PHYSICS-GUIDED VIDEO PREDICTION VIA LEARNED DYNAMICS FROM VIDEO OBSERVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Future video prediction for physical systems must preserve both visual realism and physical dynamic evolutions. Existing video generators often model motion non-physically or rely on externally user-defined manipulations, while methods that infer physical dynamics from video typically focus on identifying parameters or trajectory prediction, with limited integration into future video generation. This leaves a research gap between explicit physical prediction and physically grounded future video generation. We introduce **LPhyVideo** ("L" stands for "Learned"), a physics-guided video prediction framework via learned physical dynamics from the video, within a *video-to-physics-to-video* formulation for systems whose low-dimensional physical states are reconstructed from visual observations. LPhyVideo maps tracked key pixel locations to physically interpretable states, and rolls out via a learned state-space dynamics model; subsequently, LPhyVideo reconstructs the predicted states back into image space, and with the aid of a reference frame, converted into a video and edit mask for a constrained refinement of final generation. Across six diverse physical systems, LPhyVideo achieves the highest average Spatial IoU, Spatiotemporal IoU, Weighted Spatial IoU and the lowest MSE and Trajectory Error among comparisons.

1 INTRODUCTION

Understanding and predicting how physical systems evolve from visual observations is a fundamental problem at the intersection of computer vision, physical reasoning, and dynamical modeling (Garcia et al., 2025; Yang et al., 2025; Khanbayov et al., 2026). This is increasingly important as video generation moves toward modeling and controlling dynamic scene evolution. Diffusion-based video models now achieve high-fidelity image-to-video (I2V) generation with strong temporal coherence (Bar-Tal et al., 2024; Wu et al., 2025), and recent trajectory- and editing-based methods enable increasingly precise control over object motion and scene evolution (Mou et al., 2024; Wang et al., 2025; Chu et al., 2026). Nevertheless, photorealistic video generation can still violate the underlying physical dynamics, showing that perceptual realism does not necessarily imply physical understanding (Liu et al., 2024; Motamed et al., 2026). Most video generators predict directly in pixel or latent spaces, leaving physical regularities implicit in learned distributions (Bar-Tal et al., 2024; Wu et al., 2025; Feng et al., 2025). Editing-based generation introduces trajectories, masks, or structural conditions (Mou et al., 2024; Wang et al., 2025; Chu et al., 2026), but these controls are typically user-specified or externally provided rather than learned from the system's intrinsic dynamics.

A complementary line of work recovers interpretable physical states, parameters, and dynamics from visual observations (Watters et al., 2017; Guen & Thome, 2020; Garcia et al., 2025; Yang et al., 2025; Khanbayov et al., 2026). However, these methods typically output physical quantities, state trajectories, or dynamical models rather than photorealistic future observations. This leaves a research gap between *learning the dynamics of a physical system from video* and *using those learned dynamics to generate a visually coherent future video*. Explicitly modeling the system's true dynamics makes the predicted evolution directly inspectable, enables rollout from observed or user-specified initial conditions, and separates physical evolution from appearance synthesis. Our results further show that the method remains effective even when the camera is moving, generating physically consistent videos under viewpoint changes that differ from those seen during training.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

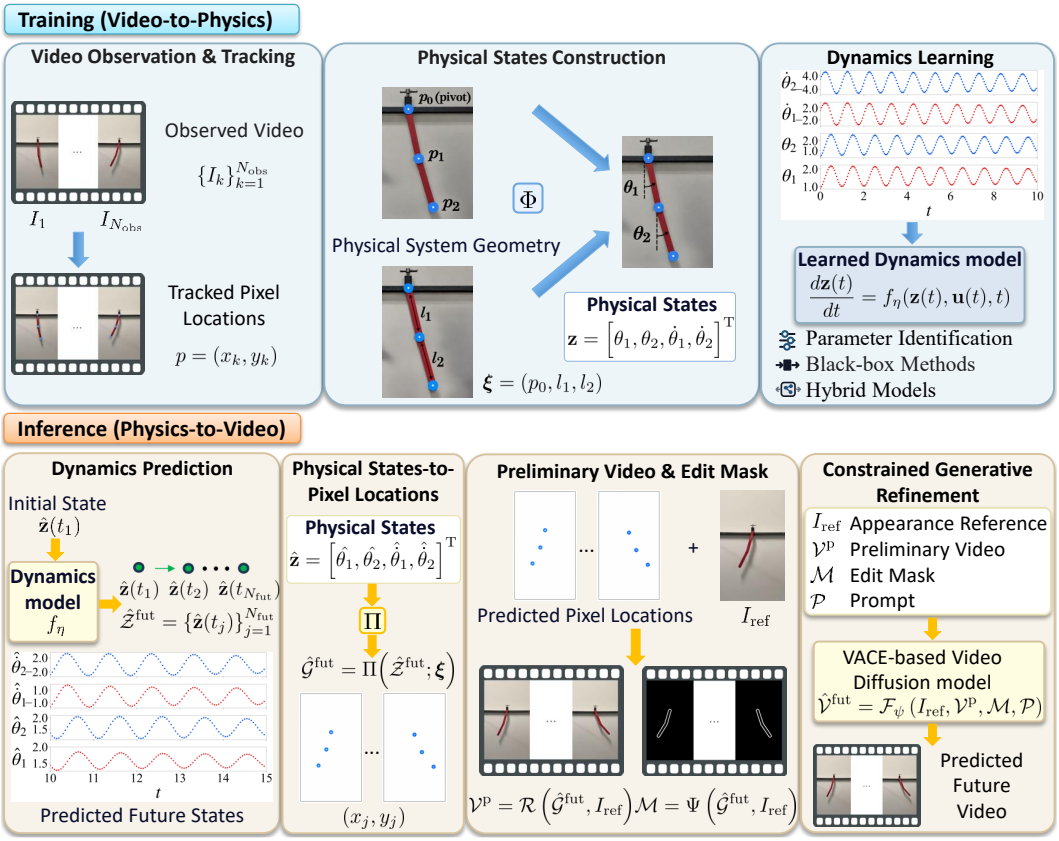

Figure 1: The proposed **LPhyVideo**. **Training (Video-to-Physics):** System-relevant points are tracked as pixel locations in the observed video and mapped to interpretable physical states, from which the dynamics model is obtained through parameter identification, black-box methods, or hybrid models. **Inference (Physics-to-Video):** Starting from an observed or specified initial physical state, the dynamics model predicts future states that are mapped back to future image space geometry. The predicted geometry and appearance reference I_{ref} are used to construct a preliminary video and a corresponding edit mask; together with the appearance reference and textual prompts, these condition a VACE-based video diffusion model to generate a refined video.

We address this gap with **LPhyVideo**, a physics-guided video prediction framework delivering a *video-to-physics-to-video* formulation (Figure 1). It targets systems whose observable kinematics can be represented by compact physical states and mapped back to image space geometry; scenes without such a representation fall outside our scope. The framework first tracks system-relevant key pixel locations and constructs interpretable physical states; then learns a state-space dynamics model from the resulting state trajectories, and rolls the model forward before mapping the predicted states back to image space and a preliminary video for final generative refinement. It is noted that the physical state representation provides a bridging interface between visual observation, dynamics learning and prediction, and future video generation.

Our contributions are threefold. i) First, we formulate future video prediction through systems' true physical states and dynamics rollout, determining physical evolution before video generation rather than relying on prescribed motion or implicit generative priors. ii) Second, we introduce preliminary video and edit mask construction that converts predicted physical evolution into dense visual guidance and localized edit regions. iii) Third, across six physical systems, LPhyVideo outperforms baselines on physics-oriented video metrics and trajectory metrics, with ablations supporting the importance of modeling systems' true dynamics.

2 Related Work

Learning Physical Dynamics from Video. Inferring physical dynamics from visual observations has long been studied as a bridge between sensing and system identification. Data-driven dynamical models such as Neural ODEs provide flexible tools for representing complex system evolution (Chen et al., 2018), while recent work has increasingly focused on recovering physically meaningful quantities directly from videos. For example, Garcia et al. (2025) estimate physical parameters from individual videos without frame-level physical supervision, and Yang et al. (2025) reconstruct nonlinear dynamics from experimental videos. More recently, IRIS Khanbayov et al. (2026) considers inverse recovery and identification of real physical systems from monocular observations.

These methods recover interpretable physical states, parameters, or dynamics from visual measurements, but generally stop at physical prediction. Our framework instead uses video-derived physical states to learn a state-space dynamics model, rolls this model forward, and then maps the predicted physical evolution back to image space for future video generation.

Physical Reasoning in Video Generation. Large-scale video generators increasingly achieve realistic appearance and motion, yet their physical reliability remains limited, with visual realism and physical understanding remaining substantially decoupled (Motamed et al., 2026). Several recent methods therefore incorporate more explicit physical structures into video generation. PhysGen combines rigid-body simulation with I2V generation (Liu et al., 2024); Neural Gaussian Force Fields learn physics-grounded 4D dynamics from Gaussian scene representations (Li et al., 2026); Feng et al. (2025) discover symbolic functions over tracked image space trajectories and use the forecast trajectories to guide I2V generation; and recent systems further explore planning-based physically grounded generation (Feng et al., 2026).

Our framework instead learns systems' true dynamics based on interpretable physical states before generating the video. This separation makes the predicted evolution directly inspectable and allows the rollout to be initialized from an observed or user-specified physical state, while allowing the generative model to focus on visual refinement under a preliminary video and localized edit mask.

Controllable Video Generation and Editing. Controllable video generation provides increasingly precise mechanisms for specifying how visual content should evolve. Modern foundation models such as Wan and HunyuanVideo provide strong image-conditioned video generation capabilities (Wan et al., 2025; Wu et al., 2025), while Video All-in-One Creation and Editing (VACE) unifies reference-conditioned, video-to-video, and masked video editing within a common conditioning framework (Jiang et al., 2025). Motion-specific approaches further improve explicit control: Any Trajectory Instruction (ATI) represents diverse motion through trajectory instructions (Wang et al., 2025), Wan-Move introduces dense point trajectories into an I2V backbone through latent trajectory guidance (Chu et al., 2026), and MotionV2V enables video motion editing through modified sparse trajectories (Burgert et al., 2026).

We adopt VACE as a generative refinement backbone, but derive our preliminary video and edit mask from the predicted physical evolution of the observed system rather than from user-specified or externally designed motion trajectories.

3 Methodology

The proposed LPhyVideo follows a *video-to-physics-to-video* formulation: during the *video-to-physics* stage, observed videos are converted into interpretable physical states from which the dynamics model is learned; at *physics-to-video* inference, the resulting dynamics are rolled forward and converted back into a preliminary video for future video generation.

The pipeline is connected by several key operators: $\mathcal{T}$ extracts tracked pixel locations from observed videos, Φ maps these observations into physical states, Π maps predicted physical states back to future image space, and $\mathcal{R}$ and Ψ construct a preliminary video and an edit mask, respectively.

3.1 PROBLEM FORMULATION

Let $\mathcal{V}^{\text{obs}}$ denote an observed video sequence consisting of N_{obs} frames:

$$\mathcal{V}^{\text{obs}} = \{I_k\}_{k=1}^{N_{\text{obs}}}, \tag{1}$$

where I_k denotes the k-th observed frame.

LPhyVideo supports both video continuation from a series of observed video frames, or future video prediction given a specified initial condition. Given an initial condition and a target horizon of N_{fut} frames, the objective is to generate a physically consistent video evolution:

$$\hat{\mathcal{V}}^{\text{fut}} = \{\hat{I}_j\}_{j=1}^{N_{\text{fut}}}. \tag{2}$$

Rather than modeling the video evolution directly in pixel, segmentation, or latent space (Bar-Tal et al., 2024; Wu et al., 2025), LPhyVideo introduces an explicit physical representation bridging between visual observation and video generation.

