization is exhaustive over the WAM census and separable from the action-coupling and backbone choices treated in Section 4.

 doi:10.20944/preprints202606.1403.v1

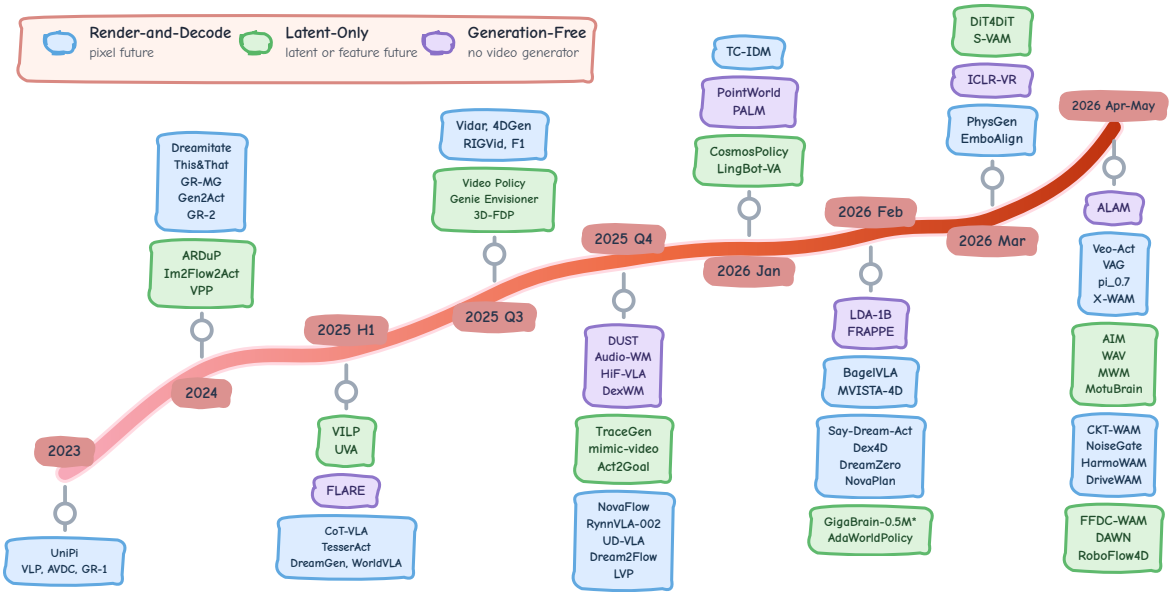

Figure 3. Chronological stream of representative WAMs grouped by design philosophy. The timeline uses coarse bins for 2023, 2024, and 2025 H1, then finer bins for the denser late-2025 and early-2026 period through May. Each date marker attaches a compact vertical stack to the curve, alternating above and below the stream. Dense periods retain early or central verified WAMs so labels remain legible. Stacks above the curve place the widest block nearest the stream, while stacks below the curve begin with the narrowest block. Render-and-Decode appears earliest, Latent-Only emerges as the field starts to drop pixel decoding from the control path, and Video-Generation-Free entries appear as LLM, VLM, JEPA, and non-video diffusion or hybrid backbones begin to carry the predictive component without a video-generation core.

3.1. Render-and-Decode World Action Models

Render-and-Decode is the most direct path from video generation to action. Its premise is that a rendered future is worth producing inside the inference-time control loop because it preserves the full visual prior learned by the video backbone. By running that backbone all the way to pixel output, these methods expose appearance, motion, contact, and scene dynamics to the actor in an inspectable form. The inference pattern is simple: condition a video generator on observation history and instruction, produce a future video, and decode the next action from that rendered future. The output may be raw RGB, a multi-view or 4D RGB-D stream, or a sequence of visual tokens that decode back to images, while the action decoder may be inverse dynamics, 6-DoF tracking, dense correspondences, or a dedicated action head. This premise gives Render-and-Decode its conceptual clarity, but also makes visual synthesis part of the policy's latency budget.

The founding branch treats rendered video itself as the plan. UniPi [4] establishes the template with text-conditioned video followed by inverse dynamics, VLP [5] adds search over language-level plans, and AVDC [6] shows that dense correspondences can replace action labels. The same interface then moves from simulated or task video to demonstrations with different sources of embodiment evidence. Dreamitate [59] tracks generated stereo video, Gen2Act [60] conditions policy learning on generated human motion, This&That [61] adds gesture coordinates, and GR-MG [62] turns a progress-aware goal image into a manipulation condition. Later variants refine how the rendered plan becomes executable. CreFlow [63] post-trains the video-as-policy generator with violation masks from a temporal-logic monitor, while MoLA [64] maps a frozen SVD rollout into semantic, depth, and flow latent-action codebooks before action decoding. The shared lesson is that the rendered future can be a common planning currency even when the recovery module changes.

Modular Render-and-Decode WAMs make that separation explicit. DreamGen [35] uses generated robot futures to obtain pseudo-actions before downstream policy training, so the coupling is offline rather than inference-time imagination. 4DGen [36], RIGVid [37], TC-IDM [65], and GraspDreamer [66] keep the world model and action recovery separate through pose tracking, VLM filtering, tool-centric

 doi:10.20944/preprints202606.1403.v1

point-cloud conversion, or human-grasp retargeting. VERA [67] makes the split even sharper by leaving the video planner action-free and training a Jacobian inverse-dynamics translator. These works are useful because the world model can be reused across embodiments, but the downstream module must absorb the gap between visual plausibility and robot executability.

A related modular pattern inserts geometric or search structure between the generated future and the action. NovaFlow [38] and Dream2Flow [39] distill generated videos into 3D object flow, while EmboAlign [68] filters rollouts with compositional constraints before trajectory optimization. NovaPlan [69] places generated video inside a closed-loop manipulation planner, 3D-ALP [43] uses a 3D world model as a rollout oracle during test-time search, and AeroPlace-Flow [70] grounds a task-complete image into object flow for aerial placement. The pixel future remains the first prediction, but control increasingly depends on the structured object recovered from it.

Joint Render-and-Decode WAMs move action prediction inside the video backbone rather than attaching a separate decoder afterwards. GR-1 [48] is the anchor for this line, because future image tokens and action tokens are learned in one autoregressive stream. GR-2 [49] scales the same video-language-action recipe to a larger pretraining and robot-trajectory mixture. WorldVLA [52] adds masking to reduce action leakage while still predicting future VQ images, RynnVLA-002 [71] refines actions through future image-state prediction, PhysGen [72] adapts a continuous autoregressive video backbone, and DriveWAM [55] interleaves driving video latents with ego-action tokens on a shared DiT. This group changes the coupling pattern, but the action token is still grounded by a pixel-decodable future.

Diffusion and hybrid variants make the same joint target less sequential. PAD [50] jointly denoises future images and actions in one DiT. UD-VLA [73], CoVAR [74], and VideoVLA [75] vary the bridge through discrete diffusion, cross-stream attention, or video-generator co-training. Motus [76] is placed here by its joint-prediction path, where sparse future video and actions are denoised together. Its scheduler can switch to action-only or inverse-dynamics use cases, but those shortcuts do not change the category assignment. These works keep the action and the future in the same generative state, but make the scheduling less rigid than autoregression.

Other joint variants expand what the generated state contains. ActionImages [77] encodes actions as multi-view RGB heatmaps so the observation and action videos share a generated channel grid. DriveVA [78] applies flow matching to driving video latents and action tokens, HarmoWAM [79] sends rendered frames and latent features to different action experts, and NoiseGate [80] learns per-latent denoising schedules around a frozen joint backbone. The family therefore grows through stronger coupling, not through removing the rendering obligation.

Other Render-and-Decode methods change what is rendered so that the future carries more task structure than RGB alone. TesserAct [81] generates RGB-D-Normal 4D video, MVISTA-4D [82] infers arbitrary-view RGB-D through test-time latent optimization, and X-WAM [83] combines multi-view RGB-D prediction with explicit 3D reconstruction. Dex4D [84] lifts generated frames into 3D object point tracks for dexterous control. DriveDreamer-Policy [85] threads depth, video, and action through modular experts, while MV-VDP [86] diffuses future multi-view RGB together with end-effector heatmap videos. These methods keep the visual future, but they make geometry or affordance structure easier for the action side to extract.

A parallel sparse-future line keeps the output pixel-grounded while reducing how much video must be produced. RoboEnvision [87] uses instruction-aligned keyframes as long-horizon plans. CoT-VLA [88] is the influential VLA-side example, because it generates a visual chain-of-thought before action prediction rather than relying on hidden language reasoning alone. pi0.7 [14] follows the same pixel-grounded logic: a BAGEL-class generator refreshes multi-view subgoal images asynchronously, and a PaliGemma action expert consumes the latest available goals. BagelVLA [89] interleaves subtask language, future keyframes, and action chunks, while SWEET [90] feeds sparse edited keyframes to a goal-conditioned diffusion policy. Here the design aim is not to avoid pixels, but to render only the visual milestones that the actor can use.

 doi:10.20944/preprints202606.1403.v1

Foundation-scale Render-and-Decode WAMs focus on whether a large video prior can be made responsive enough for closed-loop control. DreamZero [53] is the anchor: it turns a Wan-class autoregressive video diffusion backbone into an online video-action policy, uses chunked prediction with observation replacement in the KV cache, and shows that a large rendered-future WAM can stay inside a real control loop. Say-Dream-Act [91] follows the same practicality question from the distillation side, compressing Cosmos-style imagination into a few-step generator before action fusion.

Other foundation video planners keep the rendered future as a planning object and differ mainly in how they ground it. Vidar [92], LVP [93], and Veo-Act [94] use masked inverse dynamics, Wan-style image-to-video generation, or a precision VLA executor. VAG [95] and F1 [96] attach action experts to generated or pooled foresight, while EVA [97] aligns the video planner with an inverse-dynamics reward so the rendered future stays executable.

The newest foundation-scale variants spend the video prior more selectively. CKT-WAM [98] transfers context between heterogeneous WAMs, DreamAvoid [44] invokes video imagination only on critical phases, and Pelican-Unify 1.0 [99] links VLM reasoning to a Wan-style future generator. This group keeps the rendered future available when it carries task information, while shifting routine action selection toward cheaper context transfer, critical-phase triggering, or grounded VLM guidance.

The defining limitation of Render-and-Decode is the price of producing pixels. Each prediction step pays for the full denoising or autoregression schedule, even though the actor rarely consumes the rendered output in its entirety. Moreover, visual-quality metrics inherited from video generation only weakly predict downstream task success. This practical tension links the category closely to the open-loop and chunked closed-loop regimes unpacked in Section 4.5.

3.2. Latent-Only World Action Models

The Latent-Only philosophy keeps the video-world-model prior of Render-and-Decode but removes pixel decoding from the inference-time control path. Its motivation is to retain temporal and physical structure learned from video while reducing inference cost. The action signal is decoded directly from a latent, an intermediate denoising feature, a flow field, a semantic mask, a value map, or another video-derived feature future, each of which is cheaper to produce than a fully rendered video. The trade-off is that these methods give up the pixel-space supervision and visual-quality metrics that make Render-and-Decode easy to inspect. The designs differ mainly in where they intercept the predictive path: some consume the model's own video latent, some tap an intermediate denoising state, some predict a structured substrate such as object flow or masks, and some inherit a joint video-action backbone but disable pixel decoding at inference. In all cases, the key move is to keep a video-shaped dynamic prior while refusing to pay for the final renderer during action selection.

The earliest Latent-Only entries make the interception point explicit. ARDuP [100] focuses latent video diffusion on active regions, VPP [7] conditions inverse dynamics on Stable Video Diffusion representations, and VILP [101] recovers actions from time-aligned multi-view latent video. UVA [102] and Video Policy [103] show the two main training patterns: either share a video-action latent across heads, or freeze the video generator and train the policy on its intermediate features. Genie Envisioner [104] is the clearest bridge between these patterns. GE-Base remains an instruction-conditioned multi-view video DiT, while GE-Act decodes action chunks from a one-step denoised latent cache through flow matching. Act2Goal [105] reuses the same foundation for goal-conditioned transition features, and WAV [106] performs value-guided latent trajectory inference before action decoding. These methods show that the first shortcut is to stop at the representation that action already needs.

As video diffusion backbones scale, Latent-Only designs increasingly intercept the denoising trajectory rather than a finished latent. mimic-video [8] attaches an action decoder to an intermediate ODE checkpoint of Cosmos-Predict2, and DiT4DiT [107] conditions its action expert on intermediate Cosmos-Predict2.5 features in a deterministic step. S-VAM [9] self-distills multi-step denoising into one pass over geometric and semantic features. Fast-WAM [58] and GigaWorld-Policy [108] keep future-video co-training but make full generation optional or unnecessary at inference. LaMP [109] exposes a one-step 3D motion hidden state to a VLA through gated cross-attention, and VAMPO [110]

 doi:10.20944/preprints202606.1403.v1

tunes early-step VPM latents against expert visual dynamics. The practical shift is from pixel synthesis to feature extraction.

Another branch predicts structured motion before action decoding. Im2Flow2Act [111] and 3DFlowAction [112] replace appearance with object flow, while 3D-FDP [113] predicts dense 3D flow over query points before a second action diffusion model runs. TraceGen [40] predicts 3D traces that inverse kinematics converts into joint commands. This flow-and-trace group compresses the future toward motion rather than texture.

The structured-substrate branch then broadens beyond flow. 3PoinTr [41] and RoboFlow4D [42] freeze future point-track or 4D-flow predictors and train downstream policies to consume the predicted motion. MWM [114] keeps a video-diffusion architecture but predicts semantic-mask evolution, OmniVTA [115] adds tactile latents for contact-rich control, and EgoExo-WM [45] uses a DINOv3 latent regressor under model-predictive control after exo-to-ego data conversion. This branch makes the future closer to contact and task state than a rendered frame would be.

The clearest Latent-Only inference pattern appears when a joint video-action backbone is trained with visual prediction but used through a latent action path. UWM [51] is the reference case: independent diffusion timesteps let the method collapse the visual branch at inference while retaining the video-trained representation. CosmosPolicy [116], AdaWorldPolicy [117], and MotuBrain [118] follow the pixel-latent version of the same idea, where future RGB or video latents remain useful during training but the action call at inference does not require a fully rendered RGB forecast. LingBot-VA [119] exposes semantic structures before pixel reconstruction, and AIM [54] decodes action through a spatial value map rather than directly through future RGB. The design consequence is clear: visual prediction shapes the representation, but action selection does not have to render the visual prediction.

Recent Latent-Only WAMs extend the same idea across modalities and memory designs. GigaBrain-0.5M* [120] feeds the policy with future-state latents and value embeddings from a Wan2.2 world model. VTAM [121] treats camera views and GelSight tactile input as decoder-bound latent views, while CLWM [122] forecasts future DINOv3 feature maps and replaces a growing KV cache with Dual-State Test-Time Training memory.

The same inference logic also travels across domains and joint-state designs. DAWN [123] iteratively refines V-JEPA latent world tokens against DiT action hypotheses, JOPAT [124] adds point tracks and visibility to the joint denoising state, and tau0-WM [57] keeps Wan-style video latents in the joint state but decodes action without routine pixel rendering. WALL-WM [56] pushes the same idea into depth-wise cross-attention between matched video blocks and action queries. These examples make Latent-Only a design philosophy rather than a single substrate choice.

A newer branch uses latent imagination as an adaptive control signal. FFDC-WAM [125] caches Motus latent video tokens with the proposed action chunk and uses a causal attention verifier to decide when a full WAM replan is needed. tau0-WM [57] can also reactivate its action-conditioned video simulator only when candidate scoring is useful. This turns imagination into a learned controller decision: the policy invokes the heavier future model only when confidence drops.

Across this family, the central tension is clear. Video pretraining supplies a valuable dynamic prior, but full pixel rendering imposes latency and memory costs that are difficult to justify for every control step. Latent-Only methods preserve the prior while bypassing the renderer, enabling several members of the family to reach real-time inference at the cost of reduced direct visual interpretability.

3.3. Video-Generation-Free World Action Models

The Video-Generation-Free philosophy removes the video-generation backbone altogether. Here *video-generation-free* refers strictly to the absence of a pixel-level video backbone in the predictive path. The LLM, VLM, JEPA, and flow-matching components below remain generative models in the broader sense. Its motivation is to avoid the training and inference cost of video generation while retaining predictive supervision in a more compact representational space. These methods instead leverage large language models, vision-language models, joint-embedding-predictive encoders, deterministic regressors over frozen vision foundation models, or non-video diffusion and hybrid backbones over a

 doi:10.20944/preprints202606.1403.v1

compact non-pixel substrate. The dominant pattern attaches a lightweight action expert to a VLM, learns a latent-action vocabulary that bridges actionless video and robot control, or supervises a policy with an auxiliary feature-prediction loss in a vision-foundation embedding space. The family is listed in Table 2, where it complements the Render-and-Decode and Latent-Only entries in Table 1.

The feature-prediction line replaces pixel prediction with teacher-target or foundation-feature prediction. FLARE [126] is the anchor: additional policy tokens are aligned to future-observation embeddings from a frozen teacher, and the action expert consumes those predicted future tokens at inference. FRAPPE [127] extends the idea by aligning a generalist policy to future representations from multiple vision foundation models, while DexWM [128] applies deterministic future-feature regression to dexterous hand-object interaction. LDA-1B [12] moves the same principle into a Qwen3-VL stack by jointly learning dynamics, policy, and forecasting in a structured DINO latent. These works keep the predictive training signal, but place it in an embedding space whose quality is judged through action rather than photorealism.

Another Video-Generation-Free line treats the future as a learned token or latent transition. DUST [10] augments an Eagle-2 VLM with a dual-stream diffusion head that predicts future observation embedding tokens and an action together without a video-generation foundation. ALAM [129] learns latent transitions from action-free video and regularizes them with composition and reversal consistency before a flow-matching action expert consumes the codes. The shared move is to keep a future object in the inference path while choosing a representation that is cheaper than video.

Structured non-pixel futures give the family its second anchor. PointWorld [46] predicts action-conditioned 3D point flow through a deterministic Point Transformer and plans with MPPI over the compact future. PALM [130] structures foresight as affordance maps plus progress, while HiF-VLA [131] uses compact motion-vector foresight tokens and ICLR-VR [132] predicts a gripper-keypoint polyline before each action chunk. These works replace video with geometry, affordance, or trace-like variables that are closer to the control interface.

The remaining generation-free variants broaden the same principle across modalities and feedback. Audio-WM [11] predicts future Mel latents that condition a robot policy when contact sound carries task evidence. DDP [133] switches to latent imagination when a real-imagination gap indicates an out-of-distribution event, and Feedback-WM [47] treats a learned latent transition model as an observer that corrects diffusion-policy denoising online. Together, these methods show what is gained by dropping the video generator entirely, and what must be replaced by tokens, features, audio latents, affordances, or geometry.

3.4. Where the Three Philosophies Meet the Formal Anatomy

The three philosophies answer the question of *where* action prediction sits along the video-generation inference path: at the rendered pixel output for Render-and-Decode, at an intermediate latent or feature for Latent-Only, or outside the video-generation path for Video-Generation-Free. They do not yet answer *how* the model produces that prediction, where action is injected, or when the model is invoked relative to the control loop. Tables 1 and 2 place every WAM in the same census under a four-axis design space that captures these component-level choices. Section 4 introduces the formal notation behind that four-axis view and uses it to compare WAMs across substrate, backbone family, action coupling, and deployment regime. Together, the philosophy-level taxonomy of this section and the component-level anatomy of the next provide two complementary viewpoints for navigating the field.

4. What Makes a World Action Model

The term *world action model* now spans works built in different communities, with different vocabularies, and on top of different backbones. A paper-by-paper enumeration is insufficient for comparing them. Methods that share the WAM label can differ in nearly every implementation detail, whereas closely related methods may use entirely different terminology. This section takes a different route. It treats WAMs as instances of a common mathematical object: a conditional joint

 doi:10.20944/preprints202606.1403.v1

distribution over future predictions and future actions. Under this view, the main design choices specify what future is represented, how actions are coupled to that future, what model family produces the prediction, and how the resulting model is invoked during control. We refer to these choices as the *predictive substrate*, *action coupling*, *architectural backbone*, and *deployment regime*, respectively. Figure 4 previews how the four ingredients combine. Section 4.1 introduces the shared notation. Sections 4.2 through 4.5 then walk through the four ingredients one by one. They sort existing methods along the choices each ingredient offers and note the practical trade-off that each choice carries.

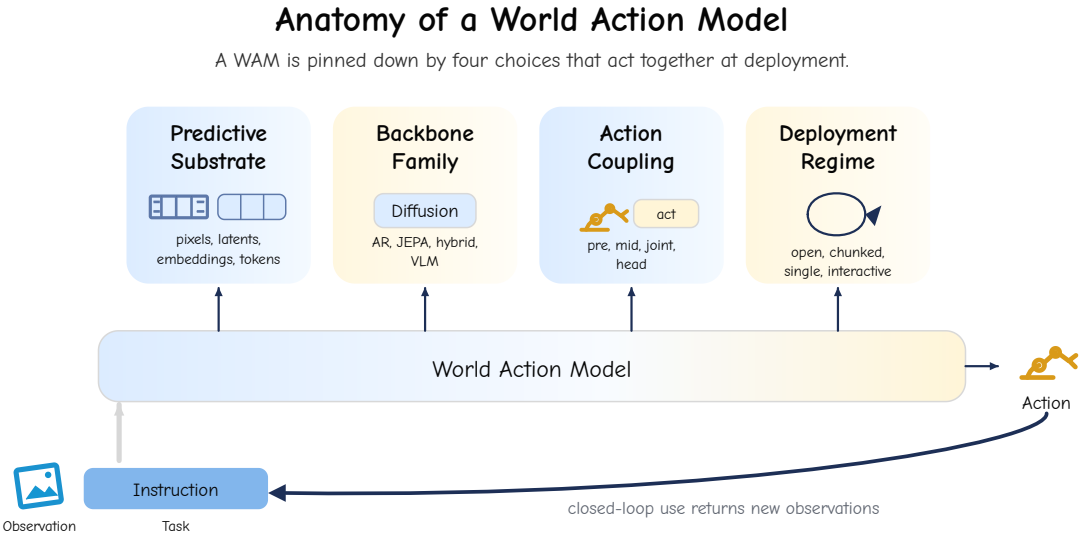

Figure 4. The anatomy of a world action model. Any existing WAM can be specified by four separable but interacting choices: what it predicts (Section 4.2), how action is coupled to that prediction (Section 4.3), the function family that produces the prediction (Section 4.4), and the deployment regime it is intended for (Section 4.5). Two methods with the same labels but different choices on any one axis behave very differently in practice.

4.1. A Unified Notation for World Action Models

At the right level of generality, WAMs can be described by a common abstract object. The object is a parameterized conditional joint distribution over a window of future predictions and a window of future actions:

$$p_\Theta(\, s_{t+1:t+H},\, a_{t:t+H-1} \mid o_{\le t},\, a_{<t},\, l\,). \tag{8}$$

Here $o_{\le t} = (o_0, \ldots, o_t)$ is the observation history, with each $o_i \in \mathcal{O}$ an observation, such as a frame, video chunk, or multimodal sensory packet, $a_{<t} = (a_0, \ldots, a_{t-1})$ is the past action history with $a_i \in \mathcal{A}$, and l is a task instruction in language, a goal image, or another auxiliary modality. The model predicts a future trajectory $s_{t+1:t+H}$ of length H in a chosen *substrate space* $\mathcal{S}$ together with a future action chunk $a_{t:t+H-1}$ of the same length. The parameter collection Θ may be one shared backbone or a retained world-model module paired with an action module. To keep later equations compact, we abbreviate the full conditioning context as

$$c \;\equiv\; (\, o_{\le t},\, a_{<t},\, l\,), \tag{9}$$

and write Equation 8 as $p_\Theta(s_{t+1:t+H},\, a_{t:t+H-1} \mid c)$ whenever no ambiguity arises.

The substrate space $\mathcal{S}$ is the space of the future representation that the WAM exposes to action production or action evaluation. We write it as the image of an interface encoder $\phi : \mathcal{O} \to \mathcal{S}$, but ϕ should not be taken as the generator's internal implementation. A pixel-grounded WAM exposes decoded future frames, or the VAE / VQ latent grid that the generator decodes to a clip, even if the action side never invokes the pixel decode at inference. A feature WAM exposes a learned hidden state or token block with no decodable visual interpretation, whether it is a frozen self-supervised teacher's target, a feature tap of a generative trunk, or a VLM future-token block. A geometric-primitive WAM

 doi:10.20944/preprints202606.1403.v1

exposes a structured object whose dimensions are physical coordinates, such as flow, point clouds, depth, pose, or motion vectors. An affordance-map WAM exposes a task-relevant label or score map, such as a value, affordance, or end-effector heatmap. The symbol p_Θ in Equation 8 is a placeholder for an abstract joint, not necessarily one monolithic network. Section 4.4 lists the five concrete families that realize its future-prediction factor in practice, each with its own parameterization. Until that point, p_Θ should be treated as the action-facing future contract that every WAM must satisfy.

The four ingredients of this section are choices about how Equation 8 is realized in practice.

