Hallucination in World Models is Predictable and Preventable

Nicklas Hansen[1] **Xiaolong Wang**[1]

[1]UC San Diego

Abstract. Modern generative world models render increasingly realistic action-controllable futures, yet they frequently *hallucinate*: rollouts remain visually fluent while drifting from the ground-truth dynamics. We hypothesize that hallucination concentrates in low-coverage regions of the state-action space, where lightweight data-centric signals can both detect it and guide mitigation. To test this, we introduce MMBench2, a 427-hour, 210-task dataset for visual world modeling with ground-truth actions, rewards, and live simulators, and train a 350M-parameter world model on it. We identify three distinct hallucination modes: perceptual, action-marginalized, and scene-diverging – each anchored to a different stage of the pipeline, and develop three signals that accurately predict where the model will fail. To close coverage gaps at training time, we develop a coverage-aware sampling technique; to close them online, our hallucination predictors serve as curiosity rewards for targeted data collection, yielding a data-efficient finetuning recipe that adapts the pretrained world model to entirely unseen environments with as few as 50 real environment trajectories. Overall, our findings reveal that hallucination in world models is inherently a data coverage issue, and that the same signals used to detect it can also be used for mitigation.

Webpage: https://nicklashansen.com/mmbench2

1 Introduction

Modern generative world models render strikingly realistic, action-controllable futures across diverse environments [Alonso et al., 2024, Valevski et al., 2024, Hafner et al., 2025, Parker-Holder et al., 2025]. However, the rollouts they produce frequently *hallucinate*: they remain visually fluent and superficially plausible while drifting away from the ground-truth dynamics [Janner et al., 2019]. The term is borrowed from the language modeling literature [Ji et al., 2023, Huang et al., 2025], where it typically denotes generation of factually incorrect text; analogous failures have also been studied in image [Li et al., 2023] and video generation [Huang et al., 2024]. In a world model, the failure is arguably more consequential: hallucinated trajectories are fed directly into downstream planners and policies [Schrittwieser et al., 2020, Hafner et al., 2023, Hansen et al., 2024], so silent hallucination during rollout translates into silently incorrect decisions during control.

Despite the increasing fidelity of these models, *where* an autoregressive rollout will hallucinate, and *why*, is poorly understood. A natural reading is that hallucination is an architectural problem to be solved by ever-larger backbones and more training compute [Hoffmann et al., 2022]. However, we argue that **hallucination in world models is primarily a data coverage problem:** it concentrates in low-coverage regions of the state-action space [Levine et al., 2020, Gadre et al., 2023] and is therefore both *predictable* from signals available at runtime [Lakshminarayanan et al., 2017, Gal and Ghahramani, 2016, Chua et al., 2018] and *preventable* by adjusting which data the model is trained on rather than architectural changes. Our experiments reveal that a single underlying cause – coverage gaps – explains failures at every stage of the model pipeline: tokenizer, action-conditioning, and multi-step rollout, and that it manifests as three distinct failure modes shown in Figure 1.

Preprint. Correspondence to Nicklas Hansen <nihansen@ucsd.edu>.

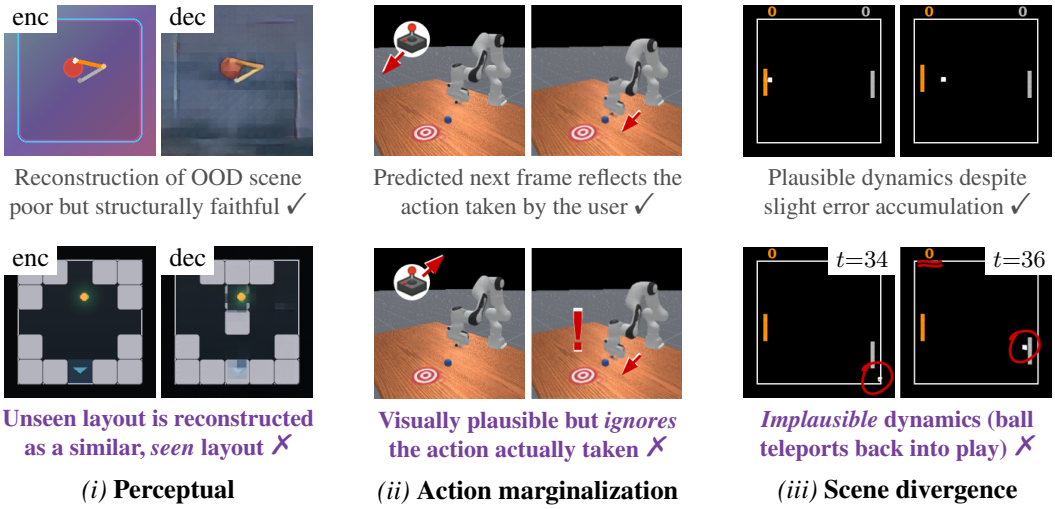


Figure 1. **Hallucination.** We categorize **hallucinations** as three distinct failure modes: *perceptual*, *action marginalization*, and *scene divergence*, and develop methods for detection and mitigation.

In this work, we set out to characterize different types of hallucinations, predict when they will occur using signals internal to the model, and use that information to close the underlying coverage gap. We explore two concrete ways in which the gap can be closed: *(i)* using coverage-aware sampling techniques during large-scale pretraining, and *(ii)* by targeted online data collection using our derived predictors of hallucination. Understanding when and why hallucinations happen requires three resources that no single benchmark currently provides: full control over the training corpus, behaviorally diverse data spanning many tasks and domains, as well as live environments to probe coverage gaps via online interaction. To address this gap we develop **MMBench2**, a massively multitask dataset for visual world modeling that extends MMBench [Hansen et al., 2026] with 65.6k mixed-quality trajectories equivalent to 427 hours of 224×224 video at 15 fps, complete with ground-truth action and reward labels and live simulators across 210 tasks spanning 10 domains. Of the 210 tasks, 200 form the pretraining corpus and 10 are held out as entirely unseen transfer tasks, allowing us to probe coverage both within and beyond the training distribution.

Using MMBench2, we train a 350M parameter Dreamer 4 [Hafner et al., 2025] world model and dissect its hallucination behavior. We identify three distinct hallucination modes, each anchored to a different stage of the pipeline: *perceptual* hallucination in the encoder-decoder pair, *action marginalization* in the dynamics model, and *scene divergence* during multi-step rollouts. We develop three predictors of hallucination: tokenizer round-trip residual, flow instability, and inter-seed denoising variance, and use them to detect hallucination at runtime, as well as curiosity reward for online data collection [Sekar et al., 2020] in seen and unseen tasks. Empirically, our hallucination-driven data collection enables adaptation to entirely unseen environments with just 50 real trajectories, approaching the effectiveness of human data collection. Together, these results show that hallucination in modern world models is, to a large extent, a coverage problem, and that the same signals that detect it can also be used to mitigate it. **We summarize our contributions as follows:**

- **MMBench2:** a large 427-hour dataset for visual world modeling with ground-truth actions and rewards, live environments, and behaviorally diverse data including human play.
- A **stage-by-stage characterization of hallucination** in generative world models tied to tokenizer, action marginalization, and rollout failures.
- **Three hallucination predictors** that detect hallucination without labels or additional training.
- A **coverage-aware training recipe** that reduces hallucination across all three predictors and improves rollout fidelity at no additional cost.
- A framework for **targeted data collection** that adapts a pretrained 350M world model to unseen environments with as few as 50 trajectories.

To support further research on generative world modeling, ***we release our full dataset, code for training and evaluation, model checkpoints, and a browser interface for open-ended interaction with the world model.*** See `https://nicklashansen.com/mmbench2` for videos and resources.

Figure 2. **MMBench2 tasks.** A sample of 36 of the 210 tasks in MMBench2, illustrating the visual and morphological diversity of the corpus. All observations are 224×224 RGB frames.

2 MMBench2: A Dataset for Visual World Modeling

To fully understand hallucination in large generative world models, it is necessary to have control over the training process as well as access to its training data, but (to our knowledge) no such resource exists today at a satisfactory scale. To address this gap, we propose **MMBench2**: a massively multitask dataset for visual world modeling. Our dataset consists of **65,600** trajectories (427 hours of 224×224 video at 15 fps for a total of 23M frames) with diverse behaviors (including human play data) across **210** different continuous control tasks, complete with ground-truth action and reward labels as well as live environments and language instructions for every task included in MMBench2. As the name implies, MMBench2 directly builds on top of MMBench which provides a collection of 200 live environments unified under a common interface. Figure 2 shows the breadth of our tasks.