3.2 TRAINING (VIDEO-TO-PHYSICS)

Video Observation & Tracking. LPhyVideo tracks system-relevant key points to recover geometric configurations and their temporal evolutions for physical state construction. For each frame I_k, the tracking procedure $\mathcal{T}$ extracts pixel locations of the keypoints:

$$\mathbf{P}^{\text{obs}} = \mathcal{T}(\mathcal{V}^{\text{obs}}) = \{p_{m,k}\} = \{(x_{m,k}, y_{m,k})\}, \;\; m = 1, \cdots, M; k = 1, \cdots, N_{\text{obs}}, \tag{3}$$

where m indexes the tracked points. The tracked quantities depend on the geometry of each system, e.g., joint locations, object center positions, interface positions, etc. These measurements preserve explicit geometric meaning while avoiding direct modeling of raw pixel intensity. Detailed tracking information is illustrated in Appendix A.4. By tracking these pixel locations across frames, LPhyVideo obtains the temporal geometric observations required for subsequent physical state construction.

Physical States Construction. The tracked pixel locations are then converted into a physical state trajectory through a system-dependent mapping Φ:

$$\mathcal{Z}^{\text{obs}} = \Phi\big(\mathbf{P}^{\text{obs}}; \boldsymbol{\xi}\big) = \{\mathbf{z}(t_k)\}_{k=1}^{N_{\text{obs}}}, \tag{4}$$

where t_k denotes the physical time associated with the k-th observed frame, and $\boldsymbol{\xi}$ denotes the system's geometric parameters used to relate the image space observations to the physical configuration. Depending on the system, $\boldsymbol{\xi}$ may include quantities such as characteristic lengths, reference locations, or other known geometric dimensions. The same geometric information is also used later to map predicted physical states back to pixel locations. The state $\mathbf{z}$ comprises the variables required to characterize the physical dynamics of the system.

In general, the mapping from pixel coordinates to physical states depends on both system geometries and camera projection models. For a planar system, if the camera is fixed and approximately fronto-parallel to the motion plane, lens distortion is negligible, and the image and physical axes are approximately aligned, the camera projection can be reduced to a simple geometry-based mapping. Under these assumptions, the relevant physical states can be recovered without explicitly estimating the full set of camera parameters. For systems involving three-dimensional (3D) motion (e.g., in our Forced 3D Robot Arm example), the mapping further relies on known system geometries and kinematic structures, together with an appropriate camera projection model, to relate image space observations to the underlying physical configuration. More generally, when these simplifying assumptions do not hold, such as under arbitrary 3D motion or a moving camera (e.g., in our Moving Camera 3D Robot Arm example), a more complete camera model, including camera intrinsics, extrinsics, and perspective projection, can be incorporated into Φ. The corresponding general formulation is provided in Appendix A.1.

Dynamics Learning. Given the observed physical state trajectories, LPhyVideo represents the system evolution through a state-space dynamics model:

$$\frac{d\mathbf{z}(t)}{dt} = f_\eta\big(\mathbf{z}(t), \mathbf{u}(t), t\big), \tag{5}$$

where f_η denotes the dynamics model, η represents the parameters identified or learned from video-derived physical state trajectories, and $\mathbf{u}(t)$ denotes an optional time-varying control or exogenous input. When no explicit control or time-varying external input is modeled, $\mathbf{u}(t)$ is omitted. It is noted that known physical models and fixed system parameters may also be encoded in the structure of f_η to form a hybrid model for f_η.

Depending on the available physical structure, LPhyVideo uses parameter identification, black-box methods, or hybrid models: these respectively estimate parameters within known equations, learn unknown dynamics, or combine known structure with learned components. In our experiments, LPhyVideo is instantiated with parameter identification models and Neural ODEs. Despite their different model forms, all dynamics models operate on the same interpretable physical state used for subsequent prediction.

3.3 Inference (Physics-to-Video)

Dynamics Prediction. During inference, the dynamics model is initialized from a physical state $\mathbf{z}_{\text{init}}$ obtained from the available observation or specified by the users. For video continuation, $\mathbf{z}_{\text{init}}$ corresponds to the state at the end of the observed interval. For prediction under a new initial condition, $\mathbf{z}_{\text{init}}$ can instead be specified.

Starting from $\hat{\mathbf{z}}(t_1) := \mathbf{z}_{\text{init}}$, the system state is rolled forward through the learned dynamics model:

$$\frac{d\hat{\mathbf{z}}(t)}{dt} = f_\eta\big(\hat{\mathbf{z}}(t), \mathbf{u}(t), t\big). \tag{6}$$

Equivalently, the predicted state at a future time t is obtained by solving an initial value problem:

$$\hat{\mathbf{z}}(t) = \mathbf{z}_{\text{init}} + \int_{t_1}^{t} f_\eta\big(\hat{\mathbf{z}}(\tau), \mathbf{u}(\tau), \tau\big)\, d\tau. \tag{7}$$

Evaluating the solution at the target future time stamps yields the future physical state trajectory:

$$\hat{\mathcal{Z}}^{\text{fut}} = \{\hat{\mathbf{z}}(t_j)\}_{j=1}^{N_{\text{fut}}}. \tag{8}$$

The future evolution is therefore determined explicitly in physical state space before video generation. In the scenarios where the system is externally controlled or driven, the corresponding known input $\mathbf{u}(t)$ is supplied during rollout; otherwise, prediction proceeds solely from the initialized state and the learned dynamics model.

Physical States-to-Pixel Locations. The predicted physical states are mapped back to the image plane to recover the future pixel locations of the points tracked during observation. We denote the resulting future geometric trajectory as:

$$\hat{\mathcal{G}}^{\text{fut}} = \Pi\Big(\hat{\mathcal{Z}}^{\text{fut}}; \boldsymbol{\xi}\Big) = \{\hat{g}_j\}_{j=1}^{N_{\text{fut}}}, \quad \hat{g}_j = \hat{p}_{m,j} = \{(\hat{x}_{m,j}, \hat{y}_{m,j})\}_{m=1}^{M}. \tag{9}$$

Thus, Π maps each predicted physical configuration to the corresponding image space locations of the tracked points.

For planar systems, this mapping can be obtained using the corresponding geometric relations introduced during physical state construction. For systems involving 3D motions or camera motions, Π additionally incorporates the system geometry and kinematic structure together with an appropriate camera projection model to obtain the corresponding pixel locations (detailed in Appendix A.1).

Preliminary Video & Edit Mask. The predicted geometric trajectory $\hat{\mathcal{G}}^{\text{fut}}$ specifies the future system evolution in image space, but it does not provide visual texture for video generation. To bridge this gap, LPhyVideo constructs a preliminary video $\mathcal{V}^{\text{p}}$ and a corresponding edit mask video $\mathcal{M}$ from $\hat{\mathcal{G}}^{\text{fut}}$ together with an appearance reference frame I_{ref}:

$$\mathcal{V}^{\text{p}} = \mathcal{R}\Big(\hat{\mathcal{G}}^{\text{fut}}, I_{\text{ref}}\Big), \tag{10}$$

$$\mathcal{M} = \Psi\Big(\hat{\mathcal{G}}^{\text{fut}}, I_{\text{ref}}\Big). \tag{11}$$

The operator $\mathcal{R}$ constructs $\mathcal{V}^{\mathrm{p}}$ frame by frame based on the predicted geometric trajectory. For each future configuration $\hat{g}_j$, $\mathcal{R}$ translates, rotates, or deforms reference image components to match the predicted geometry and places them over the scene background. Applying this procedure to all N_{fut} predicted configurations produces a temporally ordered sequence of preliminary video frames, which together form the preliminary video $\mathcal{V}^{\mathrm{p}}$.

In parallel, the operator Ψ constructs $\mathcal{M}$ frame by frame from the predicted geometric trajectory. For each future configuration, Ψ localizes the regions that require generative refinement. Rather than allowing unconstrained modification of the entire frame, Ψ restricts editing to local regions surrounding the predicted moving structures while preserving the rendered structural core and unaffected scene content. The resulting temporally aligned masks together form the edit mask video $\mathcal{M}$. Thus, $\mathcal{V}^{\mathrm{p}}$ specifies the predicted structure and motion to be followed, whereas $\mathcal{M}$ specifies where the generative model is allowed to refine the rendered guidance.

Constrained Generative Refinement. The final future video is conditioned on the appearance reference I_{ref}, preliminary video $\mathcal{V}^{\mathrm{p}}$, edit mask $\mathcal{M}$, and textual prompt $\mathcal{P}$ (details in Appendix A.3):

$$\hat{\mathcal{V}}^{\mathrm{fut}} = \mathcal{F}_{\psi}\left(I_{\mathrm{ref}}, \mathcal{V}^{\mathrm{p}}, \mathcal{M}, \mathcal{P}\right), \tag{12}$$

where $\mathcal{F}_{\psi}$ is instantiated with a pretrained VACE-based video diffusion model (Jiang et al., 2025). Since the future physical evolution has already been determined by the dynamics model and encoded into the preliminary video, $\mathcal{F}_{\psi}$ focuses on refining visual appearance and local details, while maintains the prescribed system's future dynamics.

4 Experiments

We evaluate the proposed LPhyVideo on six physical systems summarized in Table 1. For the first four 2D video datasets: *Double Pendulum* is from (Yang et al., 2025); *Rolling Orange* and *Spring Mass* are from Morpheus (Tragoudaras et al., 2025); *Torricelli Motion* is from Delfys75 (Garcia et al., 2025). The last two 3D video datasets, *Forced 3D Robot Arm* and *Moving Camera 3D Robot Arm*, are controlled 3D articulated systems based on the Franka Emika Panda model in MuJoCo (Zakka et al., 2022; Todorov et al., 2012). The Moving Camera 3D Robot Arm additionally introduces continuous camera motion. Each system instantiates the physical state, dynamics model, and state-to-pixel mapping, and the implementation details are provided in Appendix A.3.

4.1 Experimental Setup

Baselines. We compare LPhyVideo with three general video generation baselines without explicit future physics guidance: *Wan2.2 (I2V)* (Wan et al., 2025), *HunyuanVideo 1.5 (I2V)* (Wu et al., 2025), and *LTX-2.3 (Video Continuation)* (HaCohen et al., 2026). The I2V baselines use the real frame at the beginning of the prediction horizon, while the video continuation baseline uses the observed video clip in its native continuation setting.

Table 1: Overview of the six physical systems used in our experiments.

System	**Physical State z** (Sec. 3.2)	**Dynamics Model** (Eq. 5)	**Rollout Setting** (Sec. 3.3)	**Challenge**
Double Pendulum	$\left[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\right]^\top$	Neural ODEs	Video Continuation	Coupled nonlinear articulation
Rolling Orange	$[s, \dot{s}, \phi]^\top$	Parameter Identification	New Initial Condition	Translation-rotation coupling
Spring Mass	$[x_{\text{mass}}, \dot{x}_{\text{mass}}]^\top$	Parameter Identification	Video Continuation	Damped spring mass oscillation
Torricelli Motion	$\left[h, \sqrt{h}, y_{\text{water}}\right]^\top$	Parameter Identification	Video Continuation	Evolving liquid boundary
Forced 3D Robot Arm	$[q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3]^\top$	Neural ODEs	Video Continuation	Controlled 3D articulation
Moving Camera 3D Robot Arm	$[q_1^m, q_2^m, q_3^m, \dot{q}_1^m, \dot{q}_2^m, \dot{q}_3^m]^\top$	Neural ODEs	Video Continuation	Physical camera motion disentanglement

We additionally evaluate two guided generation methods using the same future physical prediction from LPhyVideo: *Wan-Move* (Chu et al., 2026) uses the predicted trajectories, and *LTX-2.3 + EditAnything* (dos Anjos, 2026) uses the predicted preliminary video. We also report *LPhyVideo before Refinement*, corresponding to the *Preliminary Video* in Figure 1 (Eq. 10) before generative refinement, to isolate the effect of the refinement stage. All text-conditioned methods use the same prompt, and all generated videos follow the same evaluation protocol.

Evaluation Metrics. We report the four Physics-IQ component metrics (three complementary metrics using Intersection over Union (IoU) and a mean squared error (MSE) metric) (Motamed et al., 2026; Rädsch et al., 2026): *Spatial IoU* for spatial motion overlap, *Spatiotemporal IoU* for frame-wise motion overlap, *Weighted Spatial IoU* for motion occupancy, and *MSE* for pixel-level discrepancy. Detailed definitions are provided in Appendix A.2.