(i) Predictive substrate.

A choice of action-facing future representation, and therefore of the space $\mathcal{S}$ in which $s_{t+1:t+H}$ lives. Section 4.2.

(ii) Action coupling.

A choice of factorization of the joint in Equation 8. Three top-level factorization families appear across existing WAMs and return in Section 4.3:

$$\text{action-conditioned rollout} : \begin{cases} q_\psi(a_{t:t+H-1} \mid c)\, p_\theta(s_{t+1:t+H} \mid c,\, a_{t:t+H-1}), & \text{chunk-level,} \\ \prod_{k=0}^{H-1} q_\psi(a_{t+k} \mid h_k, c)\, p_\theta(s_{t+k+1} \mid h_k,\, a_{t+k},\, c), & \text{step-wise,} \end{cases} \tag{10}$$

$$\text{joint generation} : p_\theta(s_{t+1:t+H},\, a_{t:t+H-1} \mid c), \tag{11}$$

$$\text{post-prediction head} : p_\theta(s_{t+1:t+H} \mid c)\, q_\psi(a_{t:t+H-1} \mid s_{t+1:t+H},\, c). \tag{12}$$

where $h_k = (s_{t+1:t+k}, a_{t:t+k-1})$ in the step-wise submode. Action-conditioned rollout composes an action source with a world predictor, either by fixing an action chunk before prediction or by updating the action after each predicted step. Joint generation produces the future substrate and the future action chunk from one coupled generative model $p_\theta(s_{t+1:t+H}, a_{t:t+H-1} \mid c)$, where θ may include a shared trunk as well as modality-specific heads or experts The post-prediction head factorization predicts the substrate first and then decodes action from it with a smaller actor q_ψ, which is often a tracker, inverse-dynamics model, optimizer, planner, or policy that can be trained separately from θ. All three families carry the past action history through the abbreviation c, so the conditioning sides remain consistent with Equation 8.

(iii) Architectural backbone.

A choice of function family that realizes the future-prediction factor of p_Θ, namely an iterative denoising network, an autoregressive decoder, a joint-embedding predictor, a hybrid that combines two of the above, or a large language or vision-language model with an attached action decoder. Each family carries its own parameterization, which we introduce only when the family is treated. Section 4.4.

(iv) Deployment regime.

A choice of when the WAM is invoked and over what window. Open-loop rollout invokes the WAM once with $H \approx T$ and specifies the whole window before execution. Chunked closed-loop invokes the WAM every K control steps with $H = K$ and replans after each chunk is executed. Single-step closed-loop invokes the WAM at every control step with $H = 1$. Interactive operation typically reuses a trained model θ and extends H indefinitely while carrying state across calls through a key-value cache or a persistent latent. Section 4.5.

This decomposition is useful not because the four axes are separable, but because they separate different questions about a WAM. The substrate asks what kind of future variable s is represented, the backbone asks how that variable is predicted, the coupling asks how actions enter the prediction or are recovered from it, and the deployment regime asks how the resulting model is invoked inside a control loop. In practice, the substrate and backbone are usually fixed by training, while deployment

 doi:10.20944/preprints202606.1403.v1

and some wrapper-level coupling choices can be adjusted at inference. Changing the coupling family itself, however, often requires an additional action head, planner, or fine-tuning stage. We use these four questions as the organizing lens for the subsections below.

Tables 1 and 2 place every WAM in our census as a 4-tuple over these choices, and the subsections that follow walk through the choices one by one and anchor each one in the same equation.

4.2. Predictive Substrate: Where WAMs Dream

The first choice in Equation 8 is where the future variable $s_{t+1:t+H}$ lives. We call this space the predictive substrate. The substrate is not determined by the backbone alone, nor by the last tensor consumed by the action head. For example, a video-diffusion model may denoise a VAE grid and skip pixel decoding during control. As long as a fixed decoder maps that grid back to video, the predicted future remains pixel-grounded. Conversely, a policy may consume an intermediate state from a video-trained trunk, but that state is a feature substrate if it has no fixed observation decoder. In this section, we therefore classify a WAM by the representation in which it forms the future used for action prediction or evaluation.

We use four substrate categories. Pixel-grounded substrates represent future observations, either as decoded RGB, RGB-D, multi-view video, or as VAE / VQ latents with a fixed video decoder. Acoustic observation latents are rare in the current literature, so we treat them as multimodal observation-latent edge cases rather than as a separate category. Feature substrates represent learned states, teacher embeddings, or VLM token blocks without a fixed observation decoder. Geometric substrates represent physical structure or motion, such as optical flow, point tracks, depth, pose, motion vectors, or polylines. Affordance substrates represent task-specific maps or score fields, such as value maps, contact maps, semantic masks, or heatmaps. Some WAMs predict more than one kind of future in the same forward process. The tables mark these joint cases with $\wedge$.

4.2.1. Pixel-Grounded Substrates

The pixel-grounded category is the direct descendant of video generation. Its decoded form is an observation tensor,

$$s_{t+1:t+H} \in \mathbb{R}^{H \times C_o \times H_{px} \times W_{px}}, \tag{13}$$

where C_o is the observation channel count. A decoder-bound latent variant uses the same category when s is a VAE, VQ, or 3D-VAE code whose fixed decoder maps it back to video. The model may act before that decoder is called, but the future still lives in an observation coordinate frame. This criterion separates decoder-bound observation latents from feature latents below.

Decoded observation futures. Early WAMs take the direct route: generate a future image or video, then decode action from it. UniPi [4] establishes this pattern with text-conditioned video plus inverse dynamics, while AVDC [6] shows that dense correspondences from the decoded video can replace action labels. The same substrate category covers cascaded designs when the action module is separate. DreamGen [35] uses decoded robot futures to obtain pseudo-actions before policy training, RIGVid [37] retargets tracked generated demonstrations, and VERA [67] lets a Jacobian inverse-dynamics model act from a generated visual lookahead. Later decoded-observation methods keep the same substrate but broaden the content of the forecast. Vidar [92], LVP [93], and EVA [97] use stronger Wan-class planners, while Veo-Act [94] uses a precision VLA executor on top of a generated visual plan. These wrappers still belong to the pixel-grounded category when geometry is recovered after the visual future has been produced.

Pixel-decodable latent futures. The second route keeps the future in the pixel-decodable latent space of a video generator. GR-1 [48] and GR-2 [49] predict VQ image tokens and action tokens in one autoregressive stream. PAD [50], UWM [51], WorldVLA [52], VideoVLA [75], DriveVA [78], and τ_0-WM [57] instead couple action to a diffusion or flow-matching trajectory over VAE video latents. Hybrid entries such as F1 [96], Motus [76], Pelican-Unify 1.0 [99], NoiseGate [80], and WALL-WM [56] keep the same substrate while changing the routing between video and action streams. CKT-WAM [98]

 doi:10.20944/preprints202606.1403.v1

and FFDC-WAM [125] add a teacher context or verifier around that substrate, but the action-facing future remains the student or Motus pixel-decodable visual forecast. These works are not feature-substrate WAMs under the present criterion because the future is still a compressed observation under a fixed observation decoder.

Sparse, multimodal, and joint observation futures. The pixel-grounded category also covers reduced, multi-channel, and rare non-visual observation displays. SWEET [90] shrinks the forecast to task-critical keyframes. Audio-WM [11] predicts future Mel latents that condition a robot policy, and it is grouped here as an acoustic observation-latent edge case because it follows the same decoder-bound criterion without changing the category name. DriveDreamer-Policy [85] predicts decoded video together with depth, while MV-VDP [86] and ActionImages [77] predict RGB together with Gaussian end-effector heatmaps. OmniVTA [115] is also an observation-future case rather than a geometric one, because its future is a pair of visual and tactile decoder-bound latents with fixed decoders, not a coordinate tensor. HarmoWAM [79] is a dual observation case because its Wan backbone produces both rendered frames and pixel-decodable latent features that feed different action experts. The common trade-off is clear: observation futures preserve appearance, contact traces, and broad video priors, but they pay for detail that many action decisions do not need.

4.2.2. Feature Substrates

The feature category covers futures that are learned states rather than decoder-bound observation states. A single sample can be written as

$$s_{t+1:t+H} \in \mathbb{R}^{H \times N_{\text{tok}} \times d_{\text{emb}}}, \tag{14}$$

with N_{tok} tokens per step and embedding dimension d_{emb}. The key property is absence of a fixed observation decoder. The state may be trained through future prediction, but it is interpreted only by the action pathway or by an embedding loss.

Encoder-only and feature-tap futures. This variant keeps the future-trained backbone but routes action through an internal state. Fast-WAM [58] is the clean encoder-only case: it keeps video co-training but masks the test-time future branch, so action is predicted from a hidden state shaped by future prediction. VPP [7], Genie Envisioner [104], VidMan [134], mimic-video [8], VAG [95], and WAV [106] keep more of the generative trajectory but decode action from latent or intermediate denoising states. LaMP [109] is the VLM-stack version of the same shortcut, exposing a motion-expert hidden state through gated cross-attention before the action expert fires. The boundary is that an auxiliary hidden state is not enough to make the substrate a feature future when the inference-time action-facing future is still a pixel-decodable visual forecast.

Teacher-target futures. A second feature route predicts the embedding of a frozen vision encoder, a multimodal encoder, or a policy/expert latent state. FLARE [126] aligns policy tokens to future-observation embeddings, and LDA-1B [12] learns dynamics, policy, and forecasting in a structured DINO feature space on a Qwen3-VL backbone. DAWN [123], CLWM [122], EgoExo-WM [45], DexWM [128], DDP [133], Feedback-WM [47], and FRAPPE [127] all keep the future in a learned feature space rather than in a decoder-bound observation latent. The benefit is compact prediction and strong semantic invariance. The cost is that the substrate has no direct visual-fidelity metric.

VLM-token futures. The third feature route represents the future as tokens in a VLM vocabulary. CoT-VLA [88] predicts visual chain-of-thought tokens before action generation. DUST [10] jointly diffuses a next-state observation token and an action token, and ALAM [129] interleaves view-specific latent-transition tokens with action tokens under a structured attention mask. These methods use tokens as future carriers, not merely as action discretisation. The feature category is therefore broad, but its practical boundary is whether the predicted future lacks a fixed observation decoder and is not a physical coordinate tensor or task map.

 doi:10.20944/preprints202606.1403.v1

4.2.3. Geometric Primitive Substrates

The third category exposes a structured object whose dimensions are physical coordinates. A common instance is dense optical flow, for which ϕ is an off-the-shelf flow operator and a single sample is

$$s_{t+1:t+H} \in \mathbb{R}^{H\times 2\times H_{\text{px}}\times W_{\text{px}}}, \tag{15}$$

namely two channels carrying the (u, v) displacement field. Variants replace the flow field with N_{kp} tracked keypoints in $\mathbb{R}^{H\times N_{\text{kp}}\times 3}$, a depth tensor, a 6-DoF pose stream, a sparse waypoint polyline, or a macroblock motion-vector grid. A method belongs here when geometry is the predicted future representation, not when a tracker merely measures geometry after a pixel-grounded future has already been chosen.

Flow and point-motion futures. Im2Flow2Act [111] and 3DFlowAction [112] motivate the category by replacing full appearance with object motion. NovaFlow [38] and Dream2Flow [39] take the cascaded route by extracting 3D object flow from generated videos before planning. TraceGen [40] predicts 3D traces, while 3PoinTr [41] and RoboFlow4D [42] feed separately predicted point tracks or flow into downstream policies. 3D-FDP [113], PointWorld [46], and Dex4D [84] extend this idea from dense image flow to sparse or scene-level 3D point trajectories. The common payoff is a future that is closer to control than RGB.

Structured physical futures. TesserAct [81] reconstructs a 4D point cloud as the control-facing representation. HiF-VLA [131] uses MPEG-4 motion vectors as a compact future-motion field, and ICLR-VR [132] reduces the future to a five-point gripper polyline. JOPAT [124] is one prominent joint case because it denoises a pixel-decodable visual future together with point tracks and visibility. Geometric futures are cheaper and often easier to supervise than pixel-grounded futures, but they lose appearance cues that are irrelevant to motion yet useful for semantic grounding.

The structured tensor is far smaller than the pixel tensor at the same spatial resolution, and it can often be trained stably on less data. The trade-off is that these predictions are most reliable when motion, contact, or geometry carries the task-relevant information. They are less suitable when appearance or semantics is essential.

4.2.4. Affordance Map Substrates

The fourth category exposes a task-relevant label or score map. A single sample is

$$s_{t+1:t+H} \in \mathbb{R}^{H\times C\times H_{\text{px}}\times W_{\text{px}}}, \tag{16}$$

with C channels carrying value, affordance, contact likelihood, segmentation, or progress information. The substrate is not an observation, a feature, or a physical coordinate field. It is a spatial answer to a task-specific question.

Value, mask, and affordance futures. AIM [54] inserts a spatial value map between future visual dynamics and action, with intent-causal attention forcing action to pass through that value representation. PALM [130] predicts a bundle of future affordance maps, including object masks, contact likelihood, placement candidates, and motion regions. MWM [114] predicts semantic-mask dynamics rather than RGB, so its future is a label map rather than a generic hidden state. ActionImages [77] and MV-VDP [86] are joint pixel-grounded-and-affordance cases because they predict RGB together with Gaussian end-effector heatmaps. Affordance futures are therefore more task-specific than observation or feature futures. They are compact and often directly useful for manipulation, but their labels must be defined for the task at hand.

4.2.5. Substrate and Coupling Are Distinct Axes

The four categories classify the space of the future variable s, while Section 4.3 classifies how s and a are tied together. The distinction matters. Fast-WAM [58] is feature-substrate and encoder-only because its future branch is masked at inference and the action expert uses a hidden state. τ_0-WM [57]

 doi:10.20944/preprints202606.1403.v1

is pixel-grounded-substrate and joint-denoising because it predicts a Wan VAE video latent together with action velocity. mimic-video [8] is feature-substrate and post-prediction-head because the action head taps an intermediate ODE state of a frozen video trunk. The substrate names where the WAM forms its future. The coupling axis names how that future reaches action. Figure 5 arranges the categories and representative members on the same panel.

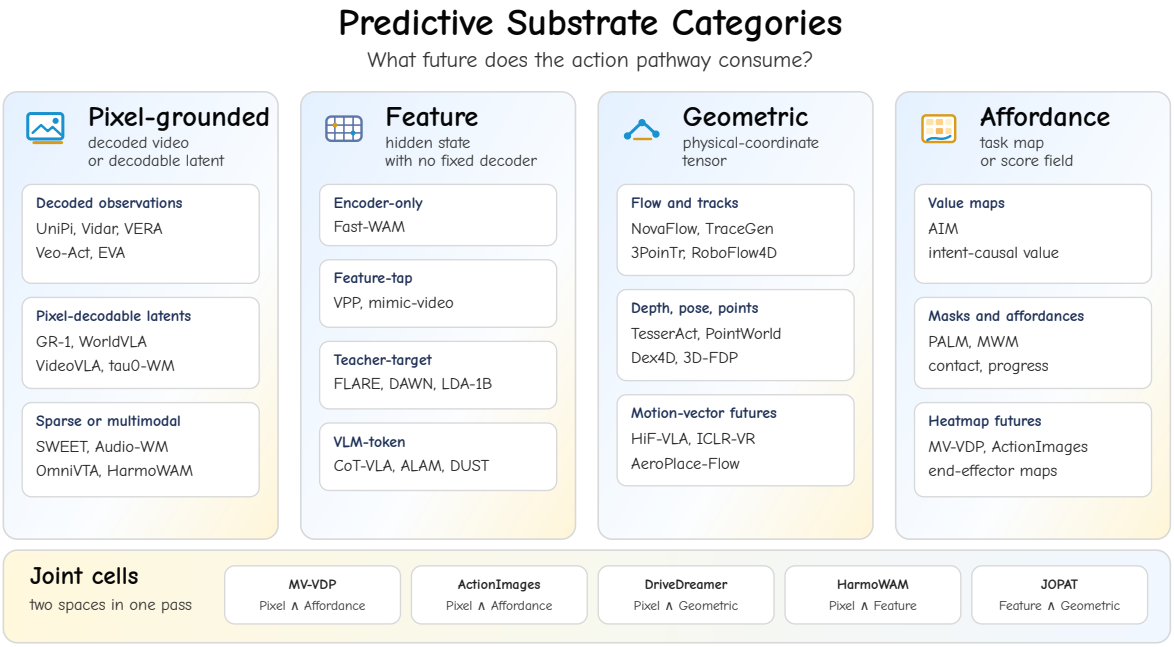

Figure 5. The four predictive-substrate categories of Section 4.2, classified by the representational space in which the WAM forms its future. Pixel-grounded substrates include decoded observations and pixel-decodable latents, with Audio-WM treated as a rare acoustic observation-latent edge case (4,11,48,52,57). Feature substrates include encoder-only, feature-tap, teacher-target, and VLM-token futures with no fixed observation decoder (7,58,123,126,129). Geometric substrates expose flow, point clouds, depth, polylines, or motion vectors (46,81,131). Affordance substrates expose value maps, masks, affordances, or heatmaps (54,114,130). Joint cells use the $\wedge$ separator when one forward pass predicts futures in two categories.

4.3. Action Coupling: How Action Enters and Leaves

Action coupling is the structural decision that turns a predictive world model into a WAM. The factorization families in Equations 10, 11, and 12 pin down the main arrangements. Action-conditioned rollout is a composite family: an outer planner, policy, candidate sampler, or teleoperation stream supplies actions through q_ψ, and an action-conditioned world model predicts their consequences. Joint generation and post-prediction heads are action-producing families, either because the future action chunk is produced by the same joint model or because it is decoded from the produced future by a separate head. Within each arrangement, existing methods further differ on how an action is represented and on how many actions are produced per forward pass. Figure 6 sketches the three top-level arrangements and shows representative candidate forms inside the first. Notely, in this section, when we refer to a method as non-census, we mean that it is cited for background but is not listed as a WAM in Tables 1 and 2.

 doi:10.20944/preprints202606.1403.v1

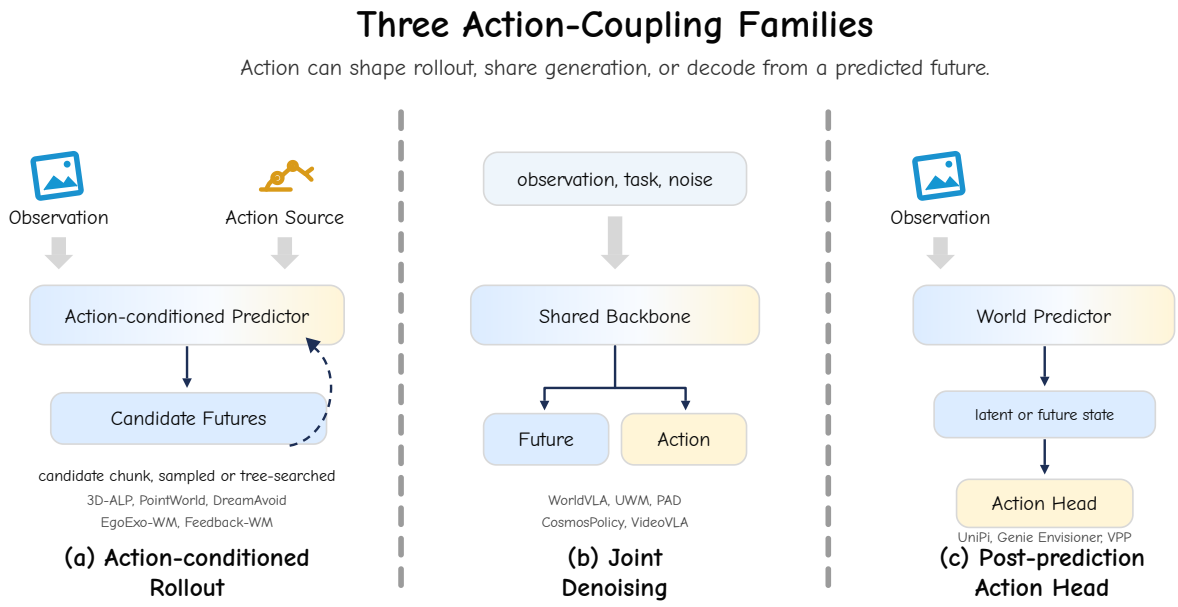

Figure 6. Three common action-coupling families in existing WAMs. The arrangements differ in where the action representation is bound to the substrate prediction, and that binding is what determines latency, controllability, and per-step inference cost.

4.3.1. Action-Conditioned Rollout

Action-conditioned rollout composes an action source with an action-conditioned world predictor. Repeating Equation 10,

$$p_{\theta,\psi}(s_{t+1:t+H}, a_{t:t+H-1} \mid c) = \begin{cases} q_\psi(a_{t:t+H-1} \mid c)\, p_\theta(s_{t+1:t+H} \mid c,\, a_{t:t+H-1}), & \text{chunk-level,} \\ \prod_{k=0}^{H-1} q_\psi(a_{t+k} \mid h_k, c)\, p_\theta(s_{t+k+1} \mid h_k,\, a_{t+k},\, c), & \text{step-wise,} \end{cases} \tag{17}$$
$$h_k = (s_{t+1:t+k}, a_{t:t+k-1}).$$

The first line is chunk-level rollout. The future action chunk is output by q_ψ and enters the substrate predictor as conditioning. The second line is step-wise rollout. It alternates action selection with action-conditioned prediction, so q_ψ can update the next action after each predicted state. Training usually applies the standard substrate-prediction loss of Section 4.4 to p_θ, with the proposed action or action chunk added to the conditioning side. The action source may be a planner, a policy, a human command stream, a fixed candidate sampler, or, in the step-wise background lineage, a diffusion policy queried after each predicted state.

In the current census, the WAMs listed in the tables in this family all sit on the chunk-level submode. The step-wise submode is retained to connect the WAM notation to the model-based RL and interactive-prediction lineage discussed below.

The chunk-level submode is the natural form when the method scores candidate futures before execution. Cosmos-Transfer1 [135] illustrates a nearby non-census case: segmentation, depth, edge, and other spatial control maps condition world generation for data generation, but the method does not use generated futures as an inference-time robot-action selector. PointWorld [46] conditions a point-flow dynamics model on a proposed sequence of robot point flows from MPPI and scores the predicted scene flow, so action search happens outside the predictor rather than inside the trunk. DreamAvoid [44] samples flow-policy action chunks only at detected critical phases, renders each chunk with DreamDojo, and executes the candidate with the highest predicted progress value. EgoExo-WM [45] conditions a DINOv3-L latent predictor on horizon-length UniEgoMotion candidate body-motion sequences, rolls each candidate forward under MPC, and selects the sequence whose predicted latent ends closest to a visual goal.

 doi:10.20944/preprints202606.1403.v1

Tree-search and feedback-guided variants keep the same family but change the cadence of candidate evaluation. 3D-ALP [43] uses MCTS to propose joint actions, converts each proposal into a 3D camera pose through kinematics, and uses a 3D-consistent renderer as a rollout oracle before executing the selected action. Feedback-WM [47] uses a lightweight latent transition model to guide a diffusion policy while that policy forms an action chunk, passes the corrected chunk to the retained world-prediction path, and then updates an auxiliary feedback state from the observation after execution. These variants explain why the family should not be split by whether actions arrive as one candidate chunk, a tree node, or a corrected action sequence. In all cases, actions are supplied to a predictor that evaluates their consequences.

The step-wise submode remains useful for background and for future WAMs whose action source reacts inside the imagined trajectory. AdaWorld [136] conditions an SVD-style latent-video diffusion predictor at each rollout step, although it is not part of the census. FlowDreamer [137] cascades a geometric-primitive flow prediction into a diffusion next-frame generator, a two-stage variant of the same arrangement. The latent-imagination Dreamer line of PlaNet [16], DreamerV3 [17], Dreamer 4 [19], and TransDreamer [18] supplies the model-based RL ancestor of this design, where a planner, actor, critic, or compact dynamics model advances imagined steps depending on the member. The non-WAM antecedent iVideoGPT [22] interleaves observation, action, and reward tokens in an autoregressive stream, while InteractiveWorldSimulator [25] and RoboScape [138] extend interactive prediction with streaming inputs or physics-aware branches.

Action-conditioned rollout is therefore a composite WAM factorization rather than an action-producing world model by itself. Its advantage is counterfactual control: candidate actions shape the future before selection. Its limitation depends on the submode. Chunk-level rollout can evaluate many candidates in parallel, but it cannot react to a control signal that changes inside the predicted window. Step-wise rollout can react after each predicted state, but it loses the parallelism of a single generated window.

4.3.2. Joint Generation

The joint factorization produces substrate and action from a single generative process on a shared backbone. Repeating Equation 11,

$$p_\theta(s_{t+1:t+H},\, a_{t:t+H-1} \mid c). \tag{18}$$

Training is often expressible as, or approximated by, a weighted sum of a substrate-side generative loss and an action-side regression or classification loss,

$$\mathcal{L}_{\text{joint}}(\theta) \;=\; \mathcal{L}_{\text{gen}}(s) \;+\; \lambda\, \mathcal{L}_{\text{act}}(a), \tag{19}$$

where $\mathcal{L}_{\text{gen}}$ is a diffusion or autoregression loss on s and $\mathcal{L}_{\text{act}}$ is a flow-matching, regression, or cross-entropy loss on a. Section 4.4 reuses Equation 19 as the canonical training objective of hybrid backbones.

