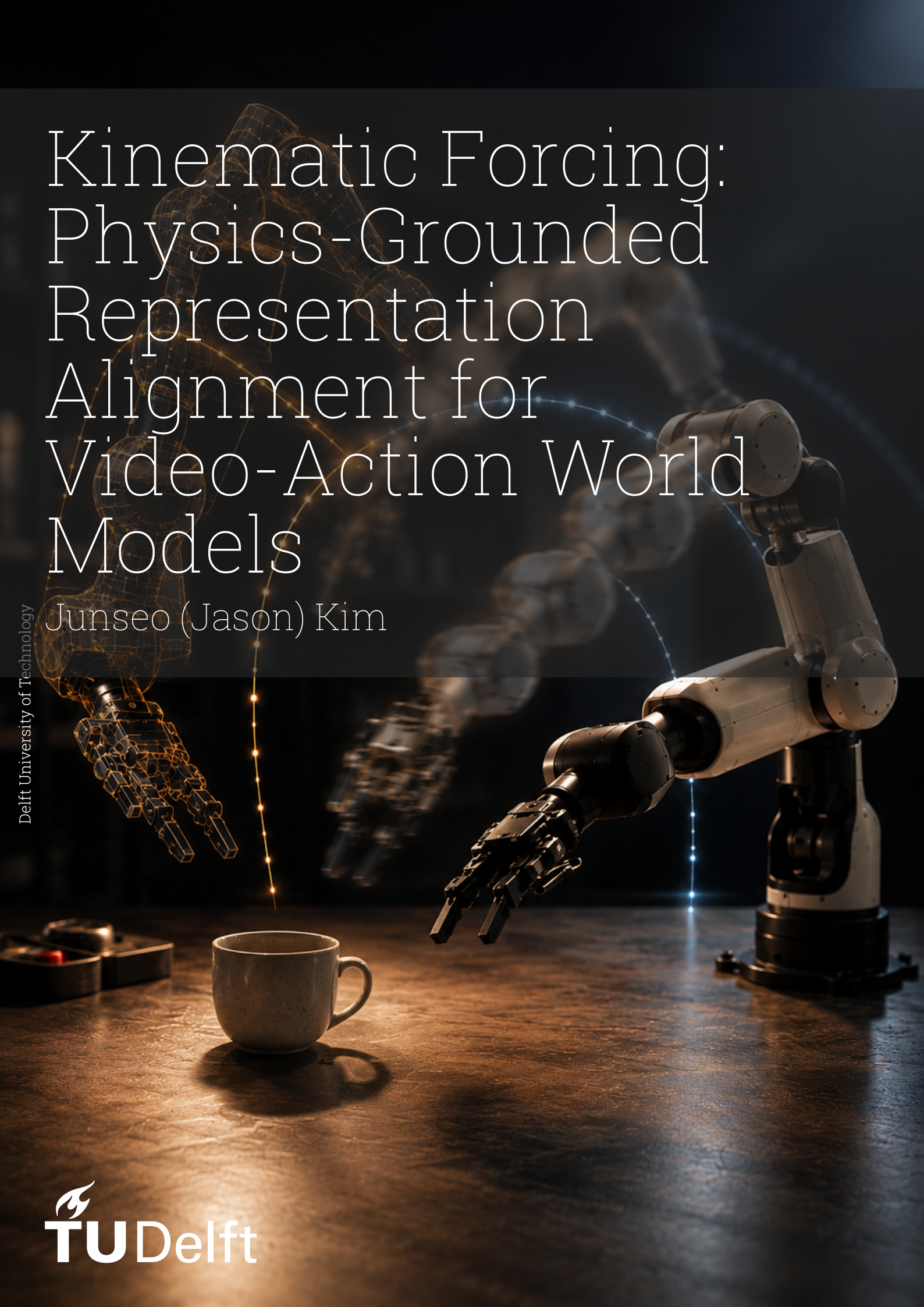
Kinematic Forcing: Physics-Grounded Representation Alignment for Video-Action World Models

Junseo (Jason) Kim



Delft University of Technology

TU Delft

Master of Science Robotics

Thesis research

Kinematic Forcing: Physics-Grounded Representation Alignment for Video-Action World Models

by

Junseo (Jason) Kim

Student Number: 6163246

Supervisors:

Dr. Javier Alonso-Mora	– Supervisor
Dr. Jean Oh	– External Supervisor

Date: July 9, 2026

Kinematic Forcing: Physics-Grounded Representation Alignment for Video-Action World Models

Junseo (Jason) Kim
Student Number: 6163246
Department of Cognitive Robotics
Delft University of Technology, The Netherlands

Abstract: Joint video-action models learn a shared representation to predict future video frames and robot actions, with decoupled heads that allow the video head to be dropped during inference for real-time policy execution. However, this decoupling means the two heads are trained independently against ground truth, with no mechanism for one head's predictions to inform the other. We observe that the video head produces structurally incorrect robot poses, especially during dynamic motion. We introduce *Kinematic Forcing* (KF), a simple training objective that grounds this shared representation in the robot's physical structure, using the action head's own predicted pose as the source of that structure. From the action head's predicted pose, we recover joint angles via inverse kinematics, render the robot with its URDF model, and composite it into the scene with an inpainter. We encode this kinematic composite using a frozen DINOv2 encoder and force the backbone's shared representation to align with the features at robot-token positions using a physics-grounded, action-derived target. The entire rendering and alignment pipeline is training-only and removed at inference, leaving the deployed model identical to the baseline. Across simulated (WidowX, Google Robot) and real-world (xArm7) tasks, KF improves video prediction quality at robot regions, with the largest gains under fast motion, while maintaining downstream policy success.

1 Introduction

Recent work has shown that training a single backbone to simultaneously predict visual scenes and robot controls produces representations that benefit both tasks [1, 2, 3]. The Unified Video Action (UVA) framework [1] takes this further with a decoupled architecture: a video head and an action head each operate on the backbone's latent $\mathbf{z}$ but can be deployed independently, allowing the video head to be discarded for real-time control. This decoupling is practical but comes at a cost: the two heads are trained with separate losses against ground truth, with no mechanism to enforce consistency between their predictions.