Task overview. MMBench2 spans **10** task domains: DMControl [Tassa et al., 2018], DMControl Extended [Hansen et al., 2026], Meta-World [Yu et al., 2019], ManiSkill3 [Tao et al., 2025], MuJoCo [Todorov et al., 2012], MiniArcade [Hansen et al., 2026], Box2D [Brockman et al., 2016], RoboDesk [Kannan et al., 2021], OGBench [Park et al., 2025], and Continuous Atari [Farebrother and Castro, 2024]. Together they cover locomotion, dexterous and tabletop manipulation, goal-conditioned navigation, and arcade-style environments, with continuous action spaces of 1 to 16 dimensions; we zero-pad every action vector to the maximum dimension (d_a=16) and supply a per-dimension validity mask so models can ignore inactive dimensions per task. Of the **210** tasks, 200 form the pretraining set and 10 (proposed in this work) are held out as *unseen* transfer tasks.

The 200-task pretraining corpus contains 260 episodes per task. However, episode lengths vary by task, ranging from 25 for certain ManiSkill3 manipulation tasks up to 1,000 for Atari games. As a result, the corpus is highly non-uniform across tasks as shown in Figure 3. Additionally, the visual diversity *within* a task also differs wildly across the corpus with, *e.g.*, Inverted Pendulum (MuJoCo) varies little whereas tasks like Dungeon Explorer (proposed in this work), Bipedal Walker (Box2D), and Krull (Atari) vary greatly between frames and episodes. Please refer to Appendix C for an overview of tasks and Appendix D for a detailed description of the data collection process.

3 Training a Large Visual World Model

We seek to answer the following question: *why* do world models hallucinate, and can we predict *when* they are likely to hallucinate? To answer this question, we choose to develop and train a 350M parameter visual world model that largely follows the overall architecture and training recipe introduced in Dreamer 4 [Hafner et al., 2025]. As such, our world model is an action-conditioned generative model that follows an increasingly common two-stage training recipe: *(1)* first, a video *tokenizer* is trained via masked auto-encoding of visual observations, and *(2)* a dynamics model implemented as a block-causal Transformer [Vaswani et al., 2017] is subsequently trained via flow-

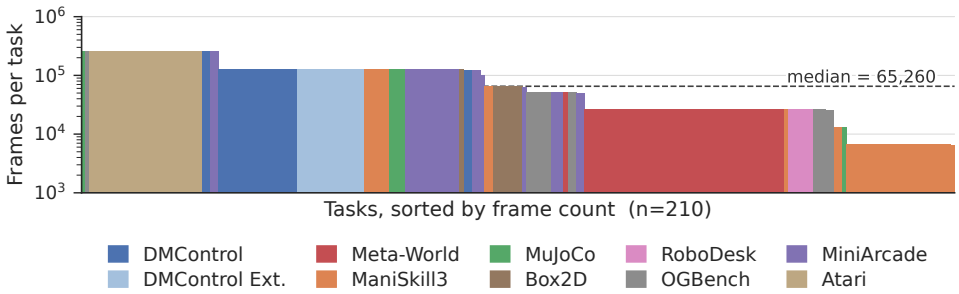


Figure 3. **Dataset composition.** Per-task frame counts across the 210 tasks in our training corpus (20M frames), sorted in descending order and colored by domain. The distribution is heavy-tailed: the top 20 tasks account for 26% of all frames (dominated by Atari) while the bottom 20 contribute only 0.7% (short-horizon manipulation tasks). The dashed line marks the per-task median (65k).

matching over spatial latent tokens produced by the frozen tokenizer with additional conditioning on action tokens. We first pretrain our world model on the MMBench2 training corpus, and then address our specific research questions by conducting a series of targeted finetuning experiments. This section aims to provide an overview of our architecture and training recipe.

Tokenizer. Our video tokenizer is instantiated as a symmetric encoder-decoder Transformer architecture. The encoder ingests 224×224 RGB frames "patchified" at stride 14 (256 patch tokens per frame), prepends 64 learnable latent queries, and projects the latent stream to a 64-dimensional bottleneck bounded by a $\tanh$ activation, producing a per-frame code $z \in [-1, 1]^{64\times64}$. The decoder then reconstructs images conditioned only on latent codes. Training uses a masked reconstruction objective [He et al., 2022] where a fraction of input patches drawn per-frame from $\mathcal{U}(0, 0.9)$ are replaced by a learned mask token, and the loss (pixel MSE + LPIPS [Zhang et al., 2018]) is computed only on masked positions. We normalize losses by their running RMS to reduce sensitivity to hyperparameters. Encoder and decoder are 50M learnable parameters each.

Dynamics model. Our dynamics model is a 250M parameter block-causal Transformer over packed tokenizer latents, trained on top of the *frozen* tokenizer introduced above. Per timestep, the input sequence consists of an action token (a 2-layer MLP over the 16-dimensional padded action), a shortcut-conditioning token (encoding noise level σ and step size d), 32 packed spatial latent tokens, 4 register tokens, and optional agent tokens used as the readout for the reward and behavior-cloning heads. The model is trained with the *shortcut flow-matching* objective of Frans et al. [2025], which interleaves an empirical one-step regression term with a self-consistency bootstrap that distills two coarser-step predictions into one finer-step target. At inference this enables next-frame sampling in as few as 4 Euler substeps.

Training recipe. The training recipe of Dreamer 4 was developed for a setting where only a small percentage of the training data has action and reward labels. However, MMBench2 provides ground-truth actions and rewards for every trajectory in the dataset which we choose to take advantage of. We first pretrain the tokenizer on the full training corpus, and then proceed to also pretrain the dynamics model, conditioning on actions. After this initial pretraining phase, we experiment with additional mid-training and finetuning recipes introduced in the following sections. As part of our experiments, we initialize additional reward prediction and behavior cloning (BC) policy heads after pretraining. The reward predictor is trained via $L{=}8$ multi-step discrete regression (using symlog two-hot encoding) with gradients backpropagated through the dynamics model. The BC policy is a deterministic Gaussian policy trained to predict ground-truth actions via an MSE loss – this is a departure from Dreamer 4 which only considered discrete actions.

4 Hallucination in World Models

A generative world model imagines a future by chaining three distinct operations: *(1)* an encoder maps each observation into a latent code, *(2)* an action-conditioned dynamics head predicts the next latent, and *(3)* a decoder renders a reconstruction back to pixel space. Each of these stages is a learned function trained on a finite slice of the state-action space, and each can therefore fail *independently* when asked to extrapolate beyond what it has seen. We use the term *hallucination* to

refer to any such failure: an output that is fluent and visually plausible, yet decoupled from reality. Crucially, because the three stages compose sequentially, a hallucination introduced early (*e.g.* a corrupted encoding) is propagated and amplified by the stages that follow, so naming *which* stage produces a given failure is a prerequisite for diagnosing and fixing it. This section explores how hallucinations can be characterized, detected, and finally mitigated.

4.1 Characterizing Hallucination

We identify three distinct ways in which a generative world model may hallucinate, each tied to a different stage of the imagination pipeline. We illustrate each type of hallucination in Figure 1 and define them as follows:

(i) **Perceptual hallucination.** The tokenizer's reconstruction of an observation already differs from the observation itself, before any dynamics rollout has been executed. Concretely, the encoder–decoder pair projects out-of-distribution scene structure onto the closest in-distribution exemplar in its learned latent "vocabulary". For example, an unseen maze layout might be reconstructed with the agent and goal in the correct positions but with the walls of an entirely *different* layout seen during training. The dynamics head then rolls out against this corrupted scene as if it were ground-truth. This failure mode is a property of the frozen encoder–decoder pair alone and persists even at horizon $H{=}0$.

(ii) **Action-marginalized (ignored) hallucination.** Conditional on a context, the predicted next latent is largely insensitive to the input action. The rollout is visually plausible but collapses onto an action-marginalized future, so the model behaves more like a video generator than a controllable world model. Operationally, we expose this mode by intervening on the action stream at evaluation time, *e.g.* by randomly shuffling actions within a batch and measuring the resulting change in flow MSE; a model that hallucinates in this way is one whose flow MSE barely moves under the intervention.

(iii) **Scene-diverging hallucination.** It is well understood that autoregressive rollouts accumulate compounding error as the prediction horizon increases. However, *scene-diverging* hallucination is a very specific failure mode where physically implausible events (such as a ball teleporting back into play when scoring in Pong) are predicted. This type of hallucination is most frequent in states with poor data coverage.

These three types of failure modes probe disjoint pieces of the model: the tokenizer, the action-conditioning of the dynamics model, and the multi-step accumulation of dynamics error.

4.2 Detecting Hallucination

Next, we develop *three* distinct metrics designed to track model hallucination. Empirically **we find that these three metrics, although mechanistically distinct, are all strong predictors of hallucination events**. We view this convergence as a feature and not a redundancy, since three mechanistically distinct metrics agreeing with respect to hallucination events is in itself evidence that the underlying signal is real. None of them require any labels nor additional training which makes them especially suitable for runtime detection.