Joint diffusion of video and action. The joint-diffusion group differs less in its factorization than in the interface through which the two streams meet. UWM [51], CosmosPolicy [116], VideoVLA [75], DriveVA [78], and DriveWAM [55] place video-side latents and action-side variables in the same denoising or flow-matching state, so one sampled trajectory carries both the imagined future and the action chunk. CoVAR [74], UD-VLA [73], GigaWorld-Policy [108], and CKT-WAM [98] keep the same joint target but change the information routing through bridge attention, discrete diffusion, action-centered masking, or teacher-context prefixes. Multisubstrate variants add another future channel without changing the coupling: AIM [54] inserts a value map, DriveDreamer-Policy [85] orders depth, video, and action inside one training pass, MV-VDP [86] and ActionImages [77] attach end-effector heatmaps to the visual stream, VTAM [121] adds a tactile force proxy, HiF-VLA [131] adds motion-vector foresight tokens, and DUST [10] applies the same idea to a VLM-substrate setting. The

 doi:10.20944/preprints202606.1403.v1

useful distinction is therefore whether the added channel changes the substrate, not whether the action is still jointly produced.

Joint autoregression of frame and action tokens. GR-1 [48] and GR-2 [49] jointly predict next-frame and next-action tokens autoregressively over a discrete codebook. WorldVLA [52] carries the same recipe to a Chameleon backbone with an action-chunk attention mask between action and frame tokens. PhysGen [72] adapts the autoregressive NOVA backbone with continuous physical tokens and a per-token diffusion detokenizer, so that autoregressive image and action streams share a joint training loss in the sense of Equation 19. ICLR-VR [132] serializes state, reasoning, and action tokens into a single Llama2-style causal stream under a combined next-token loss, with the gripper-keypoint polyline decoded immediately before each action chunk.

Mixture-of-experts hybrids that joint-denoise. The hybrid branch uses separate experts or heads so that video and action can share a trunk without forcing every layer to treat both variables identically. UVA [102] and PAD [50] are the clean baselines: one shared pass supports both visual prediction and action, and the visual branch can be bypassed when only control is needed. F1 [96], Motus [76], MotuBrain [118], RynnVLA-002 [71], BagelVLA [89], Pelican-Unify 1.0 [99], and ALAM [129] instantiate the mixture pattern with different expert allocations for understanding, video, latent transition, and action streams. A second set keeps the same joint target but changes the schedule: DreamZero [53] uses one shared timestep across a Wan-style video-action state, X-WAM [83] lets action finish before multi-view video, NoiseGate [80] learns per-latent denoising increments, τ_0-WM [57] samples video and action flow times independently, and WALL-WM [56] couples action queries into matched video blocks at every depth. CLWM [122], DAWN [123], JOPAT [124], and DDP [133] show the same coupling on feature, track, or compact latent substrates, while FFDC-WAM [125] keeps the Motus joint rollout and adds a verifier that decides when the cached future is still trustworthy. The family is diverse, but the shared claim is narrow: the action and the future substrate are trained as mutually constrained variables rather than as a planner output followed by a passive decoder.

The benefit of the family is one coupled sample or rollout that produces both predictions and a strong mutual consistency between them. The trade-off is potential training instability because the generation and action losses can pull the shared representation in different directions. Practitioners therefore often pretrain the substrate head on video and introduce the action head later, sometimes with a careful λ schedule in Equation 19.

4.3.3. Post-Prediction Action Head

The post-prediction head factorizes into a substrate generator and a smaller action expert q_ψ that decodes action from the substrate. Repeating Equation 12,

$$p_\theta(s_{t+1:t+H},\, a_{t:t+H-1} \mid c) = p_\theta(s_{t+1:t+H} \mid c)\; q_\psi(a_{t:t+H-1} \mid s_{t+1:t+H},\, c), \tag{20}$$

with θ frozen during q_ψ training in many implementations. This arrangement is common because it lets a pretrained predictor stay fixed while the action head changes across embodiments.

Video-plan then action-recovery WAMs. UniPi [4] a canonical early example: a text-conditioned diffusion planner first synthesizes a future image trajectory, and a task-specific inverse-dynamics module then recovers the low-level actions between generated frames. VLP [5] adds a VLM policy and value-guided search over abstract text actions, but the executable controls still come from goal-conditioned policies that consume the synthesized video plan. EVA [97] pairs a Wan2.1-14B planner with a frozen spatial-softmax IDM and uses the IDM-derived smoothness-and-limits reward to align the planner via GRPO without folding the IDM into the joint denoiser. CreFlow [63] post-trains the Vidar text-image-to-video trunk against an LTL-grounded corrective reflow loss and keeps the Vidar inverse-dynamics expert frozen. These methods are often described as video planning or video world modeling, but under the notation here their action coupling is the post-prediction factorization of Equation 20.

 doi:10.20944/preprints202606.1403.v1

Action expert on a latent, token, or multimodal substrate. This group keeps the action expert separate, so the key question is what future object q_ψ consumes. Some members expose a task-structured future before action decoding: $\pi_{0.7}$ [14] uses BAGEL-style subgoal images as visual targets, CoT-VLA [88] uses visual chain-of-thought tokens, PALM [130] uses affordance maps, and OmniVTA [115] uses visual and tactile latents whose fixed decoders tie them back to observation space for slow-fast contact-rich control. Feature-substrate members differ mainly in where the future feature comes from. FLARE [126], LDA-1B [12], FRAPPE [127], and GigaBrain-0.5M* [120] rely on future embeddings or value-bearing latent tokens, while VPP [7], VideoPolicy [103], Genie Envisioner [104], LaMP [109], and VAMPO [110] intercept intermediate states of a video-trained trunk. The action expert can therefore be small, but its reliability depends on whether the chosen substrate preserves the action-relevant part of the forecast.

Action expert decodes structured geometry from a generated future. The geometric post-head line uses motion or structure as the intermediate action-facing representation. Im2Flow2Act [111], 3DFlowAction [112], 3D-FDP [113], NovaFlow [38], Dream2Flow [39], and AeroPlace-Flow [70] make flow the intermediate future, while TesserAct [81], 4DGen [36], MVISTA-4D [82], TC-IDM [65], and Dex4D [84] recover pose or point structure before control. Earlier decoded-video cascades such as AVDC [6], Dreamitate [59], This&That [61], ARDuP [100], Gen2Act [60], GR-MG [62], and GraspDreamer [66] fit the same factorization even when the extracted geometry is implicit in a tracker or learned head. RoboEnvision [87], Vidar [92], LVP [93], Veo-Act [94], Say-Dream-Act [91], NovaPlan [69], and MoLA [64] keep the same post-head order but add stronger video planners, VLM scoring, or modality-specific inverse dynamics. The shared lesson is that a generated future can be action-facing even when the final action module is trained separately.

Generated future is partly or fully optional at inference. The optional-future pattern trains with a substrate generator, then shortens or bypasses that generator when the inference-time action path can use an internal state. VidMan [134] is the clearest representative: it mounts a policy head on frozen video-diffusion features rather than waiting for a rendered future. mimic-video [8], S-VAM [9], DiT4DiT [107], Fast-WAM [58], and VAG [95] make the same move through ODE checkpoints, distilled foresight states, fixed denoising timesteps, masked future-video tokens, or pooled clean latents. WAV [106], MWM [114], DexWM [128], and SWEET [90] instead keep a compact action-facing substrate, such as valued latent rollouts, semantic-mask dynamics, DINO latents, or sparse keyframes. LingBot-VA [119] and HarmoWAM [79] show the mixture-backbone version, where visual dynamics remain available but a lighter action expert or gate handles the frequent control path. This design keeps q_ψ cheap and embodiment-specific. But the risk is equally clear, since the skipped future can help only if the retained state still preserves the part of the forecast that action needs.

4.3.4. Action Representation and Chunk Size

Across the three coupling families, the action variable $a_t \in \mathcal{A}$ is parameterized differently. Three representations are common. The first is per-step continuous controls, with $\mathcal{A} = \mathbb{R}^{d_a}$ for joint velocities, end-effector deltas, or gripper commands. The second is discrete tokens from a fixed codebook, as in autoregressive joint denoisers such as WorldVLA [52], where $\mathcal{A}$ is a finite vocabulary obtained from per-dimension binning. The third is learned latent actions that act as currency between the world model and the policy. In that case $\mathcal{A} = \mathbb{R}^{d_z}$ for some learned $d_z \ll d_a$, as in Motus [76], ALAM [129], and LDA-1B [12]. The chunk size is the second nested choice. A model that produces one action per forward pass pays a low per-action latency but invokes the full backbone at the control frequency. A model that produces a long chunk per pass spreads the backbone cost over the chunk but cannot react during it. Closed-loop control on real hardware pushes methods toward smaller chunks or toward cheaper backbones, and many of the architectural choices in Section 4.4 can be reinterpreted as a response to that pressure.

 doi:10.20944/preprints202606.1403.v1

4.4. Architectural Backbone: How the Prediction Is Produced

Once a substrate space $\mathcal{S}$ is fixed and a coupling factorization is chosen, the next ingredient is the function family that realizes $p_\theta(\cdot \mid c)$. We group current backbones into five families, and for each we list the defining parameterization, the canonical training objective, and the WAMs that use it.

4.4.1. Iterative-Denoising Backbones (Video Diffusion)

The diffusion family writes p_θ as the marginal of a learned reverse process over a sequence of progressively denoised latents $s^{(N_{\text{diff}})}, s^{(N_{\text{diff}}-1)}, \ldots, s^{(0)} = s_{t+1:t+H}$:

$$p_\theta(s_{t+1:t+H} \mid c) = \int p(s^{(N_{\text{diff}})}) \prod_{n=1}^{N_{\text{diff}}} p_\theta\Big(s^{(n-1)} \mid s^{(n)}, c\Big)\, \mathrm{d}s^{(N_{\text{diff}}:1)}, \tag{21}$$

where $p(s^{(N_{\text{diff}})})$ is a Gaussian noise prior and each reverse step is parameterized by a noise-prediction network $\epsilon_\theta(s^{(n)}, n, c)$. Training minimizes the standard score-matching loss

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{n, s^{(0)}, \epsilon}\Big[\big\|\epsilon - \epsilon_\theta\Big(s^{(n)},\, n,\, c\Big)\big\|^2\Big], \quad s^{(n)} = \alpha_n s^{(0)} + \sigma_n \epsilon. \tag{22}$$

At inference, sampling draws $s^{(N_{\text{diff}})} \sim p(s^{(N_{\text{diff}})})$ and runs N_{diff} reverse steps.

Founding video-diffusion backbones. The dominant topology is a video diffusion transformer that attends jointly over space and time, as in Latte [32], CogVideoX [27], Wan [26], and Cosmos-Predict2 [28]. AnimateDiff [33] represents an earlier generation that adapts a frozen image-diffusion backbone with a learned temporal motion module.

Because standard diffusion denoises a whole window at once rather than advancing one element at a time, the coupling-type landscape is narrower than the autoregressive one. The census mainly uses diffusion in three ways:

$$(\mathcal{Y}, c_{\text{diff}}) = \begin{cases} (s_{t+1:t+H},\, (c, a_{t:t+H-1})), & \text{action-conditioned rollout,} \\ ((s_{t+1:t+H},\, a_{t:t+H-1}),\, c), & \text{joint generation,} \\ (s_{t+1:t+H},\, c) \text{ followed by } q_\psi(a_{t:t+H-1} \mid s_{t+1:t+H}, c), & \text{post-prediction head.} \end{cases} \tag{23}$$

Pure diffusion supports action-conditioned rollout most naturally when the action source proposes a whole candidate chunk before the reverse chain begins, as in DreamAvoid [44]. Step-wise rollout is less natural for a windowed reverse chain, so it appears mainly in non-census foundation or simulator works such as AdaWorld [136] and FlowDreamer [137], or in hybrid wrappers that alternate an action source and a predictor.

Post-prediction head (Eq. 12). Many diffusion WAMs first denoise the substrate $s^{(0)}$ and then decode action from it with a separate module q_ψ. The important variation is which substrate q_ψ consumes, not the mere fact that the action head is separate.

Decoded-observation members use the generated visual future as the common interface. UniPi [4] is the representative case, since a text-conditioned diffusion rollout is followed by inverse dynamics. VLP [5] changes the conditioning context before sampling, while AVDC [6], Dreamitate [59], This&That [61], Gen2Act [60], and GR-MG [62] recover actions, tool motion, or goals from the produced frames. Later visual planners mostly strengthen the generator or the search wrapper: RoboEnvision [87], Vidar [92], MVISTA-4D [82], LVP [93], Veo-Act [94], and Say-Dream-Act [91] keep the same factorization while changing horizon, view structure, executor, or sampling schedule.

Pixel-latent and geometric variants keep the post-head order but make the substrate cheaper or closer to control. ARDuP [100], VILP [101], DiT4DiT [107], and OmniVTA [115] act from latent video, intermediate denoising states, or visual-tactile decoder-bound latents. Im2Flow2Act [111] is the representative geometric instance because it denoises object flow as the action-facing future. 3DFlowAction [112] and TesserAct [81] extend the same idea to 3D flow or reconstructed point clouds.

 doi:10.20944/preprints202606.1403.v1

The trade-off is direct: these substrates reduce appearance burden, but their usefulness depends on whether the chosen geometry captures the task-relevant contact and motion.

Feature-level post heads push the shortcut further by consuming latent states rather than decoded futures. VPP [7], Video Policy [103], Genie Envisioner [104], Act2Goal [105], WAV [106], VAG [95], EnerVerse [24], VidMan [134], mimic-video [8], S-VAM [9], and Fast-WAM [58] use variations of this shortcut, such as tapping intermediate denoising states, distilling foresight features, fixing a small number of denoising steps, or bypassing explicit video generation at inference. Related compact-substrate variants, such as MWM [114], pursue a similar latency goal through task maps rather than through a pure feature tap.

Joint generation (Eq. 11). The second group of diffusion WAMs models the future substrate s and the future action a within a coupled denoising or flow-matching process, so Equation 21 operates on the joint variable (s, a). PAD [50] is a compact image-action DiT instance, while VideoVLA [75], CosmosPolicy [116], CoVAR [74], UD-VLA [73], GigaWorld-Policy [108], and CKT-WAM [98] vary the shared denoising state through a pretrained video trunk, bridge attention, discrete tokens, action-centered masking, or transfer prefixes. GigaWorld-Policy is the action-centered case in this group: its causal mask prevents future-video tokens from influencing action tokens, so deployment can sample the action chunk without instantiating future-video tokens. AIM [54], X-WAM [83], and AdaWorldPolicy [117] show the same diffusion coupling with value maps, multi-view 4D video, or force-prediction modules attached to the joint state. The benefit is a coupled generative process that encourages consistency between action and future prediction. The trade-off is potential training instability because the generation and action losses can impose competing demands on the shared representation. Practitioners therefore often pretrain the substrate head on video and introduce the action head later, sometimes with a careful λ schedule.

Flow matching as a variant of Equation 21. A large subset of the diffusion family instantiates ϵ_θ as a velocity-field predictor under flow matching rather than as a noise predictor under score matching, but both still satisfy the iterative-reverse-chain parameterization of Equation 21. Conditional flow matching learns $v_\theta(x_t, t, c)$ to regress the linear interpolation velocity $x_1 - x_0$ under $\mathcal{L}_{\text{FM}} = \mathbb{E}\,\|(x_1 - x_0) - v_\theta(x_t, t, c)\|_2^2$, and at inference the same Euler reverse integration takes the place of the noise-prediction reverse chain. The flow-matching variant is therefore not a new backbone family but a different choice of the score parameterization inside Equation 21. Among the post-prediction-head WAMs above, 3D-FDP [113], EVA [97], VAMPO [110], and MoLA [64] run flow matching as the score parameterization. Among the joint-denoising WAMs, DriveVA [78], ActionImages [77], MV-VDP [86], and CreFlow [63] take the same route on a Wan-class trunk. The choice between score matching and flow matching remains separable from the coupling family within Equation 21.

4.4.2. Autoregressive Backbones (Next-Frame, Next-Token)

The autoregressive family is a backbone parameterization, not an action-coupling regime by itself. It realizes whichever generative factor from Section 4.3 is assigned to it by serializing that factor's output into a causal stream. Let

$$y_{1:M} = \text{Serialize}(\mathcal{Y}), \tag{24}$$

where $\mathcal{Y}$ is the set of variables generated by the autoregressive backbone, and let c_{ar} denote the local context supplied to that backbone. The generic autoregressive parameterization is

$$p_\theta(y_{1:M} \mid c_{\text{ar}}) = \prod_{j=1}^{M} p_\theta(y_j \mid y_{<j}, c_{\text{ar}}), \tag{25}$$

 doi:10.20944/preprints202606.1403.v1

where y_j may be a frame, a latent-grid block, a discrete visual token, a continuous latent token, an action token, or a compact latent state. Training minimizes the corresponding next-element loss

$$\mathcal{L}_{\mathrm{ar}}(\theta) = -\sum_{j=1}^{M} \log p_\theta\big(y_j \mid y_{<j}, c_{\mathrm{ar}}\big), \tag{26}$$

with cross-entropy for discrete tokens and a continuous likelihood, regression, diffusion, or flow-matching loss for continuous elements. At inference, a key-value cache reuses prefix computation, so each newly appended element requires one forward pass over the new tokens rather than recomputation over the whole history.

The pair $(\mathcal{Y}, c_{\mathrm{ar}})$ is inherited from the coupling choice rather than from autoregression itself:

$$(\mathcal{Y}, c_{\mathrm{ar}}) = \begin{cases} (s_{t+1:t+H},\ (c, a_{t:t+H-1})) \text{ or } (s_{t+k+1},\ (h_k, a_{t+k}, c)), & \text{action-conditioned rollout,} \\ ((s_{t+1:t+H}, a_{t:t+H-1}),\ c), & \text{joint autoregression,} \\ (s_{t+1:t+H},\ c) \text{ followed by } q_\psi(a_{t:t+H-1} \mid s_{t+1:t+H}, c), & \text{post-prediction head.} \end{cases} \tag{27}$$

Equation 27 is only a backbone-level map. The full joint factorizations remain those in Section 4.3. Its purpose is to prevent the common confusion that autoregression implies joint action prediction. In many WAMs the autoregressive stream predicts only the future substrate. The future action may be supplied to the rollout, generated in the same stream, or decoded later by q_ψ.

Autoregressive video priors (not WAMs). The following examples provide the video prior but not the WAM coupling. The list is selective rather than exhaustive, because these works cover the substrate forms that later WAMs reuse. VideoGPT [139] gives the VQ-VAE plus GPT recipe over discrete video latents. VideoPoet [34] is a decoder-only multimodal video language model over tokenized visual and audio streams. NOVA [29] removes the discrete codebook and predicts continuous video latents. These works define reusable autoregressive video priors, but they are not WAMs by themselves. Earlier GAN-based video generators such as MoCoGAN [140] belong to the historical substrate discussion of Section 4.2. They are not instances of the next-element parameterization in Equation 25.

Action-conditioned rollout (Eq. 10). The following methods use autoregression for action-conditioned sequential dynamics. iVideoGPT [22] brings the VideoGPT-style token stream into interactive world modeling by interleaving observations, actions, and rewards. AdaWorld [136] is the boundary case in this group. The paper calls it an autoregressive world model because inference repeats action-conditioned next-frame prediction, but each local transition is an SVD-based denoising problem. It is therefore best understood as an autoregressive rollout with a diffusion transition, not as a homogeneous next-token decoder. PlaNet [16], DreamerV3 [17], Dreamer 4 [19], and TransDreamer [18] are latent-imagination methods based on compact dynamics models. These methods explain the sequential-dynamics side of the family, but they usually pair the dynamics model with a separate actor, planner, or critic and therefore serve here as background rather than census WAMs.

Joint autoregression (Eq. 11). The following methods use autoregression as the trunk for streams that include both future substrate elements and action-facing elements. WorldVLA [52] is the clearest discrete instance: it serializes language, image, and action tokens in one autoregressive token framework. PhysGen [72] is the continuous-latent counterpart, adapting the NOVA [29] autoregressive video prior to continuous physical tokens. GR-1 [48] establishes the GPT-style video-generative trunk for robot manipulation, and GR-2 [49] scales that trunk with larger video-language pretraining and robot-trajectory fine-tuning. From the backbone perspective, the shared property is causal next-element prediction. The exact action head, detokenizer, or joint-output factorization is the coupling detail handled in Section 4.3.

Post-prediction head (Eq. 12). The post-head line of Equation 27 is still possible for autoregressive backbones, even when it is less common than joint token prediction. LingBot-VA [119] is the closest current member because it models visual dynamics before inverse dynamics, but it uses an autoregressive diffusion mixture-of-transformers rather than the next-element backbone of Equation 25. The

 doi:10.20944/preprints202606.1403.v1

distinction matters because the backbone can be causal while the action interface remains a separate decoder over the predicted substrate.

The common pressure across the family is serial cost. Autoregression avoids the many denoising steps of diffusion and benefits from caching, but it still advances the generated stream one element at a time. For WAMs, the relevant latency is therefore governed by what the stream contains: full frames, latent blocks, visual tokens, action tokens, or compact latent states. When an autoregressive WAM misses a real-time control budget, the first places to inspect are the serialized substrate size, the action-chunk length, and whether actions are produced inside the same causal stream or decoded by a cheaper head.

4.4.3. Joint-Embedding-Predictive Backbones

The joint-embedding-predictive family realizes $p_\theta(s_{t+1:t+H} \mid c)$ as a Dirac at a deterministic predictor rather than as a stochastic density. It learns a context encoder $\mathbf{E}^{\text{ctx}}$ and an exponential-moving-average target encoder $\mathbf{E}^{\text{tgt}}$ together with a predictor f_θ^{pred} that maps context features to target features. Given a context view x_{ctx} of the past and a target view x_{tgt} of the future, the objective is

$$\mathcal{L}_{\text{jepa}}(\theta) = \mathbb{E}_{x_{\text{ctx}}, x_{\text{tgt}}}\Big[\big\| f_\theta^{\text{pred}}(\mathbf{E}^{\text{ctx}}(x_{\text{ctx}})) - \text{sg}\big[\mathbf{E}^{\text{tgt}}(x_{\text{tgt}})\big]\big\|^2\Big], \tag{28}$$

with $\text{sg}[\cdot]$ a stop-gradient. Under the unified contract of Equation 8, the substrate $s_{t+1:t+H}$ is $\mathbf{E}^{\text{tgt}}(x_{\text{tgt}})$, and the predictive distribution $p_\theta(s_{t+1:t+H} \mid c)$ collapses to a Dirac at $f_\theta^{\text{pred}}(\mathbf{E}^{\text{ctx}}(x_{\text{ctx}}))$. The JEPA backbone never inverts $\mathbf{E}^{\text{tgt}}$, which makes the inference loop cheap and is the source of the family's appeal. The backbone predicts the substrate only. Actions enter through one of two coupling patterns:

$$(\mathcal{Y}, c_{\text{jepa}}) = \begin{cases} (s_{t+k+1},\ (h_k, a_{t+k}, c)) \text{ at rollout step } k, & \text{action-conditioned rollout,} \\ (s_{t+1:t+H},\ c) \text{ followed by } q_\psi(a_{t:t+H-1} \mid s_{t+1:t+H}, c), & \text{post-prediction head.} \end{cases} \tag{29}$$

Action-conditioned rollout treats the JEPA predictor as a world model and recovers actions through planning over imagined consequences. A post-prediction head predicts the full substrate window first and then decodes action from it with a separate expert q_ψ.

I-JEPA [141] introduces the architecture on images, predicting the latent representations of large masked target blocks from a spatially distributed context block. A-JEPA [142] carries the same recipe to audio spectrograms under a curriculum time-frequency masking strategy, and MC-JEPA [143] couples the prediction objective with optical-flow learning so that one shared encoder serves both motion and content. V-JEPA [20] and V-JEPA 2 [21] extend the idea to video through pure feature prediction, and the V-JEPA 2 action-conditioned variant shows how such features can support MPC through action-conditioned latent rollout. The tabulated WAMs use this lineage in two ways. FLARE [126] pairs a deterministic JEPA predictor with a downstream action expert in the post-prediction-head factorization by aligning additional DiT tokens to future-observation embeddings from a frozen teacher. DAWN [123] keeps V-JEPA 2-style feature tokens inside a hybrid WAM that alternates compact latent-world prediction with action denoising. The common pressure is that the embedding space has no intrinsic visual quality measure, so downstream-task accuracy becomes the only standard test of the world model.