This split has a visible consequence. During fast, dynamic motion, the video head produces deformed, blurred, or ghosted robot arms, even though the coarse spatial layout stays roughly correct. The video head treats the robot as arbitrary pixels and has no notion of its physical structure. Yet a robot arm is a rigid-body kinematic chain: given joint angles, its configuration and visual appearance are fully determined. This physical prior is available through the action head's predictions, but is never used to shape the representation that drives video generation.

Our key observation is that both heads predict the same future from the shared backbone: the video head predicts the frame at $t{+}1$, while the action head predicts the action at $t{+}1$. From that predicted action, we can recover the robot's joint configuration via its kinematic chain and render exactly how the arm should appear, yielding a physics-grounded picture of what the video head is trying to draw.

We propose *Kinematic Forcing* (KF), which uses this render to ground the shared representation in physical structure.

We pass the kinematic render through a frozen DINOv2 encoder [4] and train the shared representation to match the resulting features at the robot-token positions (Sec. 4.3). Prior representation alignment methods use the clean input image as the target; instead, we use the kinematic render of the robot at the action head's predicted pose, so the target carries the robot's physical structure and depends on the predicted action. This alignment runs only during training and only in one direction: the action head's prediction shapes the representation the video head reads, and adds nothing at inference.

This raises two questions. First, can this action-derived structural signal improve the video head's ability to generate physically correct robot predictions, particularly during fast motion, where it most often fails? Second, since the aligned representation is shared with the action head, does grounding it in render structure come at a cost to downstream control? We evaluate on the WidowX arm in SimplerEnv [5] (Bridge), a long-horizon Google Robot task [6], and a real-world xArm7, and show that robot-region video quality improves while downstream policy is left unchanged (Sec. 5).

Contributions.

1. We observe that the video and action heads of decoupled video-action models make independent predictions about the same future, and propose *Kinematic Forcing*, a representation-grounding objective that bridges the action and video spaces through kinematic rendering.
2. We show that a frozen inpainter can bridge the domain gap between URDF renders and real images, enabling kinematic rendering as an alignment target.
3. We show that this physics-grounded alignment improves video quality at robot regions without systematically degrading downstream policy success.

2 Related Works

2.1 Video-Action Models for Robot Learning

Video prediction and action prediction for robot control have been coupled in two sequential directions: video-to-action, where a generated video is decoded into actions through an inverse dynamics model [7], and action-to-video, where action sequences condition video generation [8]. Both are directional pipelines in which one modality conditions the other.

A recent line of work instead trains both modalities jointly within a shared backbone. UVA [1] learns a joint video-action latent representation with two lightweight diffusion heads that can be decoupled at inference, and UWM [2] couples action and video diffusion within a single transformer with independent noise schedules. WorldVLA [3] and UD-VLA [9] unify video and action prediction inside a single autoregressive or diffusion transformer, DreamZero [10] scales this to a 14B diffusion backbone for real-time closed-loop control, and Cosmos Policy [11] adapts a pretrained video model into a single-stage policy that jointly emits actions, future states, and value estimates. In all of these models, however, the video and action heads are trained independently against the ground truth, and Kinematic Forcing addresses this gap by allowing the action head's structural prediction to shape the shared representation.

2.2 Representation Alignment in Generative Models

Joint-Embedding Predictive Architectures learn by predicting in representation space rather than reconstructing raw pixels. I-JEPA [12] predicts masked image patches in the latent space of a ViT encoder, and V-JEPA [13] extends this to video. We borrow the principle of asymmetric projection into a shared representation space for our alignment design.

REPA [14] is the most direct precedent: it aligns intermediate diffusion transformer features with a frozen DINOv2 [4] encoder through a learned projector and token-wise cosine similarity, accel-

erating convergence by over an order of magnitude. Geometry Forcing [15] adapts REPA to video diffusion by replacing DINOv2 with the 3D geometric foundation model VGGT [16]. Both align internal features against a frozen external encoder applied to the frame that the model is generating. Kinematic Forcing uses the same alignment mechanism but targets a different object: a kinematic render of the robot at the action head's predicted pose, rather than the clean frame (Sec. 4). EVA [17] also enforces consistency between a video world model and physics, but does so at the episode level with an inverse-dynamics reward rather than per-token during training.

2.3 Physical Structure and Rendering for Robot Learning

Several methods make the robot's visual appearance differentiable with respect to its control parameters. Dr.Robot [18] combines a kinematic-aware deformable model with Gaussian Splatting, and Prof.Robot [19] extends this with collision-aware regularization for visuomotor control. DynaRend [20] uses differentiable volumetric rendering of masked future frames to jointly learn semantics, geometry, and dynamics. These methods require the rendering pipeline itself to be differentiable. Kinematic Forcing takes a complementary approach: the rendering pipeline is entirely frozen and need not be differentiable, serving only to produce an alignment target. Gradients flow through the projector into the shared backbone, not through the renderer.

3 Preliminaries

Unified Video-Action Model. We build on the Unified Video Action (UVA) framework [1], which jointly predicts future video frames and robot actions from a shared Masked Autoregressive (MAR) backbone. An input sequence of T_{obs} observed frames is VAE-encoded into latent tokens $\mathbf{x} \in \mathbb{R}^{T \times S \times d_{\text{vae}}}$, where S is the number of spatial tokens per frame and d_{vae} is the VAE latent dimension. A subset of future-frame tokens is randomly masked (mask ratio 70–100%), and the backbone with 16-block encoder and 16-block decoder transformer predicts the masked tokens from visible context, producing a latent representation $\mathbf{z} \in \mathbb{R}^{T \times S \times d}$ at the decoder output. We use 128×128 inputs with a VAE stride of 16, giving $S = 8 \times 8 = 64$ tokens per frame.

Two lightweight, decoupled heads operate on $\mathbf{z}$:

- **Video head**: denoises each spatial token with diffusion, predicting future VAE latents. Trained with an ϵ-prediction MSE loss $\mathcal{L}_{\text{video}}$.
- **Action head**: aggregates spatial features and predicts a chunk of future actions with diffusion. Trained with $\mathcal{L}_{\text{action}}$.

During inference, the video head can be dropped for real-time policy execution, leaving only the action head needed.

Notation. We use $\mathbf{z}$ for the backbone decoder output (the shared representation both heads read), C for the composite image produced by the kinematic rendering pipeline (Sec. 4.1), g_ϕ for the frozen DINOv2 encoder, and f_{kin} for the learned projector that maps $\mathbf{z}$ into DINOv2 feature space where the alignment loss is computed.