We additionally report *Trajectory Error*. All generated and ground truth videos are processed by the same system-specific tracking pipeline. Let $\boldsymbol{\chi}_{\iota,j}^{\mathrm{gen}}$ and $\boldsymbol{\chi}_{\iota,j}^{\mathrm{gt}}$ denote the generated and ground truth two-dimensional (2D) landmark coordinates, with both image axes normalized to $[0, 1]$. We define:

$$\mathcal{E}_{\mathrm{raw}} = \mathrm{mean}_{\iota,j}\left(\frac{\left\|\boldsymbol{\chi}_{\iota,j}^{\mathrm{gen}} - \boldsymbol{\chi}_{\iota,j}^{\mathrm{gt}}\right\|_2}{\sqrt{2}}\right), \qquad \mathcal{E}_{\mathrm{traj}} = \frac{\mathcal{E}_{\mathrm{raw}}}{1 + \mathcal{E}_{\mathrm{raw}}}, \tag{13}$$

where the mean is over all evaluation frames and system-specific moving landmarks. Higher IoU and lower MSE and Trajectory Error indicate better performance. Methods evaluated with five random seeds (detailed in Appendix A.3) are averaged; LPhyVideo before Refinement is evaluated once.

4.2 Overall Results

We evaluate the performance of LPhyVideo across the six physical systems from three complementary perspectives: physical state trajectories, quantitative metrics, and qualitative video frames.

Figure 2 provides a representative state-space comparison for the *Double Pendulum* and *Forced 3D Robot Arm*. The recovered trajectories show that LPhyVideo closely follows the ground truth physical evolution, providing a direct view of the underlying dynamics complementary to the video-level metrics reported below. Detailed per-system state trajectories' results are provided in Appendix A.5.

Table 2 reports averaged performance across the six physical systems. LPhyVideo achieves the best Spatial IoU, Weighted Spatial IoU, and MSE, while LPhyVideo before Refinement achieves the

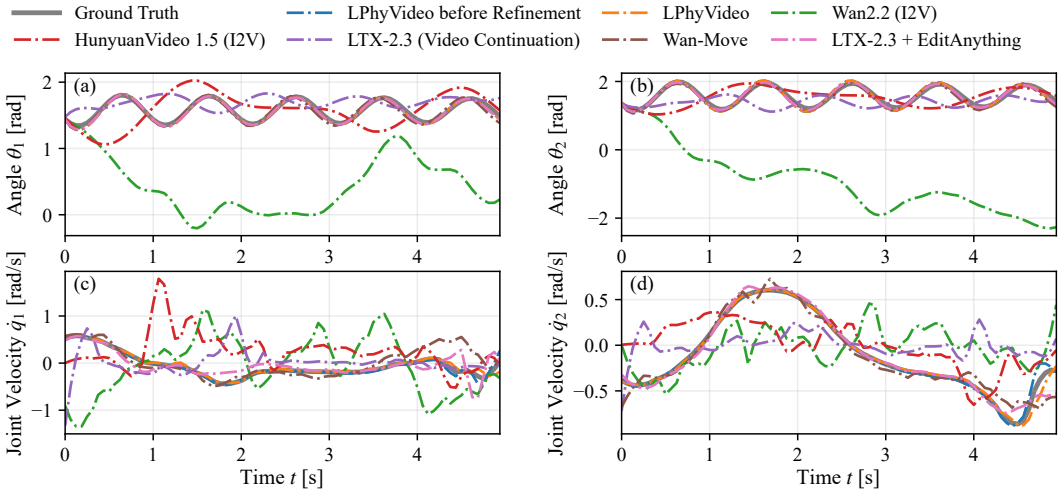

Figure 2: Representative physical state trajectories recovered from generated videos for the nonlinear Double Pendulum and Forced 3D Robot Arm. (a)&(b) show the recovered double pendulum link angles θ_1 and θ_2, while (c)&(d) show the recovered robot joint angular velocities $\dot{q}_1$ and $\dot{q}_2$. All videos are processed through the same system-specific tracking pipeline for evaluation.

Table 2: Overall quantitative comparison averaged across the six physical systems. LPhyVideo before Refinement (Figure 1 & Eq. 10) denotes the preliminary video before generative refinement.

Model	Spatial IoU ↑	Spatiotemporal IoU ↑	Weighted Spatial IoU ↑	MSE ↓	Trajectory Error ↓
Generic Video Generation					
Wan2.2 (I2V)	0.25462	0.10314	0.18459	0.01867	0.09548
HunyuanVideo 1.5 (I2V)	0.32860	0.12540	0.23908	0.01316	0.09562
LTX-2.3 (Video Continuation)	0.19506	0.09706	0.10540	0.06901	0.09445
Generation with LPhyVideo Guidance					
Wan-Move	0.45415	0.32326	0.37794	0.01062	0.02790
LTX-2.3 + EditAnything	0.57281	0.49643	0.49796	0.00959	0.01557
LPhyVideo					
LPhyVideo before Refinement	0.74569	**0.59931**	0.66546	0.00221	**0.00900**
LPhyVideo	**0.75827**	0.59605	**0.67094**	**0.00194**	0.00996

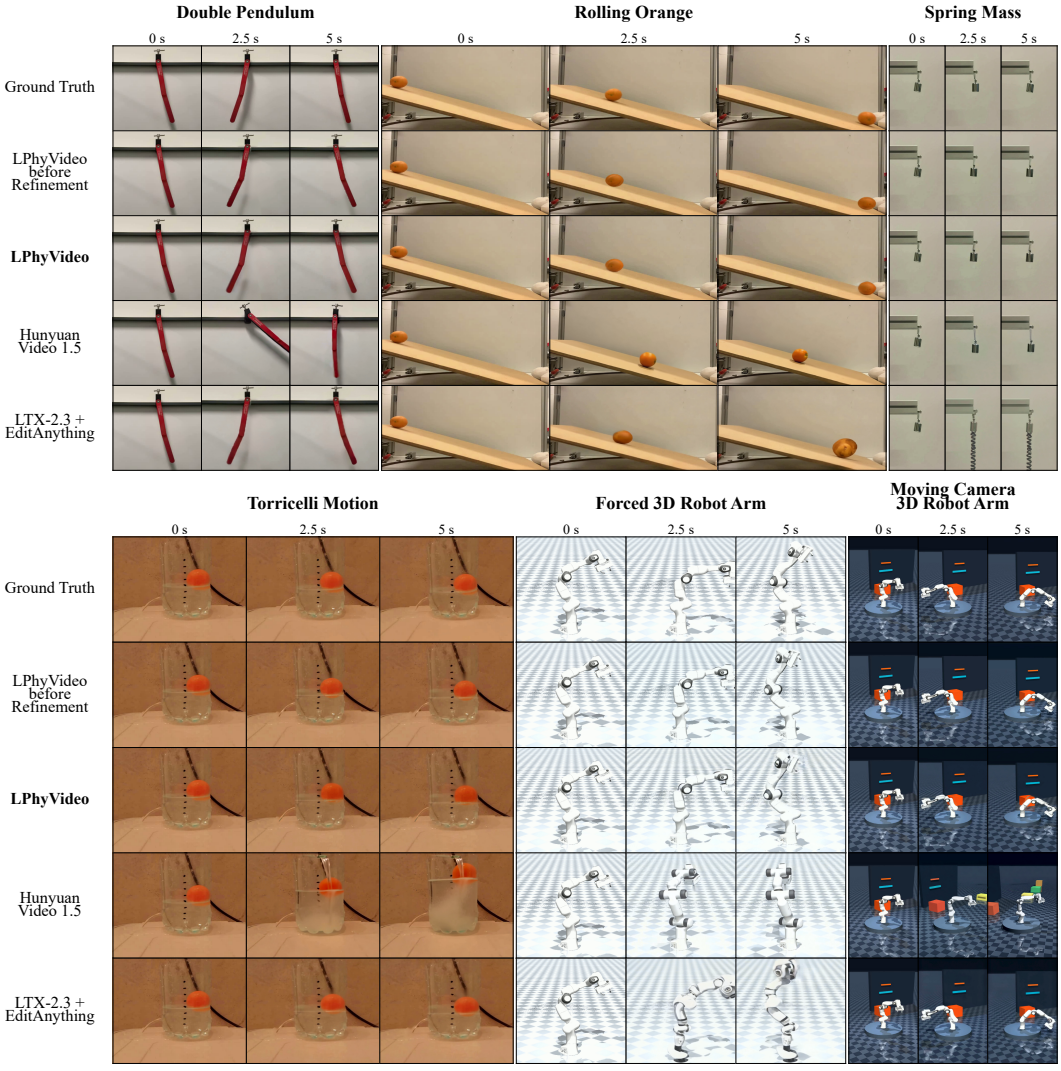

Figure 3: Qualitative comparison of six physical systems. We show frames from the ground truth, LPhyVideo before Refinement, LPhyVideo, and two representative baselines. The full video comparisons are provided in the Supplementary Material.

best Spatiotemporal IoU and Trajectory Error. It shows that the preliminary video generated from the physical rollout already captures the predicted motion and geometry, while generative refinement further improves spatial and pixel-level agreement, with only small changes in Spatiotemporal IoU and Trajectory Error. Detailed per-system quantitative results are provided in Appendix A.5.1.

Figure 3 shows representative predictions across the six systems. LPhyVideo before Refinement follows the predicted motion and geometry directly, while LPhyVideo preserves this guidance and improves visual quality. In contrast, HunyuanVideo 1.5 often exhibits substantial motion drift, particularly for the nonlinear double pendulum and controlled 3D robot arm cases. LTX-2.3 + EditAnything benefits from the same predicted preliminary video but can still introduce structural or appearance artifacts in challenging cases such as the spring mass and robot systems. These observations are consistent with Table 2. Detailed frame comparisons are provided in Appendix A.5.

4.3 Ablation Studies

Component Ablation. We ablate three components while keeping the generative backbone and evaluation protocol fixed: *w/o Dynamics Rollout* removes the learned dynamics prediction and instead constructs future states directly from the observed boundary state or recent observed trend, without using the dynamics model; *w/o Preliminary Video* replaces the temporally varying preliminary video with a static reference frame; and *w/o Edit Mask* uses a full frame editable region. Table 3 shows that removing the dynamics rollout or preliminary video causes substantial degradation, while removing the edit mask produces a smaller performance drop.

Table 3: Effect of component ablation of LPhyVideo averaged across the six physical systems.

Variant	Spatial IoU ↑	Spatiotemporal IoU ↑	Weighted Spatial IoU ↑	MSE ↓	Trajectory Error ↓
w/o Dynamics Rollout	0.28803	0.20504	0.20024	0.00822	0.11861
w/o Preliminary Video	0.21600	0.24394	0.17593	0.00981	0.10777
w/o Edit Mask	0.70603	0.55488	0.62460	0.00363	0.01337
LPhyVideo	**0.75827**	**0.59605**	**0.67094**	**0.00194**	**0.00996**

Generative Backbone Scale. We compare LPhyVideo with VACE-1.3B and VACE-14B (Jiang et al., 2025), keeping all other settings unchanged. As shown in Table 4, increasing backbone scale does not consistently improve performance. Detailed results are in Appendix A.5.8.

Table 4: Effect of generative backbone scale of LPhyVideo across the six physical systems.