4.4.4. Hybrid Backbones (Generative Head Plus Action Head)

The hybrid family runs a generative head and an action head on the same backbone. Schematically,

$$p_\theta(s_{t+1:t+H},\, a_{t:t+H-1} \mid c) \ \propto\ g_\theta(c) \ \longrightarrow\ \{h_\theta^s,\, h_\theta^a\}, \tag{30}$$

where g_θ is a shared encoder-decoder trunk and h_θ^s, h_θ^a are two output heads. Training minimizes the joint coupling loss of Equation 19 from Section 4.3, namely $\mathcal{L}_{\text{gen}}(s) + \lambda \mathcal{L}_{\text{act}}(a)$, with the substrate head and action head sharing the trunk g_θ.

 doi:10.20944/preprints202606.1403.v1

Because the shared trunk g_θ feeds both heads in one place, the hybrid family is the backbone in which every coupling family of Section 4.3 can appear, and which one a given model realizes is fixed by how the substrate head h^s_θ and the action head h^a_θ are ordered around that trunk:

$$(h^s_\theta,\, h^a_\theta) = \begin{cases} (s, a) \text{ jointly from } g_\theta(c), & \text{joint generation,} \\ s \text{ from } h^s_\theta, \text{ then } p^a_\theta(a \mid s, c), & \text{post-prediction head,} \\ q_\psi(a \mid c) \text{ or external search, then } p^s_\theta(s \mid c, a), & \text{action-conditioned rollout.} \end{cases} \tag{31}$$

Here p^s_θ and p^a_θ denote the substrate-prediction and action-prediction distributions implemented by h^s_θ and h^a_θ when a shared trunk feeds multiple heads. The action-producing couplings place the action inside or after the shared pass, whereas action-conditioned rollout wraps the same trunk with a planner, sampler, tree search, or feedback-guided action proposal.

Joint generation (Eq. 11). Joint generation is the native arrangement of the family, because a single pass of $g_\theta(c)$ produces the future substrate s and the future action a under the loss of Equation 19. The plain shared-trunk form is easiest to see in UVA [102]: the video head shapes the representation during training, and the action head can be used alone at inference. UWM [51], DUST [10], VTAM [121], DDP [133], DriveDreamer-Policy [85], and DriveWAM [55] keep that contract while changing the substrate from pixel-decodable latents to VLM tokens, tactile force proxies, compact latent states, or driving-specific video and depth.

Mixture-of-Transformer variants keep joint generation but separate expert routes so that video and action do not have to share every layer. F1 [96] is the representative mixture case, with generation and action experts coupled by next-scale prediction. Motus [76], MotuBrain [118], X-WAM [83], and RynnVLA-002 [71] vary the same idea through optical-flow latent actions, Vidu latents, asynchronous denoising, or an autoregressive-image plus continuous-action split. BagelVLA [89], CLWM [122], Pelican-Unify 1.0 [99], DAWN [123], NoiseGate [80], WALL-WM [56], and τ_0-WM [57] then vary the schedule and exchange pattern through same-timestep attention, feature flow matching, iterative refinement, per-latent timestep gates, depth-wise cross-attention, or propose-rank-rectify control.

FFDC-WAM [125] sits at the boundary between cached chunked rollout and closed-loop replanning: the Motus joint rollout remains the action-facing substrate, while a lightweight verifier decides when cached imagination should be trusted. Across the group, the shared benefit is consistency between the predicted world and the chosen action. The cost is that the action and future heads must train together without one objective overwhelming the other.

Post-prediction head (Eq. 12). A different arrangement keeps the shared trunk but orders the heads, so h^s_θ completes the substrate window before h^a_θ decodes the action from it. FRAPPE [127] is the teacher-feature representative, routing multiple predicted future embeddings into one action transformer. Genie Envisioner [104], GigaBrain-0.5M* [120], and LaMP [109] instead expose video-trained latent, value, or motion-expert states to the action expert. LingBot-VA [119] and HarmoWAM [79] keep a visual-dynamics branch active but hand frequent control to inverse-dynamics or gated action experts. Dex4D [84] and DexWM [128] show the same order on geometric or DINO-latent futures. The action expert here conditions only on the predicted substrate, so it stays cheap and can be retrained per embodiment without touching the trunk, provided that substrate is action-informative.

Action-conditioned rollout (Eq. 10). Action-conditioned rollout enters the family when an action source is specified before or during consequence prediction. 3D-ALP [43] is the pixel-grounded tree-search version, where MCTS proposes joint actions, kinematics converts them into 3D camera poses, and a 3D-consistent world model renders candidate futures for scoring. PointWorld [46] is the geometric chunk-level version, composing an MPPI sampler over end-effector trajectories with an action-conditioned predictor over a robot point-flow substrate. EgoExo-WM [45] uses horizon-length body-motion candidates from UniEgoMotion to condition a DINOv3-L latent predictor under MPC. Feedback-WM [47] uses the retained predictor to guide or correct proposed action chunks after the

 doi:10.20944/preprints202606.1403.v1

action source has formed them. The retained world-prediction module therefore scores or corrects proposed actions instead of producing them as an unconstrained action head.

Across the hybrid backbone family, the main cost is maintaining both a world-prediction path and an action path in the same runtime design. The benefit depends on the coupling. Joint generation encourages consistency between predicted futures and actions within a coupled generative process. Post-prediction heads keep the action expert relatively cheap once the substrate has been produced. Action-conditioned rollout supports counterfactual scoring, but repeated candidate evaluation can dominate inference. For joint members, the main optimization challenge is that substrate and action objectives can impose competing demands on the shared trunk. Practitioners therefore often pretrain the substrate path on video before introducing the action path, sometimes with a scheduled weight on the action loss in Equation 19.

4.4.5. LLM- and VLM-Backbone WAMs

The fifth family builds a WAM directly on a large language or vision-language backbone without using a video-generation trunk as the main policy backbone. The natural substrate is the VLM-token variant of the feature category in Equation 14, since the future is described as a token block in the VLM's vocabulary, and several members of the family reach across into the geometric or affordance categories of Section 4.2. The backbone is the VLM itself. The contract of Equation 8 is realized by composing a VLM forward pass with an optional action expert under the post-prediction-head factorization:

$$s_{t+1:t+H} \sim \mathrm{VLM}_\theta(\mathrm{prompt}(c)), \qquad a_{t:t+H-1} \sim q_\psi(\,\cdot \mid s_{t+1:t+H}, c). \tag{32}$$

The factorization in Equation 32 realizes a WAM when a future substrate is part of the control pathway, either produced by the LLM/VLM itself or supplied by an attached world-model module before action decoding. A VLM policy that maps observations directly to action tokens, without predicting or consuming such a future substrate, falls outside this family under our definition. This subsection therefore covers LLM- and VLM-backbone designs in which future tokens, subgoal images, motion representations, affordance maps, or related latent transitions are present in the inference-time control context.

Among the π-series models, $\pi_{0.7}$ [14] satisfies this gate by conditioning the policy on generated subgoal images. At runtime, a lightweight BAGEL-initialized world model produces multi-view near-future visual subgoals, which are encoded together with observations by the Gemma3-based VLA before its 860M-parameter flow-matching action expert predicts the action chunk. CoT-VLA [88] writes an explicit visual chain-of-thought into the substrate before action is decoded. LDA-1B [12] learns dynamics, policy, and forecasting jointly in a structured DINO-based latent on a Qwen3-VL backbone. HiF-VLA [131] adopts a pretrained VLA trunk with DINOv2 and SigLIP encoders that produces both motion-vector substrate tokens and action latents in one inference pass. PALM [130] carries the family down to a GPT-2-class transformer over CLIP text, MAE vision, and MLP-encoded state, with a diffusion transformer attached as the action expert. ICLR-VR [132] adopts a Llama2-style causal transformer with a pretrained ViT image encoder and decodes a future-third-view gripper-keypoint polyline as the structured substrate before the action chunk fires. ALAM [129] pairs a PaliGemma-2B trunk with a Gemma-300M flow-matching expert, with an algebraic consistency regulariser living entirely in the frozen pretraining encoder. These examples keep the VLM family inside the WAM gate because the future substrate remains in the inference forward path.

The whole family inherits the cost of the VLM stack rather than that of video diffusion, which is why many members use action chunking and a frozen backbone at inference.

4.4.6. The Five Families Together

The five families are not mutually exclusive and the dividing lines move every few months. Equations 21, 25, 28, 30, and 32 are the canonical parameterizations and they appear across substrates. A practitioner who fixes the substrate in Section 4.2 has narrowed the backbone choices by construction,

 doi:10.20944/preprints202606.1403.v1

since not every substrate is compatible with every family. The remaining freedom is mostly about where the prediction lands relative to the control loop, which is the subject of Section 4.5.

4.5. Deployment Regime: Interactive, Rollout, Open Loop, Closed Loop

The final ingredient is the regime the WAM is meant to run in. It is the choice of horizon H in Equation 8 and of the cadence with which the WAM is invoked relative to the control loop. Let T be the task length in control steps, let $N_{\text{fwd}}(H)$ be the cost of one forward pass that produces a substrate trajectory of length H, and let f_{ctrl} be the control frequency. The four regimes below differ primarily in how they allocate N_{fwd} over the task of length T and how frequently new observations can revise the current plan.

4.5.1. Open-Loop Rollout

Open-loop rollout sets $H \approx T$ and invokes the WAM once. A long window of future substrate is generated once before execution and a separate actor consumes it. Total compute spent on the WAM is

$$\mathcal{C}_{\text{open}}(T) = N_{\text{fwd}}(T), \tag{33}$$

which is paid once per scenario and does not scale with f_{ctrl}. UniPi [4], AVDC [6], Dreamitate [59], This&That [61], ARDuP [100], Gen2Act [60], RoboEnvision [87], LVP [93], and MVISTA-4D [82] run the WAM open-loop in their original setting, with the actor consuming the offline rollout through tracking or inverse dynamics. PointWorld [46] marks the open-loop limit of planning: MPPI evaluates chunked point-flow predictions over the horizon before execution, but the reported experiments do not replan during execution. Data-generation uses of Cosmos-Transfer1 [135] and Sora-class video models [30,31] illustrate a related but non-census use of generated futures, where the future model supplies synthetic experience rather than acting as an inference-time WAM. The benefit of the regime is that the cost is paid once per scenario. Its main limitation is that the rollout cannot react to real-world deviations, so the actor that consumes it must absorb every distribution shift on its own.

4.5.2. Chunked Closed-Loop Control

Chunked closed-loop sets $H = K$ for some replan period K and invokes the WAM every K control steps. The WAM produces an action chunk of length K, the actor executes the chunk in K/f_{ctrl} seconds, and the next invocation begins. Total compute over a task of length T is

$$\mathcal{C}_{\text{chunk}}(T, K) = \lceil T/K \rceil \, N_{\text{fwd}}(K), \tag{34}$$

which interpolates between open-loop ($K \to T$) and single-step ($K \to 1$). It is feasible as long as $N_{\text{fwd}}(K)/K < 1/f_{\text{ctrl}}$, so that the next chunk is prepared before the previous one is consumed. This is a common choice for real-robot deployment because it amortizes a large backbone over multiple control ticks.

VLM-stack chunked controllers. In VLM-stack WAMs, chunked control appears when a future-substrate token block or structured future is refreshed before an action chunk is decoded. $\pi_{0.7}$ [14] produces a chunked action from its Gemma3 flow-matching expert after a separate BAGEL-class world model refreshes its multi-view subgoal images every four seconds, making it the π-series member that satisfies the inference-time gate. CoT-VLA [88] writes a visual chain-of-thought before each chunk is decoded. HiF-VLA [131] models motion-vector substrate tokens and action latents in parallel on a pretrained VLA trunk. PALM [130] and ICLR-VR [132] place a diffusion or autoregressive action expert on top of an affordance map or future-third-view gripper-keypoint polyline substrate. ALAM [129] uses algebraically consistent latent transitions on a PaliGemma-family trunk. FRAPPE [127], LDA-1B [12], DUST [10], FLARE [126], and GR-2 [49] also use chunked control, with the substrate supplied by a future embedding, a dual-stream diffusion head, or autoregression over frame and action tokens.

 doi:10.20944/preprints202606.1403.v1

Pixel-latent and feature chunked controllers. WAMs that avoid decoded observation output use the same amortization principle, but differ in how much of the future substrate remains active at inference. UVA [102] and UWM [51] are the representative head-skipping cases, because the video side helps training while the inference call can produce only the action chunk. F1 [96], Motus [76], MotuBrain [118], BagelVLA [89], NoiseGate [80], Pelican-Unify 1.0 [99], τ_0-WM [57], and WALL-WM [56] keep the same chunked regime inside MoT trunks. Genie Envisioner [104], mimic-video [8], RynnVLA-002 [71], UD-VLA [73], and CKT-WAM [98] show adjacent ways to reduce per-call work, through a slow latent refresh, intermediate features, continuous or discrete joint autoregression, or a frozen student with updated context tokens. VideoVLA [75], CoVAR [74], CosmosPolicy [116], AdaWorldPolicy [117], DriveVA [78], DriveWAM [55], and DriveDreamer-Policy [85] keep chunked joint generation active in manipulation or driving settings.

Other chunked designs place the chunk behind a planning or abstraction stage rather than a bare action head. VLP [5] is an early decoded-video case, using value-guided search over abstract actions and replanning before goal-conditioned policies execute low-level controls. ActionImages [77], MV-VDP [86], CreFlow [63], and EVA [97] decode chunks from pixel-substrate plans. MoLA [64], VAMPO [110], LaMP [109], CLWM [122], DDP [133], and DAWN [123] do the same from latent or JEPA-style substrate plans. Among action-conditioned rollouts, 3D-ALP [43], DreamAvoid [44], and Feedback-WM [47] keep the execution window chunked, but differ in whether candidates come from tree search, critical-phase sampling, or feedback-guided denoising. EgoExo-WM [45] is the horizon-ranking boundary case, since it scores chunk-length body-motion candidates but reports open-loop rollout evaluation. JOPAT [124] and 3D-FDP [113] move the chunk to joint track or 3D-flow substrates, while Vidar [92], GR-MG [62], VILP [101], VideoPolicy [103], Act2Goal [105], Im2Flow2Act [111], TesserAct [81], and 3DFlowAction [112] show that the same chunked-cost equation can sit on pixel-grounded, feature, or geometric futures.

Compact or optional-substrate chunked controllers. Several chunked WAMs reduce the per-call burden by making the future sparse, latent, or optional. Veo-Act [94] and Say-Dream-Act [91] still decode from generated visual plans, while VAG [95], S-VAM [9], DiT4DiT [107], Fast-WAM [58], WAV [106], and DexWM [128] move the action path to feature or teacher-target futures. MWM [114] and AIM [54] replace appearance with task maps, and GigaWorld-Policy [108] and X-WAM [83] keep a joint video-action model but let the inference action path avoid the full visual schedule when possible.

Variants of chunked closed-loop control. Some WAMs keep Equation 34 but make the chunk schedule task-aware. FFDC-WAM [125] is the representative adaptive case, since a lightweight verifier compares predicted and observed latents and triggers an early replan when imagination drifts. NoiseGate [80] adapts denoising increments, WALL-WM [56] ties the chunk window to the next semantic event, and PALM [130] gates subtask progress inside the chunk. HarmoWAM [79] splits reactive and predictive loops, while SWEET [90] plus BagelVLA [89] reduce the substrate per call to task-critical keyframes. These variants still pay the chunked cost, but they spend it when the forecast is likely to change the action.

The benefit of the chunked regime is that it spreads the cost of a large backbone over the chunk and tolerates moderate backbone latency. Its main limitation is the staleness of a long chunk, which becomes visible when the world changes during chunk execution.

4.5.3. Single-Step Closed-Loop Control

Single-step closed-loop sets $H = 1$ and invokes the WAM at every control step. The model produces one substrate prediction and one action per call, and per-step compute is

$$\mathcal{C}_{\text{single}}(T) = T\, N_{\text{fwd}}(1), \tag{35}$$

subject to the hard constraint $N_{\text{fwd}}(1) < 1/f_{\text{ctrl}}$. The iVideoGPT [22] antecedent illustrates the autoregressive version of this regime, while current census members such as WorldVLA [52], VidMan [134], GR-1 [48], PAD [50], VPP [7], DreamZero [53], PhysGen [72], and OmniVTA [115] sit closest to the

 doi:10.20944/preprints202606.1403.v1

single-step WAM pattern. Dex4D [84] runs one forward pass per control tick on the student backbone to output one action and one next-joint prediction, with the goal pointer advancing only when the point-to-point distance falls below a threshold. The latent-imagination Dreamer line [17–19] remains background for this regime when the agent acts at every imagination step. The regime gives the highest reactivity but pays the highest per-step inference cost. It is feasible mainly when the substrate has been chosen sparsely, in the feature, geometric, or affordance categories of Section 4.2, or in the pixel-decodable latent variant of the pixel-grounded category. In that case one forward pass can fit inside the control period.

4.5.4. Interactive Simulator Operation

The fourth regime is the truly interactive simulator, in which a stream of user or actor inputs continuously shapes the ongoing generation and there is no fixed endpoint. The WAM is invoked at every step but reuses the past via a key-value cache or a persistent latent of state size $\mathcal{M}(t)$. Per-step compute is

$$\mathcal{C}_{\text{int}}(t) = N_{\text{fwd}}^{\text{cached}}(\mathcal{M}(t)), \tag{36}$$

where $\mathcal{M}(t)$ grows with the horizon. For attention-based backbones, the cached forward pass is linear in $\mathcal{M}(t)$ at each step and cumulative cost is quadratic in t. InteractiveWorldSimulator [25], EnerVerse [24], and LingBot-VA [119] approach this regime among the cited interactive works, while Dreamer 4 [19] remains a background lineage point for persistent latent imagination. The latent-imagination lineage from PlaNet [16] and DreamerV3 [17] through TransDreamer [18] also fits when the actor and the world model run inside a shared rollout loop. The defining property is that the model can be steered mid-stream and the same cache or persistent state carries the long-horizon trajectory. The cost pressure lands on memory rather than on per-step compute. Section 5 returns to that pressure under the heading of persistence.

4.6. Putting the Four Ingredients Together

Sections 4.1 through 4.5 let us pin down any existing WAM by a compact 4-tuple

$$\text{WAM} \;\cong\; \big(\Phi,\ \mathcal{F},\ \mathcal{B},\ \mathcal{D}\big), \tag{37}$$

whose four coordinates take values matching the table columns below. The substrate Φ from Section 4.2 takes one of {pixel-grounded, feature, geometric, affordance}. The table cells record narrower variants where they matter, with decoded or latent for pixel-grounded entries, audio-latent for the acoustic observation entry, and encoder-only, tap, teacher, or VLM token for feature. Here *Pixel (latent)* abbreviates the pixel-decodable latent route above. The action coupling $\mathcal{F}$ from Section 4.3 takes one of {action-conditioned rollout, joint generation, post-prediction head}. The backbone $\mathcal{B}$ from Section 4.4 takes one of {diffusion, autoregressive, joint-embedding, hybrid, LLM/VLM}. The deployment regime $\mathcal{D}$ from Section 4.5 takes one of {open, chunked, single, interactive}.

Example placements show how the tuple is used. F1 [96] is (pixel-decodable latent grid, joint generation, hybrid, chunked). WorldVLA [52] is (pixel-decodable latent grid, joint generation, autoregressive, single). FLARE [126] is (latent teacher-target, post-prediction head, joint-embedding, chunked). UWM [51] is (pixel-decodable latent grid, joint generation, hybrid, chunked). Frame-level video-diffusion works such as Wan [26] and CogVideoX [27] are *not* WAMs on their own because they fix only Φ and $\mathcal{B}$. They become a WAM only once $\mathcal{F}$ and $\mathcal{D}$ are added, which is exactly the wrapper-around-a-frozen-backbone pattern that runs through CosmosPolicy [116], VidMan [134], and VideoVLA [75] on top of related backbones.

The 4-tuple also clarifies the current direction of WAM design. The first and third coordinates, Φ and $\mathcal{B}$, receive substantial attention because they are training-time choices and because choosing a pretrained backbone is often the expensive design decision. The middle coordinate $\mathcal{F}$ and the closing coordinate $\mathcal{D}$ are more flexible at inference time, and this flexibility is where many recent mechanisms appear. The census reinforces this asymmetry. Substrate choices remain inside the four

 doi:10.20944/preprints202606.1403.v1

categories and backbone choices cluster around diffusion and hybrid families, while sharper variation appears in coupling mechanisms such as iterative-interactive refinement, layer-wise integration, and asynchronous-adaptive denoising, and in runtime mechanisms such as adaptive-K chunking, event-triggered invocation, and dual-frequency slow-fast loops. The expensive corner of the design space combines a decoded pixel-grounded substrate, joint generation, a diffusion backbone, and single-step closed-loop deployment. Few methods can use that corner directly in production. Designs that appear to use it usually trade one coordinate for simulation, an offline rollout, or a smaller model used at inference. Section 5 examines the same methods through the five properties that embodiment demands of them. Section 6 examines the data and the evaluation practices the field has settled on to make these design choices empirically meaningful.

Tables 1 and 2 place every WAM in our census on the 4-tuple of Equation 37. Table 1 gathers the works in the Render-and-Decode and Latent-Only philosophies of Section 3, while Table 2 gathers the Video-Generation-Free works that drop the video-generation backbone from the predictive path. Within each band, rows follow arXiv first-submission date so that the family's evolution unfolds top to bottom. The substrate column reports where the WAM represents its future, following the four categories of Section 4.2. The backbone column refers to the families of Section 4.4. The action coupling column refers to the three families of Equations 10, 11, and 12, with *Cond. rollout* used as the compact table label for action-conditioned rollout. The deployment column refers to the regimes of Section 4.5.

Table 1. Census of World Action Models in the Render-and-Decode and Latent-Only philosophies of Section 3. The upper band lists Render-and-Decode methods and the lower band lists Latent-Only methods. Columns give the four-axis anatomy of Section 4, locating each work on the design tuple of Equation 37, and within each band rows follow arXiv first-submission date.