Proposed metric 1: tokenizer round-trip residual. $u_r = \|\hat{z} - \mathrm{Encode}(\mathrm{Decode}(\hat{z}))\|$, the latent-space residual of a single decode–encode round-trip of the dynamics-predicted next latent $\hat{z}$, targets perceptual hallucination by measuring the very symptom that defines it: a predicted latent whose decoded frame falls off the tokenizer's manifold, *e.g.* a corrupted scene layout or a fabricated object, does not survive re-encoding and produces a large u_r.

Proposed metric 2: flow instability. u_f, the *flow instability* of the dynamics head at a given $(\text{context}, \text{action})$ pair, measures how much the denoiser's clean-target prediction $\hat{x}_1$ moves between successive Euler integration substeps, averaged over the second half of substeps. A sharp, well-conditioned dynamics head converges quickly to a stable $\hat{x}_1$ (low u_f); a head whose conditioning provides little signal keeps oscillating across substeps (high u_f).

Proposed metric 3: inter-seed variance. u_s, the *inter-seed variance* of the next-latent prediction across N independent denoising trajectories at fixed past and action, targets scene-diverging hallucination by measuring epistemic uncertainty across noise seeds: regions where seeds disagree are precisely those where multi-step rollouts will fan out.

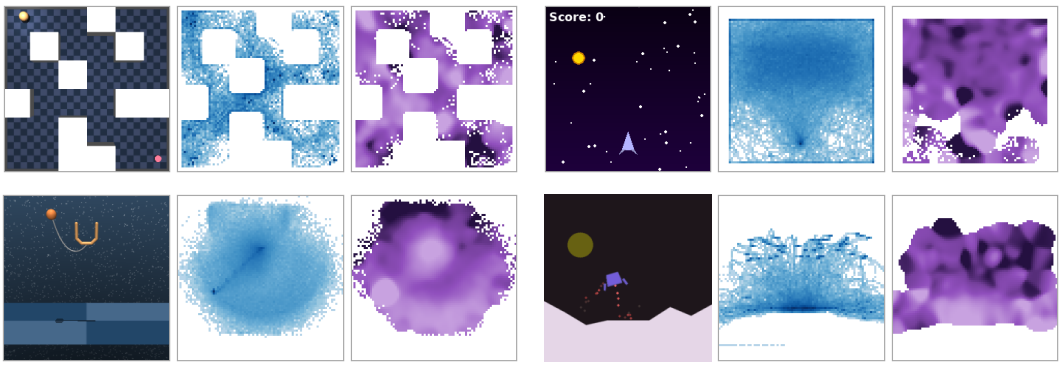


Figure 4. **Data coverage and hallucinations.** From left to right: sample frame, state density of key agent/object position, and tokenizer round-trip residual u_r of the world model. We use **dark blue** and **dark purple** to denote **high state density** and **high $\mathbf{u_r}$**, respectively. Hallucinations concentrate in low-coverage regions, especially near the periphery of the visited state distribution.

Controlling for scene motion. In practice, we find that a naive use of the three signals above is confounded by scene activity: high-motion transitions inflate tokenizer residuals, flow instability, and seed-to-seed denoising dispersion alike. To remove this confound we instead report dynamism-normalized variants $u^{\text{norm}} \doteq u/m$ where m is the per-step RMS change of the latent representation, computed as a per-task average over the dataset, or as a running estimate when used online for data collection. Normalizing by scene motion means that each metric tracks model uncertainty *relative to how much is happening in the scene*. We treat $u_r^{\text{norm}}, u_f^{\text{norm}}, u_s^{\text{norm}}$ as our primary hallucination predictors in all subsequent analysis.

4.3 Mitigating Hallucination

The taxonomy and predictors above suggest a single data-centric lens on hallucination: each of the three failure modes are, mechanistically, a consequence of the model having seen too little of some region of the state-action space. A perceptual hallucination is a coverage gap in the tokenizer's reconstruction distribution, an action-marginalized hallucination is a coverage gap in action-conditional transitions, and a scene-diverging hallucination is a coverage gap along the trajectory the model is asked to imagine. Figure 4 shows the relationship between data coverage and hallucination on four of our tasks. Two interventions follow naturally:

Coverage-aware training. We propose to resample the existing dataset to upweight under-represented regions of the state-action space, then ask whether closing those gaps at training time reduces all three failure modes simultaneously. Because the lens above identifies coverage as the underlying lever for every type of hallucination, a single reweighting recipe is expected to move all three signals in the right direction at once, rather than requiring a separate intervention per type. In practice, we rebalance training data by adjusting sampling to be uniform across *tasks* rather than *frames*. We also experiment with loss re-weighting but find interventions in sampling to be superior.

Targeted data collection. When the existing dataset does not have enough coverage of a region for re-weighting alone to help, the hallucination predictors can themselves act as an objective for targeted, curiosity-driven data collection. Specifically, during interaction with a live environment, candidate trajectories are rolled out in the world model, scored by predicted hallucination, and the highest-ranked trajectory is executed in the environment, producing data that, by construction, covers transitions that previously caused the model to hallucinate. In practice, we collect data in a closed-loop manner with a prediction horizon of $H = 32$ and replanning every $K = 16$ steps.

5 Experiments

We train a 350M parameter action-conditioned world model on the training corpus of MMBench2. Our training data consists of approximately 20M frames at 224×224 resolution across 200 distinct continuous control tasks, and we reserve the remaining 3M frames for testing. Although low-dimensional state information is readily available in MMBench2, we choose to strictly consider RGB observations in this work. We describe our experimental setup below.

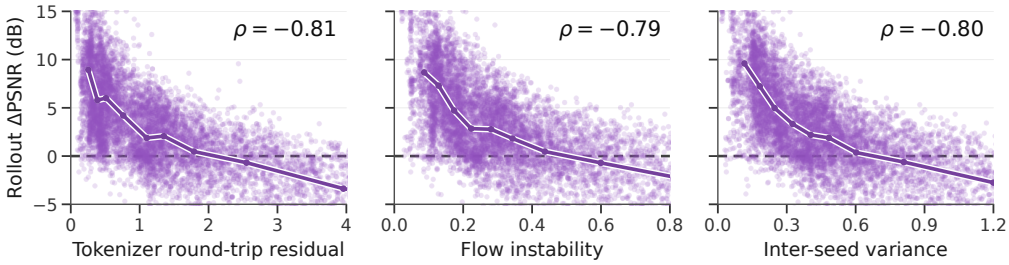


Figure 5. **Our three hallucination predictors track the same realized rollout error.** Each point corresponds to one of 9k held-out 24-frame sequences from any of 200 training tasks, computed using our pretrained base model; the purple curve shows the median for each of 8 bins.

Evaluation. We evaluate our world model and its variants across four successive training phases: *(1)* action-conditioned pretraining *without* reward labels, *(2)* coverage-aware "mid-training" that extends the previous phase with our proposed reweighted sampling, *(3)* action-conditioned world modeling *with* reward labels, and *(4)* finetuning on seen and unseen tasks via targeted data collection. Our key evaluation metrics include:

(i) **Reconstruction PSNR** ↑ evaluates encoder/decoder quality without considering dynamics.

(ii) **Rollout PSNR gain (dB)** ↑ evaluates quality of generated rollouts relative to a baseline that repeats the last frame across the entire rollout horizon. Although naive, this baseline can be surprisingly strong depending on the task. We consider a scene *divergent* when $\Delta\text{PSNR} \leq 0$.

(iii) **Action shuffle ratio** ↑ evaluates action sensitivity in the dynamics model by measuring one-step teacher-forced flow MSE relative to batch-shuffled actions. We consider actions to be *ignored* (marginalized) when this ratio ≤ 1.1.

(iv) **Downstream task performance (normalized score)** ↑ evaluates how useful the world model is for downstream tasks via closed-loop MPC using the Cross-Entropy Method (CEM) with horizon $H = 32$ and replanning every 16 steps. We report normalized score $s \in [0, 1]$ rather than raw reward since reward scales differ significantly (by several orders of magnitude) across tasks.

Implementation details. The encoder and decoder each have 50M learnable parameters, and the dynamics model (including prediction heads) has 250M parameters. We use $8\times$ NVIDIA H100 GPUs for training. The tokenizer is pretrained for 300k steps (14 GPU days), and the dynamics model is pretrained for 180k steps (24 GPU days). Our final checkpoints (post-finetuning) are trained for 380k and 210k steps, respectively, for a total of 58 GPU days using a context length of $T = 24$. The reward head and BC policy are conditioned on CLIP-ViT/B embeddings of per-task language instructions. See Appendix F for more implementation details.

We seek to answer the following questions:

(Q1) **Hallucination detection.** Do our proposed metrics accurately predict model hallucination?

(Q2) **Pretraining.** Does coverage-aware training meaningfully reduce hallucination?