Backbone	GPU	Time	Spatial IoU ↑	Spatiotemporal IoU ↑	Weighted Spatial IoU ↑	MSE ↓	Trajectory Error ↓
LPhyVideo (VACE-1.3B)	1 × A40	12′11″	0.75827	0.59605	0.67094	0.00194	0.00996
LPhyVideo (VACE-14B)	8 × A40	10′10″	0.75433	0.58210	0.66055	0.00197	0.01054

5 Conclusion

We presented LPhyVideo, a physics-guided video prediction framework that determines future physical evolution through systems' true dynamics rollout before video generation. The predicted physical states are mapped back to image space geometry and converted into preliminary video and edit masks for constrained generative refinement. Across six physical systems, LPhyVideo (including LPhyVideo before refinement) achieves the best metrics of Spatial IoU, Spatiotemporal IoU, Weighted Spatial IoU, MSE and Trajectory Error in the main comparison. Together with the ablation results, these findings show that explicit dynamics rollout provides the foundation for physically consistent prediction, while structured preliminary video guidance effectively preserves the predicted evolution during visual synthesis. More broadly, LPhyVideo separates physical evolution from appearance synthesis, allowing what should happen to be determined before how it should look is generated. The current framework focuses on systems with compact interpretable state representations and primarily stable backgrounds; extending it to richer visual environments and higher-dimensional physical systems remains future work.

Reproducibility Statement

Both the data and code will be released. Detailed experimental and implementation information, including prediction protocols, state and dynamics implementation, preliminary video construction, generative refinement, baseline settings, text prompts, evaluation protocols, and visual observation and tracking, is provided in Appendices A.3 and A.4.

AI Use Statement

In this work, we used OpenAI ChatGPT (GPT-5.6) for related work retrieval, manuscript drafting and language polishing, software implementation and debugging, assistance with experimental design and evaluation, and presentation-level editing of scientific figures and tables. We did not use generative AI to formulate the research problem, propose the core LPhyVideo methodology, or determine its claimed contributions. The authors manually reviewed, executed, and validated AI-assisted suggestions, code, and manuscript content for technical correctness and consistency with the actual experimental results. We take full responsibility for the final content of this work.

References

Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Optical pattern recognition XII*, volume 4387, pp. 95–102. SPIE, 2001.

Ryan Burgert, Charles Herrmann, Forrester Cole, Michael S Ryoo, Neal Wadhwa, Andrey Voynov, and Nataniel Ruiz. Motionv2v: Editing motion in a video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35988–35997, 2026.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Ruihang Chu, Yefei He, Zhekai Chen, Shiwei Zhang, Xiaogang Xu, Dingdong WANG, Hongwei Yi, Xihui Liu, Hengshuang Zhao, Yu Liu, et al. Wan-move: Motion-controllable video generation via latent trajectory guidance. *Advances in Neural Information Processing Systems*, 38:404–432, 2026.

Alisson Pereira dos Anjos. Edit anything: Experimental ltx-2 video editing loras. Hugging Face model repository, 2026. URL `https://huggingface.co/Alissonerdx/EditAnything`.

Tao Feng, Xianbing Zhao, Zhenhua Chen, Tien Tsin Wong, Hamid Rezatofighi, Gholamreza Haffari, and Lizhen Qu. Physics-grounded motion forecasting via equation discovery for trajectory-guided image-to-video generation. *arXiv preprint arXiv:2507.06830*, 2025.

Yuxiang Feng, Juncheng Wang, Chao Xu, Yijie Qian, Huihan Wang, Wenlong Hou, Yang Liu, Baigui Sun, Yong Liu, and Shujun Wang. Newton: Agentic planning for physically grounded video generation. *arXiv preprint arXiv:2605.18396*, 2026.

Alejandro Castaneda Garcia, Jan Warchocki, Jan Van Gemert, Daan Brinks, and Nergis Tomen. Learning physics from video: Unsupervised physical parameter estimation for continuous dynamical systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27924–27933, 2025.

Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11474–11484, 2020.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, et al. Ltx-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2026.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.

Joseph Howse. *OpenCV computer vision with python*, volume 27. Packt Publishing Birmingham, UK, 2013.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17191–17202. IEEE, 2025.

Rasul Khanbayov, Mohamed Rayan Barhdadi, Erchin Serpedin, and Hasan Kurban. Iris: A real-world benchmark for inverse recovery and identification of physical dynamic systems from monocular video. *arXiv preprint arXiv:2603.16432*, 2026.

Vincent Lepetit, Francesc Moreno-Noguer, and P Fua. Epnp: Efficient perspective-n-point camera pose estimation. *Int. J. Comput. Vis*, 81(2):155–166, 2009.

Shiqian Li, Ruihong Shen, Junfeng Ni, Chang Pan, Chi Zhang, and Yixin Zhu. Learning physics-grounded 4d dynamics with neural gaussian force fields. In *The Fourteenth International Conference on Learning Representations*, 2026.

Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. Physgen: Rigid-body physics-grounded image-to-video generation. In *European Conference on Computer Vision*, pp. 360–378. Springer, 2024.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 948–958. IEEE, 2026.

Chong Mou, Mingdeng Cao, Xintao Wang, Zhaoyang Zhang, Ying Shan, and Jian Zhang. Revideo: Remake a video with motion and content control. *Advances in Neural Information Processing Systems*, 37:18481–18505, 2024.

Tim Rädsch, Yuki M Asano, Hilde Kuehne, Stefan Bauer, Priyank Jaini, Robert Geirhos, and Carsten T Lüth. Physics-iq verified. *arXiv preprint arXiv:2606.18943*, 2026.

Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

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*, pp. 5026–5033. IEEE, 2012.

Antonios Tragoudaras, Chenyu Zhang, Daniil Cherniavskii, Antonios Vozikis, Thijmen Nijdam, Derck WE Prinzhorn, Mark Bodracska, Nicu Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Evaluating newtonian mechanics in video generative models with real physical systems. *arXiv preprint arXiv:2504.02918*, 2025.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Angtian Wang, Haibin Huang, Jacob Zhiyuan Fang, Yiding Yang, and Chongyang Ma. Ati: Any trajectory instruction for controllable video generation. *arXiv preprint arXiv:2505.22944*, 2025.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. *Advances in neural information processing systems*, 30, 2017.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, et al. Hunyuanvideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025.

Antony Yang, Joar Axås, Fanni Kádár, Gábor Stépán, and George Haller. Modeling nonlinear dynamics from videos. *Nonlinear Dynamics*, 113(10):10881–10909, 2025.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. Mujoco menagerie: A collection of high-quality simulation models for mujoco, 2022. URL `https://github.com/google-deepmind/mujoco_menagerie`.

Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

A Appendix

A.1 Camera Geometry in the Video-to-Physics-to-Video Mapping

The mappings Φ and Π introduced in Sec. 3 generally depend on the physical structure of the system and the camera projection model relating physical configurations to image space observations. This section makes this camera component explicit through a general calibrated projection formulation and then describes its realization in our experiments.

For the first four planar systems (Double Pendulum, Rolling Orange, Spring Mass and Torricelli Motion), the fixed approximately fronto-parallel camera setup simplifies the projection sufficiently that the camera parameters need not be estimated explicitly in Φ and Π. In contrast, both Forced 3D robot arm and Moving Camera experiments retain the calibrated camera projection together with the known kinematic structure to recover physical states from image observations and to project predicted 3D configurations back to the image plane; the moving-camera variant additionally uses time-varying camera extrinsics at each frame.

General Camera Projection. We adopt the standard pinhole camera model and projective formulation (Hartley & Zisserman, 2003). Consider a physical point expressed in a world coordinate frame as:

$$\boldsymbol{\rho} = [\rho_1 \quad \rho_2 \quad \rho_3]^\top . \tag{14}$$

Let $\mathbf{Q}_{\text{cam}} \in \mathbb{R}^{3\times3}$ denote the camera rotation matrix and $\boldsymbol{\delta}_{\text{cam}} \in \mathbb{R}^3$ denote the translation vector. Together, they transform the point from the world coordinate frame to the camera coordinate frame:

$$\boldsymbol{\gamma} = \mathbf{Q}_{\text{cam}}\boldsymbol{\rho} + \boldsymbol{\delta}_{\text{cam}}, \qquad \boldsymbol{\gamma} = [\gamma_1 \quad \gamma_2 \quad \gamma_3]^\top . \tag{15}$$

Perspective projection first maps the camera frame point to normalized image coordinates:

$$\boldsymbol{\nu} = \begin{bmatrix} \gamma_1/\gamma_3 \\ \gamma_2/\gamma_3 \end{bmatrix} . \tag{16}$$

Lens distortion can be represented by a distortion operator $\mathcal{D}_{\text{cam}}$ with calibrated distortion coefficients $\boldsymbol{\kappa}_{\text{cam}}$:

$$\boldsymbol{\nu}^{\,d} = \mathcal{D}_{\text{cam}}\left(\boldsymbol{\nu}; \boldsymbol{\kappa}_{\text{cam}}\right). \tag{17}$$

The distorted normalized coordinates are finally converted to pixel coordinates through the intrinsic matrix:

$$\mathbf{K}_{\text{cam}} = \begin{bmatrix} \alpha_1 & 0 & \zeta_1 \\ 0 & \alpha_2 & \zeta_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

such that:

$$\begin{bmatrix} \varpi_1 \\ \varpi_2 \\ 1 \end{bmatrix} = \mathbf{K}_{\text{cam}} \begin{bmatrix} \nu_1^d \\ \nu_2^d \\ 1 \end{bmatrix} . \tag{19}$$

Here, α_1 and α_2 are the effective focal lengths in pixel units, while (ζ_1, ζ_2) denotes the principal point.

When lens distortion is neglected, Eqs. 15-19 can be written in the standard homogeneous form:

$$\lambda \begin{bmatrix} \varpi_1 \\ \varpi_2 \\ 1 \end{bmatrix} = \mathbf{K}_{\text{cam}} \left[\mathbf{Q}_{\text{cam}} \mid \boldsymbol{\delta}_{\text{cam}}\right] \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ 1 \end{bmatrix} , \tag{20}$$

where λ is the projective scale, equal to the point depth in the ideal pinhole model. Thus, the complete image formation process consists of a world-to-camera rigid transformation, perspective division, lens distortion, and intrinsic pixel conversion.

Camera Calibration and Pose Estimation. The intrinsic parameters $\mathbf{K}_{\text{cam}}$ and $\boldsymbol{\kappa}_{\text{cam}}$ can be calibrated from known reference geometry observed from multiple views, following standard camera calibration procedures such as the planar calibration method of Zhang (2000). Given known reference points and their detected image locations, calibration can be written generally as a reprojection error minimization:

$$\min_{\mathbf{K}_{\text{cam}},\boldsymbol{\kappa}_{\text{cam}},\{\mathbf{Q}_{\text{cam}}^{(b)},\boldsymbol{\delta}_{\text{cam}}^{(b)}\}} \sum_b \sum_a \left\| \boldsymbol{\varpi}_a^{(b)} - \mathcal{C}\left(\boldsymbol{\rho}_a; \mathbf{K}_{\text{cam}}, \boldsymbol{\kappa}_{\text{cam}}, \mathbf{Q}_{\text{cam}}^{(b)}, \boldsymbol{\delta}_{\text{cam}}^{(b)}\right) \right\|_2^2, \tag{21}$$

where $\mathcal{C}$ denotes the complete projection defined by Eqs. 15-19. Once the intrinsic parameters are fixed, the camera pose relative to a particular physical system can be estimated from known 3D-to-2D correspondences through the Perspective-n-Point (PnP) formulation (Lepetit et al., 2009), yielding $\mathbf{Q}_{\text{cam}}$ and $\boldsymbol{\delta}_{\text{cam}}$. Therefore, calibration determines the image formation properties of the camera, whereas pose estimation determines the camera's position and orientation relative to the physical reference frame.

Planar Reduction and the First Four Experiments. For a physical system constrained to a plane, the world frame can be chosen such that $\rho_3 = 0$. Let $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ denote the first two columns of $\mathbf{Q}_{\text{cam}}$. Ignoring distortion for clarity, Eq. 20 becomes:

$$\lambda \begin{bmatrix} \varpi_1 \\ \varpi_2 \\ 1 \end{bmatrix} = \underbrace{\mathbf{K}_{\text{cam}} \begin{bmatrix} \boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \boldsymbol{\delta}_{\text{cam}} \end{bmatrix}}_{\mathbf{L}_{\text{cam}}} \begin{bmatrix} \rho_1 \\ \rho_2 \\ 1 \end{bmatrix}, \tag{22}$$

where $\mathbf{L}_{\text{cam}}$ is the planar homography, i.e., the projective transformation between the physical motion plane and the image plane (Hartley & Zisserman, 2003). Thus, a general oblique view of a planar physical system can already be handled using a calibrated planar mapping without reconstructing unconstrained 3D geometry.