WAM	Date	Substrate	Backbone	Action coupling	Deployment
UniPi [4]	2023.02	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
AVDC [6]	2023.10	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
VLP [5]	2023.10	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
GR-1 [48]	2023.12	Pixel (latent)	Autoregressive	Joint generation	Single-step
Dreamitate [59]	2024.06	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
This&That [61]	2024.07	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
GR-MG [62]	2024.08	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
Gen2Act [60]	2024.09	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
GR-2 [49]	2024.10	Pixel (latent)	Autoregressive	Joint generation	Chunked
PAD [50]	2024.11	Pixel (latent)	Diffusion	Joint generation	Single-step
CoT-VLA [88]	2025.03	Feature (VLM token)	LLM/VLM	Post-prediction head	Chunked
TesserAct [81]	2025.04	Geometric	Diffusion	Post-prediction head	Chunked
DreamGen [35]	2025.05	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
WorldVLA [52]	2025.06	Pixel (latent)	Autoregressive	Joint generation	Single-step
RoboEnvision [87]	2025.06	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
4DGen [36]	2025.07	Pixel (decoded) ∧ Geometric	Diffusion	Post-prediction head	Open-loop
RIGVid [37]	2025.07	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
Vidar [92]	2025.07	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
F1 [96]	2025.09	Pixel (latent)	Hybrid	Joint generation	Chunked
NovaFlow [38]	2025.10	Geometric	Diffusion	Post-prediction head	Open-loop
UD-VLA [73]	2025.11	Pixel (latent)	Diffusion	Joint generation	Chunked
RynnVLA-002 [71]	2025.11	Pixel (latent)	Hybrid	Joint generation	Chunked
VideoVLA [75]	2025.12	Pixel (latent)	Diffusion	Joint generation	Chunked
Motus [76]	2025.12	Pixel (latent)	Hybrid	Joint generation	Chunked
LVP [93]	2025.12	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
CoVAR [74]	2025.12	Pixel (latent)	Diffusion	Joint generation	Chunked
Dream2Flow [39]	2025.12	Geometric	Diffusion	Post-prediction head	Open-loop
TC-IDM [65]	2026.01	Geometric	Diffusion	Post-prediction head	Open-loop
BagelVLA [89]	2026.02	Pixel (latent)	Hybrid	Joint generation	Chunked
MVISTA-4D [82]	2026.02	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
Say-Dream-Act [91]	2026.02	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
Dex4D [84]	2026.02	Geometric	Hybrid	Post-prediction head	Single-step
DreamZero [53]	2026.02	Pixel (latent)	AR / Diffusion	Joint generation	Single-step
NovaPlan [69]	2026.02	Geometric	Hybrid	Post-prediction head	Chunked
PhysGen [72]	2026.03	Pixel (latent)	Autoregressive	Joint generation	Single-step

 doi:10.20944/preprints202606.1403.v1

Table 1. *Cont.*

WAM	Date	Substrate	Backbone	Action coupling	Deployment
EmboAlign [68]	2026.03	Pixel (decoded)	Diffusion	Post-prediction head	Open-loop
AeroPlace-Flow [70]	2026.03	Geometric	Diffusion	Post-prediction head	Open-loop
EVA [97]	2026.03	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
DriveDreamer-Policy [85]	2026.04	Pixel (decoded) ∧ Geometric	Hybrid	Joint generation	Chunked
MV-VDP [86]	2026.04	Pixel (decoded) ∧ Affordance	Diffusion	Joint generation	Chunked
DriveVA [78]	2026.04	Pixel (latent)	Diffusion	Joint generation	Chunked
Veo-Act [94]	2026.04	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
ActionImages [77]	2026.04	Pixel (decoded) ∧ Affordance	Diffusion	Joint generation	Chunked
GraspDreamer [66]	2026.04	Pixel (decoded) ∧ Geometric	Diffusion	Post-prediction head	Open-loop
3D-ALP [43]	2026.04	Pixel (decoded)	Hybrid	Cond. rollout	Chunked
VAG [95]	2026.04	Feature (tap)	Diffusion	Post-prediction head	Chunked
$\pi_{0.7}$ [14]	2026.04	Pixel (decoded)	Hybrid	Post-prediction head	Chunked
X-WAM [83]	2026.04	Pixel (latent)	Hybrid	Joint generation	Chunked
CKT-WAM [98]	2026.05	Pixel (latent)	Diffusion	Joint generation	Chunked
NoiseGate [80]	2026.05	Pixel (latent)	Hybrid	Joint generation	Chunked
HarmoWAM [79]	2026.05	Pixel (decoded / latent)	Hybrid	Post-prediction head	Chunked
DreamAvoid [44]	2026.05	Pixel (decoded)	Diffusion	Cond. rollout	Chunked
MoLA [64]	2026.05	Pixel (latent)	Diffusion	Post-prediction head	Chunked
CreFlow [63]	2026.05	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
Pelican-Unify 1.0 [99]	2026.05	Pixel (latent)	Hybrid	Joint generation	Chunked
SWEET [90]	2026.05	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
DriveWAM [55]	2026.05	Pixel (latent)	Hybrid	Joint generation	Chunked
VERA [67]	2026.05	Pixel (decoded)	Diffusion	Post-prediction head	Chunked
Latent-Only					
ARDuP [100]	2024.06	Pixel (latent)	Diffusion	Post-prediction head	Open-loop
Im2Flow2Act [111]	2024.07	Geometric	Diffusion	Post-prediction head	Chunked
VPP [7]	2024.12	Feature (tap)	Diffusion	Post-prediction head	Single-step
VILP [101]	2025.02	Pixel (latent)	Diffusion	Post-prediction head	Chunked
UVA [102]	2025.03	Pixel (latent)	Hybrid	Joint generation	Chunked
UWM [51]	2025.04	Pixel (latent)	Hybrid	Joint generation	Chunked
3DFlowAction [112]	2025.06	Geometric	Diffusion	Post-prediction head	Chunked
Video Policy [103]	2025.08	Feature (tap)	Diffusion	Post-prediction head	Chunked
Genie Envisioner [104]	2025.08	Feature (tap)	Diffusion	Post-prediction head	Chunked
3D-FDP [113]	2025.09	Geometric	Diffusion	Post-prediction head	Chunked
TraceGen [40]	2025.11	Geometric	Diffusion	Post-prediction head	Chunked
mimic-video [8]	2025.12	Feature (tap)	Diffusion	Post-prediction head	Chunked
Act2Goal [105]	2025.12	Feature (tap)	Diffusion	Post-prediction head	Chunked
CosmosPolicy [116]	2026.01	Pixel (latent)	Diffusion	Joint generation	Chunked
LingBot-VA [119]	2026.01	Pixel (latent)	Hybrid	Post-prediction head	Interactive
GigaBrain-0.5M* [120]	2026.02	Pixel (latent)	Hybrid	Post-prediction head	Chunked
AdaWorldPolicy [117]	2026.02	Pixel (latent)	Hybrid	Joint generation	Chunked
3PoinTr [41]	2026.03	Geometric	Hybrid	Post-prediction head	Chunked
DiT4DiT [107]	2026.03	Pixel (latent)	Diffusion	Post-prediction head	Chunked
S-VAM [9]	2026.03	Feature (tap)	Diffusion	Post-prediction head	Chunked
Fast-WAM [58]	2026.03	Feature (encoder-only)	Diffusion	Post-prediction head	Chunked
GigaWorld-Policy [108]	2026.03	Pixel (latent)	Diffusion	Joint generation	Chunked
OmniVTA [115]	2026.03	Pixel (latent)	Diffusion	Post-prediction head	Single-step
VAMPO [110]	2026.03	Feature (tap)	Diffusion	Post-prediction head	Chunked
VTAM [121]	2026.03	Pixel (latent)	Hybrid	Joint generation	Chunked
LaMP [109]	2026.03	Feature (tap)	Hybrid	Post-prediction head	Chunked
AIM [54]	2026.04	Affordance	Diffusion	Joint generation	Chunked
WAV [106]	2026.04	Feature (tap)	Diffusion	Post-prediction head	Chunked
CLWM [122]	2026.04	Feature (teacher)	Hybrid	Joint generation	Chunked
MWM [114]	2026.04	Affordance	Diffusion	Post-prediction head	Chunked
MotuBrain [118]	2026.04	Pixel (latent)	Hybrid	Joint generation	Chunked
FFDC-WAM [125]	2026.05	Pixel (latent)	Hybrid	Joint generation	Chunked
DAWN [123]	2026.05	Feature (teacher)	Hybrid	Joint generation	Chunked
EgoExo-WM [45]	2026.05	Feature (teacher)	Hybrid	Cond. rollout	Open / Chunked
RoboFlow4D [42]	2026.05	Geometric	Diffusion	Post-prediction head	Chunked
JOPAT [124]	2026.05	Pixel (latent) ∧ Geometric	Diffusion	Joint generation	Chunked
τ_0-WM [57]	2026.06	Pixel (latent)	Hybrid	Joint generation	Chunked
WALL-WM [56]	2026.06	Pixel (latent)	Hybrid	Joint generation	Chunked

 doi:10.20944/preprints202606.1403.v1

Table 2. Census of Video-Generation-Free World Action Models, the third design philosophy of Section 3. No video-generation backbone is in the predictive path. Columns match Table 1, locating each work on the design tuple of Equation 37, and rows follow arXiv first-submission date.

WAM	Date	Substrate	Backbone	Action coupling	Deployment
			Video-Generation-Free		
FLARE [126]	2025.05	Feature (teacher)	Joint-embedding	Post-prediction head	Chunked
DUST [10]	2025.10	Feature (VLM token)	Hybrid	Joint generation	Chunked
Audio-WM [11]	2025.12	Audio (latent)	Diffusion	Post-prediction head	Chunked
HiF-VLA [131]	2025.12	Geometric	LLM/VLM	Joint generation	Chunked
DexWM [128]	2025.12	Feature (teacher)	Hybrid	Post-prediction head	Chunked
PointWorld [46]	2026.01	Geometric	Hybrid	Cond. rollout	Open / Chunked
PALM [130]	2026.01	Affordance	LLM/VLM	Post-prediction head	Chunked
LDA-1B [12]	2026.02	Feature (teacher)	LLM/VLM	Post-prediction head	Chunked
FRAPPE [127]	2026.02	Feature (teacher)	Hybrid	Post-prediction head	Chunked
ICLR-VR [132]	2026.03	Geometric	LLM/VLM	Joint generation	Chunked
DDP [133]	2026.03	Feature (teacher)	Hybrid	Joint generation	Chunked
ALAM [129]	2026.05	Feature (VLM token)	LLM/VLM	Joint generation	Chunked
Feedback-WM [47]	2026.05	Feature (teacher)	Hybrid	Cond. rollout	Chunked

5. Core Properties of World Action Models

The four-axis anatomy of Section 4 explains how a WAM is assembled. This section asks a different question. Once that assembly is placed inside a control loop and made to run, what must remain true of it? We focus on five properties. A WAM must be **interactable**, so that control signals can shape the predicted future rather than only be decoded after generation. It must be **causal**, so that future information cannot leak into the action being executed. It must be **persistent**, so that the predicted state remains coherent as the robot acts, observes, and replans. It must be **physically plausible**, so that the future is realizable by the embodiment rather than only visually convincing. It must be **generalizable**, so that the same predictive-action contract remains useful when tasks, objects, scenes, cameras, or embodiments change. These properties are not independent. A design that improves one property often shifts cost, latency, memory, or error into another, which is why WAM design is best understood as a set of trade-offs among prediction quality, runtime feasibility, and control bandwidth.

5.1. Interactability

A video world model is only weakly interactable by default. Prompted to show a cup being lifted, it will render a plausible lift, but nothing ties that imagined motion to the trajectory the controller actually intends to send. The action is not yet the variable that governs the continuation. A WAM becomes interactable by deciding where the action enters the prediction loop. The factorization families in Equations 10, 11, and 12 lay out this design space. This subsection walks from post-prediction decoding, where action is recovered only after the future has been produced, through action-conditioned rollout, where proposed actions shape candidate futures, to joint prediction, where action and future are produced together.

Post-prediction decoding. In this form, the model synthesizes a whole future first and only then recovers control from it. UniPi, AVDC, and VLP establish this form by generating visual plans before extracting control [4–6]. Dreamitate, Gen2Act, and RoboEnvision extend the same post-prediction interface to tool use, human video, and long-horizon keyframes [59,60,87]. This end of the spectrum is clearest in cascaded WAMs: NovaFlow, Dream2Flow, and TC-IDM keep the future predictor separate, then convert generated motion into object flow, tool trajectories, or inverse-dynamics controls [38,39,65]. Audio-WM shows the same pattern outside vision, where generated future sound becomes the policy condition [11]. This interface is easy to attach to a pretrained video generator, but controllability arrives only after the full generation cost has been paid.

In-generation control. A more integrated interface pushes control into the generative process itself, and the works here line up as a near-monotone progression in how early the action binds. This&That starts the progression by conditioning video diffusion on spatial language and paired gesture coordinates, with ablations showing the gesture channel is essential for precise manipulation [61]. ARDuP

 doi:10.20944/preprints202606.1403.v1

narrows that conditioning to automatically discovered active regions and decodes action from latent representations rather than rendered pixels [100]. More integrated still, CoVAR and VAG run a dedicated action branch alongside video denoising and couple the two through bridge attention or step-synchronized denoising, passing only compact pooled video context to the action side [74,95]. UWM, UVA, F1, Motus, and Cosmos Policy drop the separate branch and share substrate prediction and action heads on a single backbone [51,76,96,102,116]. AdaWorld binds earliest of all, injecting learned latent actions at every denoising step [117,136]. Taken in order, the progression is a dial. The later the action binds, the cheaper and more modular the model, and the earlier it binds, the more the predicted future is genuinely shaped by the act it must inform.

Bottlenecked action pathways. The most bottlenecked designs restrict the information that the action pathway can consume. AIM exposes the future through a predicted spatial value map rather than raw future-pixel tokens [54]. GigaWorld-Policy makes action the primary prediction target and treats video as an optional action-conditioned by-product [108]. From the subgoal side, $\pi_{0.7}$ supplies sparse multi-view goal images to the policy context rather than a dense rollout [14]. Narrowing the channel this way is exactly what buys cheap, controllable inference, and it is also what caps the approach. The action can react only to whatever future variables survive the bottleneck.

5.2. Causality

Causality governs both correctness and latency. A standard video diffusion model denoises an entire clip jointly, so future frames can influence the representation that later drives the current action. This bidirectional pattern improves temporal smoothness, but it prevents the model from acting before imagination has finished. Worse, a model that quietly attends to the next frame can grasp flawlessly inside its own rollout and then close on empty air at inference, because the cue it leaned on was never available in real time. A WAM must instead produce predictions that depend only on past observations and the chosen action, and it must expose the relevant part of that prediction quickly enough for control.

Causal token streams. Causal streaming makes the ordering explicit: future and action are decoded as a token stream, and past computation is reused through a key-value cache [22,48,49,52,72]. PhysGen makes the mechanism explicit by combining causal masking, lookahead multi-token prediction, and key-value caching. It predicts several future action tokens at each step, while execution consumes only the leading token [72]. The gain comes from avoiding repeated computation over context that has already been fixed.

Leakage-free denoising. When prediction remains chunked, the key constraint is leakage control inside the chunk. Action tokens are forbidden from attending to future-pixel tokens or later actions, so the action head cannot benefit from information that would be unavailable at inference time [52,73,88]. WorldVLA introduced this mask to control error propagation in autoregressive action chunks, and the same mask also removes expensive attention between action tokens and future-pixel tokens [52]. CoT-VLA uses causal attention to generate a discrete visual chain-of-thought, then switches to full attention only for the short action sequence that reaches the goal [88].

Action-prioritized inference. Once leakage is controlled, latency is what is left to fight. Intermediate ODE checkpoints, single forward passes, and early denoising exits work because the controller usually needs the next action more than it needs a fully refined future video [8,9,58,73,107]. UD-VLA formalizes this point with synchronous joint discrete denoising and reports roughly fourfold faster inference than autoregressive counterparts [73]. The saved computation is mainly the tail of the denoising trajectory, where distant pixels are refined but the next action has already become stable.

The same prioritization also overlaps prediction with execution. LingBot-VA, MotuBrain, and DreamZero hide part of world-model latency behind the execution of the previous action chunk [53,118,119]. LingBot-VA pairs a Mixture-of-Transformers backbone, which gives shared and modality-specific tokens their own expert streams, with an asynchronous pipeline that runs action prediction and motor execution at the same time [119]. DreamZero pushes this overlap furthest, sustaining closed-loop control from a large autoregressive video-diffusion model through the cache-level re-grounding we

 doi:10.20944/preprints202606.1403.v1

examine under persistence in Section 5.3 [53]. Diffusion forcing supplies a related middle point: it assigns low noise to the near-certain past and high noise to the uncertain future under causal masking, so diffusion quality is retained where uncertainty remains [93]. Causal structure thus does two jobs at once. It blocks the future from leaking into the current action, and it decides where computation is worth spending, on the part of the rollout that can still move the next decision.

5.3. Persistence

Persistence is coherence under repeated action, observation, and replanning. It fails through drift, cost growth, and forgetting. Drift appears because each predicted step conditions the next, so small errors compound until the rollout leaves the data manifold. Cost grows when a model attends to full history, since memory and attention increase with the episode horizon. Forgetting appears when a finite context discards scene identity, so an occluded object can reappear in the wrong place. These failures pull against each other. Full-history attention fights drift and forgetting but raises cost. Sliding context bounds cost but discards old evidence.

Observation replacement. The cheapest persistence mechanism is re-grounding. After each executed chunk, the method replaces imagined observations with measured observations and updates the cache before forecasting again. DreamZero applies this idea through key-value-cache observation replacement and reaches 7 Hz closed-loop control with a 14B autoregressive video-diffusion model [53]. LingBot-VA follows the same principle with a closed-loop rollout that re-grounds on feedback before predicting the next chunk [119]. Re-grounding is cheap because it uses the robot's new observation rather than generating a longer imagined future.

Bounded memory. Re-grounding limits drift, but memory still needs a fixed budget. DexWM predicts future latent states from finger-keypoint actions and maintains bounded test-time memory instead of an unbounded cache, which lets it train and operate over 900 hours of human and robot interaction [128]. Act2Goal uses multi-scale temporal hashing to keep near-future frames dense while sparsifying distal frames, preserving long-horizon goal consistency at bounded cost [105]. LingBot-VA and MotuBrain add per-step cost control through Mixture-of-Transformers streams, which cap the cost of processing history even when a longer context is available [118,119].

5.4. Physical Plausibility

The video-world-model instinct equates a good future with a photorealistic one. A WAM needs something stricter. Its future exists only to constrain action, so what matters is whether the predicted dynamics are realizable by the body that will execute them, not whether they look convincing. Contact, force, and kinematics are what decide that, and three kinds of physical constraint push prediction toward it.

Visual-geometric abstraction ladder. Prediction targets form a ladder, and a WAM can choose how far down it to predict. At the top sits full RGB. One rung down, RGB-D, surface normals, and 4D point maps add explicit geometry to generated video, as in WAMs that reconstruct structure from RGB-D-Normal rollouts [81–83]. Below that, optical flow and 3D object flow keep only motion, the variable that transfers most cleanly across humans, robots, simulation, and reality. Im2Flow2Act and 3DFlowAction use flow to bridge domains and embodiments [111,112], and FlowDreamer carries the same abstraction into flow-conditioned generation [137]. Lower still, semantic masks discard photometric detail and keep only what the policy acts on. MWM makes the case for this rung directly, arguing that RGB prediction is misaligned with control because masks preserve the geometry while discarding the photometric nuisance [114].

At the bottom, the future stops being rendered at all. VPP and S-VAM decode action from predicted or distilled foresight features, S-VAM foreseeing geometric and semantic representations in a single forward pass rather than a video [7,9]. GigaWorld-Policy, Fast-WAM, WAV, and DexWM keep the future latent or action-centered at inference [58,106,108,128], and FLARE and LDA-1B push the target into frozen or structured embedding spaces [12,126]. The descent obeys a single logic. Each

 doi:10.20944/preprints202606.1403.v1

rung down trades away appearance the controller never needed and forces capacity toward geometry, contact, and dynamics, at the price of a future that is harder to inspect.

Richer physical modalities. Geometry narrows the target, but vision alone does not expose the variables that decide contact-rich manipulation, such as contact state, friction, slip, and normal force. A WAM that predicts only pixels is therefore fragile in the regime where manipulation is hardest. OmniVTA predicts future tactile contact states and pairs a slow policy with a high-rate reflexive controller that corrects discrepancies between predicted and observed tactile signals [115]. AdaWorldPolicy adds force prediction as a dedicated branch alongside the world model and the action expert, then uses force-torque mismatch for online adaptation [117]. DexWM predicts finger-level interaction rather than coarse end-effector motion [128], the same fine-grained bet that buys it bounded memory under persistence (Section 5.3). These modalities are low-dimensional, but they carry dense information exactly where contact determines success. They often offer a cheaper and more action-relevant target than the pixels of the same event.

Proprioceptive and kinematic-chain coherence. Contact signals still need to be grounded in the body that executes the action. A predicted future can look plausible and still be impossible for a particular robot. The arm may pass through itself, the end-effector may be unreachable, or the hand configuration may violate joint limits. Physical plausibility therefore also requires embodiment coherence. PAD, GR-1, Cosmos Policy, and X-WAM condition prediction on proprioception so that the forecast is tied to joint state [48,50,83,116]. Cosmos Policy goes further by treating future proprioception and image observations as jointly modeled latent frames [116]. DexWM adds a hand-consistency loss that penalizes kinematically inconsistent predictions [128]. MVISTA-4D uses trajectory-level inference with a residual inverse model to project an imagined future onto an action the embodiment can execute [82]. Each design shrinks the effective search space to the robot's feasible manifold.

These constraints all draw on physical priors learned from internet-scale video pretraining [26–28,30–32]. The main lesson is that how useful the imagined future is for the action predicts task success better than how good it looks [58,75,106,108]. A WAM should therefore predict the coarsest embodiment-grounded representation that still constrains the action, and flow, masks, tactile and force signals, and proprioceptively consistent latents often fill that role better than photorealism.

5.5. Generalization

Generalization in robotics is a bundle of shifts rather than a single axis: new tasks, objects, appearances, rooms, cameras, embodiments, and action spaces. Under the WAM contract, generalization requires both sides of the model to survive the shift. The predictive substrate must remain useful, and the action pathway must still execute the prediction. This is stricter than VLA-style semantic generalization and stricter than video generalization, because an action-relevant future is required in addition to a plausible instruction-to-action mapping or a plausible video.

Video-prior transfer. The broadest transfer source is the video prior itself. GR-1 and GR-2 show that large-scale language-conditioned video pretraining improves manipulation in unseen scenes and novel scenarios once action prediction is fine-tuned [48,49]. VideoVLA ties the prior directly to action by jointly denoising future frame latents and action tokens, improving novel-object and cross-embodiment performance while retaining the cost of multi-step diffusion sampling [75]. DreamZero makes a similar argument with an autoregressive video-action model and reports unseen-motion gains over VLA baselines, together with robot-to-robot and human-to-robot transfer from short video-only adaptation sets [53]. LVP keeps the connection cascaded: it trains a large video planner on human and robot clips, then extracts executable motion through pose estimation and retargeting for zero-shot real-robot tasks [93]. RoboEnvision targets task transfer by turning instructions into future keyframes that condition the downstream policy, so unseen tasks can still be routed through a visual subgoal interface [87]. $\pi_{0.7}$ uses generated futures as subgoal context rather than as the online action generator [14]. The common thread is that scale helps only once the future is connected to the action, and making that connection usually costs an inverse model, a subgoal generator, or a large video backbone.

 doi:10.20944/preprints202606.1403.v1

Substrate transfer. When pixels carry too much appearance-specific detail, transfer shifts to the substrate. Flow-based methods treat object motion as the transferable variable. Im2Flow2Act uses object flow to bridge human-to-robot and simulation-to-real gaps before a flow-conditioned policy acts [111]. 3DFlowAction lifts the same idea into 3D object flow, separating object dynamics from the robot that later executes the trajectory [112]. What flow buys is suppression of texture, camera style, and part of the embodiment gap, but it leaves the hard part to the executor, which still has to solve grasping, collision avoidance, and contact.

Mask, latent, and feature substrates make the same trade-off at a deeper level. MWM replaces RGB prediction with semantic-mask dynamics, improving robustness to background, lighting, and object-color shifts [114]. S-VAM distills a diffusion backbone into geometric and semantic foresight features, so inference no longer depends on full video denoising [9]. Fast-WAM and GigaWorld-Policy keep video prediction as a training signal but remove future-video generation from part or all of inference [58,108]. The deepest version of the move forecasts in feature space rather than pixels. FRAPPE and LDA-1B predict in self-supervised vision-foundation features, where appearance variation washes out and what survives the shift is the action-relevant structure [12,127]. These substrates make generalization less dependent on photorealistic rendering, but they also make the predicted future harder to inspect and harder to evaluate with video metrics.

Action-abstraction transfer. Substrate transfer still needs an executor, so another line of work transfers the action abstraction and the data recipe. LDA-1B scales this separation with universal embodied data ingestion, a hand-centric action space, and DINO latent forecasting across high-quality demonstrations, low-quality trajectories, and actionless videos [12]. DUST shows the diffusion-side version of the point: action-free BridgeV2 video pretraining improves a dual-stream policy even when future observations are predicted only in an embedding space [10]. ALAM adds an algebraic consistency constraint over latent action transitions, so the abstract action space keeps a predictable composition law while the action expert remains tied to the embodiment [129]. The shared move is to separate what should transfer from what should stay local. The substrate carries task and environment dynamics. The action decoder absorbs embodiment-specific kinematics. The data recipe supplies enough variation that neither side memorizes the training platform.

What falls out in practice is a single principle. Predict at the invariant level that still constrains control. Pixels transfer semantic and physical priors but carry appearance cost. Flow and masks transfer object motion and geometry but require a reliable executor. Latent actions and feature forecasts scale to action-free video and heterogeneous data, but they need careful grounding before they become precise control. No design dominates across all shifts. A WAM that claims generalization should therefore state which shift it targets, which design axis is meant to transfer, and which adapter remains specific to the robot that executes the action. This framing turns generalization from a broad claim into a checkable property.

Across the five properties, one lesson outweighs the rest. They are not a checklist but a set of competing pressures. The stronger action interface that makes a model interactable also narrows the channel that causality must keep leak-free. The bounded memory that buys persistence is the same budget that starves long-horizon plausibility. The abstraction that lets a substrate transfer is what makes its predicted future hard to evaluate. A WAM is therefore never tuned for one property alone. Every design sits at a point where a gain in one property is paid for in another, and turning that balance into something measurable, rather than asserted, is the task we take up in Section 6.

6. Data and Evaluation

A WAM can only preserve the properties of Section 5 when its training data and evaluation protocol expose those properties. Data determines which future variables the model can learn, which action labels it can trust, and which embodiments it can transfer across. Evaluation determines whether those learned variables are useful in a control loop rather than only plausible as video. The two questions are therefore coupled. This section first organizes WAM data by the trade-off between

 doi:10.20944/preprints202606.1403.v1

scale, action-label quality, and embodiment match. It then organizes evaluation by the trade-off between visual fidelity, closed-loop success, physical plausibility, and evaluation cost.

6.1. Data Sources

WAM training data falls into five groups. Each group provides a different compromise between scale, action-label fidelity, physical grounding, and access.

Robot teleoperation.

Robot teleoperation gives the cleanest action-conditioned trajectories because a human operator drives the embodiment and the command stream is recorded with the visual observation. Open X-Embodiment [144] aggregates this recipe across laboratories and robot platforms, making it the canonical cross-embodiment resource for robot policy training. RoboMIND [145], RoboSet [146], RoboTurk [147], and Robo360 [148] extend the same supervision pattern with different platform mixes, task suites, and camera configurations. Human2Robot [149] pairs each teleoperated trajectory with a synchronized video of a bare human hand performing the same task, adding a human-to-robot correspondence on top of the exact action labels. RoboNet [150] departs from human teleoperation by collecting real-robot trajectories through scripted and random policies, trading demonstration precision for throughput while still logging exact action commands. The benefit of this group is label fidelity. The cost is that every additional hour consumes robot time, operator time, or both.

Portable human demonstrations.