4 Kinematic Forcing

We now describe Kinematic Forcing in detail: how the action head's prediction is converted into a visual reference (Sec. 4.1, 4.2), how that reference becomes a DINOv2 alignment target for the shared representation (Sec. 4.3), why we pair the alignment with a spatial video head (Sec. 4.4), and how the combined objective is structured (Sec. 4.5).

4.1 Kinematic Rendering Pipeline

Given the action head's predicted end-effector pose, we construct a scene-consistent image of the robot at that pose on the real scene background.

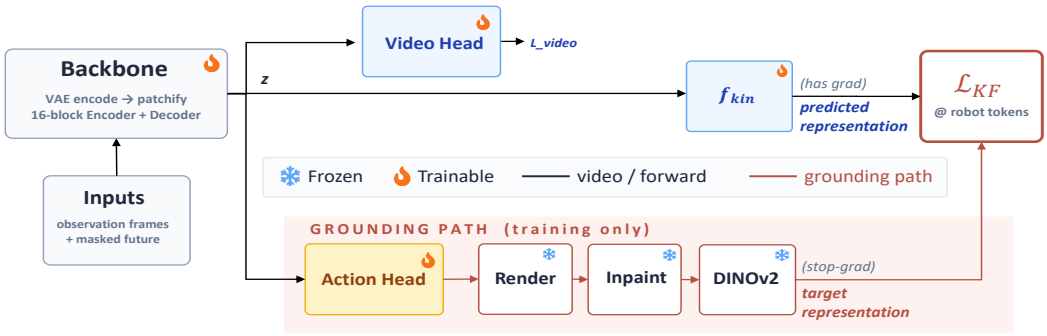


Figure 1: Overview of Kinematic Forcing. The grounding path (red box) renders the robot at the action head's predicted pose, composites it into the scene with a frozen inpainter, and encodes it with a frozen DINOv2 to form the target $\mathbf{c} = g_\phi(C)$. The backbone's shared representation $\mathbf{z}$ is projected via a learned projector f_{kin} and aligned with $\mathbf{c}$ at robot-token positions using cosine similarity.

Action to joint angles. The action head produces a predicted action trajectory via diffusion sampling from $\mathbf{z}$. From each predicted end-effector pose, we recover joint angles through numerical inverse kinematics (IK).

URDF rendering. The solved joint angles are passed to a rasterizer (nvdiffrast [21]) that renders the robot's URDF mesh, producing a rendered RGB image R and a binary silhouette mask M^{pred}.

4.2 Scene-consistent Compositing via a Frozen Inpainter

The kinematic alignment loss (Sec. 4.3) requires a target image depicting the robot at the action-predicted pose superimposed on the real scene. The naive approach overlays the URDF render directly onto the ground-truth future frame. We find this insufficient and replace the overlay with a lightweight frozen inpainter.

Setup. For each training sample, we take a ground-truth future frame $I \in [0,1]^{3\times H\times W}$ and the action head's predicted end-effector pose. Inverse kinematics recovers the corresponding joint angles, and nvdiffrast [21] rasterizes the robot URDF, yielding a rendered RGB $R \in [0,1]^{3\times H\times W}$ and a binary silhouette $M^{\text{pred}} \in \{0,1\}^{H\times W}$. We also render the robot at the dataset's recorded joint angles to obtain the ground-truth pose silhouette M^{gt}, which indicates where the real robot appears in I and must be removed before compositing.

Why not direct overlay? The URDF render R is geometrically correct, but visually inconsistent with I: it lacks scene-consistent lighting, shadows, and surface reflections. Direct alpha compositing produces a visible domain discontinuity at the silhouette boundary. When this composite is VAE-encoded and compared with the video head's prediction, the domain gap contaminates the alignment signal, and the loss penalizes appearance mismatch rather than structural mismatch.

Union mask removal. Before inpainting, we must remove the real robot from I so it does not interfere with the rendered robot at its predicted pose. We dilate M^{gt} by a 3×3 kernel to account for silhouette boundary differences between the rasterizer and the simulator, then form the union mask $M^{\cup} = \text{clamp}(M^{\text{gt}}_{\text{dilated}} + M^{\text{pred}},\ 0,\ 1)$. The masked frame $\tilde{I} = I \odot (1 - M^{\cup})$ removes both the real and predicted robot regions.

Inpainter architecture. We train a lightweight UNet to reconstruct the GT frame I within $M^{\cup}$. The inpainter receives the masked background, the rendered robot, and the mask, and outputs a visually plausible completion that preserves the render's geometric structure while matching the

scene's visual distribution. Training uses an L1 reconstruction loss, with the masked region weighted at full strength and a 0.1-weighted full-image term for global coherence: $\mathcal{L}_{\text{inp}} = \|(\hat{I} - I) \odot M^{\cup}\|_1 + 0.1 \|\hat{I} - I\|_1$.

The inpainter is trained once on GT joint angles and frozen during $\mathcal{L}_{\text{KF}}$ training. During $\mathcal{L}_{\text{KF}}$ training, it receives predicted pose renders that may differ from the GT pose; it generalizes because it has learned appearance transfer (lighting, texture) rather than pose-specific patterns. Details of the architecture and training details are in Appendix B.

Final composite. The composite target is:

$$C = \tilde{I} + \hat{I} \odot M^{\cup} \tag{1}$$

where $\hat{I}$ is the inpainter's output. C is a scene-consistent image with the robot at the action-predicted pose on the real background. C is VAE-encoded and patchified to produce target tokens $\mathbf{c} \in \mathbb{R}^{S \times d_{\text{vae}}}$. Each step of the inpainter is shown in Figure 2.

4.3 Loss Formulation

Aligning the shared representation. Kinematic Forcing aligns the backbone's shared representation $\mathbf{z}$, which both heads read. Because the signal enters the representation that drives both video and action prediction, a single alignment term grounds the structure that the video head must render and the action head must read.

Frozen DINOv2 target. The composite C is resized to 224×224, normalized, and encoded by a frozen DINOv2 [4]. We take the patch tokens, reshape them to a spatial grid, and average pool to the backbone's token grid, yielding target features $\mathbf{c} = \text{pool}\big(g_\phi(C)\big) \in \mathbb{R}^{S \times d_\phi}$ with $d_\phi = 768$. We use a pretrained representation encoder rather than the model's own VAE latents or raw pixels, which entangle the robot's structure with its appearance; this follows the representation-alignment rationale of REPA and Geometry Forcing [14, 15]. The encoder is frozen, and with a stable reference, the alignment cannot collapse.