The first four experiments in this work admit a further simplification. Their motion is planar, the camera is fixed, and approximately fronto-parallel to the motion plane, lens distortion is negligible over the relevant image region, and the image axes are approximately aligned with the physical motion axes. Under these conditions, that is:

$$\mathbf{Q}_{\text{cam}} \approx \mathbf{I}, \qquad \boldsymbol{\delta}_{\text{cam}} \approx \begin{bmatrix} 0 & 0 & d_{\text{cam}} \end{bmatrix}^\top, \qquad \boldsymbol{\kappa}_{\text{cam}} \approx \mathbf{0}, \tag{23}$$

and all points on the motion plane have approximately the same camera depth. The projection therefore reduces to:

$$\varpi_1 = \frac{\alpha_1}{d_{\text{cam}}} \rho_1 + \zeta_1, \qquad \varpi_2 = \frac{\alpha_2}{d_{\text{cam}}} \rho_2 + \zeta_2. \tag{24}$$

For approximately equal horizontal and vertical image scales, $\alpha_1 \approx \alpha_2$, this is a 2D similarity transformation:

$$\Delta \boldsymbol{\varpi} \approx \varsigma \, \Delta \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix}, \tag{25}$$

for a constant scale ς.

Eq. 25 explains why the calibrated camera geometry does not need to appear explicitly in Φ and Π for these experiments. Constant image translation disappears when relative positions are used, while the common scale cancels from directional quantities such as link orientation. For translational quantities, the remaining constant scale is absorbed into the fixed image space reference geometry, the chosen state units, or the corresponding identified physical coefficients. Consequently, the effect that would otherwise be represented explicitly by: $\mathbf{K}_{\text{cam}}$, $\mathbf{Q}_{\text{cam}}$, and $\boldsymbol{\delta}_{\text{cam}}$ is implicit in the system-specific geometry used by Φ and Π.

For example, in the Double Pendulum, relative vectors between the pivot, intermediate joint, and endpoint determine the link orientations; the constant image offset and isotropic scale therefore cancel when the angular states are constructed. For Rolling Orange, motion is measured along a fixed projected slope direction, with the tracked center and radius providing the corresponding image space geometry. For Spring Mass, the motion is restricted to the aligned vertical direction relative to the fixed upper attachment. For Torricelli Motion, the liquid height and waterline are measured relative to fixed container and outlet geometry. In these cases, using a fixed image space scale for a translational coordinate is equivalent to choosing a constant unit conversion: the state remains an

interpretable low-dimensional physical coordinate rather than a generic pixel-space representation, while any constant rescaling is absorbed consistently by the system-specific geometric or identified dynamics parameters.

The same simplification applies in the reverse direction. After the dynamics rollout, Π uses the same fixed geometry and coordinate convention to reconstruct the predicted state into image space locations. Because both the video-to-physics and physics-to-video mappings use the same fixed projection geometry, no explicit camera parameter estimation is required for these four settings.

Calibrated Projection for the Forced 3D Robot Arm. The Forced 3D Robot Arm does not satisfy the constant depth planar approximation. Its image formation therefore retains the full calibrated projection in Eq. 20. Let $\boldsymbol{\vartheta}$ denote the active joint configuration. Given the known Panda kinematic structure, the 3D location of each tracked visual fiducial can be written as $\boldsymbol{\rho}_a(\boldsymbol{\vartheta})$. The configuration can then be recovered from the tracked RGB observations by minimizing the reprojection error:

$$\widehat{\boldsymbol{\vartheta}} = \arg\min_{\boldsymbol{\vartheta}} \sum_a \left\| \boldsymbol{\varpi}_a^{\mathrm{obs}} - \mathcal{C}\left(\boldsymbol{\rho}_a(\boldsymbol{\vartheta}); \mathbf{K}_{\mathrm{cam}}, \boldsymbol{\kappa}_{\mathrm{cam}}, \mathbf{Q}_{\mathrm{cam}}, \boldsymbol{\delta}_{\mathrm{cam}}\right) \right\|_2^2 . \tag{26}$$

Thus, Φ does not perform unconstrained monocular 3D reconstruction. Instead, it estimates a low-dimensional physical configuration whose known 3D kinematics best explain the observed image space landmarks under the calibrated camera model. The resulting configuration trajectory provides the physical states used for dynamics learning.

Conversely, after dynamics prediction, Π applies the known forward kinematics to the predicted joint configuration and projects the resulting 3D geometry through the same calibrated camera model. Hence, for the Forced 3D Robot Arm, the two operators can be viewed schematically as:

$$\begin{aligned} &\Phi : \text{image landmarks} \rightarrow \text{constrained 3D configuration}, \\ &\Pi : \text{3D configuration} \rightarrow \text{projected image geometry}. \end{aligned} \tag{27}$$

Calibrated Projection for the Moving Camera 3D Robot Arm. The Moving Camera 3D Robot Arm extends the preceding setting by introducing time-varying camera extrinsics, thereby separating physical motion from camera-induced image motion. If the lens configuration remains unchanged while the camera moves, $\mathbf{K}_{\mathrm{cam}}$ and $\boldsymbol{\kappa}_{\mathrm{cam}}$ remain fixed, whereas the extrinsic parameters become frame dependent:

$$\mathbf{Q}_{\mathrm{cam}} \rightarrow \mathbf{Q}_{\mathrm{cam}}^{(b)}, \qquad \boldsymbol{\delta}_{\mathrm{cam}} \rightarrow \boldsymbol{\delta}_{\mathrm{cam}}^{(b)}. \tag{28}$$

During video-to-physics processing, these frame wise camera poses can be estimated from static scene landmarks, a calibration target, or standard visual odometry and camera pose estimation procedures (Scaramuzza & Fraundorfer, 2011). Φ then removes the camera motion by reconstructing all observations in a common physical reference frame before constructing the physical state trajectory.

For physics-to-video prediction, Π additionally requires the camera pose over the future prediction interval. In our Moving Camera 3D Robot Arm experiment, the future camera trajectory is prescribed. If the future camera trajectory is prescribed or otherwise available, the predicted physical geometry is simply projected using the corresponding frame-dependent extrinsics. If the future camera trajectory is unknown, it must be predicted separately; without a future camera pose, the future image space location of a correctly predicted 3D physical configuration is not uniquely determined.

A.2 PHYSICS-IQ

Physics-IQ (Motamed et al., 2026) evaluates physical consistency in generated videos through three complementary metrics using Intersection over Union (IoU): Spatial IoU, Spatiotemporal IoU, Weighted Spatial IoU, and a mean squared error (MSE) metric. The first three metrics operate on motion activation maps derived from temporal changes between neighboring video frames, while MSE directly compares the generated and ground truth pixel values. Physics-IQ Verified (Rädsch et al., 2026) retains these four basic metrics while refining the benchmark data and the aggregation used to construct the final Physics-IQ score.

Let $\mathbf{V}^{\mathrm{gt}}$ and $\mathbf{V}^{\mathrm{gen}}$ denote the ground truth and generated videos, respectively. Following Physics-IQ, temporal frame differences are converted into a binary spatiotemporal motion-activation map $\mathbf{A}^{\mathrm{ST}} \in \{0,1\}^{H\times W\times N_{\mathrm{eval}}}$, where H and W are the image height and width, N_{eval} is the number of evaluated time steps, and $A^{\mathrm{ST}}_{\ell_1,\ell_2,\ell_3} = 1$ indicates detected motion at row ℓ_1, column ℓ_2, and time index ℓ_3.

From this representation, two additional spatial maps are constructed:

$$A^{\mathrm{SP}}_{\ell_1,\ell_2} = \max_{\ell_3} A^{\mathrm{ST}}_{\ell_1,\ell_2,\ell_3}, \qquad A^{\mathrm{WS}}_{\ell_1,\ell_2} = \frac{1}{N_{\mathrm{eval}}} \sum_{\ell_3=1}^{N_{\mathrm{eval}}} A^{\mathrm{ST}}_{\ell_1,\ell_2,\ell_3}. \tag{29}$$

The binary spatial map $\mathbf{A}^{\mathrm{SP}}$ records whether motion occurs at a location at any time during the video, whereas the weighted spatial map $\mathbf{A}^{\mathrm{WS}}$ records the fraction of frames in which motion occurs at that location.

Spatial IoU. Spatial IoU measures *where* motion occurs, independent of its precise timing. It compares the binary spatial motion maps obtained by taking the maximum activation over time:

$$\mathrm{IoU}_{\mathrm{SP}} = \frac{\left|\mathbf{A}^{\mathrm{SP}}_{\mathrm{gt}} \cap \mathbf{A}^{\mathrm{SP}}_{\mathrm{gen}}\right|}{\left|\mathbf{A}^{\mathrm{SP}}_{\mathrm{gt}} \cup \mathbf{A}^{\mathrm{SP}}_{\mathrm{gen}}\right|}. \tag{30}$$

A high Spatial IoU therefore indicates that the generated motion occupies the correct overall spatial region, even if the timing of that motion is not perfectly reproduced.

Spatiotemporal IoU. Spatiotemporal IoU additionally accounts for *when* motion occurs. Instead of reducing the activation map over time, it computes the overlap of the generated and ground truth binary motion masks at each frame and averages the result over the video:

$$\mathrm{IoU}_{\mathrm{ST}} = \frac{1}{N_{\mathrm{eval}}} \sum_{\ell_3=1}^{N_{\mathrm{eval}}} \frac{\left|\mathbf{A}^{\mathrm{ST}}_{\mathrm{gt},\ell_3} \cap \mathbf{A}^{\mathrm{ST}}_{\mathrm{gen},\ell_3}\right|}{\left|\mathbf{A}^{\mathrm{ST}}_{\mathrm{gt},\ell_3} \cup \mathbf{A}^{\mathrm{ST}}_{\mathrm{gen},\ell_3}\right|}. \tag{31}$$

Consequently, a prediction may achieve a relatively high Spatial IoU but a lower Spatiotemporal IoU if the motion follows approximately the correct spatial path but occurs at incorrect times.

Weighted Spatial IoU. Weighted Spatial IoU measures both *where* motion occurs and *how frequently* each spatial location is activated. Using the temporally averaged activation maps in Eq. **??**, it is defined as:

$$\mathrm{IoU}_{\mathrm{WS}} = \frac{\sum_{\ell_1=1}^{H}\sum_{\ell_2=1}^{W} \min\left(A^{\mathrm{WS}}_{\mathrm{gt},\ell_1,\ell_2}, A^{\mathrm{WS}}_{\mathrm{gen},\ell_1,\ell_2}\right)}{\sum_{\ell_1=1}^{H}\sum_{\ell_2=1}^{W} \max\left(A^{\mathrm{WS}}_{\mathrm{gt},\ell_1,\ell_2}, A^{\mathrm{WS}}_{\mathrm{gen},\ell_1,\ell_2}\right)}. \tag{32}$$

Unlike Spatial IoU, which only records whether motion ever occurs at a location, Weighted Spatial IoU distinguishes between transient motion and repeatedly occupied motion regions. This is particularly relevant for systems with oscillatory or recurrent trajectories.

MSE. The above three IoU-based metrics primarily characterize motion localization and temporal occupancy. MSE instead provides a strict pixel level comparison between the generated and ground

truth videos. For videos normalized to $[0, 1]$, let C be the number of color channels and ℓ_4 the channel index. We compute:

$$\mathrm{MSE} = \frac{1}{HWN_{\mathrm{eval}}C} \sum_{\ell_3=1}^{N_{\mathrm{eval}}} \sum_{\ell_1=1}^{H} \sum_{\ell_2=1}^{W} \sum_{\ell_4=1}^{C} \left(V^{\mathrm{gt}}_{\ell_1,\ell_2,\ell_3,\ell_4} - V^{\mathrm{gen}}_{\ell_1,\ell_2,\ell_3,\ell_4} \right)^2. \tag{33}$$

MSE is sensitive not only to object motion but also to appearance, geometry, and other pixel-level differences, and therefore complements the three activation based metrics.