Portable demonstrations increase throughput by moving data collection from a robot platform to a human-worn or handheld capture setup. EgoMimic [151] records egocentric video and three-dimensional hand tracks with a lightweight wearable rig. EgoVerse [152] scales the same idea with wearable and phone-based capture, then retargets the demonstrations across robot embodiments. EgoDex [153] carries the head-worn route to large scale, recording dexterous bimanual hand-pose tracks on Apple Vision Pro with no robot in the loop. This group buys scale because human demonstrations are faster to collect than robot teleoperation. It pays for that scale with an embodiment gap that must be closed before the data can train a robot policy.

Internet-scale egocentric and instructional video.

Internet video gives WAMs their scale, but it usually lacks robot actions. Ego4D [154], Ego-Exo4D [155], and EgoPAT3D [156] emphasize first-person activity and hand motion. EPIC-KITCHENS [157], HD-EPIC [158], EgoVid-5M [159], OpenEgo [160], IndEgo [161], EgoScale [162], and Egocentric-10k [**?**] broaden the pool with instructional, household, and large-scale egocentric video. This group is valuable because it exposes broad visual variation, object dynamics, and human manipulation patterns. Its limitation is the missing action channel. One route uses such video, together with broader internet video, to pretrain a video backbone, as in V-JEPA 2 [21] and Wan [26]. Another route attaches an inverse-dynamics or latent-action model that recovers a control proxy from video. AVDC [6] recovers flow-conditioned action from unlabeled video, while LDA-1B [12] and DUST [10] use action-free video to strengthen latent future prediction before action decoding. Internet video is therefore a scale source only after the action-label problem has been made explicit.

Simulation.

Simulation provides exact action labels, controlled curricula, and low marginal cost after the environment has been authored. Classical physics platforms for manipulation include Robosuite [163], ManiSkill 2 [164], ManiSkill 3 [165], MetaWorld [166], and LIBERO [167]. General simulation backends such as IsaacSim [168], CoppeliaSim [169], and SimplerEnv [170] provide the execution substrate for broader embodied tasks. Household and digital-twin datasets build on these engines to generate demonstrations programmatically, including RoboCasa [171], RoboTwin [172], and RoboTwin 2 [173]. RoboCerebra [174] instead collects human-teleoperated trajectories inside the simulator for long-

 doi:10.20944/preprints202606.1403.v1

horizon evaluation. Aggregators sit above both lines: RoboVerse [175] unifies several simulator backends, and RoboData [176] merges simulation benchmarks with a real-robot set. The benefit is scalable label quality. The limitation is the sim-to-real gap, which a WAM often tries to close with an internet-pretrained visual prior or a more abstract substrate.

Synthetic data from WAMs themselves.

A WAM can also become a data engine. IRASim [177] follows this pattern from the simulator side by replacing a physics engine with a learned action-conditioned diffusion model whose rollouts stand in for simulated trajectories. Cosmos-Transfer1 [135] and InteractiveWorldSimulator [25] support similar data-generation use, and Sora-class video models are sometimes used to seed downstream policies [30,31]. DreamGen [35] makes the cascaded WAM version concrete by generating robot videos, recovering pseudo-actions or latent actions, and training the final policy from those neural trajectories. RIGVid [37] and GraspDreamer [66] use generated demonstrations in a more direct action-recovery path, where tracking and retargeting replace manual action labels. Synthetic neural trajectories sit between simulation and real teleoperation. They inherit the realism of internet pretraining, but they also inherit the generator's failure modes.

In practice, WAM training mixes these sources. Internet video supplies visual and physical priors, teleoperation supplies trusted action labels, portable human data adds scale, simulation supplies controllable coverage, and synthetic trajectories fill gaps that are expensive to collect physically. The unresolved question is how to choose the mixture for a target deployment regime. Access is a second constraint. Sora-class generators [30,31] and the EgoScale release [162] are closed or not yet public, which limits independent verification of their contribution.

6.2. Evaluation

Evaluation must judge both prediction and action. Video-generation metrics ask whether the future looks plausible. Robot-learning benchmarks ask whether the policy succeeds in the loop. A WAM needs both perspectives, but neither is sufficient on its own. A useful protocol must also ask whether the prediction is physically realizable, whether coherence survives long horizons, and what compute, memory, and latency are required to obtain the result.

Visual fidelity metrics.

The video-generation lineage contributes FVD [178], FID, LPIPS [179], PSNR, SSIM [180], and DreamSim [181]. These metrics are cheap and familiar, but they reward visual realism rather than action utility. A crisp rollout can still be useless when contact, geometry, or proprioception is wrong. A visually sparse representation can still drive a successful policy when it captures the variables needed for control. Fast-WAM [58] and GigaWorld-Policy [108] bear this out by dropping the rendered forecast at inference without losing control, while WorldScore [182] and WorldSimBench [183] fold visual fidelity into broader assessments that pair perceptual scores against closed-loop action success.

Closed-loop benchmarks.

Closed-loop evaluation measures whether the policy actually completes the task. These benchmarks differ mainly in where the rollout comes from, which sets a throughput-versus-validity trade-off. Simulation sits at the high-throughput end, where benchmarks probe different competences. The LIBERO family [167,184–186] and RoboMME [187] focus on language-conditioned manipulation and multimodal evaluation. robomimic [188], MetaWorld [166], ManiSkill 2 [164], and ManiSkill 3 [165] provide manipulation suites with different levels of control diversity and physics coverage. HomeRobot [189], VLABench [190], RoboEval [191], and RoboVerse [175] broaden the benchmark space toward navigation, embodied reasoning, and multi-backend evaluation. Real-robot arenas such as RoboArena [192] and RoboChallenge [193] trade throughput for physical validity by running the policy on hardware. DWorldEval [194] closes the loop inside a learned world model instead, scoring a policy from imagined rollouts whose success correlates with real execution. Cascaded WAMs add

 doi:10.20944/preprints202606.1403.v1

another evaluation layer because the generated future and the action translator can each fail separately. Video-to-flow and video-to-trajectory methods therefore need both substrate checks and execution checks, as illustrated by NovaFlow, Dream2Flow, TraceGen, and RoboFlow4D [38–40,42]. Closed-loop evaluation measures the right outcome, but every trial consumes a robot, a simulator, or a learned generator for the duration of the task.

Physical plausibility and long-horizon coherence.

Some WAM properties are visible in a rollout but poorly captured by a single success number. Physical plausibility can be evaluated through tactile and force prediction errors, as in OmniVTA [115] and AdaWorldPolicy [117], or through kinematic and geometry-consistency checks, as in DexWM [128] and MVISTA-4D [82]. A standard metric has not yet emerged. Long-horizon coherence is similarly under-specified. Reported numbers usually reduce to task success after a fixed number of replans, and current benchmarks do not test hour-scale rollout at control frequency. RoboScape [138] brings physics signals into evaluation, while PhysWorld [195] points to the broader problem of judging physical world models.

What current evaluation misses.

Current evaluation has a predictable ordering. Visual-fidelity metrics are cheap but weakly tied to downstream success. Closed-loop simulation is more predictive but consumes task-time rollouts. Real-robot arenas are physically valid but expensive to repeat. Physical-plausibility and long-horizon metrics remain underdeveloped. The same ordering appears in data collection. A teleoperation hour costs a robot and an operator, a simulator hour costs compute, an internet-video hour costs curation, and a learned-model rollout costs a forward pass. This is why proxy evaluators are attractive despite depending on the generator's own fidelity. The WAMs that matter for real use are also the hardest to evaluate, because chunked and interactive closed-loop control require longer horizons, stricter latency constraints, and real-environment variation. The field is therefore moving toward a two-stage protocol: inexpensive visual and representation screens, followed by selective closed-loop tests under explicit compute, memory, and latency budgets. The conversion factor between the cheap screen and the expensive test remains the central open question in WAM evaluation, and it motivates the frontiers in Section 7.

7. Open Challenges

Sections 4 through 6 define the design axes, core properties, data sources, and evaluation practice of current World Action Models. The open challenges below use those axes to separate problems that are often discussed together. The boundaries are analytical rather than absolute: data affects generalization, action abstraction affects physics, and inference cost affects almost every design choice. Still, each subsection asks a different question. We first ask how much future computation a WAM should build into its design or spend at runtime, then ask which data should train each stage. We then move through memory, generalization, action grounding, physical plausibility, and evaluation.

7.1. Dream More or Act More?

Prediction helps action only when its latency fits the decision loop. Richer futures, larger backbones, and deeper denoising can improve the substrate available to the action side, but they also lengthen the path from observation to control. Current WAMs therefore split between computing a detailed future and extracting just enough future-trained structure for action. S-VAM distills video diffusion into geometric and semantic foresight features, Fast-WAM removes the future-video branch at inference, GigaWorld-Policy masks attention so future video can be skipped, and X-WAM lets action denoising finish before video denoising completes [9,58,83,108]. These methods suggest that the video objective can be more important than a fully rendered test-time forecast.

The other side keeps stronger imagination in the control path and pays for it in latency. DreamZero brings a large video backbone into closed-loop control through a custom optimization stack, Cosmo-

 doi:10.20944/preprints202606.1403.v1

sPolicy uses repeated model and value queries for planning, and NovaPlan places video generation before flow extraction and geometric grounding [53,69,116]. The open problem is not simply to make every model smaller. It is to expose a controllable fidelity-latency curve, so a WAM can choose how much future information the action actually needs.

That choice should also become a runtime decision. A video policy may run the same denoising schedule at every step, and a latent planner may fix its horizon and search depth before it knows whether the next state is routine transit or contact-sensitive manipulation [16,75]. Coarse switches help but remain limited. A forecast-confidence method can choose when to replan, yet a binary execute-or-replan decision cannot decide how much of the remaining horizon should be regenerated [125]. DreamAvoid, SANTS, NoiseGate, and HarmoWAM show that the trigger can be learned from task phase, denoising state, or process cues [44,79,80,196]. The open problem is to spend generative compute by expected action value: more when uncertainty, contact, or irreversible error is near, and less when the next action is routine.

This budget question also clarifies the relation between video world models and WAMs. Video generation has pushed toward longer and more detailed futures, while WAMs increasingly try to compute only the future information needed for action. The two lines now meet at the substrate level of Section 4.2: video generators add action conditioning, and WAMs borrow latent or joint-embedding substrates from video backbones [25,28,99,135,197]. The practical obstacle is that a video-scale model reaches the control rate only after substantial optimization, while skipping video tokens recovers speed by giving up the rendered forecast [53,58,108]. A stronger design would train one backbone with video objectives, then let the action path choose how much of that backbone to run at inference.

7.2. What Data Should Each Stage Learn from?

Section 6 organizes WAM data by source, but the harder question is where each source belongs in training. A WAM usually needs visual dynamics, embodiment alignment, executable action, and sometimes a post-training controller or scheduler. These variables are not learned from the same evidence. Internet video can teach broad visual regularities, egocentric or third-person human demonstrations can align motion with intent, robot teleoperation can ground the final action decoder, and rollout data can train gates or budget controllers. Pooling these sources hides the variable each source supervises. The open challenge is to assign each data source to a stage, an objective, and a model component.

The pretraining stage is the clearest starting point and also the easiest stage to overgeneralize. GR-2 and DreamDojo show the appeal of broad video pretraining before robot-specific adaptation, while VPP shows that a video prior can be decisive for downstream action when the robot data alone is too narrow [7,49,198]. Yet robot-domain video and general human video are not interchangeable. VidMan reports that robot-domain pretraining helps while a mismatched egocentric source can hurt the same downstream setting, and VLA-JEPA finds that human-video gains depend on the evaluation split rather than appearing uniformly [134,199]. The question is therefore not whether video pretraining helps. The question is which kind of video teaches the dynamics that the later action path will need.

The alignment stage is less settled. Egocentric human data, paired human-robot demonstrations, and wearable capture can bridge the gap between passive video and robot control, but each format carries a different bias. EgoScale separates broad human pretraining from an aligned mid-training stage, EgoMimic uses first-person human demonstrations to improve robot imitation, and EgoVerse finds that scene diversity can matter more than raw volume [151,152,162]. FRAPPE makes the same point from a different angle: web-scale egocentric data, task-specific egocentric data, and robot data play distinct roles in transfer [127]. These results argue for a staged curriculum, but they do not yet tell us the minimum capture hardware, viewpoint match, or data quality needed for each embodiment.

The final action stage is where the data bottleneck becomes sharpest. Cross-platform corpora and simulation benches provide a practical base for action supervision, but their action spaces remain uneven across arms, hands, mobile bases, and deformable-object tasks [144,145,171,173–175]. Action-free recipes try to reduce this bottleneck by discovering latent actions, flow actions, or structured

 doi:10.20944/preprints202606.1403.v1

transition codes from video, as in LAPA, DUST, ALAM, and LDA-1B [10,12,129,200]. These recipes reduce the amount of labeled robot control needed, but they do not remove it. Every abstract action still needs a decoder, an inverse-dynamics model, or a robot-specific calibration step before it can become executable control.

Synthetic trajectories add a further stage rather than a complete replacement for action labels. DreamGen uses generated futures to produce neural trajectories, and physical-world-model pipelines can turn video generation into policy training data, but both still inherit generator errors and still need grounding in robot dynamics [35,195]. Human2Robot shows the other side of the same problem: paired human-robot data can be valuable, but contact-rich skills remain hard to capture even with specialized teleoperation hardware [149]. The missing result is a joint scaling law over source, stage, quality, model size, and action grounding. Without such a law, adding more data remains an engineering choice rather than a predictable design decision.

7.3. Can Memory Keep up?

Section 5 catalogs mechanisms for persistence, including observation replacement, bounded memory, and spatially indexed state. What remains open is not simply a longer prediction horizon. A useful WAM must maintain task state while actions are being produced, and its memory cost should grow with scene complexity rather than with episode length. Autoregressive prediction exposes the failure mode directly: IRASim accumulates temporal drift despite overlapping context, PhysGen is bounded by a fixed context of physical packages, and sparse-memory chunking still degrades once the scene becomes dynamic [24,72,177]. These are memory failures, not only prediction-quality failures.

Promising ingredients exist in separate places. A test-time memory can keep per-step recall bounded, but its update remains sensitive enough to risk forgetting earlier state [122]. Dreamer 4 shows that compact latent state can support continual imagination under closed-loop control, but the abstraction leaves open how to preserve object-level spatial detail across long manipulation episodes [19]. The challenge is to combine bounded memory, spatial indexing, and observation replacement into one method that remains reactive over long tasks.

7.4. How Can WAMs Generalize?

Section 5 treats generalization as a family of shifts, not as one scalar property. A WAM therefore cannot be made generalizable by adding scale alone. The target shift has to be named first, because visual appearance, object dynamics, morphology, action space, camera placement, and contact regime stress different parts of the model. Domain randomization remains a sharp test of this separation. Act2Goal drops steeply under harder randomized conditions, and FRAPPE remains far below a practical rate even when it leads the compared baselines [105,127]. Transfer to new motions and objects is also bounded: a video-conditioned policy improves over baselines, but the hardest motion and object splits still leave a large gap [60]. The lesson is that future prediction helps only when the predicted substrate matches the shift that the action path must absorb.

The path toward generalizable WAMs is to match data, substrate, and adaptation to the shift being targeted. Cross-embodiment transfer is the clearest case, because the action space itself changes. DreamZero reports transfer from short video-only adaptation sets, but the need for adaptation shows that morphology shift is not solved by scale alone [53]. Removing video pretraining from a strong policy sharply reduces task completion, which shows that the borrowed video prior matters, but it does not explain which embodiment changes the prior can absorb [7]. EgoScale offers one positive sign through a clean scaling law for dexterous pretraining volume [162]. The open challenge is to design for a declared shift before training, then test whether the chosen substrate and action decoder generalize across that shift rather than reporting transfer after the fact.

7.5. What Grounds Abstract Action?

The action-coupling axis of Section 4 often replaces raw control with an abstract target such as a latent code, a flow field, a point track, or a structured transition. This substitution is useful because it

 doi:10.20944/preprints202606.1403.v1

lets a method learn from action-free or weakly labeled video. It is also risky because the action can lose a physical handle. One failure mode is estimator dependence. Two-dimensional object flow cannot represent depth translation, out-of-plane rotation, or contact force, and a rigid-transform decoder breaks when the object is articulated or deformable [6,111]. Point-track and flow-based adapters make the action interface inspectable, but they inherit errors from trackers, depth estimators, and local linearity assumptions [41,67,124].

The second failure mode is opacity. LAPA learns discrete latent actions from video and improves action-free pretraining, but fine grasping remains difficult when the codebook is too coarse [200]. ALAM adds algebraic consistency to latent transitions, yet the resulting structure is still a composition law rather than a force, torque, or contact explanation [129]. LDA-1B learns a broad latent action space, but execution still depends on the decoder that maps that space back to an embodiment-specific command [12]. The open challenge is to keep the data advantage of abstract actions while adding a physical grounding signal and an inspection handle for failure analysis.

7.6. When Is a Future Physical?

Section 5 frames physical plausibility through visual-geometric abstraction, tactile and force prediction, and proprioceptive or kinematic coherence. The difficulty is that a future can look plausible while failing the embodiment. Momentum is a simple example. A flow-based world model conditioned on one frame cannot encode object velocity, so push tasks fail when the object keeps moving after contact ends [137]. Contact and force are harder still. A tactile world model derives a force proxy from tactile-flow divergence, but the proxy is uncalibrated, and a point-cloud substrate omits force magnitude, friction, and surface compliance [46,121]. Even an executability reward can miss contact if it only scores smoothness and joint limits [97].

Some methods point toward a better standard by making physics part of the substrate or loss. RoboScape and RoboScape-R add geometric and consistency constraints, OmniVTA and DexWM bring tactile or dexterous state into the prediction problem, and AdaWorldPolicy includes action-conditioned future structure in its training objective [115,117,128,138,201]. These examples remain partial. The open challenge is to train and evaluate futures for being executable by the embodiment, not only visually convincing.

7.7. What Should Evaluation Report?

Section 6 describes visual screens, simulator tests, and closed-loop robot evaluation. The open challenge is to report the variables that make these tiers comparable. World-model quality is not yet a reliable proxy for policy utility, as MotuBrain finds only a weak correlation between its world-model score and downstream success [118]. A single success number can also hide opposite regimes. HarmoWAM shows that a schedule useful for transit can fail at precise interaction, which means that averaging the phases removes the failure mode the evaluation should expose [79].

The missing protocol is an accuracy-at-budget report. A WAM that succeeds slowly is not equivalent to one that acts at the control rate, and a WAM that succeeds over a short horizon is not equivalent to one that keeps state over a long task. Some papers report latency, but often apart from accuracy and on different hardware or task settings [5,74,86,107,202]. Better examples are emerging: Fast-WAM isolates the value of test-time video generation once video co-training is preserved, while SANTS and NoiseGate report success together with learned reductions in denoising cost [58,80,196]. Evaluation should place success, latency, sustained horizon, peak memory, and contact-sensitive failure tags on the same axis.

7.8. The Challenges Are Coupled

These challenges are coupled even when the boundaries above separate their main questions. A memory fix that deepens search raises the compute budget. A richer substrate can improve contact realism while making evaluation and inference more expensive. Latent actions reduce the need for labeled robot control, but the same abstraction weakens physical grounding unless the decoder is

 doi:10.20944/preprints202606.1403.v1

calibrated. Contact-aware prediction requires force, tactile, or failure data that cheap visual screens do not provide. Upstream estimator errors can appear as both grounding failures and transfer failures. The shared lesson is that a WAM should be advanced as one coupled design space, where progress on one challenge changes the practical trade-offs of the others.

8. Conclusion

This survey has treated World Action Models as a contract between prediction and control: given past observations, past actions, and a task context, a WAM must anticipate the actionable future while producing the action that should follow. This framing separates WAMs from broad world models, video generation models, and plain VLAs, and it explains why works with different names can still belong to the same family. We organized existing works through two complementary views. The philosophy-level view asks what a method is required to generate before action is decoded, while the component-level anatomy locates the same method by predictive substrate, backbone, action coupling, and deployment regime. These views then support the core properties that matter after a model enters a control loop: interactability, causality, persistence, physical plausibility, and generalization. They also show why data and evaluation are not secondary details, since a benchmark can only validate the properties it exposes and a dataset can only teach the action variables it records or infers. Across the works considered here, the common trajectory is not simply toward larger video prediction, but toward more selective imagination that keeps the parts of the future needed for action and discards the rest. The remaining challenges therefore lie in choosing the right substrate, coupling action without leakage, preserving long-horizon state, grounding predictions in physical constraints, and reporting capability together with compute, memory, and latency. By giving these choices a shared vocabulary, this survey aims to make future WAMs easier to compare and to help the field generate less of the future while preserving more of what embodied action requires.

References

1. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Chen, X.; Choromanski, K.; Ding, T.; Driess, D.; Dubey, A.; Finn, C.; et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023, [arXiv:cs.RO/2307.15818].
2. Kim, M.J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; et al. OpenVLA: An Open-Source Vision-Language-Action Model, 2024, [arXiv:cs.RO/2406.09246].
3. Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; et al. π_0: A Vision-Language-Action Flow Model for General Robot Control, 2024, [arXiv:cs.RO/2410.24164].
4. Du, Y.; Yang, M.; Dai, B.; Dai, H.; Nachum, O.; Tenenbaum, J.B.; Schuurmans, D.; Abbeel, P. Learning Universal Policies via Text-Guided Video Generation. In Proceedings of the Advances in Neural Information Processing Systems, 2023, [2302.00111].
5. Du, Y.; Yang, M.; Florence, P.; Xia, F.; Wahid, A.; Ichter, B.; Sermanet, P.; Yu, T.; Abbeel, P.; Tenenbaum, J.B.; et al. Video Language Planning. In Proceedings of the International Conference on Learning Representations, 2024.
6. Ko, P.C.; Mao, J.; Du, Y.; Sun, S.H.; Tenenbaum, J.B. Learning to Act from Actionless Videos through Dense Correspondences. In Proceedings of the International Conference on Learning Representations, 2024, [2310.08576].
7. Hu, Y.; Guo, Y.; Wang, P.; Chen, X.; Wang, Y.J.; Zhang, J.; Sreenath, K.; Lu, C.; Chen, J. Video Prediction Policy: A Generalist Robot Policy with Predictive Visual Representations. *arXiv preprint arXiv:2412.14803* **2024**.
8. Pai, J.; Achenbach, L.; Montesinos, V.; Forrai, B.; Mees, O.; Nava, E. mimic-video: Video-Action Models for Generalizable Robot Control Beyond VLAs. *arXiv preprint arXiv:2512.15692* **2025**.
9. Yan, H.; Zhong, Z.; Zhu, J.; He, J.; Yuan, W.; Song, W.; Gong, X.; Cai, Y.; Zhao, G.; Yan, X.; et al. S-VAM: Shortcut Video-Action Model by Self-Distilling Geometric and Semantic Foresight, 2026, [arXiv:cs.CV/2603.16195].
10. Won, J.; Lee, K.; Jang, H.; Kim, D.; Shin, J. Dual-Stream Diffusion for World-Model Augmented Vision-Language-Action Model, 2025, [2510.27607].
11. Zhang, F.; Gienger, M. Learning Robot Manipulation from Audio World Models, 2025, [2512.08405].