Asymmetric projection and cosine alignment. The predictor learns a mapping into the target's space while the target stays fixed. The predictor projects $\mathbf{z}$ into DINOv2 space through a small learned MLP f_{kin} ($d \to d \to d \to d_\phi$); the target side has no learned parameters. We align with cosine similarity, which is scale-invariant and therefore insensitive to magnitude differences between the representation and DINOv2 features. The loss is applied only at spatial positions corresponding to the rendered robot mask M:

$$\mathcal{L}_{\text{KF}} = \frac{1}{|M|} \sum_{i \in M} \Big(1 - \cos\big(f_{\text{kin}}(\mathbf{z}_i),\ \mathbf{c}_i\big)\Big) \tag{2}$$

Restricting the loss to robot tokens focuses the signal where the kinematic prior is informative; background tokens carry no render structure, and the composite background matches the ground truth in those regions.

4.4 Spatial Video Head

A structurally richer $\mathbf{z}$ only improves generated video if the video head can render that structure. The default per-token head (Sec. 3) decodes each 8×8 token independently and cannot compose neighboring tokens into a coherent arm, so the enriched representation has no spatial pathway into the output. We therefore add lightweight intra-frame spatial mixing only to the video head. The added layer is zero-initialized, so the head starts as an exact identity copy of the baseline, and training begins from the same point. The action head and the deployed inference path are untouched; the spatial mixing runs only inside the video head, which is dropped at inference. We pair this added capacity with Kinematic Forcing, which delivers its largest gains in fast motion (Sec. 5.3).

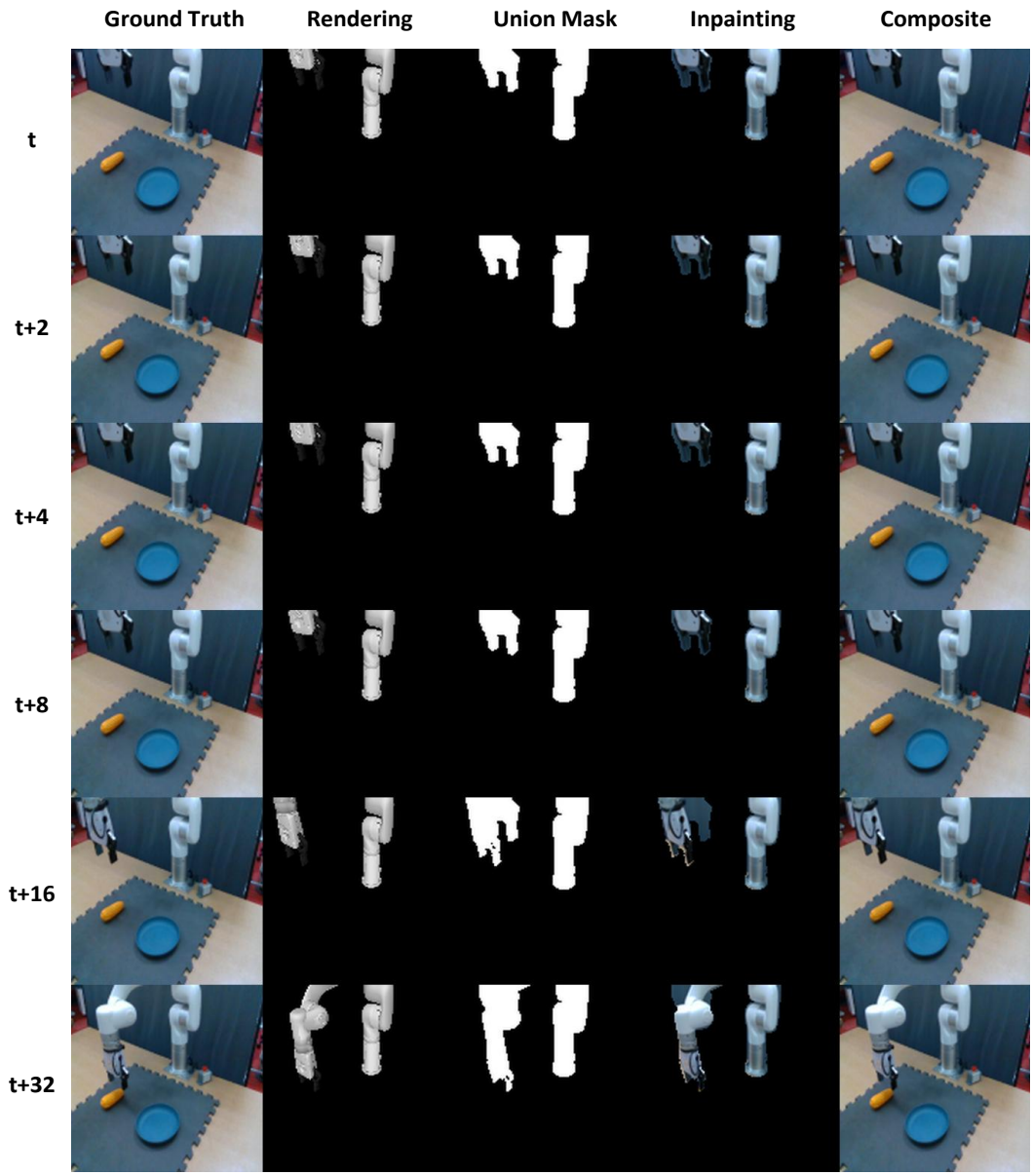


Figure 2: Compositing pipeline. Each column represents ground truth, rendering, union mask against timestep t, inpainting with respect to timestep t, and composite with ground truth background at corresponding timestep. Each row shows a different timestep from t to $t + 32$.

4.5 Training Objective and Gradient Flow

The total training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{video}} + \mathcal{L}_{\text{action}} + \lambda \mathcal{L}_{\text{KF}} \tag{3}$$

where λ controls the alignment strength and $\mathcal{L}_{\text{video}}$, $\mathcal{L}_{\text{action}}$ are unchanged from the baseline. Because the composite is rendered at the action head's predicted pose, which is unreliable early in joint training, we linearly warm λ from 0 over the first 30 epochs.