Higher Spatial IoU, Spatiotemporal IoU, and Weighted Spatial IoU indicate better agreement with the ground truth physical evolution, whereas lower MSE is better. As emphasized in Physics-IQ, these metrics are complementary proxies for physical consistency rather than direct measurements of the governing physical laws.

Why We Report Component Metrics. The original Physics-IQ additionally combines the four components into a single normalized score. Its normalization uses the physical variation estimated by comparing two independent real takes of the same physical experiment. Physics-IQ Verified similarly uses a second real take to define the physical variation reference, while revising the aggregation to operate at the sample level and weight the four components equally. Because our evaluation provides only one ground truth future realization for each physical system prediction setting, such a physical variation estimate is unavailable. We therefore report the four component metrics directly rather than constructing an aggregate Physics-IQ score.

A.3 Experimental Protocol and Implementation Details

Prediction protocol. We evaluate LPhyVideo on six physical systems with two prediction protocols. The Double Pendulum, Spring Mass, Torricelli Motion, Forced 3D Robot Arm, and Moving Camera 3D Robot Arm are video continuation tasks, where an observed prefix is first converted into a physical state and the subsequent 10–15 s interval is predicted by initializing the rollout from the final observed state. Rolling Orange is a new initial condition task: the first frame provides the initial position and object scale, while the initial velocity is estimated from the first two frames; the resulting physical state is then rolled out over 0-5 s.

States and Dynamics implementation. The dynamics models follow the system formulations described below. For the four unforced systems (Double Pendulum, Rolling Orange, Spring Mass and Torricelli Motion), $\mathbf{u}(t)$ is omitted.

For the *Double Pendulum* dataset of Yang et al. (2025), we use physical states $\mathbf{z} = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^\top$, where θ_i and $\dot{\theta}_i$ denote the link angles and angular velocities. The dynamics are modeled with a Neural ODE (Chen et al., 2018), whose acceleration model is implemented as a four-layer MLP with 160 hidden units per layer and `tanh` activations. Training proceeds through one-step acceleration fitting, short-horizon rollout training, and task-horizon fine-tuning, with maximum optimization budgets of 5000, 3000, and 1000 steps, respectively. Validation rollout error is used for model selection, and future states are integrated with RK4. For state-to-pixel mapping, $\boldsymbol{\xi} = (p_0, l_1, l_2)$ contains the fixed pivot and link lengths. Figure 10 shows state trajectories over the prediction interval.

For the *Rolling Orange* experiment from Morpheus (Tragoudaras et al., 2025), we use $\mathbf{z} = [s, \dot{s}, \phi]^\top$, where s is the slope-aligned position and ϕ is the rotation angle, with $\dot{\mathbf{z}} = [\dot{s},\ \eta,\ \sigma\dot{s}/R]^\top$. Here, $\eta = a$ is the scalar translational acceleration, R is the tracked radius, and σ specifies the rolling direction; $\dot{\phi} = \sigma\dot{s}/R$ explicitly enforces no-slip rolling. The acceleration is estimated from the training trajectories by least squares and kept fixed during the test rollout. The test sequence uses a new initial condition, with $s(0)$ obtained from the first frame, $\dot{s}(0)$ estimated from the first few frames, and $\phi(0) = 0$; the rollout is then evaluated analytically. For state-to-pixel mapping, $\boldsymbol{\xi} = c_0$ contains the first frame orange center, while the predicted s and ϕ determine its translation and rotation along the slope. Figure 12 shows the state trajectories over the prediction interval.

For the *Spring Mass* experiment from Morpheus (Tragoudaras et al., 2025), we use $\mathbf{z} = [x_{\text{mass}}, \dot{x}_{\text{mass}}]^\top$, where x_{mass} denotes the mass position, with $\dot{\mathbf{z}} = [\dot{x}_{\text{mass}}, -2\beta\dot{x}_{\text{mass}} - \nu_0^2(x_{\text{mass}} - x_{\text{eq}})]^\top$. Here, x_{eq} is the equilibrium position, while β and ν_0 denote the damping coefficient and natural frequency, respectively. The parameters $\eta = [\beta, \nu_0, x_{\text{eq}}]^\top$ are identified independently from each observed sequence. For fixed damping and damped frequency, the remaining linear coefficients are solved by least squares, while the nonlinear parameters are obtained by deterministic coarse search followed by bounded nonlinear least squares refinement. The resulting analytic oscillator is propagated over the prediction interval. For state-to-pixel mapping, $\boldsymbol{\xi}$ contains the fixed upper attachment, and Π maps the predicted displacement to the corresponding mass and spring geometry in image space. Figure 14 shows state trajectories over the prediction interval.

For the *Torricelli Motion* experiment from Delfys75 (Garcia et al., 2025), we use $\mathbf{z} = [h, \sqrt{h}, y_{\text{water}}]^\top$, where h is the liquid height above the outlet and y_{water} is the image space waterline position, with $\dot{\mathbf{z}} = [-\eta\sqrt{h},\ -\eta/2,\ \eta\sqrt{h}]^\top$. Here, $\eta > 0$ is the Torricelli coefficient, which is estimated in the linearized $\sqrt{h}$ domain from the observed data and subsequently held fixed during analytic future rollout. For state-to-pixel mapping, $\boldsymbol{\xi}$ contains the fixed image space geometry, and Π maps the predicted waterline position to the corresponding liquid boundary. Figure 16 shows state trajectories over the prediction interval.

For the *Forced 3D Robot Arm* based on the Franka Emika Panda system (Zakka et al., 2022; Todorov et al., 2012), we activate three joints and use $\mathbf{z} = [q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3]^\top$. The dynamics are modeled with a control-affine Neural ODE (Chen et al., 2018) under the known open loop torque input $\mathbf{u}(t) = \boldsymbol{\tau}(t) = [\tau_1, \tau_2, \tau_3]^\top$ in N·m. The model combines a linear acceleration baseline with a nonlinear state-dependent drift and a state-dependent correction to the torque gain, while fixed gravity compensation and joint damping are included in the system dynamics. It is trained on designated training sequences, selected using validation rollouts, frozen before test prediction, and integrated with RK4 while interpolating the prescribed torque within each integration step. For state-to-pixel

mapping, $\boldsymbol{\xi}$ contains the known Panda kinematics and calibrated camera parameters. Figure 18 shows state trajectories over the prediction interval.

For the *Moving Camera 3D Robot Arm*, we use the same three active Panda joints (Zakka et al., 2022; Todorov et al., 2012) and the world-space state $\mathbf{z} = [q_1^m, q_2^m, q_3^m, \dot{q}_1^m, \dot{q}_2^m, \dot{q}_3^m]^\top$. The Neural ODE is learned entirely in world coordinates under known open-loop forcing, with acceleration combining restoring, damping, and nonlinear coupling terms with a compact learned residual. The model is integrated with RK4 while evaluating the prescribed forcing at the integration substeps. Camera pose is never provided to the dynamics model; instead, $\boldsymbol{\xi}$ contains the known Panda kinematics and time-varying calibrated camera parameters, and the predicted world-space states are projected to pixels only afterwards using the time-varying camera extrinsics. This design separates physical motion from viewpoint-induced image motion. Figure 20 shows state trajectories over the prediction interval.

Preliminary video construction. The preliminary video in Figure 1 (Eq. 10) is constructed by combining the predicted physical states with appearance information extracted from the available observation frames. Future motion is therefore determined by the corresponding dynamics model, while the rendering strategy is adapted to the visual structure of each system.

For the *Double Pendulum*, the appearance of the two rods and joints is extracted from the $10\,\mathrm{s}$ boundary frame and rigidly transformed according to the predicted link orientations.

For the *Rolling Orange*, the orange appearance and radius are obtained from the first test frame. A static background is reconstructed from the training sequences, while the object is translated according to the predicted center trajectory and its surface texture is rotated according to the predicted rolling phase.

For the *Spring Mass*, an appearance reference is selected from the observed $10\,\mathrm{s}$ frame. The mass, lower joint, and visible connector are then repositioned according to the predicted vertical displacement while preserving the fixed upper attachment.

For the *Torricelli Motion*, the $10\,\mathrm{s}$ frame provides the appearance of the water surface, floating ball, and outflow jet. The predicted water level determines the vertical motion of the liquid surface and floating ball, while the jet geometry is adjusted according to the predicted $\sqrt{h}$ evolution.

For the *Forced 3D Robot Arm*, the $10\,\mathrm{s}$ frame provides the visual reference after removing the synthetic tracking fiducials. The subsequent articulated motion is rendered from the predicted joint configurations using the same Panda geometry and calibrated camera model.

For the *Moving Camera 3D Robot Arm*, the $10\,\mathrm{s}$ frame likewise provides the visual reference without synthetic tracking fiducials. The subsequent articulated motion is rendered from the predicted world-space joint configurations using the same Panda geometry, while the camera continues to move according to the prescribed time-varying extrinsics.

LPhyVideo generative refinement. Generative refinement uses Wan2.1-VACE-1.3B conditioned on the appearance reference, preliminary video, edit mask and prompt. Prompt extension is set to `plain`. For results reported over multiple stochastic runs, we use seeds $\{2025, 2026, 2027, 2028, 2029\}$. The inference settings used by the current evaluation scripts are summarized in Table 5.

Table 5: LPhyVideo generative refinement settings. "Frames/FPS" reports the final number of frames and playback frame rate. Shift and Guidance denote `sample_shift` and `sample_guide_scale`, respectively.

System	Resolution	Frames/FPS	Solver	Shift	Guidance	Steps
Double Pendulum	480×832	81/16	UniPC	16	5.0	50
Rolling Orange	832×480	81/16	UniPC	16	5.0	50
Spring Mass	480×832	81/16	UniPC	16	5.0	50
Torricelli Motion	832×480	81/16	UniPC	16	5.0	50
Forced 3D Robot Arm	480×832	81/16	UniPC	16	5.0	50
Moving Camera 3D Robot Arm	480×832	81/16	UniPC	16	5.0	50

Baseline implementation. All methods are evaluated at the same target resolution, video duration, and frame rate, and use the same prompt and evaluation protocol. For methods evaluated over multiple runs, we use the same five random seeds $\{2025, 2026, 2027, 2028, 2029\}$. The generic baselines use their native conditioning interfaces. *Wan2.2 (I2V)* and *HunyuanVideo 1.5 (I2V)* receive the real prediction boundary frame, whereas *LTX-2.3 (Video Continuation)* receives the observed video clip. Wan2.2 uses 50 denoising steps with UniPC, guidance scales $3.5/3.5$ for its two guidance phases, and flow shift 5. HunyuanVideo 1.5 uses 50 steps, guidance scale 6.0, and flow shift 5. LTX-2.3 Video Continuation uses 50 steps. The guided baselines use the same predicted physical evolution as LPhyVideo: *Wan-Move* receives the prediction boundary frame and predicted trajectories and uses 50 steps with guidance scale 5.0, while *LTX-2.3 + EditAnything* receives the appearance reference and preliminary video and uses 50 steps.

Text prompts. The exact prompts used by LPhyVideo inference scripts are listed below.