 doi:10.20944/preprints202606.1403.v1

12. Lyu, J.; Liu, K.; Zhang, X.; Liao, H.; Feng, Y.; Zhu, W.; Shen, T.; Chen, J.; Zhang, J.; Dong, Y.; et al. LDA-1B: Scaling Latent Dynamics Action Model via Universal Embodied Data Ingestion, 2026, [2602.12215].
13. Intelligence, P.; Black, K.; Brown, N.; Darpinian, J.; Dhabalia, K.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization, 2025, [arXiv:cs.LG/2504.16054].
14. Intelligence, P.; Ai, B.; Amin, A.; Aniceto, R.; Balakrishna, A.; Balke, G.; Black, K.; Bokinsky, G.; Cao, S.; Charbonnier, T.; et al. $\pi_{0.7}$: a Steerable Generalist Robotic Foundation Model with Emergent Capabilities, 2026, [arXiv:cs.LG/2604.15483].
15. Hou, B.; Li, G.; Jia, J.; An, T.; Guo, X.; Leng, S.; Geng, H.; Ze, Y.; Harada, T.; Torr, P.; et al. World Model for Robot Learning: A Comprehensive Survey, 2026, [arXiv:cs.RO/2605.00080].
16. Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; Davidson, J. Learning Latent Dynamics for Planning from Pixels, 2019, [arXiv:cs.LG/1811.04551].
17. Hafner, D.; Pasukonis, J.; Ba, J.; Lillicrap, T. Mastering Diverse Domains through World Models, 2023, [arXiv:cs.AI/2301.04104].
18. Chen, C.; Wu, Y.F.; Yoon, J.; Ahn, S. TransDreamer: Reinforcement Learning with Transformer World Models, 2022, [arXiv:cs.LG/2202.09481].
19. Hafner, D.; Yan, W.; Lillicrap, T. Training Agents Inside of Scalable World Models, 2025, [arXiv:cs.AI/2509.24527].
20. Bardes, A.; Garrido, Q.; Ponce, J.; Chen, X.; Rabbat, M.; LeCun, Y.; Assran, M.; Ballas, N. Revisiting Feature Prediction for Learning Visual Representations from Video, 2024, [arXiv:cs.CV/2404.08471].
21. Assran, M.; Bardes, A.; Fan, D.; Garrido, Q.; Howes, R.; Komeili, M.; Muckley, M.; Rizvi, A.; Roberts, C.; Sinha, K.; et al. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning, 2025, [arXiv:cs.AI/2506.09985].
22. Wu, J.; Yin, S.; Feng, N.; He, X.; Li, D.; Hao, J.; Long, M. iVideoGPT: Interactive VideoGPTs are Scalable World Models. In Proceedings of the Advances in Neural Information Processing Systems, 2024, [arXiv:cs.CV/2405.15223].
23. Zhou, S.; Du, Y.; Chen, J.; Li, Y.; Yeung, D.Y.; Gan, C. RoboDreamer: Learning Compositional World Models for Robot Imagination, 2024, [arXiv:cs.RO/2404.12377].
24. Huang, S.; Chen, L.; Zhou, P.; Chen, S.; Jiang, Z.; Hu, Y.; Liao, Y.; Gao, P.; Li, H.; Yao, M.; et al. EnerVerse: Envisioning Embodied Future Space for Robotics Manipulation, 2025, [arXiv:cs.RO/2501.01895].
25. Wang, Y.; Syed, R.; Wu, F.; Zhang, M.; Onol, A.; Barreiros, J.; Nayyeri, H.; Dear, T.; Zhang, H.; Li, Y. Interactive World Simulator for Robot Policy Training and Evaluation, 2026, [arXiv:cs.RO/2603.08546].
26. Wan, T.; Wang, A.; Ai, B.; Wen, B.; Mao, C.; Xie, C.W.; Chen, D.; Yu, F.; Zhao, H.; Yang, J.; et al. Wan: Open and Advanced Large-Scale Video Generative Models, 2025, [arXiv:cs.CV/2503.20314].
27. Yang, Z.; Teng, J.; Zheng, W.; Ding, M.; Huang, S.; Xu, J.; Yang, Y.; Hong, W.; Zhang, X.; Feng, G.; et al. CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer, 2025, [arXiv:cs.CV/2408.06072].
28. NVIDIA. Cosmos-Predict2: World Simulation Model for Physical AI, 2025.
29. Deng, H.; Pan, T.; Diao, H.; Luo, Z.; Cui, Y.; Lu, H.; Shan, S.; Qi, Y.; Wang, X. Autoregressive Video Generation without Vector Quantization, 2024, [2412.14169].
30. OpenAI. Video generation models as world simulators. https://openai.com/index/video-generation-models-as-world-simulators/, 2024. Accessed: 2026-03-31.
31. OpenAI. Sora 2. https://openai.com/sora, 2025. Accessed: 2026-04-08.
32. Ma, X.; Wang, Y.; Chen, X.; Jia, G.; Liu, Z.; Li, Y.F.; Chen, C.; Qiao, Y. Latte: Latent Diffusion Transformer for Video Generation, 2024, [arXiv:cs.CV/2401.03048].
33. Guo, Y.; Yang, C.; Rao, A.; Liang, Z.; Wang, Y.; Qiao, Y.; Agrawala, M.; Lin, D.; Dai, B. AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning, 2024, [arXiv:cs.CV/2307.04725].
34. Kondratyuk, D.; Yu, L.; Gu, X.; Lezama, J.; Huang, J.; Schindler, G.; Hornung, R.; Birodkar, V.; Yan, J.; Chiu, M.C.; et al. VideoPoet: A Large Language Model for Zero-Shot Video Generation, 2024, [arXiv:cs.CV/2312.14125].
35. Jang, J.; Ye, S.; Lin, Z.; Xiang, J.; Bjorck, J.; Fang, Y.; Hu, F.; Huang, S.; Kundalia, K.; Lin, Y.C.; et al. DreamGen: Unlocking Generalization in Robot Learning through Video World Models, 2025, [arXiv:cs.RO/2505.12705].
36. Liu, Z.; Li, S.; Cousineau, E.; Feng, S.; Burchfiel, B.; Song, S. Geometry-aware 4D Video Generation for Robot Manipulation, 2025.
37. Patel, S.; Mohan, S.; Mai, H.; Jain, U.; Lazebnik, S.; Li, Y. Robotic Manipulation by Imitating Generated Videos Without Physical Demonstrations. *arXiv preprint arXiv:2507.00990* **2025**.

 doi:10.20944/preprints202606.1403.v1

38. Li, H.; Sun, L.; Hu, Y.; Ta, D.; Barry, J.; Konidaris, G.; Fu, J. NovaFlow: Zero-Shot Manipulation via Actionable Flow from Generated Videos. *arXiv preprint arXiv:2510.08568* **2025**.
39. Dharmarajan, K.; Huang, W.; Wu, J.; Fei-Fei, L.; Zhang, R. Dream2Flow: Bridging Video Generation and Open-World Manipulation with 3D Object Flow. *arXiv preprint arXiv:2512.24766* **2025**.
40. Lee, S.; Jung, Y.; Chun, I.; Lee, Y.C.; Cai, Z.; Huang, H.; Talreja, A.; Dao, T.D.; Liang, Y.; Huang, J.B.; et al. TraceGen: World Modeling in 3D Trace Space Enables Learning from Cross-Embodiment Videos, 2025, [2511.21690].
41. Hung, A.; Duisterhof, B.P.; Ichnowski, J. 3PoinTr: 3D Point Tracks for Learning Manipulation from Unconstrained Human Videos, 2026, [2603.08485].
42. Lin, S.; Chen, J.; Xu, H.; Li, Z.; Wang, G.; Jing, Y.; Xu, S.; Zhao, R.; Sheil, B.; Chau, L.P.; et al. RoboFlow4D: A Lightweight Flow World Model Toward Real-Time Flow-Guided Robotic Manipulation, 2026, [2605.17522].
43. Sidik, B.; Mizrahi, D. 3D-Anchored Lookahead Planning for Persistent Robotic Scene Memory via World-Model-Based MCTS, 2026, [2604.11302].
44. Fan, X.; Lu, Y.; Gao, S.; Wu, X.; Han, R.; Li, M.; Zhao, H. DreamAvoid: Critical-Phase Test-Time Dreaming to Avoid Failures in VLA Policies, 2026, [2605.11750].
45. Tran, D.; Martín-Martín, R.; Grauman, K. EgoExo-WM: Unlocking Exo Video for Ego World Models, 2026, [2605.15477].
46. Huang, W.; Chao, Y.W.; Mousavian, A.; Liu, M.Y.; Fox, D.; Mo, K.; Fei-Fei, L. PointWorld: Scaling 3D World Models for In-The-Wild Robotic Manipulation, 2026, [2601.03782].
47. An, T.; Jia, J.; Li, G.; Li, J.; Zhou, C.; Liu, P.; Lyu, B.; Bai, J.; Guo, X.; Li, G.; et al. Feedback World Model Enables Precise Guidance of Diffusion Policy, 2026, [2605.15705].
48. Wu, H.; Jing, Y.; Cheang, C.; Chen, G.; Xu, J.; Li, X.; Liu, M.; Li, H.; Kong, T. Unleashing Large-Scale Video Generative Pre-training for Visual Robot Manipulation, 2023, [arXiv:cs.RO/2312.13139].
49. Cheang, C.L.; Chen, G.; Jing, Y.; Kong, T.; Li, H.; Li, Y.; Liu, Y.; Wu, H.; Xu, J.; Yang, Y.; et al. GR-2: A Generative Video-Language-Action Model with Web-Scale Knowledge for Robot Manipulation, 2024, [arXiv:cs.RO/2410.06158].
50. Guo, Y.; Hu, Y.; Zhang, J.; Wang, Y.J.; Chen, X.; Lu, C.; Chen, J. Prediction with Action: Visual Policy Learning via Joint Denoising Process, 2024, [2411.18179].
51. Zhu, C.; Yu, R.; Feng, S.; Burchfiel, B.; Shah, P.; Gupta, A. Unified World Models: Coupling Video and Action Diffusion for Pretraining on Large Robotic Datasets, 2025, [2504.02792].
52. Cen, J.; Yu, C.; Yuan, H.; Jiang, Y.; Huang, S.; Guo, J.; Li, X.; Song, Y.; Luo, H.; Wang, F.; et al. WorldVLA: Towards Autoregressive Action World Model, 2025, [2506.21539].
53. Ye, S.; Ge, Y.; Zheng, K.; Gao, S.; Yu, S.; Kurian, G.; Indupuru, S.; Tan, Y.L.; Zhu, C.; Xiang, J.; et al. World Action Models are Zero-shot Policies, 2026, [2602.15922].
54. Fan, L.; Xu, Z.; Cao, C.; Zhang, W.; Yuan, M.; Chen, J. AIM: Intent-Aware Unified world action Modeling with Spatial Value Maps, 2026, [arXiv:cs.RO/2604.11135].
55. Shi, C.; Xu, J.; Shi, S.; Sheng, K.; Zhang, B.; Jiang, L. DriveWAM: Video Generative Priors Enable Scalable World-Action Modeling for Autonomous Driving, 2026, [2605.28544].
56. Li, S.; Yao, V.; Yang, C.; Qu, T.; Cheng, R.; Yu, R.; Lu, H.; Von, N.; Chen, V.; Tang, Y.; et al. WALL-WM: Carving World Action Modeling at the Event Joints, 2026, [2606.01955].
57. Zhou, P.; Chen, S.; Chen, D.; Wang, J.; Jin, R.; Zhu, B.; Pan, Y.; Gu, S.; Wang, K.; Nan, S.; et al. τ_0-WM: A Unified Video-Action World Model for Robotic Manipulation, 2026, [2606.01027].
58. Yuan, T.; Dong, Z.; Liu, Y.; Zhao, H. Fast-WAM: Do World Action Models Need Test-time Future Imagination?, 2026, [arXiv:cs.CV/2603.16666].
59. Liang, J.; Liu, R.; Ozguroglu, E.; Sudhakar, S.; Dave, A.; Tokmakov, P.; Song, S.; Vondrick, C. Dreamitate: Real-World Visuomotor Policy Learning via Video Generation. In Proceedings of the Conference on Robot Learning, 2024, [2406.16862].
60. Bharadhwaj, H.; Dwibedi, D.; Gupta, A.; Tulsiani, S.; Doersch, C.; Xiao, T.; Shah, D.; Xia, F.; Sadigh, D.; Kirmani, S. Gen2Act: Human Video Generation in Novel Scenarios Enables Generalizable Robot Manipulation. In Proceedings of the CoRL Workshop on Cross-Embodiment, 2024, [2409.16283].
61. Wang, B.; Sridhar, N.; Feng, C.; Van der Merwe, M.; Fishman, A.; Fazeli, N.; Park, J.J. This&That: Language-Gesture Controlled Video Generation for Robot Planning. *arXiv preprint arXiv:2407.05530* **2024**.
62. Li, P.; Wu, H.; Huang, Y.; Cheang, C.; Wang, L.; Kong, T. GR-MG: Leveraging Partially-Annotated Data via Multi-Modal Goal-Conditioned Policy. *IEEE Robotics Autom. Lett.* **2025**, *10*, 1912–1919, [2408.14368]. https://doi.org/10.1109/LRA.2025.3526436.

 doi:10.20944/preprints202606.1403.v1

63. Ni, Z.; Li, Y.; Jiao, R.; Zhan, S.S.; Chen, S.; Yin, Z.; Chen, M.; Torr, P.; Wang, Z.; Zhu, Q. CreFlow: Corrective Reflow for Sparse-Reward Embodied Video Diffusion RL, 2026, [2605.14274].
64. Li, Y.; Zhang, B.; Gu, C.; Ma, Z.; Zhang, J.; Deng, J.; Zhu, X.; Zhang, L. From Imagined Futures to Executable Actions: Mixture of Latent Actions for Robot Manipulation, 2026, [2605.12167].
65. Mi, W.; Bao, Y.; Chi, X.; Ju, X.; Qin, Z.; Ge, K.; Tang, K.; Jia, P.; Zhang, S.; Tang, J. TC-IDM: Grounding Video Generation for Executable Zero-shot Robot Motion, 2026, [2601.18323].
66. Tang, C.; Xu, J.; Lu, H.; Zou, B.; Dong, W.; Zhang, H.; Kragic, D. Grasp as You Dream: Imitating Functional Grasping from Generated Human Demonstrations, 2026, [2604.07517].
67. Li, S.L.; Kim, E.; Bai, X.; Zhao, T.; Pang, T.; Simchowitz, M.; Sitzmann, V. Turning Video Models into Generalist Robot Policies, 2026, [2605.27817].
68. Zhang, G.; Ni, Z.; Mohapatra, P.; Liu, H.; Zhang, R.; Zhu, Q. EmboAlign: Aligning Video Generation with Compositional Constraints for Zero-Shot Manipulation, 2026, [2603.05757].
69. Fu, J.; Nan, J.; Sun, L.; Li, H.; Qian, J.; Barry, J.L.; Kitani, K.; Konidaris, G. NovaPlan: Zero-Shot Long-Horizon Manipulation via Closed-Loop Video Language Planning, 2026, [2602.20119].
70. Mishra, S.; Yadav, R.D.; Nair, N.; Pan, W.; Roy, S. AeroPlace-Flow: Language-Grounded Object Placement for Aerial Manipulators via Visual Foresight and Object Flow, 2026, [2603.07744].
71. Cen, J.; Huang, S.; Yuan, Y.; Li, K.; Yuan, H.; Yu, C.; Hou, B.; Jiang, Y.; Guo, J.; Li, X.; et al. RynnVLA-002: A Unified Vision-Language-Action and World Model. *CoRR* **2025**, *abs/2511.17502*, [2511.17502]. https://doi.org/10.48550/ARXIV.2511.17502.
72. Song, Z.; Li, Q.; Qin, S.; Chen, Y.; Chen, T.; Lin, L.; Wang, G. Learning Physics from Pretrained Video Models: A Multimodal Continuous and Sequential World Interaction Models for Robotic Manipulation, 2026, [2603.00110].
73. Chen, J.; Song, W.; Ding, P.; Zhou, Z.; Zhao, H.; Tang, F.; Wang, D.; Li, H. Unified Diffusion VLA: Vision-Language-Action Model via Joint Discrete Denoising Diffusion Process, 2026, [arXiv:cs.RO/2511.01718].
74. Yang, L.; Bai, Y.; Eskandar, G.; Shen, F.; Altillawi, M.; Chen, D.; Liu, Z.; Valada, A. CoVAR: Co-generation of Video and Action for Robotic Manipulation via Multi-Modal Diffusion, 2025, [2512.16023].
75. Shen, Y.; Wei, F.; Du, Z.; Liang, Y.; Lu, Y.; Yang, J.; Zheng, N.; Guo, B. VideoVLA: Video Generators Can Be Generalizable Robot Manipulators, 2025, [2512.06963].
76. Bi, H.; Tan, H.; Xie, S.; Wang, Z.; Huang, S.; Liu, H.; Zhao, R.; Feng, Y.; Xiang, C.; Rong, Y.; et al. Motus: A Unified Latent Action World Model, 2025, [2512.13030].
77. Zhen, H.; Gao, Z.; Sun, Q.; Zhao, Y.; Yang, Y.; Du, Y.; Guo, P.; Wang, T.H.; Qiao, Y.L.; Gan, C. Action Images: End-to-End Policy Learning via Multiview Video Generation, 2026, [2604.06168]. arXiv:2604.06168.
78. Liu, M.; Zhang, D.; Liu, J.; Cui, J.; Xie, H.; Chen, G.; Ye, H.; Yang, M.Y.; Nex, F.; Cheng, H. DriveVA: Video Action Models are Zero-Shot Drivers, 2026, [2604.04198].
79. Feng, Q.; Yu, J.; Liu, J.; Jia, Y.; Wu, Z.; Chen, H.; Qian, Z.; Gu, S.; Jia, P.; Ma, S.; et al. HarmoWAM: Harmonizing Generalizable and Precise Manipulation via Adaptive World Action Models, 2026, [2605.10942].
80. Huang, W.; Sun, H.; Guo, Y.; Ma, Y.; Li, H.; Long, J.; Mo, Z.; Guan, Z.; Guo, Y.; Di, S.; et al. NoiseGate: Learning Per-Latent Timestep Schedules as Information Gating in World Action Models, 2026, [2605.07794].
81. Zhen, H.; Sun, Q.; Zhang, H.; Li, J.; Zhou, S.; Du, Y.; Gan, C. TesserAct: Learning 4D Embodied World Models, 2025, [arXiv:cs.CV/2504.20995].
82. Wang, J.; Jiang, Y.; He, T.; Sun, J.; Zhang, Q.; He, J.; Cao, J.; Gan, Z.; Sun, M.; Shao, Q.; et al. MVISTA-4D: View-Consistent 4D World Model with Test-Time Action Inference for Robotic Manipulation. *arXiv preprint arXiv:2602.09878* **2026**.
83. Guo, J.; Li, Q.; Li, P.; Chen, Z.; Sun, N.; Su, Y.; Wang, H.; Zhang, Y.; Li, X.; Liu, H. Unified 4D World Action Modeling from Video Priors with Asynchronous Denoising, 2026, [arXiv:cs.RO/2604.26694].
84. Kuang, Y.; Park, S.; Fragkiadaki, K.; Tulsiani, S. Dex4D: Task-Agnostic Point Track Policy for Sim-to-Real Dexterous Manipulation, 2026, [2602.15828].
85. Zhou, Y.; Wang, X.; Shao, H.; Wang, L.; Zhao, G.; Shao, J.; Zhu, J.; Yu, T.; Zhu, Z.; Huang, G.; et al. DriveDreamer-Policy: A Geometry-Grounded World-Action Model for Unified Generation and Planning, 2026, [2604.01765].
86. Li, P.; Chen, Y.; Xu, Y.; Yang, J.; Wu, X.; Guo, J.; Sun, N.; Qian, L.; Li, X.; Xiao, X.; et al. Multi-View Video Diffusion Policy: A 3D Spatio-Temporal-Aware Video Action Model, 2026, [2604.03181].
87. Yang, L.; Bai, Y.; Eskandar, G.; Shen, F.; Altillawi, M.; Chen, D.; Majumder, S.; Liu, Z.; Kutyniok, G.; Valada, A. RoboEnvision: A Long-Horizon Video Generation Model for Multi-Task Robot Manipulation. *arXiv preprint arXiv:2506.22007* **2025**.

 doi:10.20944/preprints202606.1403.v1

88. Zhao, Q.; Lu, Y.; Kim, M.J.; Fu, Z.; Zhang, Z.; Wu, Y.; Li, Z.; Ma, Q.; Han, S.; Finn, C.; et al. CoT-VLA: Visual Chain-of-Thought Reasoning for Vision-Language-Action Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025. Computer Vision Foundation / IEEE, 2025, pp. 1702–1713, [2503.22020]. https://doi.org/10.1109/CVPR52734.2025.00166.
89. Hu, Y.; Zhang, J.; Luo, Y.; Guo, Y.; Chen, X.; Sun, X.; Feng, K.; Lu, Q.; Chen, S.; Zhang, Y.; et al. BagelVLA: Enhancing Long-Horizon Manipulation via Interleaved Vision-Language-Action Generation, 2026, [2602.09849]. arXiv:2602.09849.
90. Song, Y.; Wang, Y.; Deng, X.; Yan, Z.; Shou, M.Z. SWEET: Sparse World Modeling with Image Editing for Embodied Task Execution, 2026, [2605.19319].
91. Gu, S.; Cai, Y.; Wang, T.; Wu, S.; Fu, Y. Say, Dream, and Act: Learning Video World Models for Instruction-Driven Robot Manipulation. *arXiv preprint arXiv:2602.10717* **2026**.
92. Feng, Y.; Tan, H.; Mao, X.; Xiang, C.; Liu, G.; Huang, S.; Su, H.; Zhu, J. Vidar: Embodied Video Diffusion Model for Generalist Manipulation, 2025, [arXiv:cs.LG/2507.12898].
93. Chen, B.; Zhang, T.; Geng, H.; Zhang, C.; Li, P.; Song, K.; Freeman, W.T.; Malik, J.; Abbeel, P.; Tedrake, R.; et al. Large Video Planner Enables Generalizable Robot Control. *arXiv preprint arXiv:2512.15840* **2025**.
94. Zhang, Z.; Yang, C.; Lu, Q.; Guo, Y.; Zhang, J.; Hu, Y.; Chen, J. Veo-Act: How Far Can Frontier Video Models Advance Generalizable Robot Manipulation?, 2026, [arXiv:cs.RO/2604.04502].
95. Lang, X.; Wang, Y.; Zhou, Y.; Ni, C.; Li, K.; Zhu, J.; Liu, T.; Lv, J.; Zuo, X.; Ye, Y.; et al. VAG: Dual-Stream Video-Action Generation for Embodied Data Synthesis, 2026, [arXiv:cs.RO/2604.09330].
96. Lv, Q.; Kong, W.; Li, H.; Zeng, J.; Qiu, Z.; Qu, D.; Song, H.; Chen, Q.; Deng, X.; Pang, J. F1: A Vision-Language-Action Model Bridging Understanding and Generation to Actions. *CoRR* **2025**, *abs/2509.06951*, [2509.06951]. https://doi.org/10.48550/ARXIV.2509.06951.
97. Wang, R.; Liu, Q.; Deng, Y.; Liu, G.; Liu, Z.; Jia, K. EVA: Aligning Video World Models with Executable Robot Actions via Inverse Dynamics Rewards, 2026, [2603.17808].
98. Jiang, Y.; Guo, Y.; Yang, H.; Lei, G.; Chen, N.; Zhang, Y.; Yan, S.; Lin, B.; Gao, F.; Qi, B. CKT-WAM: Parameter-Efficient Context Knowledge Transfer Between World Action Models, 2026, [2605.06247].
99. Zhang, Y.; Chen, Y.; Liu, C.; Ding, Z.; Xu, J.; Zou, S.; Liao, J.; Hu, J.; Ren, X.; Zhang, X.; et al. Pelican-Unify 1.0: A Unified Embodied Intelligence Model for Understanding, Reasoning, Imagination and Action, 2026, [2605.15153].
100. Huang, S.; Levy, M.; Jiang, Z.; Anandkumar, A.; Zhu, Y.; Fan, L.; Huang, D.A.; Shrivastava, A. ARDuP: Active Region Video Diffusion for Universal Policies. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2024, pp. 8465–8472, [2406.13301].
101. Xu, Z.; Qiu, Q.; She, Y. VILP: Imitation Learning with Latent Video Planning. *IEEE Robotics and Automation Letters* **2025**, *10*, 3350–3357.
102. Li, S.; Gao, Y.; Sadigh, D.; Song, S. Unified Video Action Model, 2025, [2503.00200].
103. Liang, J.; Tokmakov, P.; Liu, R.; Sudhakar, S.; Shah, P.; Ambrus, R.; Vondrick, C. Video Generators are Robot Policies. *arXiv preprint arXiv:2508.00795* **2025**.
104. Liao, Y.; Zhou, P.; Huang, S.; Yang, D.; Chen, S.; Jiang, Y.; Hu, Y.; Cai, J.; Liu, S.; Luo, J.; et al. Genie Envisioner: A Unified World Foundation Platform for Robotic Manipulation. *CoRR* **2025**, *abs/2508.05635*, [2508.05635]. https://doi.org/10.48550/arXiv.2508.05635.
105. Zhou, P.; Chen, L.; Chen, S.; Chen, D.; Zhao, W.; Jin, R.; Ren, G.; Luo, J. Act2Goal: From World Model To General Goal-conditioned Policy, 2025, [2512.23541].
106. Li, R.; Zhang, H.; Jin, J.; Zeng, Q.; Zhuang, Z.; Tang, Y.; Lyu, S.; Wang, D. World-Value-Action Model: Implicit Planning for Vision-Language-Action Systems, 2026, [arXiv:cs.RO/2604.14732].
107. Ma, T.; Zheng, J.; Wang, Z.; Jiang, C.; Cui, A.; Liang, J.; Yang, S. DiT4DiT: Jointly Modeling Video Dynamics and Actions for Generalizable Robot Control, 2026, [2603.10448].
108. Ye, A.; Wang, B.; Ni, C.; Huang, G.; Zhao, G.; Li, H.; Li, H.; Li, J.; Lv, J.; Liu, J.; et al. GigaWorld-Policy: An Efficient Action-Centered World–Action Model, 2026, [arXiv:cs.CV/2603.17240].
109. Wang, X.; Wang, C.; Xu, Y.; Ye, M.; Zhang, F.C.; Tian, J.; Zhan, X.; Zhu, L.; Lu, C.; Yang, L. LaMP: Learning Vision-Language-Action Policies with 3D Scene Flow as Latent Motion Prior, 2026, [2603.25399].
110. Ge, Z.; Ding, P.; Yin, B.; Wang, Q.; Xie, Z.; Wang, Y.; Wang, J.; Li, H.; Suo, R.; Song, W.; et al. VAMPO: Policy Optimization for Improving Visual Dynamics in Video Action Models, 2026, [2603.19370].
111. Xu, M.; Xu, Z.; Xu, Y.; Chi, C.; Wetzstein, G.; Veloso, M.; Song, S. Flow as the Cross-Domain Manipulation Interface. In Proceedings of the 8th Annual Conference on Robot Learning, 2024, [2407.15208].