(Q3) **Targeted data collection.** Can model hallucination be mitigated by collecting additional data? Given a fixed budget, which type of data improves the world model the most?

(Q4) **Generalization.** How can a pretrained world model be adapted to unseen tasks? Does our world model generalize zero-shot or is finetuning necessary? When does it fail to transfer?

5.1 Main results

Detecting hallucination events. To determine whether our proposed hallucination metrics are predictive of hallucination and by extension model performance, we compute Spearman correlations between open-loop rollout ΔPSNR and each of our metrics; results are shown in Figure 5. We find a strong negative correlation ($\rho \approx 0.80$) across all three metrics, which indicates that they all track the same underlying error. We additionally measure their per-task aggregated AUROC against two binary hallucination labels, *action ignored* (action shuffle ratio ≤ 1.1) and *scene divergent* (rollout ΔPSNR vs. frame-repeating baseline ≤ 0); results are shown in Appendix E. We find that our metrics consistently outperform their unnormalized counterparts (*i.e.*, u_f^{norm} is a better predictor than u_f) as well as baselines that rely on latent scene motion m or per-task frame count.

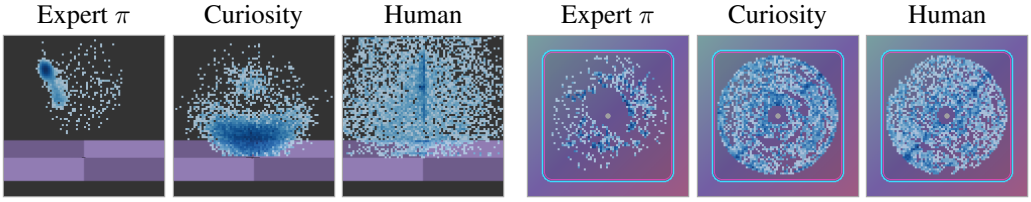


Figure 6. **Data coverage by collection method.** We show state densities for three online data collection policies (expert, curiosity, and human) across two tasks in the unseen task set.

Table 2. **Targeted data collection for finetuning on 10 unseen tasks.** We finetune our world model on a set of 10 seen + 10 unseen tasks, varying data source and finetuning strategy. Each data source contains 50 trajectories per task. Offline metrics were computed using a test set of expert trajectories and human play data in equal amount. Task performance is measured via closed-loop MPC using CEM; we report mean performance normalized to $[0, 1]$ across 3 episodes per task.

Method	Tok FT	Dyn FT	Recon PSNR ↑	Rollout ΔPSNR ↑	Action shuf. ↑	u_r^{norm} ↓	Task perf. (MPC) ↑
Random policy			—	—	—	—	0.118
Base	✗	✗	17.37	−12.44	1.12	3.860	—
Coverage-aware	✗	✗	17.21	−12.52	1.29	3.769	0.276
No-op actions	✗	✓	17.21	−11.66	1.41	4.175	—
No-op actions	✓	✓	34.74	+0.66	1.55	1.486	0.163
Random policy	✗	✓	17.21	−11.29	1.73	4.167	—
Random policy	✓	✓	35.81	+2.66	2.00	1.201	0.228
Expert policy	✓	✓	35.86	+2.84	2.04	1.131	0.362
Human play	✓	✓	37.11	+3.89	**2.42**	1.002	0.362
Curiosity (u_r^{norm})	✓	✓	36.05	+3.00	2.00	1.144	0.325
All (combined)	✓	✓	**37.91**	**+4.02**	2.34	**0.975**	**0.390**

Coverage-aware training. We evaluate the efficacy of our proposed coverage-aware training by extending the pretraining of our base model by an additional 30k steps for the tokenizer and dynamics model each, while varying the sampling method: uniform sampling across *frames* (default), and uniform sampling across *tasks* (ours). Results are shown in Table 1. We find that tokenizer and dynamics model both benefit from coverage-aware training, and that adopting it for both achieves best overall performance.

Table 1. **Coverage-aware training, by stage.** Mean change with coverage-aware training vs. the base model on held-out trajectories across 200 tasks. *Tok ft* finetunes the tokenizer for 30k steps using coverage-aware training and then finetunes the dynamics model for 30k steps using default sampling, *Dyn ft* reverses this, and *Both* finetunes both for 30k steps each using coverage-aware training. Best is in bold.

Metric	Tok ft	Dyn ft	**Both**
Recon PSNR (dB) ↑	**+0.46**	−0.01	+0.44
Action-shuffle ratio ↑	+0.02	**+0.27**	+0.29
Rollout ΔPSNR (dB) ↑	+0.42	+0.68	**+0.88**
u_r^{norm} ↓	−0.07	−0.16	**−0.20**
u_f^{norm} ↓	−0.03	−0.06	**−0.07**
u_s^{norm} ↓	−0.06	−0.13	**−0.14**

Mitigating hallucination via targeted data collection. Since we observe a strong relationship between data coverage and hallucination, a question that naturally follows is *can hallucination be mitigated by filling the gaps in coverage via targeted data collection?* To answer this question, we construct a task set that consists of 10 tasks seen during pretraining and 10 completely unseen tasks, and then collect 50 episodes for each of these tasks varying only the behavior policy used. Specifically, we evaluate the following approaches: no-op (all-zero) actions, random actions, expert policies, human play (via keyboard), and curiosity-based exploration [Sekar et al., 2020] using our proposed hallucination metric u_r^{norm} as objective. State densities for different collection policies are shown in Figure 6. We finetune tokenizer and dynamics model for 50k steps and 30k steps, respectively, and report offline performance met-

Table 3. **Comparison to off-the-shelf tokenizers.** Reconstruction PSNR and LPIPS for our tokenizer as well as four off-the-shelf tokenizers, evaluated on the 10 "seen" and "unseen" task sets used in our finetuning experiments. Our tokenizers outperform the best off-the-shelf tokenizer, Wan 2.1 VAE, by a large margin on tasks in the training set but generalizes poorly to new tasks without additional finetuning. Best result in bold; arrows indicate whether higher (↑) or lower (↓) is better.

			PSNR (dB) ↑			LPIPS ↓	
Tokenizer	Params	Latent/frame	Seen	Unseen	Δ_{S-U}	Seen	Unseen
Ours							
Base	102 M	4096	38.29	17.34	+20.95	0.011	0.389
Coverage-aware	102 M	4096	38.93	17.12	+21.81	0.008	0.348
Post-FT	102 M	4096	**39.66**	**38.04**	**+1.62**	**0.007**	**0.010**
Off-the-shelf							
SD-VAE-MSE	84 M	3136	33.32	32.39	+0.93	0.031	0.030
Cosmos-CV8x8x8 (1.0)	106 M	2048	32.80	32.72	+0.08	0.050	0.042
Wan 2.1 VAE	127 M	4096	36.45	36.62	−0.17	0.010	0.010
DC-AE-f32c32	323 M	2048	31.49	32.15	−0.66	0.035	0.031

rics and downstream task performance via closed-loop MPC (planning with CEM). Results for the 10 unseen tasks are shown in Table 2, and additional results can be found in Appendix E. Pretraining transfers, to some extent, zero-shot (0.276, 2.3× the 0.118 random policy baseline), and collecting just 50 trajectories per task using u_r^{norm}-based curiosity lifts performance to 0.325, within $\sim 90\%$ of the expert/human oracles (0.362) despite using no privileged behavior.

5.2 Discussion

Empirically, we find that hallucination can be mitigated, to a great extent, by targeted data collection, but that not all data sources are equally informative. We believe that there are two concrete reasons for this: *(i)* world modeling requires broad state-action coverage which, *e.g.*, a random policy does not guarantee, and *(ii)* downstream tasks are goal-directed and have a much narrower state-action distribution than the general space, so *e.g.* our closed-loop MPC evaluations mainly measure model accuracy around that narrower subspace. That is, arguably, by design: there are certain behaviors we are more interested in modeling than others, but it raises a broader question about how we can best evaluate world models moving forward.

Do off-the-shelf tokenizers improve perception? One concrete way in which perceptual hallucination can be mitigated on unseen tasks is by leveraging off-the-shelf tokenizers trained on datasets several orders of magnitude larger than the one considered in this work. To investigate the viability of this approach, we compare our trained tokenizers (before and after finetuning on the unseen task set) against four off-the-shelf tokenizers; results are shown in Table 3. We find that off-the-shelf tokenizers such as Wan 2.1 VAE (the strongest) underperform compared to our in-domain tokenizer when evaluated on tasks from the 200-task training corpus, but perform significantly better on the unseen task set unless we finetune our model in which case our in-domain tokenizer wins again. These results suggest that off-the-shelf tokenizers indeed are promising for world modeling but that there are still tangible benefits to in-domain finetuning when possible.