The gradient of $\mathcal{L}_{\text{KF}}$ follows the path:

$$\mathcal{L}_{\text{KF}} \xrightarrow{\nabla} f_{\text{kin}} \rightarrow \mathbf{z} \rightarrow \text{backbone}$$

The gradient passes through the projector f_{kin} and into the shared representation $\mathbf{z}$, never through the renderer or DINOv2. f_{kin} learns to map $\mathbf{z}$ into DINOv2 space, while the backbone adapts $\mathbf{z}$ to satisfy $\mathcal{L}_{\text{video}}$, $\mathcal{L}_{\text{action}}$, and the structural alignment simultaneously. The video head then reads this enriched $\mathbf{z}$ and, given the spatial head's capacity, renders the structure into the predicted frame.

4.6 Compute and Memory Overhead

Kinematic Forcing adds only the projector f_{kin} as a trainable component, negligible relative to the backbone. The rendering pipeline, DINOv2, and inpainter are entirely frozen and require no optimizer states or gradient buffers. Table 1 summarizes the additional memory, and Table 2 breaks down per-step training latency.

Table 1: Additional memory from Kinematic Forcing components. Only f_{kin} is trainable; all other components are frozen with no optimizer states.

Component	Parameters	VRAM (fp32)	Trainable
f_{kin} projector (MLP)	1.8M	∼7 MB	Yes
DINOv2 (target encoder)	86M	344 MB	No
Inpainter (UNet)	81.9M	328 MB	No
Renderer (mesh buffers)	—	∼5 MB	No
IK solver (Pinocchio [22], CPU)	—	0 MB	No
Total	**169.7M**	**∼684 MB**	

Table 2: Per-step training latency breakdown. All rendering and encoding operations are forward-only (no gradients are computed through them). Measured on two NVIDIA Blackwell RTX6000 Pro with batch size 64.

Operation	Latency (ms)
Action diffusion sampling	400–440
Inverse kinematics	220–270
URDF rendering (nvdiffrast)	60–70
Inpainter forward	3–5
DINOv2 encode composite	∼80
Total $\mathcal{L}_{\text{KF}}$ overhead	**∼800**

The dominant costs are action diffusion sampling and inverse kinematics, both of which are forward-only.

5 Experiments

We evaluate Kinematic Forcing in three settings: (i) the WidowX arm in SimplerEnv [5], trained on Bridge V2 [23]; (ii) a long-horizon Google Robot task [6]; and (iii) a real-world xArm7. In each, we measure video prediction quality at robot regions and downstream policy success. We include the spatial head as a separate configuration to separate its capacity contribution from that of the alignment loss:

- **UVA**: the baseline joint video-action model (per-token video head, no alignment).
- **+ Spatial head**: UVA with the spatial video head (Sec. 4.4) but no alignment loss. Isolates the capacity contribution.
- **+ KF (ours)**: the full method with spatial head and Kinematic Forcing's kinematic grounding.

5.1 Setup

Environments and data. For WidowX, we train on Bridge V2 [23] and evaluate in SimplerEnv [5] on the "Put Carrot on Plate" task, which requires reaching, grasping, and precise placement. For the

Google Robot, we evaluate a long-horizon "Place an apple inside a closed drawer" task in SimplerEnv. In the real-world setting, we use an xArm7 to perform a tabletop task of "Put Corn on Plate". The frozen inpainter is trained separately for each domain on the same data, using ground-truth joint angles (Sec. 4.2).

Training protocol. All models are trained with the action and video heads jointly optimized via an action-conditioned video-prediction mode, so the backbone already learns representations that support both action prediction and action-conditioned video generation. The spatial-head and Kinematic Forcing models are warm-started from the same video only checkpoint. This makes the three configurations a controlled comparison that isolates the spatial head and the alignment loss.

Evaluation. We report success rate for policy and, for video, robot-region full-reference metrics together with a distribution-level FVD (Appendix A): FVD; robot structural similarity (StructSim); robot PSNR, robot SSIM, robot LPIPS; and robot edge F-score. Robot regions are localized from URDF-rendered ground-truth joint masks.

5.2 Video Prediction Quality

Standard pixel metrics favor blurry predictions that hedge across plausible configurations; Kinematic Forcing instead produces sharper, more physically grounded robot structure. We therefore report a complementary set of robot-region metrics that probe quality from different angles: FVD (distribution-level realism), StructSim (semantic robot structure in DINOv2 feature space), and PSNR, SSIM, LPIPS, and edge-F (pixel-accurate, perceptual, and boundary reconstruction). Kinematic Forcing improves over the spatial head on most of them, with its advantage concentrated in fast motion.

Table 3: Video prediction quality at robot regions, SimplerEnv `PutCarrotOnPlate` (WidowX). The spatial head improves all metrics over the baseline; Kinematic Forcing improves further on all of them.

Configuration	FVD ↓	StructSim ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Edge-F ↑
UVA (baseline)	117.9	0.733	19.55	0.636	0.0773	0.773
+ Spatial head	110.5	0.752	19.73	0.643	0.0711	0.779
+ KF (ours)	**109.4**	**0.758**	**20.04**	**0.655**	**0.0675**	**0.795**

Table 4: Video prediction quality at robot regions, Google Robot long-horizon task, and real-world xArm7. The largest single jump is the spatial head's FVD reduction.

Domain	Configuration	FVD ↓	StructSim ↑	PSNR ↑	SSIM ↑	LPIPS ↓	Edge-F ↑
Google	UVA (baseline)	462.4	0.590	9.95	0.516	0.1733	0.619
	+ Spatial head	**377.1**	0.624	9.98	0.526	0.1638	0.621
	+ KF (ours)	379.0	**0.625**	**10.00**	**0.528**	**0.1626**	**0.621**
xArm7	UVA (baseline)	320.4	0.657	17.59	0.550	0.1108	0.733
	+ Spatial head	288.8	0.669	17.58	0.551	0.1081	0.732
	+ KF (ours)	**283.8**	**0.671**	**17.65**	**0.554**	**0.1070**	**0.734**

Across all three domains (Tables 3, 4), the spatial head improves over the baseline, most visibly on FVD and StructSim, and Kinematic Forcing adds a further improvement on most metrics. On WidowX, the full method improves over the spatial head alone on all six metrics, e.g. PSNR by 0.31 dB and edge-F by 1.6 points. On the real xArm7, it achieves the best FVD (283.8 vs. 320.4 baseline) and the best score across all metrics. The visual comparison in Figure 3 shows the structural difference directly.