 doi:10.20944/preprints202606.1403.v1

112. Zhi, H.; Chen, P.; Zhou, S.; Dong, Y.; Wu, Q.; Han, L.; Tan, M. 3DFlowAction: Learning Cross-Embodiment Manipulation from 3D Flow World Model. *arXiv preprint arXiv:2506.06199* **2025**, [arXiv:cs.RO/2506.06199].
113. Noh, S.; Nam, D.; Kim, K.; Lee, G.; Yu, Y.; Kang, R.; Lee, K. 3D Flow Diffusion Policy: Visuomotor Policy Learning via Generating Flow in 3D Space, 2025, [2509.18676].
114. Lou, Y.; Chi, X.; Zhang, X.; Qian, Z.; Li, C.; Zhang, R.; Lyu, Y.; Song, G.; Fu, C.; Xu, H.; et al. Mask World Model: Predicting What Matters for Robust Robot Policy Learning, 2026, [arXiv:cs.RO/2604.19683].
115. Zheng, Y.; Gu, S.; Li, W.; Zheng, Y.; Zang, Y.; Tian, S.; Li, X.; Hao, C.; Gao, C.; Liu, S.; et al. OmniVTA: Visuo-Tactile World Modeling for Contact-Rich Robotic Manipulation, 2026, [arXiv:cs.RO/2603.19201].
116. Kim, M.J.; Gao, Y.; Lin, T.Y.; Lin, Y.C.; Ge, Y.; Lam, G.; Liang, P.; Song, S.; Liu, M.Y.; Finn, C.; et al. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning, 2026, [2601.16163].
117. Yuan, G.; Qiao, Q.; Zhang, J.; Xu, D. AdaWorldPolicy: World-Model-Driven Diffusion Policy with Online Adaptive Learning for Robotic Manipulation, 2026, [2602.20057].
118. MotuBrain Team.; Xiang, C.; Bao, F.; Liu, H.; Tan, H.; Bi, H.; Li, J.; Liu, J.; Pang, J.; Jing, K.; et al. MotuBrain: An Advanced World Action Model for Robot Control, 2026, [arXiv:cs.RO/2604.27792].
119. Li, L.; Zhang, Q.; Luo, Y.; Yang, S.; Wang, R.; Han, F.; Yu, M.; Gao, Z.; Xue, N.; Zhu, X.; et al. Causal World Modeling for Robot Control, 2026, [2601.21998].
120. Wang, B.; Li, B.; Ni, C.; Huang, G.; Zhao, G.; Li, H.; Li, J.; Lv, J.; Liu, J.; Feng, L.; et al. GigaBrain-0.5M*: a VLA That Learns From World Model-Based Reinforcement Learning, 2026, [2602.12099].
121. Yuan, H.; Yi, W.; Zhang, Z.; Chen, W.; Mo, Y.; Yin, J.; Li, X.; Zeng, X.; Wen, C.; Lu, C.; et al. VTAM: Video-Tactile-Action Models for Complex Physical Interaction Beyond VLAs, 2026, [2603.23481].
122. Deng, Y.; Liu, G.; Jia, K. DexWorldModel: Causal Latent World Modeling towards Automated Learning of Embodied Tasks, 2026, [2604.16484].
123. Lu, H.; Yao, L.; He, C.; Wang, H.; Gu, X.; Li, X.; Liao, W.; He, T.; Peng, P. The DAWN of World-Action Interactive Models, 2026, [2605.11550]. arXiv:2605.11550.
124. Guan, J.; Zhao, W.; Pei, Y.; Chen, Z.; Solin, A.; Kannala, J. Point Tracking Improves World Action Models, 2026, [2605.23856].
125. Wang, R.; Zhang, Y.; Lin, J.; Luo, K.; Wang, J.; Wang, Z.; Qi, X. When to Trust Imagination: Adaptive Action Execution for World Action Models, 2026, [2605.06222].
126. Zheng, R.; Wang, J.; Reed, S.; Bjorck, J.; Fang, Y.; Hu, F.; Jang, J.; Kundalia, K.; Lin, Z.; Magne, L.; et al. FLARE: Robot Learning with Implicit World Modeling, 2025, [2505.15659].
127. Zhao, H.; Wang, J.; Song, W.; Chen, S.; Liu, Y.; Wang, Y.; Li, H.; Wang, D. FRAPPE: Infusing World Modeling into Generalist Policies via Multiple Future Representation Alignment, 2026, [2602.17259].
128. Goswami, R.G.; Bar, A.; Fan, D.; Yang, T.Y.; Zhou, G.; Krishnamurthy, P.; Rabbat, M.; Khorrami, F.; LeCun, Y. World Models for Learning Dexterous Hand-Object Interactions from Human Videos, 2025, [arXiv:cs.RO/2512.13644].
129. Tang, Z.; Liu, H.; Chang, X.; Wu, C.; Huo, D.; Yang, Y.; Liu, B.; Cai, Z.; Xiong, F.; Xu, M.; et al. ALAM: Algebraically Consistent Latent Action Model for Vision-Language-Action Models, 2026, [2605.10819]. arXiv:2605.10819.
130. Liu, Y.; Zhu, J.; Mo, Y.; Li, G.; Cao, X.; Jin, J.; Shen, Y.; Li, Z.; Yu, T.; Yuan, W.; et al. PALM: Progress-Aware Policy Learning via Affordance Reasoning for Long-Horizon Robotic Manipulation, 2026, [2601.07060].
131. Lin, M.; Ding, P.; Wang, S.; Zhuang, Z.; Liu, Y.; Tong, X.; Song, W.; Lyu, S.; Huang, S.; Wang, D. HiF-VLA: Hindsight, Insight and Foresight through Motion Representation for Vision-Language-Action Models, 2025, [2512.09928].
132. Nguyen, T.; Yuan, W.; Wei, S.; Li, H.; Seita, D.; Wang, Y. ICLR: In-Context Imitation Learning with Visual Reasoning, 2026, [2603.07530].
133. Hu, Z.; Yao, X.; Meng, Y.; Bing, Z.; Knoll, A. Dreaming the Unseen: World Model-regularized Diffusion Policy for Out-of-Distribution Robustness, 2026, [2603.21017].
134. Wen, Y.; Lin, J.; Zhu, Y.; Han, J.; Xu, H.; Zhao, S.; Liang, X. VidMan: Exploiting Implicit Dynamics from Video Diffusion Model for Effective Robot Manipulation, 2024, [arXiv:cs.RO/2411.09153].
135. NVIDIA.; Alhaija, H.A.; Alvarez, J.; Bala, M.; Cai, T.; Cao, T.; Cha, L.; Chen, J.; Chen, M.; Ferroni, F.; et al. Cosmos-Transfer1: Conditional World Generation with Adaptive Multimodal Control, 2025, [arXiv:cs.CV/2503.14492].
136. Gao, S.; Zhou, S.; Du, Y.; Zhang, J.; Gan, C. AdaWorld: Learning Adaptable World Models with Latent Actions, 2025, [arXiv:cs.AI/2503.18938].

 doi:10.20944/preprints202606.1403.v1

137. Guo, J.; Ma, X.; Wang, Y.; Yang, M.; Liu, H.; Li, Q. FlowDreamer: A RGB-D World Model with Flow-based Motion Representations for Robot Manipulation, 2025, [arXiv:cs.RO/2505.10075].
138. Shang, Y.; Zhang, X.; Tang, Y.; Jin, L.; Gao, C.; Wu, W.; Li, Y. RoboScape: Physics-informed Embodied World Model, 2025, [arXiv:cs.CV/2506.23135].
139. Yan, W.; Zhang, Y.; Abbeel, P.; Srinivas, A. VideoGPT: Video Generation using VQ-VAE and Transformers, 2021, [arXiv:cs.CV/2104.10157].
140. Tulyakov, S.; Liu, M.Y.; Yang, X.; Kautz, J. MoCoGAN: Decomposing Motion and Content for Video Generation, 2017, [arXiv:cs.CV/1707.04993].
141. Assran, M.; Duval, Q.; Misra, I.; Bojanowski, P.; Vincent, P.; Rabbat, M.; LeCun, Y.; Ballas, N. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture, 2023, [arXiv:cs.CV/2301.08243].
142. Fei, Z.; Fan, M.; Huang, J. A-JEPA: Joint-Embedding Predictive Architecture Can Listen, 2024, [arXiv:cs.SD/2311.15830].
143. Bardes, A.; Ponce, J.; LeCun, Y. MC-JEPA: A Joint-Embedding Predictive Architecture for Self-Supervised Learning of Motion and Content Features, 2023, [arXiv:cs.CV/2307.12698].
144. Collaboration, O.X.E.; O'Neill, A.; Rehman, A.; Gupta, A.; Maddukuri, A.; Gupta, A.; Padalkar, A.; Lee, A.; Pooley, A.; Gupta, A.; et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, 2023, [arXiv:cs.RO/2310.08864].
145. Wu, K.; Hou, C.; Liu, J.; Che, Z.; Ju, X.; Yang, Z.; Li, M.; Zhao, Y.; Xu, Z.; Yang, G.; et al. RoboMIND: Benchmark on Multi-embodiment Intelligence Normative Data for Robot Manipulation. In Proceedings of the Robotics: Science and Systems XXI. Robotics: Science and Systems Foundation, 2025, RSS2025, [2412.13877]. https://doi.org/10.15607/rss.2025.xxi.152.
146. Bharadhwaj, H.; Vakil, J.; Sharma, M.; Gupta, A.; Tulsiani, S.; Kumar, V. RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking, 2023, [arXiv:cs.RO/2309.01918].
147. Mandlekar, A.; Booher, J.; Spero, M.; Tung, A.; Gupta, A.; Zhu, Y.; Garg, A.; Savarese, S.; Fei-Fei, L. Scaling Robot Supervision to Hundreds of Hours with RoboTurk: Robotic Manipulation Dataset through Human Reasoning and Dexterity, 2019, [arXiv:cs.RO/1911.04052].
148. Liang, L.; Bian, L.; Xiao, C.; Zhang, J.; Chen, L.; Liu, I.; Xiang, F.; Huang, Z.; Su, H. Robo360: A 3D Omnispective Multi-Material Robotic Manipulation Dataset, 2023, [arXiv:cs.CV/2312.06686].
149. Xie, S.; Cao, H.; Weng, Z.; Xing, Z.; Chen, H.; Shen, S.; Leng, J.; Wu, Z.; Jiang, Y.G. Human2Robot: Learning Robot Actions from Paired Human-Robot Videos, 2025, [arXiv:cs.RO/2502.16587].
150. Dasari, S.; Ebert, F.; Tian, S.; Nair, S.; Bucher, B.; Schmeckpeper, K.; Singh, S.; Levine, S.; Finn, C. RoboNet: Large-Scale Multi-Robot Learning, 2020, [arXiv:cs.RO/1910.11215].
151. Kareer, S.; Patel, D.; Punamiya, R.; Mathur, P.; Cheng, S.; Wang, C.; Hoffman, J.; Xu, D. EgoMimic: Scaling Imitation Learning via Egocentric Video, 2024, [arXiv:cs.RO/2410.24221].
152. Punamiya, R.; Kareer, S.; Liu, Z.; Citron, J.; Qiu, R.Z.; Cai, X.; Gavryushin, A.; Chen, J.; Liconti, D.; Zhu, L.Y.; et al. EgoVerse: An Egocentric Human Dataset for Robot Learning from Around the World, 2026, [arXiv:cs.RO/2604.07607].
153. Hoque, R.; Huang, P.; Yoon, D.J.; Sivapurapu, M.; Zhang, J. EgoDex: Learning Dexterous Manipulation from Large-Scale Egocentric Video, 2026, [arXiv:cs.CV/2505.11709].
154. Grauman, K.; Westbury, A.; Byrne, E.; Chavis, Z.; Furnari, A.; Girdhar, R.; Hamburger, J.; Jiang, H.; Liu, M.; Liu, X.; et al. Ego4D: Around the World in 3,000 Hours of Egocentric Video, 2022, [arXiv:cs.CV/2110.07058].
155. Grauman, K.; Westbury, A.; Torresani, L.; Kitani, K.; Malik, J.; Afouras, T.; Ashutosh, K.; Baiyya, V.; Bansal, S.; Boote, B.; et al. Ego-Exo4D: Understanding Skilled Human Activity from First- and Third-Person Perspectives, 2024, [arXiv:cs.CV/2311.18259].
156. Li, Y.; Cao, Z.; Liang, A.; Liang, B.; Chen, L.; Zhao, H.; Feng, C. Egocentric Prediction of Action Target in 3D, 2022, [arXiv:cs.CV/2203.13116].
157. Damen, D.; Doughty, H.; Farinella, G.M.; Fidler, S.; Furnari, A.; Kazakos, E.; Moltisanti, D.; Munro, J.; Perrett, T.; Price, W.; et al. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset, 2018, [arXiv:cs.CV/1804.02748].
158. Perrett, T.; Darkhalil, A.; Sinha, S.; Emara, O.; Pollard, S.; Parida, K.; Liu, K.; Gatti, P.; Bansal, S.; Flanagan, K.; et al. HD-EPIC: A Highly-Detailed Egocentric Video Dataset, 2025, [arXiv:cs.CV/2502.04144].
159. Wang, X.; Zhao, K.; Liu, F.; Wang, J.; Zhao, G.; Bao, X.; Zhu, Z.; Zhang, Y.; Wang, X. EgoVid-5M: A Large-Scale Video-Action Dataset for Egocentric Video Generation, 2024, [arXiv:cs.CV/2411.08380].
160. Jawaid, A.; Xiang, Y. OpenEgo: A Large-Scale Multimodal Egocentric Dataset for Dexterous Manipulation, 2025, [arXiv:cs.CV/2509.05513].

 doi:10.20944/preprints202606.1403.v1

161. Chavan, V.; Imgrund, Y.; Dao, T.; Bai, S.; Wang, B.; Lu, Z.; Heimann, O.; Krüger, J. IndEgo: A Dataset of Industrial Scenarios and Collaborative Work for Egocentric Assistants, 2025, [arXiv:cs.CV/2511.19684].
162. Zheng, R.; Niu, D.; Xie, Y.; Wang, J.; Xu, M.; Jiang, Y.; Castañeda, F.; Hu, F.; Tan, Y.L.; Fu, L.; et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710* **2026**.
163. Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Lin, K.; Maddukuri, A.; Nasiriany, S.; Zhu, Y. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning, 2020, [arXiv:cs.RO/2009.12293].
164. Gu, J.; Xiang, F.; Li, X.; Ling, Z.; Liu, X.; Mu, T.; Tang, Y.; Tao, S.; Wei, X.; Yao, Y.; et al. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills, 2023, [arXiv:cs.RO/2302.04659].
165. Tao, S.; Xiang, F.; Shukla, A.; Qin, Y.; Hinrichsen, X.; Yuan, X.; Bao, C.; Lin, X.; Liu, Y.; Chan, T.k.; et al. ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI. *arXiv preprint arXiv:2410.00425* **2024**, [arXiv:cs.RO/2410.00425].
166. Yu, T.; Quillen, D.; He, Z.; Julian, R.; Narayan, A.; Shively, H.; Bellathur, A.; Hausman, K.; Finn, C.; Levine, S. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In Proceedings of the Conference on Robot Learning, 2019, [1910.10897].
167. Liu, B.; Zhu, Y.; Gao, C.; Feng, Y.; Liu, Q.; Zhu, Y.; Stone, P. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning, 2023, [arXiv:cs.AI/2306.03310].
168. NVIDIA. NVIDIA Isaac Sim. https://developer.nvidia.com/isaac/sim. NVIDIA Developer. Accessed: 2026-04-14.
169. Rohmer, E.; Singh, S.; Freese, M. V-REP: A versatile and scalable robot simulation framework. 11 2013, pp. 1321–1326. https://doi.org/10.1109/IROS.2013.6696520.
170. Li, X.; Hsu, K.; Gu, J.; Pertsch, K.; Mees, O.; Walke, H.R.; Fu, C.; Lunawat, I.; Sieh, I.; Kirmani, S.; et al. Evaluating Real-World Robot Manipulation Policies in Simulation. In Proceedings of the Conference on Robot Learning, 2024, [2405.05941].
171. Nasiriany, S.; Maddukuri, A.; Zhang, L.; Parikh, A.; Lo, A.; Joshi, A.; Mandlekar, A.; Zhu, Y. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots, 2024, [arXiv:cs.RO/2406.02523].
172. Mu, Y.; Chen, T.; Chen, Z.; Peng, S.; Lan, Z.; Gao, Z.; Liang, Z.; Yu, Q.; Zou, Y.; Xu, M.; et al. RoboTwin: Dual-Arm Robot Benchmark with Generative Digital Twins, 2025, [arXiv:cs.RO/2504.13059].
173. Chen, T.; Chen, Z.; Chen, B.; Cai, Z.; Liu, Y.; Li, Z.; Liang, Q.; Lin, X.; Ge, Y.; Gu, Z.; et al. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation, 2025, [arXiv:cs.RO/2506.18088].
174. Han, S.; Qiu, B.; Liao, Y.; Huang, S.; Gao, C.; Yan, S.; Liu, S. RoboCerebra: A Large-scale Benchmark for Long-horizon Robotic Manipulation Evaluation, 2025, [arXiv:cs.RO/2506.06677].
175. Geng, H.; Wang, F.; Wei, S.; Li, Y.; Wang, B.; An, B.; Cheng, C.T.; Lou, H.; Li, P.; Wang, Y.J.; et al. RoboVerse: Towards a Unified Platform, Dataset and Benchmark for Scalable and Generalizable Robot Learning, 2025, [2504.18904].
176. Yan, F.; Liu, F.; Zheng, L.; Zhong, Y.; Huang, Y.; Guan, Z.; Feng, C.; Ma, L. RoboTron-Mani: All-in-One Multimodal Large Model for Robotic Manipulation, 2025, [arXiv:cs.RO/2412.07215].
177. Zhu, F.; Wu, H.; Guo, S.; Liu, Y.; Cheang, C.; Kong, T. IRASim: A Fine-Grained World Model for Robot Manipulation, 2025, [arXiv:cs.RO/2406.14540].
178. Unterthiner, T.; Van Steenkiste, S.; Kurach, K.; Marinier, R.; Michalski, M.; Gelly, S. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717* **2018**.
179. Zhang, R.; Isola, P.; Efros, A.A.; Shechtman, E.; Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 586–595.
180. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* **2004**, *13*, 600–612.
181. Fu, S.; Tamir, N.; Sundaram, S.; Chai, L.; Zhang, R.; Dekel, T.; Isola, P. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In Proceedings of the Advances in Neural Information Processing Systems, 2023.
182. Duan, H.; Yu, H.X.; Chen, S.; Fei-Fei, L.; Wu, J. WorldScore: A Unified Evaluation Benchmark for World Generation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, [arXiv:cs.GR/2504.00983].
183. Qin, Y.; Shi, Z.; Yu, J.; Wang, X.; Zhou, E.; Li, L.; Yin, Z.; Liu, X.; Sheng, L.; Shao, J.; et al. WorldSimBench: Towards Video Generation Models as World Simulators, 2024, [arXiv:cs.CV/2410.18072].

 doi:10.20944/preprints202606.1403.v1

184. Zhou, X.; Xu, Y.; Tie, G.; Chen, Y.; Zhang, G.; Chu, D.; Zhou, P.; Sun, L. LIBERO-PRO: Towards Robust and Fair Evaluation of Vision-Language-Action Models Beyond Memorization. *arXiv preprint arXiv:2510.03827* **2025**.
185. Wang, G.; Zhang, C.; Liu, Q.; Zhang, J.; Cai, J.; Liu, J.; Liu, X. LIBERO-X: Robustness Litmus for Vision-Language-Action Models. *arXiv preprint arXiv:2602.06556* **2026**.
186. Fei, S.; Wang, S.; Shi, J.; Dai, Z.; Cai, J.; Qian, P.; Ji, L.; He, X.; Zhang, S.; Fei, Z.; et al. LIBERO-Plus: In-Depth Robustness Analysis of Vision-Language-Action Models. *arXiv preprint arXiv:2510.13626* **2025**.
187. Dai, Y.; Fu, H.; Lee, J.; Liu, Y.; Zhang, H.; Yang, J.; Finn, C.; Fazeli, N.; Chai, J. RoboMME: Benchmarking and Understanding Memory for Robotic Generalist Policies. In Proceedings of the International Conference on Machine Learning, 2026.
188. Mandlekar, A.; Xu, D.; Wong, J.; Nasiriany, S.; Wang, C.; Kulkarni, R.; Fei-Fei, L.; Savarese, S.; Zhu, Y.; Martín-Martín, R. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In Proceedings of the Conference on Robot Learning, 2021, [2108.03298].
189. Yenamandra, S.; Ramachandran, A.; Yadav, K.; Wang, A.; Khanna, M.; Gervet, T.; Yang, T.Y.; Jain, V.; Clegg, A.W.; Turner, J.; et al. HomeRobot: Open-Vocabulary Mobile Manipulation. In Proceedings of the Conference on Robot Learning, 2023, [2306.11565].
190. Zhang, S.; Xu, Z.; Liu, P.; Yu, X.; Li, Y.; Gao, Q.; Fei, Z.; Yin, Z.; Wu, Z.; Jiang, Y.G.; et al. VLABench: A Large-Scale Benchmark for Language-Conditioned Robotics Manipulation with Long-Horizon Reasoning Tasks, 2024, [2412.18194].
191. Wang, Y.R.; Ung, C.; Tan, C.; Tannert, G.; Duan, J.; Li, J.; Le, A.; Oswal, R.; Grotz, M.; Pumacay, W.; et al. RoboEval: Where Robotic Manipulation Meets Structured and Scalable Evaluation. *arXiv preprint arXiv:2507.00435* **2025**, [arXiv:cs.RO/2507.00435].
192. Atreya, P.; Pertsch, K.; Lee, T.; Kim, M.J.; Jain, A.; Kuramshin, A.; Eppner, C.; Neary, C.; Hu, E.; Ramos, F.; et al. RoboArena: Distributed Real-World Evaluation of Generalist Robot Policies. In Proceedings of the Conference on Robot Learning, 2025, [2506.18123].
193. Yakefu, A.; Xie, B.; Xu, C.; Zhang, E.; Zhou, E.; Jia, F.; Yang, H.; Fan, H.; Zhang, H.; Peng, H.; et al. RoboChallenge: Large-Scale Real-Robot Evaluation of Embodied Policies. *arXiv preprint arXiv:2510.17950* **2025**.
194. Li, Y.; Zhou, Z.; Chen, Y.; Xue, Y.; Zhu, Y. dWorldEval: Scalable Robotic Policy Evaluation via Discrete Diffusion World Model, 2026, [arXiv:cs.RO/2604.22152].
195. Mao, J.; He, S.; Wu, H.N.; You, Y.; Sun, S.; Wang, Z.; Bao, Y.; Chen, H.; Guibas, L.; Guizilini, V.; et al. Robot Learning from a Physical World Model, 2025, [arXiv:cs.RO/2511.07416].
196. Sun, Y.; Zhuge, G.; Liu, K.; Gu, J.; Bing, X.; Gan, Z.; Tian, C. SANTS: A State-Adaptive Scheduler for World Action Models, 2026, [2605.27947].
197. Wang, Y.; Li, X.; Wang, W.; Zhang, J.; Li, Y.; Chen, Y.; Wang, X.; Zhang, Z. Unified Vision-Language-Action Model, 2025, [arXiv:cs.RO/2506.19850].
198. Gao, S.; Liang, W.; Zheng, K.; Malik, A.; Ye, S.; Yu, S.; Tseng, W.C.; Dong, Y.; Mo, K.; Lin, C.H.; et al. DreamDojo: A Generalist Robot World Model from Large-Scale Human Videos, 2026, [arXiv:cs.RO/2602.06949].
199. Sun, J.; Zhang, W.; Qi, Z.; Ren, S.; Liu, Z.; Zhu, H.; Sun, G.; Jin, X.; Chen, Z. VLA-JEPA: Enhancing Vision-Language-Action Model with Latent World Model, 2026, [arXiv:cs.RO/2602.10098].
200. Ye, S.; Jang, J.; Jeon, B.; Joo, S.J.; Yang, J.; Peng, B.; Mandlekar, A.; Tan, R.; Chao, Y.; Lin, B.Y.; et al. Latent Action Pretraining from Videos. In Proceedings of the The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025, [2410.11758].
201. Tang, Y.; Shang, Y.; Chen, Y.; Wei, B.; Zhang, X.; Yu, S.; Shi, L.; Yu, C.; Gao, C.; Wu, W.; et al. RoboScape-R: Unified Reward-Observation World Models for Generalizable Robotics Training via RL, 2025, [arXiv:cs.RO/2512.03556].
202. Huang, X.; Gai, W.; Wu, T.; Wang, C.; Liu, Z.; Zhou, X.; Wu, Y.; Gao, F. NavDreamer: Video Models as Zero-Shot 3D Navigators, 2026, [2602.09765].