Related work. We include a detailed discussion of related work in Appendix A, and position MM-Bench2 vs. existing datasets for world modeling in Appendix B.

In conclusion, we argue that hallucination in generative world models is, first and foremost, a data-coverage problem, and identify three failure modes: perceptual, action marginalization, and scene divergence. Our predictors track them with $\rho \approx 0.80$ against rollout ΔPSNR and yield two complementary recipes: coverage-aware sampling reduces all three failure modes simultaneously, and using the predictors as curiosity rewards adapts our pretrained model to entirely unseen tasks.

Limitations. Our study operates at the 350M parameter scale across 210 simulated control tasks. Whether our findings translate to billion-parameter models or to real robot data with sensor noise and partial observability remains an open empirical question. We also acknowledge the significant computational cost of training large world models, as well as the need for more diverse datasets.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Thirty-eighth Conference on Neural Information Processing Systems*, 2024.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations (ICLR)*, 2024.

Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, pages 885–897. PMLR, 2019.

Jesse Farebrother and Pablo Samuel Castro. Cale: Continuous arcade learning environment. *Advances in Neural Information Processing Systems*, 37:134927–134946, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36:27092–27112, 2023.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, 2016.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S. Merel, Daniel J. Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matthew Hoffman, Nicolas Heess, and Nando de Freitas. RL unplugged: A suite of benchmarks for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2026.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2): 1–55, 2025. doi: 10.1145/3703155.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12): 1–38, 2023. doi: 10.1145/3571730.

Harini Kannan, Danijar Hafner, Chelsea Finn, and Dumitru Erhan. Robodesk: A multi-task reinforcement learning benchmark. https://github.com/google-research/robodesk, 2021.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=1QqIfGZOWu.

Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi "Jim" Fan. Nitrogen: An open foundation model for generalist gaming agents, 2026. URL https://arxiv.org/abs/2601.02427.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

NVIDIA. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

Jack Parker-Holder, Shlomi Fruchter, and Google DeepMind. Genie 3: A new frontier for world models. Google DeepMind Blog, August 2025. URL https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Julian Quevedo, Quinn McIntyre, Spruce Campbell, Xinlei Chen, Robert Wachen, Decart, and Etched. Oasis: A universe in a transformer. Decart, October 2024. URL https://oasis-model.github.io/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Rami. Lucid v1: A world model that does go brrr on consumer hardware. Substack, November 2024. URL https://ramimo.substack.com/p/lucid-v1-a-world-model-that-does.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8583–8592. PMLR, 13–18 Jul 2020.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, et al. Deepmind control suite. Technical report, DeepMind, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines, 2024. URL https://arxiv.org/abs/2408.14837.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

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

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A Related Work

World models for control. World models that learn environment dynamics from data have a long history in model-based RL, ranging from abstract latent dynamics models [Ha and Schmidhuber, 2018, Hafner et al., 2020, Hansen et al., 2022, 2024] to high-capacity generative models that render full pixel observations [Micheli et al., 2023, Alonso et al., 2024, Valevski et al., 2024]. Recent work scales these models to heterogeneous video corpora [Bruce et al., 2024, NVIDIA, 2025, Wan et al., 2025] and to real-time, playable neural environments [Quevedo et al., 2024, Rami, 2024, Parker-Holder et al., 2025]. We build on Dreamer 4 [Hafner et al., 2025], an integrated tokenizer-plus-dynamics architecture with strong action conditioning, but the signals and interventions we propose are largely model-agnostic and apply, in principle, to any modern generative world model. Despite striking visual fidelity, these models continue to hallucinate under distribution shift — a failure mode that we set out to characterize, predict, and mitigate.

Hallucination and uncertainty in generative models. Hallucination has been studied extensively in language models [Ji et al., 2023, Huang et al., 2025] and, increasingly, in image [Li et al., 2023] and video generation [Huang et al., 2024], but the question of *where* an autoregressive world model will fail along a rollout has received comparatively little attention. A separate but related literature studies uncertainty estimation in deep networks through deep ensembles [Lakshminarayanan et al., 2017], MC dropout [Gal and Ghahramani, 2016], and post-hoc out-of-distribution detectors [Lee et al., 2018, Liu et al., 2020], with applications in offline model-based RL via uncertainty-penalized policies [Yu et al., 2020, Kidambi et al., 2020]. Closest in spirit, prior work uses ensemble disagreement as an exploration signal in single-task RL [Pathak et al., 2019, Sekar et al., 2020]. In contrast, our proposed metrics are derived directly from the existing world model and target the three distinct stages at which a generative world model can hallucinate (encoder, dynamics, decoder), without auxiliary networks or labels.

Coverage-aware training and data collection. A growing body of work argues that data scale and composition are first-order levers for generative model performance [Hoffmann et al., 2022, Gadre et al., 2023]. In offline RL specifically, data coverage is known to bound policy improvement [Levine et al., 2020, Kumar et al., 2020], motivating curated datasets such as ExoRL [Yarats et al., 2022], RL Unplugged [Gulcehre et al., 2020], and V-D4RL [Lu et al., 2023]. Curiosity-driven exploration is the natural online counterpart: agents are incentivized to visit states with high prediction error [Pathak et al., 2017], high feature-network novelty [Burda et al., 2019], or high model disagreement [Pathak et al., 2019], scaled to imagined trajectories in Plan2Explore [Sekar et al., 2020]. Whereas prior curiosity work uses these signals to drive *single-task policy exploration*, we adapt them to *data-collection for generative world modeling* in two complementary ways: *(i)* uniform-task resampling closes a substantial fraction of the hallucination gap at no additional data cost, and *(ii)* using our proposed metrics as curiosity rewards yields a data-efficient finetuning recipe that generalizes a 350M-parameter world model to entirely unseen environments with as few as 50 trajectories from the target task.

B Comparison to Existing Datasets

Table 4 compares MMBench2 to a representative set of prior datasets used for offline reinforcement learning, robot imitation learning, and large-scale generative video and world modeling. The offline RL datasets we compare against include RL Unplugged [Gulcehre et al., 2020], V-D4RL [Lu et al., 2023], ExoRL [Yarats et al., 2022], the TD-MPC2 multi-task dataset [Hansen et al., 2024], and the Atari DQN Replay dataset [Agarwal et al., 2020]. The robot imitation learning datasets we compare against are RoboNet [Dasari et al., 2019], BridgeData V2 [Walke et al., 2023], DROID [Khazatsky et al., 2024], and Open X-Embodiment [Open X-Embodiment Collaboration et al., 2023]. For large-scale video pretraining corpora with pseudo-labeled actions, we compare against VPT [Baker et al., 2022] and NitroGen [Magne et al., 2026]. Finally, we include MMBench [Hansen et al., 2026], the multi-task benchmark on which MMBench2 builds. Across these datasets, MMBench2 is the only corpus that simultaneously offers ground-truth action *and* reward labels, live simulators for every task, mixed-quality behavior, and broad coverage across both task domains and embodiments.

Table 4. **MMBench2 vs. existing datasets.** Our dataset consists of mixed-quality data spanning a larger number of tasks and domains, complete with ground-truth action and reward labels as well as live environments. We summarize key characteristics of MMBench2 and existing datasets below. [a]Per-trajectory binary success flag only. [b]Actions are pseudo-labels. [c]Estimated from reported 40k hours.

Dataset	Tasks	Domains	Trajs	Frames	Resolution	Action	Reward	Live env.	Behavior
RL Unplugged	66	4	—	12B	Varies	✓	✓	✓	Replay
V-D4RL	3	1	—	6M	64×64	✓	✓	✓	Mixed
ExoRL	11	1	—	45M	State	✓	✓	✓	Exploratory
TD-MPC2	80	2	—	545M	State	✓	✓	✓	Replay
Atari DQN Replay	60	1	—	15B	84×84	✓	✓	✓	Replay
RoboNet	—	1	162k	15M	128×128	✓	✗	✗	Scripted
BridgeData V2	13	1	60k	2.3M	640×480	✓	✗	✗	Demos
DROID	86	1	76k	19M	1280×720	✓	∼[a]	✗	Demos
Open X-Embod.	527	22	1M+	—	Varies	✓	✗	✗	Demos
VPT	—	1	—	5B	128×128	∼[b]	✗	✓	Human
NitroGen	1,000+	—	39k	4B[c]	256×256	∼[b]	✗	✓	Human
MMBench	200	10	4k	1.8M	224×224	✓	✓	✓	Demos
MMBench2	**210**	**10**	**65.6k**	**23M**	**224×224**	✓	✓	✓	**Mixed**

C Task Domains

We consider **210** tasks across **10** domains. Our task set is comprised of diverse continuous control tasks spanning robot manipulation, locomotion, navigation, arcade games, and classic control problems, each varying in task complexity, time horizon, observation and action space dimensionality, and reward formulation. Table 5 provides an overview of our task domains.