Double Pendulum. *"A real fixed camera video of a red double pendulum swinging naturally. Keep the camera viewpoint and background unchanged. Preserve the red two link pendulum, rigid link lengths, joint structure, and the motion specified by the source video. Make the result realistic, temporally smooth, and visually consistent."*

Rolling Orange. *"A real fixed camera video of a natural orange rolling down an inclined ramp. Keep the camera viewpoint, ramp geometry, lighting, and background unchanged. Do not replace the rolling motion with translational sliding. Do not freeze the surface texture. Keep the orange in continuous plausible contact with the ramp, without floating, teleporting, shape deformation, or splitting."*

Spring Mass. *"A real fixed camera video of a shiny metal mass attached to a vertical helical metal coil spring. The mass moves only up and down with natural damped spring-mass oscillation. Keep the camera viewpoint, support rail, fixed upper attachment point, and background unchanged. Preserve the metal mass shape and metallic texture. Keep clearly visible helical spring turns continuously attached between the fixed upper support and the mass. Make the result realistic, temporally smooth, mechanically consistent, and visually stable."*

Torricelli Motion. *"A real fixed camera laboratory video of Torricelli outflow from a transparent water container. The water surface is a subtle thin continuous transparent interface. A single very thin transparent gravity driven water stream exits from the outlet, with natural transparency, and soft refraction."*

Forced 3D Robot Arm. *"A realistic fixed camera video of a robotic arm moving smoothly under controlled external actuation in a clean laboratory style scene with a gray checkerboard floor. Keep the camera viewpoint, framing, floor, lighting, and background unchanged. Do not add, remove, duplicate, or reshape robot links. Do not add colored tracking markers, dots, labels, or overlays. Make the result realistic, temporally smooth, physically coherent, and visually consistent."*

Moving Camera 3D Robot Arm. *"A realistic video of a white Franka Emika Panda robotic arm moving smoothly under controlled external actuation in a dark industrial workcell. The camera smoothly orbits throughout the shot with no cuts or sudden viewpoint changes. Preserve the circular platform, dark checkerboard floor, colored workcell props, lighting, and reflections. Preserve the exact articulated robot structure, rigid link geometry, joint layout, white robot appearance, end effector, and the motion specified by the source video. Do not add, remove, duplicate, or reshape robot links. Do not add colored tracking markers, dots, labels, or overlays. Make the result realistic, temporally smooth, physically coherent, and visually consistent."*

Evaluation protocol. All generated videos, including LPhyVideo and the baselines, are evaluated from their rendered RGB outputs using the same system-specific tracking and measurement pipeline. Each output is first aligned to the corresponding physical prediction interval and is then evaluated using the Physics-IQ component metrics and Trajectory Error defined in the main paper.

A.4 Visual Observation and Tracking

The tracking operator $\mathcal{T}$ is implemented with system-specific image measurements using OpenCV (Howse, 2013). We combine color-based segmentation, template matching (Briechle & Hanebeck, 2001), and geometry-constrained tracking to extract system-relevant image space observations. When required, a small set of points or regions is manually initialized in the first frame, after which tracking proceeds automatically across subsequent frames using temporal continuity and the known geometric structure of the system. For the synthetic robot arm systems, color-coded visual fiducials are detected in RGB and interpreted together with the known kinematics and calibrated camera parameters. The resulting tracked quantities provide the geometric observations used to construct the corresponding physical states. Figures 4–9 illustrate the measured geometry, and Sec. A.3 describes its conversion into physical states.

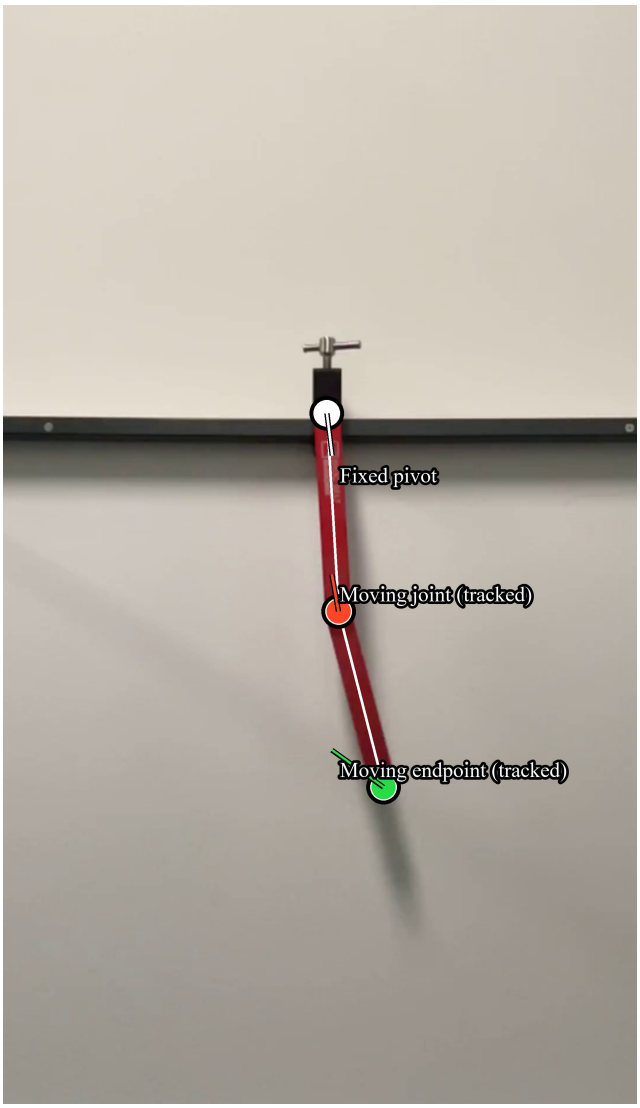

Figure 4: **Double Pendulum tracking.** The fixed pivot, intermediate joint, and endpoint define the two link directions. Their tracked image space trajectories are converted into the link angles and angular velocities for physical state construction.



Figure 5: **Rolling Orange tracking.** The orange center and radius are tracked together with a salient surface feature on the orange. The center trajectory characterizes the translational motion along the ramp, while the displacement of the surface feature relative to the center provides a visual cue for the rotational phase, allowing rolling motion to be distinguished from pure translation.

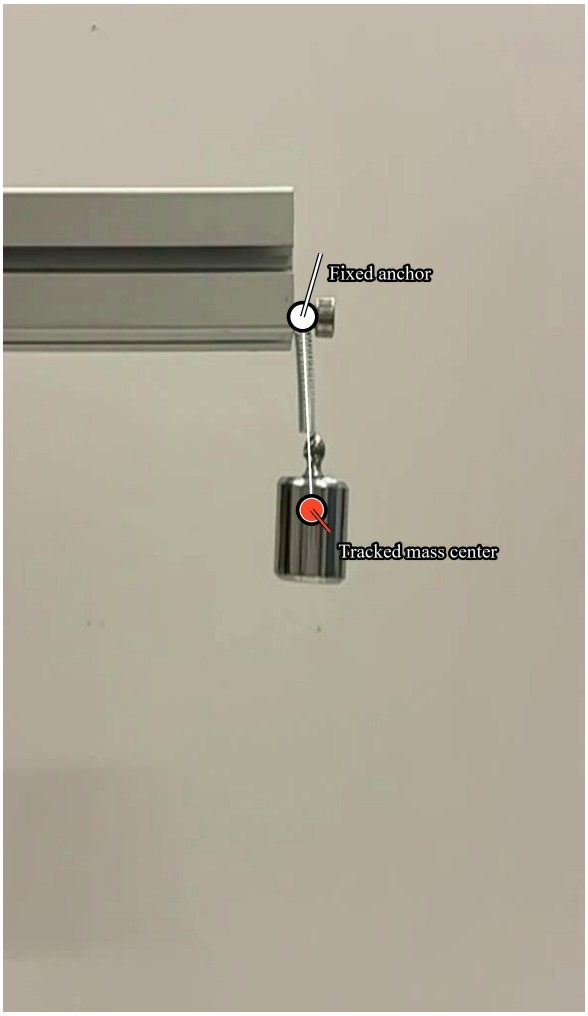

Figure 6: **Spring Mass tracking.** The moving mass is tracked relative to the fixed upper attachment. Their vertical separation defines the displacement used by the damped oscillator model.

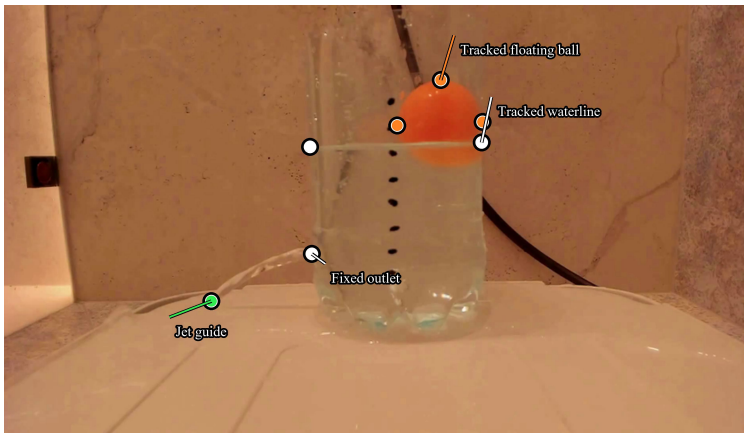

Figure 7: **Torricelli Motion tracking.** The initialization identifies the visible water surface span, fixed outlet, initial jet direction, and floating ball. Subsequent tracking provides the water surface position and floating ball geometry used to construct the water head state and rendering constraints.

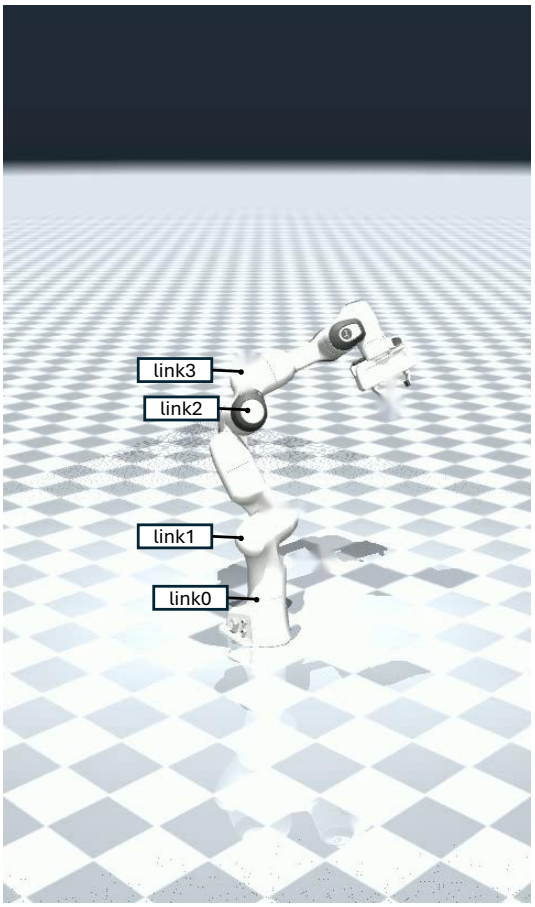

Figure 8: **Forced 3D Robot Arm tracking.** The three active joints correspond to the rotations between consecutive robot links from the base toward the end effector. The joint configuration is recovered from tracked visual observations using the known robot kinematics.

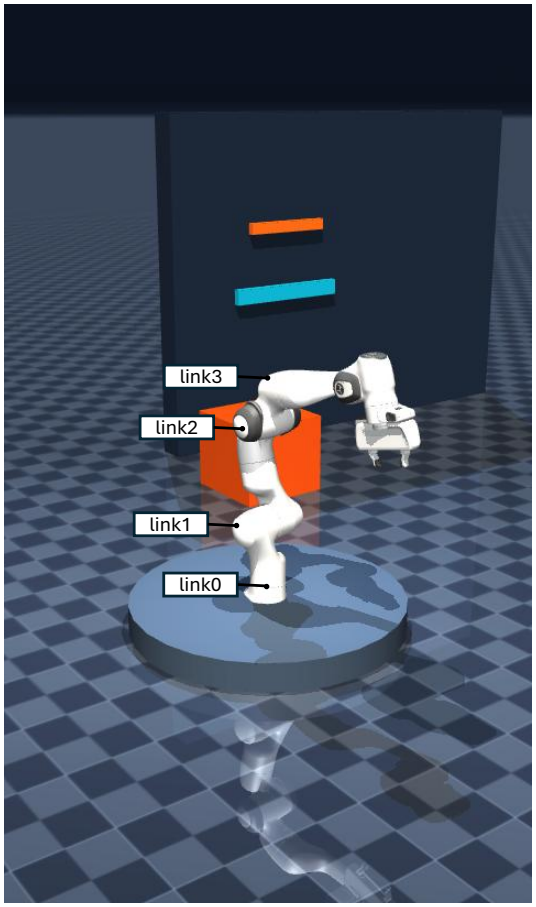

Figure 9: **Moving Camera 3D Robot Arm tracking.** The three active joints correspond to the rotations between consecutive robot links from the base toward the end effector. The joint configuration is recovered from tracked visual observations using the known robot kinematics and time-varying camera geometry.