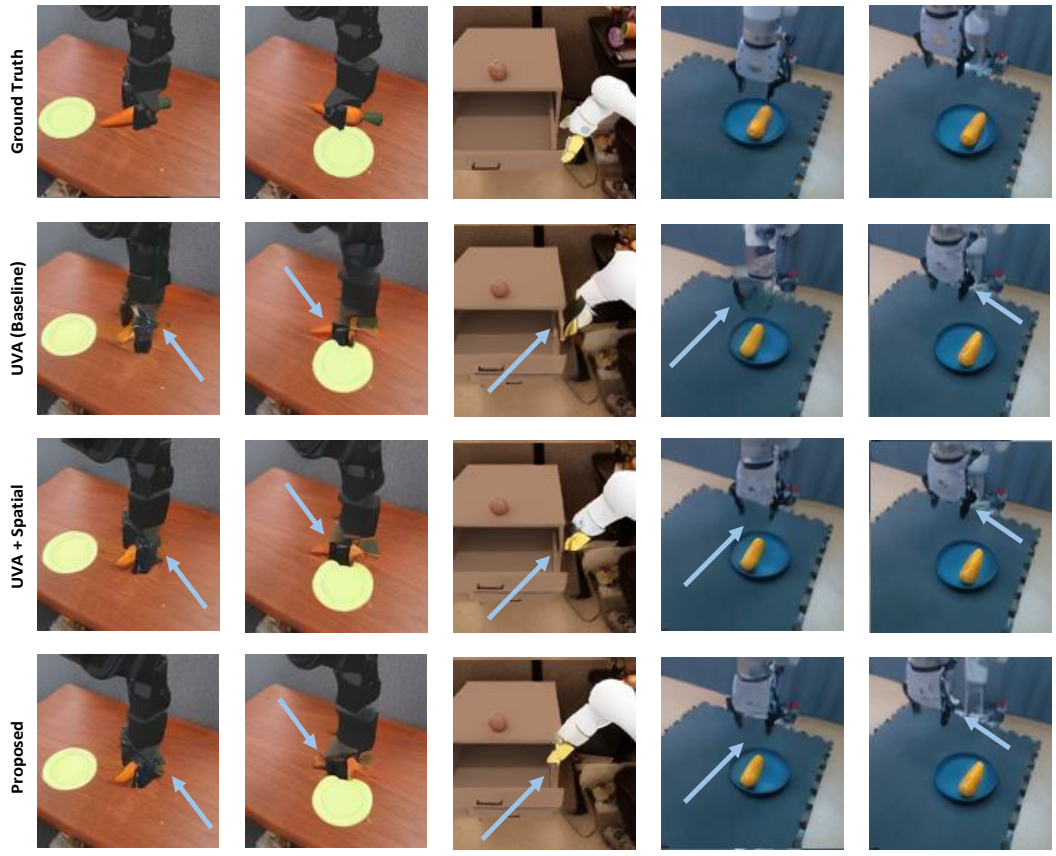


Figure 3: Visual comparison between Ground Truth, Baseline, Baseline with Spatial head, and ours showing diffusion-sampled generated frames. The proposed method produces sharper, more structurally correct robot arms. Arrows mark main structural signals across different settings.

5.3 Per velocity gain

The aggregate numbers understate the extent to which Kinematic Forcing helps. We stratify the WidowX evaluation by end-effector velocity into quartiles (Q1: slowest, Q4: fastest) and report fidelity within each quartile (Table 5). In the slow quartiles, all three models are close: when the arm barely moves, copying the previous frame is already near-optimal, and there is little structure to get wrong. In the fastest quartile, the baseline and spatial-head models degrade sharply, while Kinematic Forcing holds up: it improves Q4 PSNR by $+1.02$ dB, SSIM by $+0.042$, and edge-F by $+0.062$ over the spatial head, and by $+1.25$ dB PSNR over the baseline. This is the signature we expect from a physics-grounded target: the kinematic render carries exactly the structural information the frames lack when the arm sweeps quickly, which is also where the per-token head fails most.

5.4 Downstream Policy Performance

The shared representation is read by the action head, so a natural concern is whether grounding $\mathbf{z}$ in render structure costs control performance. Table 6 reports success rates. The spatial head substantially improves policy over the baseline, freeing the shared representation from the need to encode fine-grained robot structure via a weak per-token head, thereby improving both video and control. Adding Kinematic Forcing on top maintains this performance.

Table 5: WidowX video fidelity stratified by end-effector velocity quartile (Q1 slowest, Q4 fastest). The advantage of Kinematic Forcing is small at low speeds and large at high speeds.

PSNR ↑	Q1	Q2	Q3	Q4
UVA (baseline)	18.60	20.13	20.82	18.63
+ Spatial head	18.84	20.26	20.93	18.86
+ KF (ours)	19.04	20.17	21.04	**19.88**
Edge-F ↑	Q1	Q2	Q3	Q4
UVA (baseline)	0.732	0.830	0.814	0.717
+ Spatial head	0.741	0.833	0.817	0.724
+ KF (ours)	0.747	0.826	0.821	**0.786**

Table 6: Downstream policy success rate (%). WidowX and Google Robot were evaluated on the SimplerEnv [5] environment over 100 episodes; xArm7 is a real-world rollout.

	WidowX	Google Robot	xArm7
UVA (baseline)	59	46	60
+ Spatial head	67	**53**	**70**
+ KF (ours)	**70**	51	**70**

6 Ablations

The spatial head's role. The comparison in every results table is itself the spatial-head ablation: the + *Spatial head* row is Kinematic Forcing's architecture without the alignment loss, and the *UVA* row removes the spatial head entirely. Two observations follow. First, the spatial head adds the intra-frame mixing that the per-token head lacks, giving the enriched representation a spatial pathway into the output; empirically, it accounts for most of the aggregate gain over the baseline, above all on FVD and StructSim (Table 7). Second, the spatial head also drives the policy gain (Table 6), consistent with it offloading robot-structure capacity from the shared representation. Kinematic Forcing then supplies the geometric grounding on top of that capacity.

Where each component acts. The two components contribute in different regimes (Table 7). The spatial head supplies most of the aggregate gain, dominating the FVD and StructSim improvements. Kinematic Forcing adds a smaller but consistent gain across metrics in aggregate that grows sharply under fast motion, where pose error is largest and the kinematic render is most informative: its PSNR gain roughly triples from $+0.31$ overall to $+1.02$ in the fastest-motion quartile. This matches the design: the spatial head adds capacity, and the alignment loss adds physics-grounded structure for that capacity to express.