Table 5. **Overview of task domains.** Our dataset covers a wide range of task types, state and action dimensionalities, time horizons, and reward formulations. Table courtesy of Hansen et al. [2026].

Task domain	Tasks	Observation	Action dim		Ep. length		Reward
			min	max	min	max	
DMControl	23	224 × 224	1	12	500	500	dense/sparse
DMControl Ext.	16	224 × 224	1	7	500	500	dense/sparse
Meta-World	49	224 × 224	4	4	100	100	dense
ManiSkill3	37	224 × 224	1	12	25	500	dense/sparse
MuJoCo	6	224 × 224	1	8	50	1000	dense/sparse
MiniArcade	24	224 × 224	1	2	200	500	dense/sparse
Box2D	8	224 × 224	2	4	500	500	dense
RoboDesk	6	224 × 224	5	5	100	100	dense
OGBench	14	224 × 224	2	8	100	1000	dense
Atari	27	224 × 224	3	3	1000	1000	sparse

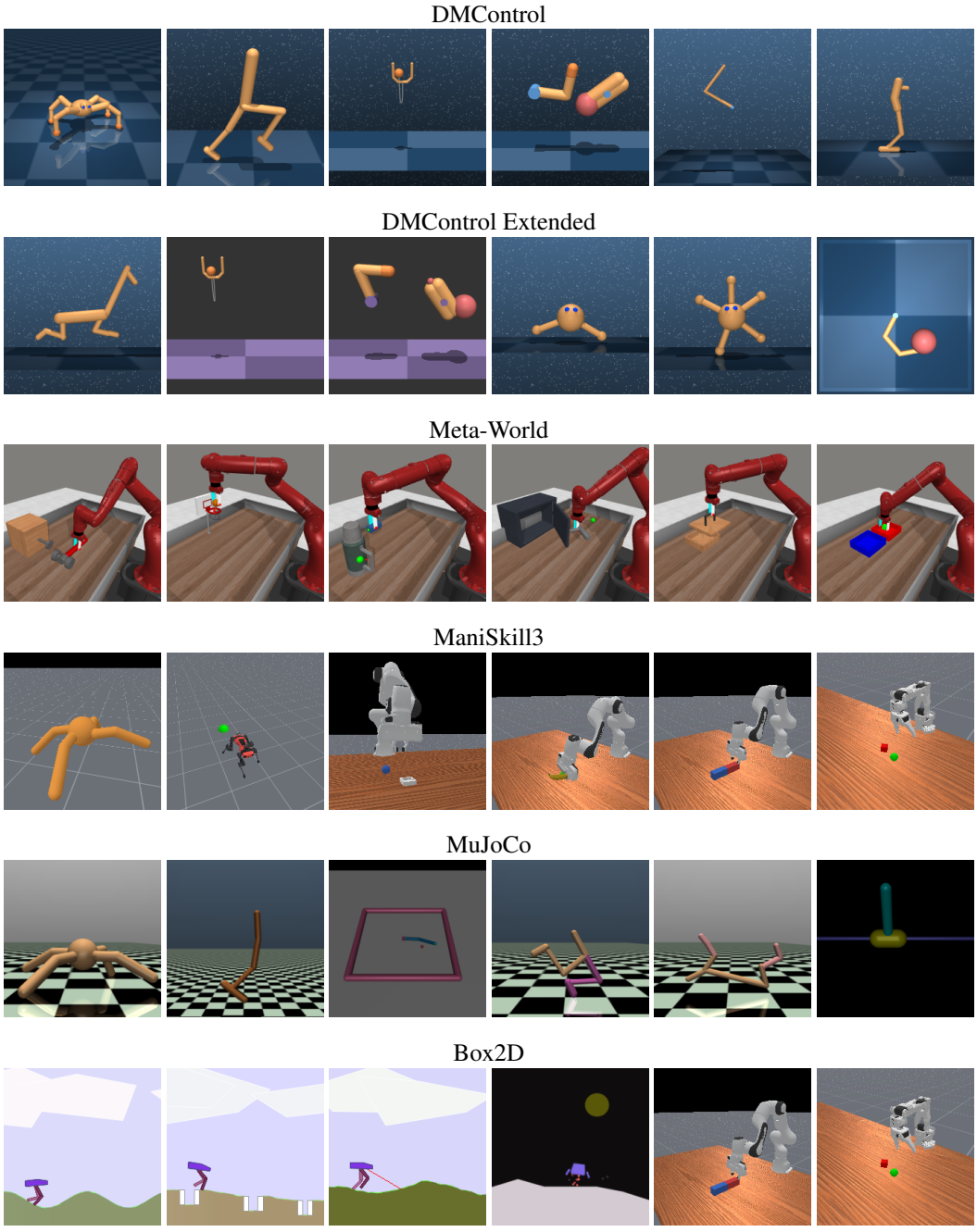


Figure 7. **Visualization of task domains (1 of 2).** We show sample tasks from each of the 10 task domains that we consider.

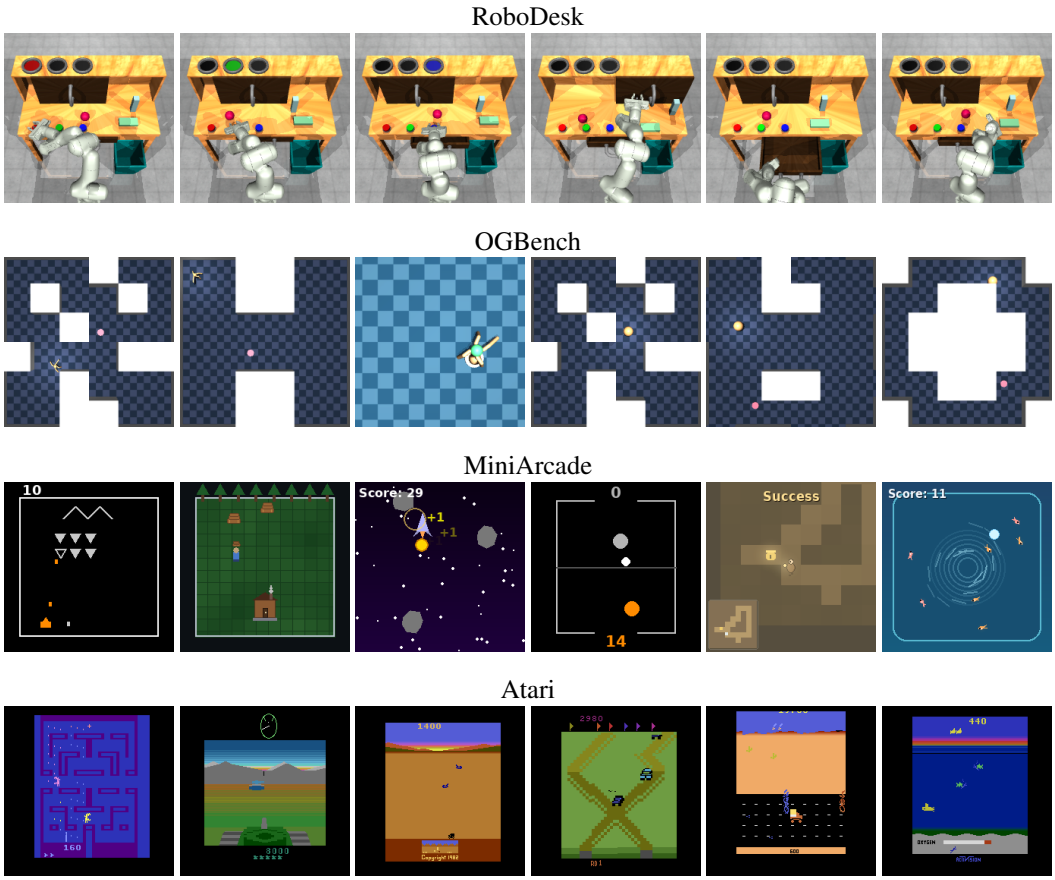


Figure 8. **Visualization of task domains (2 of 2).** We show sample tasks from each of the 10 task domains that we consider.

Table 6. **Task list for finetuning experiments.** We finetune on a set of 10 seen (also used in pretraining) + 10 unseen (held out) tasks. The two task sets are listed below. Note that unseen tasks are developed specifically for MMBench2.

Task	Domain
Seen task set	
cup-catch	DMControl
finger-turn-easy	DMControl
mw-push	Meta-World
ms-push-cube	ManiSkill
lunarlander-hover	Box2D
og-point-maze	OGBench
og-point-bottleneck	OGBench
pygame-point-maze-var1	MiniArcade
pygame-pong	MiniArcade
pygame-bird-attack	MiniArcade
Unseen task set	
cup-catch-var1	DMControl
finger-turn-easy-var1	DMControl
ms-push-banana	ManiSkill
og-point-var1	OGBench
og-point-var2	OGBench
pygame-point-maze-var4	MiniArcade
pygame-reacher-easy	MiniArcade
pygame-dungeon-explorer1	MiniArcade
pygame-foraging	MiniArcade
pygame-whirlpool	MiniArcade

D Data Collection

We base our data collection on that of MMBench, a 200-task benchmark designed primarily for online RL; it provides a total of 4k expert demonstrations collected via single-task expert policies, as well as live environments with ground-truth action and reward labels for all tasks. While MMBench is well suited for the multi-task online RL setting it was originally developed for, a dataset that consists solely of expert demonstrations lacks diversity in terms of behavior which, as our experiments show, is a significant source of hallucination in world models.

The overarching goal of MMBench2 is to produce a large, diverse dataset for visual world modeling and research in common failure modes such as hallucination. To do so, we use the single-task expert policies $\pi^\star$ of MMBench as a basis for our data collection, but crucially augment policies to generate diverse behaviors, and also collect data via non-expert policies, including humans. Our methods of data collection can be summarized as follows:

- **Random policy.** Sample actions uniformly in $[-1, 1]$. Diverse actions; poor task performance.
- **No-op actions.** Set actions $\mathbf{a} = \mathbf{0}$. Models dynamics without agent interference.
- **Expert actions.** Sample from the expert policy $\pi^\star$ without added noise. High task performance; low data diversity despite it being a stochastic policy.
- **Transformed expert actions.** Sample from $\pi^\star$, then apply any of five transforms with some probability: scale by $\varepsilon \in [0, 1]$, action dropout, flip sign, all-zero action, repeat previous action. The same transform may be applied for multiple steps. Provides counterfactual transitions.
- **Structured noise.** Sample from $\pi^\star$, then apply any of three noise types with parameters sampled on a per-episode basis: Gaussian noise, Ornstein-Uhlenbeck noise (temporally correlated), convex mixture with random policy. Mixed performance; improves data diversity.
- **Curiosity-driven.** Sample trajectories that maximize our developed hallucination metric u_r^{norm}, selected by CEM-based planning with the pretrained world model.
- **Human play data.** Actions are selected by a human interacting with the environment via a keyboard interface. High data diversity; not task-driven.

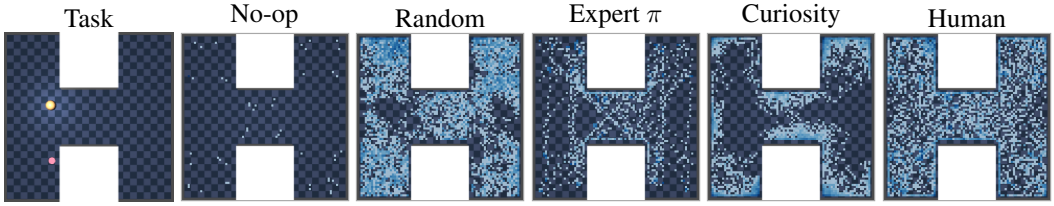
With the exception of human play data, we frequently switch between any of the above behaviors within a single episode to maximize diversity of our training data. For example, we may apply structured noise at random for a number of steps to visit a less frequently visited part of the state-action space and then switch back to the expert policy to generate recovery behavior. Data is collected across all 210 live environments, and ground-truth actions (the action selected by the behavior policy) and reward labels (task reward obtained from the environment) are logged. We collect pretraining data for 200 tasks, and additional data for targeted finetuning across 10 seen (included in pretraining) as well as 10 unseen tasks, with data partitions separated by behavior policy: *random*, *noop*, *expert*, *mixed*, *curiosity*, and *human*. Figure 9 shows a screenshot of the interface used to collect human play data.

Figure 9. **Web interface for human play data collection.** We develop a simple web interface for data collection. A human user interacts with the environment using the keyboard, and interaction data is saved and later used for world model training. We collect a total of $1{,}400$ trajectories using this interface.

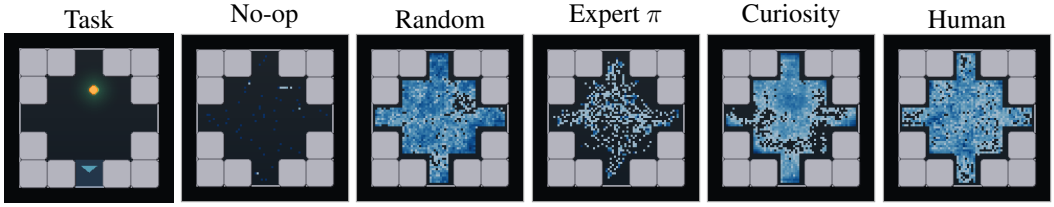
E Additional Results

Table 7. **Detecting hallucination events.** Per-task AUROC against two hallucination labels (action-ignored, scene-diverging) on held-out test data from all 200 pretraining tasks. Our three proposed metrics $u_r^{\text{norm}}, u_f^{\text{norm}}, u_s^{\text{norm}}$ reliably predict hallucinations. Higher is better (↑).

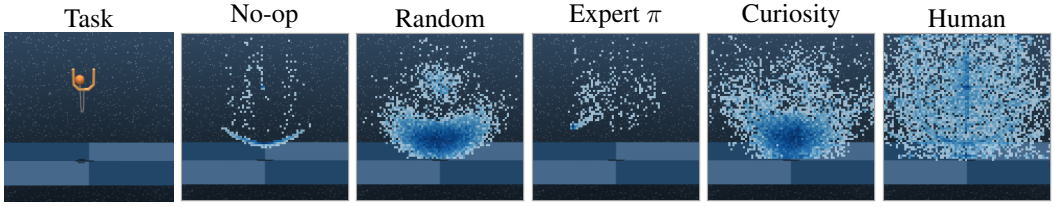
Predictor	Action ignored	Scene divergent
Tokenizer residual u_r^{norm}	**0.887**	0.919
Flow instability u_f^{norm}	0.868	**0.939**
Inter-seed variance u_s^{norm}	0.873	0.934
Scene motion (latent) m	0.803	0.927
kNN distance (global)	0.814	0.731
Flow instability u_f (raw)	0.752	0.854
n_{frames} (baseline)	0.596	0.534



Point Maze Bottleneck (OGBench)



Point Maze Variation 4 (MiniArcade)



Cup Catch (DMControl)

Figure 10. **Data coverage by collection method.** We show state densities for five online data collection policies (no-op, random, expert, curiosity, and human) across three additional tasks besides those in Figure 6.

Table 8. **Targeted data collection for finetuning on 10 unseen tasks (*expert* test set).** We finetune our world model on a set of 10 seen + 10 unseen tasks, varying data source and finetuning strategy. Each data source contains 50 trajectories per task. The offline results in Table 2 are computed over a test set that consists of trajectories from both expert policies and human play; this table complements those results by evaluating only on the *expert* trajectories. We find that *the policy that aligns best with the test set performs the best in evaluations*. Task performance is measured via closed-loop MPC using CEM; we report mean performance normalized to $[0, 1]$ across 3 episodes per task.

Method	Tok FT	Dyn FT	Recon PSNR ↑	Rollout ΔPSNR ↑	Action shuf. ↑	u_r^{norm} ↓	Task perf. (MPC) ↑
Random policy			—	—	—	—	0.118
Base	✗	✗	17.29	−14.19	1.08	4.884	—
Ours, pre-FT	✗	✗	17.07	−14.22	1.20	4.591	0.276
No-op actions	✗	✓	17.09	−13.40	1.27	5.052	—
No-op actions	✓	✓	34.73	−0.09	1.38	1.843	0.163
Random policy	✗	✓	17.07	−13.03	1.50	5.058	—
Random policy	✓	✓	35.59	+1.74	1.72	1.423	0.228
Expert policy	✓	✓	37.05	+2.64	1.95	1.281	0.362
Human play	✓	✓	36.37	+2.20	1.90	1.185	0.362
Curiosity (u_r^{norm})	✓	✓	36.63	+2.66	1.75	1.342	0.325
All (combined)	✓	✓	**38.20**	**+3.41**	**2.11**	**1.090**	**0.390**

Table 9. **Targeted data collection for finetuning on 10 unseen tasks (*human play* test set).** We finetune our world model on a set of 10 seen + 10 unseen tasks, varying data source and finetuning strategy. Each data source contains 50 trajectories per task. The offline results in Table 2 are computed over a test set that consists of trajectories from both expert policies and human play; this table complements those results by evaluating only on the *human play* trajectories. We find that *the policy that aligns best with the test set performs the best in evaluations*. Task performance is measured via closed-loop MPC using CEM; we report mean performance normalized to $[0, 1]$ across 3 episodes per task.