Table 7: Per-component contribution on WidowX (SimplerEnv, epoch 100), reported as the change each component adds on top of the previous one. The spatial head contributes most of the aggregate gain (above all FVD and StructSim); Kinematic Forcing adds a consistent further improvement that is small in aggregate but grows sharply in the fastest-motion quartile (Q4). FVD is a set-level metric and is not defined per quartile.

Component	FVD ↓	StructSim ↑	PSNR ↑	SSIM ↑	LPIPS ↓	edge-F ↑
+ Spatial head (Δ vs UVA)	-7.4	$+0.019$	$+0.18$	$+0.007$	-0.006	$+0.006$
+ KF (Δ vs Spatial head)	-1.1	$+0.006$	$+0.31$	$+0.012$	-0.004	$+0.016$
+ KF (Q4) (Δ vs Spatial head)	-1.1	$+0.018$	$+1.02$	$+0.042$	-0.010	$+0.062$

7 Conclusion

We introduced **Kinematic Forcing**, which grounds its shared representation in rigid-body physics: it aligns that representation to a frozen DINOv2 encoding of a kinematic render of the action head's predicted pose. The method requires no changes to the deployed model: a small projector, a video head, and a frozen rendering pipeline are used only during training and discarded at inference. Across simulated and real tasks, Kinematic Forcing improves video prediction quality in robot regions, with the largest gains during high-motion phases, where pixel-only prediction is most challenging.

8 Limitations and Future Work

Kinematic Forcing inherits a dependency on the action head: the kinematic composite is only as accurate as the predicted pose. When the action head's predictions are poor, often during the same fast or complex motions that the video head struggles with, the alignment target becomes less reliable; the warmup schedule mitigates this early in training but does not directly address action error.

The rendering pipeline requires a calibrated URDF model and camera parameters for each embodiment, and the inpainter must be trained separately for each domain, so extending Kinematic Forcing to a new robot entails a nontrivial setup cost.

Our policy result is that alignment does not degrade control; whether to convert the improved video representation into a control *gain* remains an open question. A promising direction is to close the loop in the other direction by decoding generated video back into actions through an inverse-dynamics model, so that the action-to-video and video-to-action paths reinforce each other rather than only sharing a representation.

References

[1] S. Li, Y. Gao, D. Sadigh, and S. Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[2] C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[3] J. Cen, C. Yu, H. Yuan, Y. Jiang, S. Huang, J. Guo, X. Li, Y. Song, H. Luo, F. Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[4] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[5] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. B. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=bo8q5MRcwy.

[8] F. Zhu, H. Wu, S. Guo, Y. Liu, C. Cheang, and T. Kong. Irasim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9834–9844, 2025.

[9] J. Chen, W. Song, P. Ding, Z. Zhou, H. Zhao, F. Tang, D. Wang, and H. Li. Unified diffusion vla: Vision-language-action model via joint discrete denoising diffusion process. *arXiv preprint arXiv:2511.01718*, 2025.

[10] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan, C. Zhu, J. Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[11] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[12] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15619–15629, 2023.

[13] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. V-jepa: Latent video prediction for visual representation learning. 2023.

[14] S. Yu, S. Kwak, H. Jang, J. Jeong, J. Huang, J. Shin, and S. Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

[15] H. Wu, D. Wu, T. He, J. Guo, Y. Ye, Y. Duan, and J. Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling. *arXiv preprint arXiv:2507.07982*, 2025.

[16] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[17] R. Wang, Q. Liu, Y. Deng, G. Liu, Z. Liu, and K. Jia. Eva: Aligning video world models with executable robot actions via inverse dynamics rewards. *arXiv preprint arXiv:2603.17808*, 2026.

[18] R. Liu, A. Canberk, S. Song, and C. Vondrick. Differentiable robot rendering. *arXiv preprint arXiv:2410.13851*, 2024.

[19] Q. Ruan, J. Lei, W. Yuan, Y. Zhang, D. Lu, G. Liu, and K. Jia. Prof. robot: Differentiable robot rendering without static and self-collisions. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22562–22572, 2025.

[20] J. Tian, L. Wang, S. Zhou, S. Wang, and G. Hua. Dynarend: Learning 3d dynamics via masked future rendering for robotic manipulation. *Advances in Neural Information Processing Systems*, 38:77292–77318, 2026.

[21] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6):1–14, 2020.

[22] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.

[23] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[25] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. 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.

[26] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[27] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan. What is a good evaluation measure for semantic segmentation?. In *Bmvc*, volume 27, pages 10–5244. Bristol, 2013.

[28] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

A Video Quality Metrics

We evaluate video prediction quality using full-reference metrics computed on robot-region pixels, along with a distribution-level FVD. Let $\hat{I}_t$ and I_t denote the predicted and ground-truth frame at horizon step $t = 1, \ldots, T$.

Robot mask. The robot region $M \subseteq \{1, \ldots, H\} \times \{1, \ldots, W\}$ is obtained by rendering the URDF at the dataset's ground-truth joint angles, rasterizing the silhouette, taking the union over the clip, and dilating it by 2 pixels (a 5×5 structuring element) to absorb rasterizer/simulator boundary mismatch. The same mask is used by all models, so the comparison is model-independent.

Robot structural similarity (StructSim). We measure whether the predicted robot is rendered in the correct pose using a frozen DINOv2 [4]. Each frame is resized to 224×224 and encoded into a 16×16 grid of patch tokens; let $\mathbf{f}^{\hat{I}}_{t,i}$ and $\mathbf{f}^{I}_{t,i}$ be the token at patch i of the predicted and ground-truth frame t. The robot patch set $\mathcal{P}$ contains every patch whose footprint is at least 25% robot pixels under M. We report the mean cosine similarity between *position-matched* patches of the predicted and ground-truth frame:

$$\text{StructSim} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}}\frac{\mathbf{f}^{\hat{I}}_{t,i}\cdot\mathbf{f}^{I}_{t,i}}{\|\mathbf{f}^{\hat{I}}_{t,i}\|\,\|\mathbf{f}^{I}_{t,i}\|}. \tag{4}$$

Because the comparison is patch-for-patch against the same ground-truth frame, the metric is pose-specific: it rewards a robot rendered with the correct articulated configuration, not merely a plausible-looking arm. It is high when the predicted robot region contains a clear, correctly posed structure, and low when the region is blurred, ghosted, or misplaced. This evaluation metric uses the native 16×16 DINOv2 patch grid, whereas the training-time alignment target (Sec. 4.3) pools the same features to the backbone's 8×8 token grid; the two are computed independently and serve different roles (evaluation vs. representation alignment).