Method	Tok FT	Dyn FT	Recon PSNR ↑	Rollout ΔPSNR ↑	Action shuf. ↑	u_r^{norm} ↓	Task perf. (MPC) ↑
Random policy			—	—	—	—	0.118
Base	✗	✗	17.44	−10.70	1.16	2.835	—
Ours, pre-FT	✗	✗	17.35	−10.82	1.39	2.947	0.276
No-op actions	✗	✓	17.32	−9.92	1.56	3.298	—
No-op actions	✓	✓	34.75	+1.41	1.71	1.128	0.163
Random policy	✗	✓	17.35	−9.55	1.96	3.276	—
Random policy	✓	✓	36.04	+3.59	2.29	0.980	0.228
Expert policy	✓	✓	34.68	+3.04	2.14	0.982	0.362
Human play	✓	✓	**37.84**	**+5.58**	**2.93**	**0.820**	0.362
Curiosity (u_r^{norm})	✓	✓	35.47	+3.35	2.24	0.946	0.325
All (combined)	✓	✓	37.61	+4.63	2.57	0.861	**0.390**

Table 10. **Targeted data collection for finetuning on 10 *seen* tasks.** We finetune our world model on a set of 10 seen + 10 unseen tasks, varying data source and finetuning strategy. Each data source contains 50 trajectories per task. Offline metrics were computed using a test set of expert trajectories and human play data in equal amount. Task performance is measured via closed-loop MPC using CEM; we report mean performance normalized to $[0, 1]$ across 3 episodes per task.

Method	Tok FT	Dyn FT	Recon PSNR ↑	Rollout ΔPSNR ↑	Action shuf. ↑	u_r^{norm} ↓	Task perf. (MPC) ↑
Random policy			—	—	—	—	0.083
Base	✗	✗	37.70	+3.00	1.59	1.325	—
Ours, pre-FT	✗	✗	38.18	+4.24	1.84	1.133	0.256
No-op actions	✗	✓	38.18	+4.87	2.11	1.113	—
No-op actions	✓	✓	39.04	**+5.30**	2.04	1.039	0.316
Random policy	✗	✓	38.14	+4.46	2.07	1.063	—
Random policy	✓	✓	39.09	+5.17	2.07	1.057	0.286
Expert policy	✓	✓	38.71	+5.21	**2.16**	**0.982**	0.355
Human play	✓	✓	39.00	+5.05	**2.16**	1.036	**0.384**
Curiosity (u_r^{norm})	✓	✓	38.96	+5.24	2.13	1.015	0.332
All (combined)	✓	✓	**39.21**	+5.09	2.08	1.019	0.308

Table 11. **Effect of reward finetuning.** Results for two variants that both extend the pretrained base model with 30k additional dynamics steps; one jointly trains the dynamics model and a reward head (backpropagating gradients from the reward back into dynamics), and the other is a reward-free control. Mean over all 200 pretraining tasks on a held-out test set. We do not observe a significant difference in results as a result of finetuning with rewards.

Metric	w/o reward	**w/ reward**	Δ_{rew}
Recon PSNR (dB) ↑	35.67	35.68	+0.01
Action-shuffle ratio ↑	1.67	1.62	−0.05
Rollout ΔPSNR (dB) ↑	3.01	3.14	+0.13
u_r^{norm} ↓	1.306	1.318	+0.012
u_f^{norm} ↓	0.304	0.286	−0.018
u_s^{norm} ↓	0.515	0.510	−0.005

F Implementation Details

Our Dreamer 4 world model is a reproduction of the original method as described in Hafner et al. [2025]. This section provides an overview of our implementation and design choices.

Language embeddings. For task conditioning we use frozen text embeddings from `openai/clip-vit-base-patch32` (CLIP; Radford et al. [2021]), which produces 512-dimensional continuous embeddings of per-task language instructions. Only the reward prediction and BC policy heads are conditioned on language embeddings.

Block-causal Transformer backbone. Each Transformer is a stack of block-causal layers consisting of *(i)* space self-attention over the per-frame token sequence, *(ii)* causal time self-attention along the temporal axis, and *(iii)* a SiLU-gated MLP with ratio 4. Attention uses RoPE [Su et al., 2024] on Q/K with a KV-cache-aware offset, QK-normalization [Henry et al., 2020], and RMSNorm [Zhang and Sennrich, 2019] pre-norm with no biases on the normalization layers.

Modality-aware mask. Within space self-attention, we apply a modality-aware mask that depends on the role of each token. In the tokenizer encoder, latent queries attend to all tokens while patch queries only attend within the image modality, preventing patch tokens from mixing across modalities before being bottlenecked through the latents; in the decoder, the directions are reversed so patch queries can read from the latent bottleneck but not from each other directly. In the dynamics model, action, shortcut, spatial, and register tokens are mutually visible while agent tokens (reward head and BC policy) are asymmetrically isolated: agent queries attend to everything, but non-agent queries do not see agent keys.

Spatial packing. The tokenizer produces per-frame latents of shape $(n_L, d_b) = (64, 64)$. Before being fed to the dynamics, these are spatially packed at factor $k{=}2$ to shape $(n_{\text{spatial}}, d_{\text{spatial}}) = (32, 128)$, halving attention cost along the spatial axis at the cost of doubling channel dimension; the inverse `unpack` is applied at decode time. Concretely, the dynamics model sees the following token layout per timestep: `[ACTION x 1, SHORTCUT x 1, SPATIAL x 32, REGISTER x 4, AGENT x 4]` where the action token is produced by a 2-layer MLP, the shortcut-conditioning token concatenates two embeddings of the discretized noise level σ and step size $d{=}1/2^{\text{step}}$, register tokens [Darcet et al., 2024] are 4 learnable "sink" tokens, and agent tokens are initialized from the per-task CLIP embedding broadcast over time.

Loss normalization. Pixel MSE and LPIPS terms in the tokenizer, the empirical and self-consistency branches of the dynamics objective, and reward two-hot cross-entropy are each separately normalized by their own running RMS before weighting. This decouples loss weights from absolute scale and removes the need to tune them when the dataset, resolution, or backbone changes.

Shortcut flow matching. The dynamics model is trained with the shortcut flow-matching objective of Frans et al. [2025]. The discretized noise level σ is indexed by an integer in $\{0, \dots, k_{\max}\}$ (with $k_{\max}{=}64$ in our experiments; 0 is pure noise, $k_{\max}$ is clean) and the step is indexed by an integer in $\{0, \dots, \log_2(k_{\max})\}$ corresponding to step size $d{=}1/2^{\text{step}}$. For a fraction $\rho_{\text{self}}{=}0.25$ of each batch we apply a self-consistency bootstrap: at (σ, step), two coarser half-steps at $\text{step}{+}1$ are run under `no_grad` and their averaged velocity is used as a stop-gradient target for the current step's predicted velocity. The remaining 0.75 of the batch uses the empirical one-step regression term at the finest step.

Reward and BC heads. Both heads read from the agent tokens via attention pooling against a learnable query. The reward head predicts $L{=}8$ multi-step symlog two-hot distributions over 255 bins on the range $[-10, 10]$. The BC head is a deterministic Gaussian policy with diagonal covariance trained via an MSE loss on the ground-truth 16-dimensional padded action.

Sampling. At inference we run a shortcut Euler integrator with step size $d{=}0.125$ (*i.e.*, $K{=}8$ substeps) for each new frame: starting from $z \sim \mathcal{N}(0, I)$, we solve $b = (\hat{x}_1 - z)/(1 - \sigma)$ followed by $z \leftarrow z + b \cdot d$. We apply context corruption to past tokens.

Hyperparameters. We summarize our hyperparameters in Table 12.

Table 12. **Hyperparameters.** Key hyperparameters used to train our world model.

Hyperparameter	Value
Data	
Image resolution	$224 \times 224 \times 3$
Action dim. (zero-padded)	16
Tokenizer (~100M)	
Patch size	14
Embedding dim. (d_{model})	512
Heads	8
Depth	12
MLP ratio	4
Number of latents (n_L)	64
Bottleneck dim. (d_b)	64
MAE keep range	$[0.1, 1.0]$
LPIPS weight	0.2
Dynamics (~230M)	
Embedding dim. (d_{model})	1024
Heads	8
Depth	16
MLP ratio	4
Spatial packing factor	2
Register tokens	4
Agent tokens	4
Self-consistency fraction	0.25
Context corruption (τ_{ctx})	0.1
Reward + BC heads (~20M)	
Multi-step horizon (L)	8
Reward bins	255
Reward symlog range	$[-10, 10]$
Optimization	
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	1×10^{-2}
Sequence length	24
Effective batch size	Tok: 96 / Dyn: 512
Sampling (inference)	
Integrator schedule	Shortcut
Step size (d)	0.125
Planning (CEM)	
Plan horizon (H)	32
Replan every (K)	16
CEM Iterations	3
Population size	32
Rollouts per candidate	2
Warm start (mean)	BC prior