Robot PSNR. PSNR over the masked robot pixels (all three channels), averaged over horizons:

$$\text{MSE}_t = \frac{1}{3\,|M|}\sum_{(x,y)\in M}\sum_{c=1}^{3}\big(\hat{I}_t(x,y,c) - I_t(x,y,c)\big)^2, \qquad \text{PSNR} = \frac{1}{T}\sum_{t=1}^{T} 10\log_{10}\frac{255^2}{\text{MSE}_t}. \tag{5}$$

Robot SSIM / LPIPS. SSIM [24] and LPIPS [25] are windowed/perceptual measures that are ill-defined on a ragged mask, so we compute them on the tight robot bounding-box crop $B = \text{bbox}(M)$:

$$\text{SSIM} = \frac{1}{T}\sum_{t=1}^{T}\text{SSIM}\big(\hat{I}_t|_B,\, I_t|_B\big), \qquad \text{LPIPS} = \frac{1}{T}\sum_{t=1}^{T}\text{LPIPS}\big(\hat{I}_t|_B,\, I_t|_B\big), \tag{6}$$

with a 7×7 Gaussian window for SSIM and crops resized to 64×64 for LPIPS. Lower LPIPS is better.

Robot edge F-score. We measure whether the arm's structure is drawn in the correct place and is tolerant of texture and brightness variations, adapting the boundary F-measure used in contour-detection and segmentation benchmarks [26, 27] to robot-region edges. We extract Canny edges (hysteresis thresholds $50/150$) from the grayscale predicted and ground-truth frames, restrict them to the robot region M, and match them with a tolerance of $\tau = 1$ pixel. With $E^{\hat{I}}_t$ and E^{I}_t the masked predicted/ground-truth edge sets and $\mathcal{N}_\tau(\cdot)$ a τ-pixel dilation,

$$\text{prec}_t = \frac{|E^{\hat{I}}_t \cap \mathcal{N}_\tau(E^{I}_t)|}{|E^{\hat{I}}_t|}, \quad \text{rec}_t = \frac{|E^{I}_t \cap \mathcal{N}_\tau(E^{\hat{I}}_t)|}{|E^{I}_t|}, \quad \text{edge-F} = \frac{1}{T}\sum_{t=1}^{T}\frac{2\,\text{prec}_t\,\text{rec}_t}{\text{prec}_t + \text{rec}_t}. \tag{7}$$

Higher is better; blurred or ghosted arms produce diffuse, misaligned edges and score low.

FVD. We report Fréchet Video Distance over the full predicted clips using I3D features [28]. Fitting Gaussians (μ_r, Σ_r) and (μ_g, Σ_g) to the I3D embeddings of the real and generated clips,

$$\mathrm{FVD} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2\,(\Sigma_r \Sigma_g)^{1/2}\Big). \tag{8}$$

Motion stratification. For the fast-motion analysis (Sec. 5.3), we classify each sample by its mean ground-truth end-effector velocity and partition the evaluation set into velocity quartiles (Q1 slowest, Q4 fastest). The quartile boundaries are computed over the pooled evaluation set, which is shared across all models and seed-independent, so Q4 denotes the same set of samples for every model.

B Inpainter Architecture and Training Details

In this section, we provide implementation details for the inpainter h_ψ introduced in Sec. 4.2.

Architecture. The inpainter is a single-pass U-Net that takes the seven-channel input defined in Sec. 4.2 (the masked background frame, the URDF render at the predicted pose, and the union mask) and produces a three-channel completion at the same resolution. The encoder downsamples through four stages, and the symmetric decoder upsamples with skip connections. Self-attention is added at the bottleneck and at the two stages on each side of it, where the spatial resolution is coarsest. The output passes through a sigmoid, so all predictions lie in $[0, 1]$. The network is fully convolutional and operates on any input resolution divisible by 16. The configuration used in our experiments is given in Table 8.

Table 8: Inpainter layer stack at input resolution $H \times H$. We use $H = 128$.

Stage	Resolution	Channels
Input	$H \times H$	7
Encoder 1	$H \times H$	96
Encoder 2	$H/2 \times H/2$	192
Encoder 3 (attention)	$H/4 \times H/4$	384
Encoder 4 (attention)	$H/8 \times H/8$	768
Bottleneck (attention)	$H/16 \times H/16$	768
Decoder 4 (attention)	$H/8 \times H/8$	384
Decoder 3 (attention)	$H/4 \times H/4$	192
Decoder 2	$H/2 \times H/2$	96
Decoder 1	$H \times H$	96
Output	$H \times H$	3

Training data. Training pairs are constructed offline from any robot-manipulation dataset that provides synchronized RGB frames and joint-angle (or end-effector pose) annotations. For each sampled frame I, we render two URDF silhouettes: M^{cur} at the frame's recorded joint angles, and M^{other} at a different pose drawn either from another timestep of the same episode or sampled near the current pose. With probability $p_\cup = 0.8$ the input mask is the union $M^{\text{cur}} \cup M^{\text{other}}$; otherwise it is M^{cur} alone. The union regime matches the inference-time use of h_ψ, where two robot regions must be erased at once. The conditioning RGB R is the URDF render at M^{other}, so the network learns to insert the robot at the shown pose rather than memorizing the training pose distribution.

Optimization. The network is trained with AdamW under a cosine schedule. Supervision is the original frame I, and the loss is computed only on the inpainted region, so the network is graded on what it had to synthesize rather than on background pixels it merely had to copy. Episodes are partitioned into train and validation splits to prevent temporally adjacent frames from leaking across the split. Hyperparameters are summarized in Table 9.

Table 9: Inpainter training hyperparameters.

Hyperparameter	Value
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	1×10^{-4}
LR schedule	Cosine
Batch size	32
Max epochs	400
Early-stopping patience	60 epochs
Reconstruction loss	L1 on masked region
Train / val split	$95/5$, by episode
Frame stride	2
Union-mask probability $p_{\cup}$	0.8
Precision	FP32