A.5 Detailed Experimental Results

The state trajectory plots below use the same coordinate conventions as the corresponding dynamics models.

For the four planar systems (Double Pendulum, Rolling Orange, Spring Mass and Torricelli Motion), translational quantities derived from image geometry are reported in fixed image space units (pixels, with corresponding velocities in pixels per second), while angular quantities are represented in radians and radians per second. These coordinates should not be confused with unconstrained pixel space representations: they are low-dimensional physical coordinates constructed by Φ from tracked geometry. Under the camera settings used in these experiments, the conversion between the translational coordinates of the relevant image space and their counterparts in the physical space differs only by a fixed scale, which is consistently absorbed into the geometric representation and the identified dynamics parameters, as discussed in the Appendix A.1.

For both 3D robot arm systems (Forced 3D robot arm and Moving Camera 3D Robot Arm), the states are recovered from tracked visual observations using the known robot kinematics and calibrated camera model, and are reported directly in joint coordinates; for the moving camera variant, the time-varying camera extrinsics are explicitly accounted for so that viewpoint-induced image motion does not enter the physical state representation.

A.5.1 Per-System Quantitative Results

Table 6: Per-system quantitative comparison. Metric definitions are given in Sec. 4.1 and Appendix A.2. For methods evaluated over five random seeds, results are reported as mean $\pm$ standard deviation. LPhyVideo before Refinement is evaluated once.

System	Model	Spatial IoU ↑	Spatiotemporal IoU ↑	Weighted Spatial IoU ↑	MSE ↓	Trajectory Error ↓
Double Pendulum	Wan2.2 (I2V)	0.28827 ± 0.10108	0.12905 ± 0.04553	0.29062 ± 0.10581	0.01825 ± 0.01060	0.15969 ± 0.05615
	HunyuanVideo 1.5 (I2V)	0.52616 ± 0.13675	0.15613 ± 0.02403	0.42021 ± 0.08122	0.01198 ± 0.00419	0.10684 ± 0.01606
	LTX-2.3 (Video Continuation)	0.17562 ± 0.05421	0.08837 ± 0.01840	0.02251 ± 0.01045	0.21980 ± 0.00048	0.08757 ± 0.01295
	Wan-Move	0.67684 ± 0.03370	0.44144 ± 0.03984	0.53642 ± 0.02846	0.00571 ± 0.00163	0.01677 ± 0.00307
	LTX-2.3 + EditAnything	0.68465 ± 0.02567	0.49241 ± 0.01331	0.56391 ± 0.01604	0.00895 ± 0.00014	0.00947 ± 0.00016
	LPhyVideo before Refinement	0.72963	0.48524	0.62193	0.00253	**0.00897**
	LPhyVideo	$\mathbf{0.80072} \pm 0.02430$	$\mathbf{0.51232} \pm 0.01273$	$\mathbf{0.65995} \pm 0.00933$	$\mathbf{0.00238} \pm 0.00006$	0.00905 ± 0.00010
Rolling Orange	Wan2.2 (I2V)	0.33526 ± 0.27172	0.09517 ± 0.10362	0.28486 ± 0.27583	0.02064 ± 0.01802	0.10481 ± 0.02957
	HunyuanVideo 1.5 (I2V)	0.65680 ± 0.07507	0.16077 ± 0.08197	0.55528 ± 0.05302	0.00259 ± 0.00070	0.08176 ± 0.04571
	LTX-2.3 (Video Continuation)	0.25533 ± 0.20349	0.04023 ± 0.01820	0.13845 ± 0.14659	0.09070 ± 0.01376	0.15949 ± 0.03295
	Wan-Move	0.39284 ± 0.19392	0.22641 ± 0.10741	0.34930 ± 0.16844	0.01705 ± 0.01060	0.03962 ± 0.02871
	LTX-2.3 + EditAnything	0.71602 ± 0.11349	0.44069 ± 0.06649	0.58762 ± 0.10288	0.00913 ± 0.00036	0.01713 ± 0.00218
	LPhyVideo before Refinement	0.90233	0.69105	0.76958	**0.00039**	**0.00432**
	LPhyVideo	$\mathbf{0.91114} \pm 0.00491$	$\mathbf{0.71355} \pm 0.01513$	$\mathbf{0.80580} \pm 0.02070$	0.00042 ± 0.00003	0.00478 ± 0.00008
Spring Mass	Wan2.2 (I2V)	0.16728 ± 0.03498	0.12019 ± 0.06632	0.08994 ± 0.03000	0.00926 ± 0.00243	0.06873 ± 0.00910
	HunyuanVideo 1.5 (I2V)	0.42697 ± 0.11501	0.13850 ± 0.03954	0.19906 ± 0.05032	0.00504 ± 0.00023	0.03520 ± 0.00753
	LTX-2.3 (Video Continuation)	0.16169 ± 0.09787	0.10924 ± 0.04711	0.13778 ± 0.06324	0.01718 ± 0.00300	0.06896 ± 0.04213
	Wan-Move	0.60546 ± 0.24045	0.58763 ± 0.05823	0.54685 ± 0.12584	0.00341 ± 0.00197	0.01251 ± 0.00128
	LTX-2.3 + EditAnything	0.24997 ± 0.25871	0.46100 ± 0.10111	0.17248 ± 0.23539	0.01776 ± 0.01102	0.01883 ± 0.00851
	LPhyVideo before Refinement	**0.75319**	**0.62641**	0.55873	0.00286	**0.01090**
	LPhyVideo	0.74493 ± 0.01979	0.61915 ± 0.01815	$\mathbf{0.56886} \pm 0.01144$	$\mathbf{0.00218} \pm 0.00008$	0.01097 ± 0.00005
Torricelli Motion	Wan2.2 (I2V)	0.21078 ± 0.06845	0.12750 ± 0.02607	0.15952 ± 0.04033	0.00085 ± 0.00025	0.04879 ± 0.01853
	HunyuanVideo 1.5 (I2V)	0.08306 ± 0.03621	0.05822 ± 0.01866	0.05651 ± 0.02830	0.00463 ± 0.00408	0.13558 ± 0.02928
	LTX-2.3 (Video Continuation)	0.09912 ± 0.07868	0.11011 ± 0.04386	0.06610 ± 0.04089	0.00388 ± 0.00138	0.04742 ± 0.02348
	Wan-Move	0.05295 ± 0.02544	0.04785 ± 0.04685	0.05183 ± 0.03268	0.00381 ± 0.00057	0.02054 ± 0.02718
	LTX-2.3 + EditAnything	0.52829 ± 0.03264	0.43999 ± 0.03172	0.46764 ± 0.03903	0.00108 ± 0.00024	0.00542 ± 0.00065
	LPhyVideo before Refinement	0.59184	**0.55246**	**0.58044**	0.00034	**0.00406**
	LPhyVideo	$\mathbf{0.59500} \pm 0.02596$	0.48186 ± 0.04111	0.52755 ± 0.04573	$\mathbf{0.00027} \pm 0.00004$	0.00750 ± 0.00154
Forced 3D Robot Arm	Wan2.2 (I2V)	0.13450 ± 0.07538	0.07024 ± 0.03914	0.08020 ± 0.07121	0.01165 ± 0.00413	0.05892 ± 0.02460
	HunyuanVideo 1.5 (I2V)	0.16332 ± 0.06867	0.20353 ± 0.13240	0.11209 ± 0.08717	0.01145 ± 0.00192	0.05825 ± 0.01251
	LTX-2.3 (Video Continuation)	0.08978 ± 0.06738	0.19670 ± 0.12245	0.05934 ± 0.07133	0.01090 ± 0.00921	0.03454 ± 0.00750
	Wan-Move	0.35028 ± 0.02980	0.31640 ± 0.04597	0.31881 ± 0.02830	0.00750 ± 0.00048	0.01428 ± 0.00039
	LTX-2.3 + EditAnything	0.33938 ± 0.13496	0.43963 ± 0.08431	0.37076 ± 0.20683	0.00734 ± 0.00228	0.01486 ± 0.00064
	LPhyVideo before Refinement	0.55521	0.49388	**0.59324**	0.00276	**0.01327**
	LPhyVideo	$\mathbf{0.56544} \pm 0.01777$	$\mathbf{0.49939} \pm 0.00446$	0.58478 ± 0.00796	$\mathbf{0.00218} \pm 0.00004$	0.01376 ± 0.00010
Moving Camera 3D Robot Arm	Wan2.2 (I2V)	0.39165 ± 0.07236	0.07667 ± 0.02296	0.20238 ± 0.06264	0.05136 ± 0.01956	0.13195 ± 0.00827
	HunyuanVideo 1.5 (I2V)	0.11530 ± 0.07923	0.03527 ± 0.02685	0.09136 ± 0.08174	0.04328 ± 0.00475	0.15606 ± 0.03852
	LTX-2.3 (Video Continuation)	0.38880 ± 0.08893	0.03771 ± 0.01652	0.20820 ± 0.03747	0.07160 ± 0.01073	0.16872 ± 0.02557
	Wan-Move	0.64651 ± 0.20399	0.31984 ± 0.14554	0.46443 ± 0.14171	0.02622 ± 0.00644	0.06367 ± 0.04354
	LTX-2.3 + EditAnything	0.91854 ± 0.00367	0.70486 ± 0.00330	0.82537 ± 0.00302	0.01327 ± 0.00011	0.02770 ± 0.00064
	LPhyVideo before Refinement	**0.94192**	0.74680	0.86883	0.00436	**0.01249**
	LPhyVideo	0.93237 ± 0.00233	$\mathbf{0.75003} \pm 0.00373$	$\mathbf{0.87872} \pm 0.00248$	$\mathbf{0.00420} \pm 0.00005$	0.01372 ± 0.00050

A.5.2 DOUBLE PENDULUM RESULTS

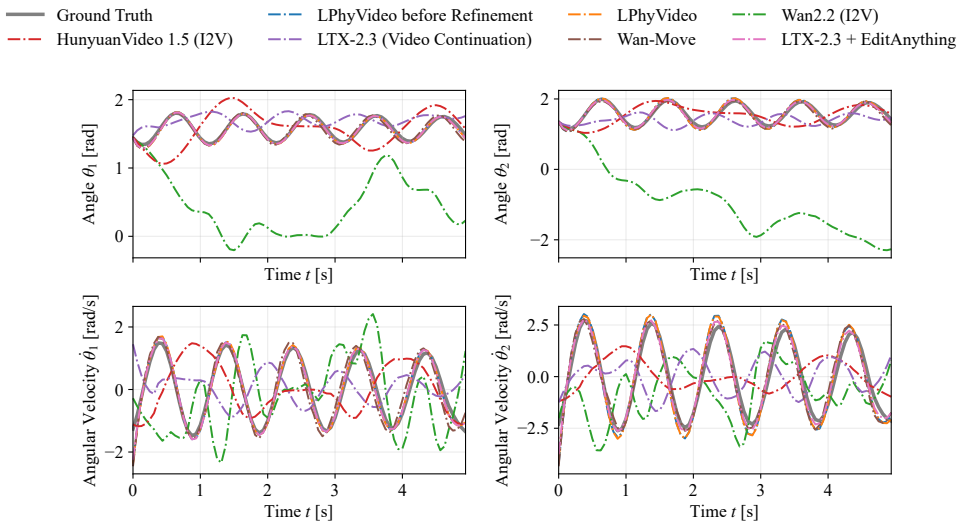

Figure 10: Double Pendulum state trajectories over the 5 s prediction interval. We compare the link angles θ_1, θ_2 and angular velocities $\dot{\theta}_1, \dot{\theta}_2$ recovered from generated videos against the ground truth.

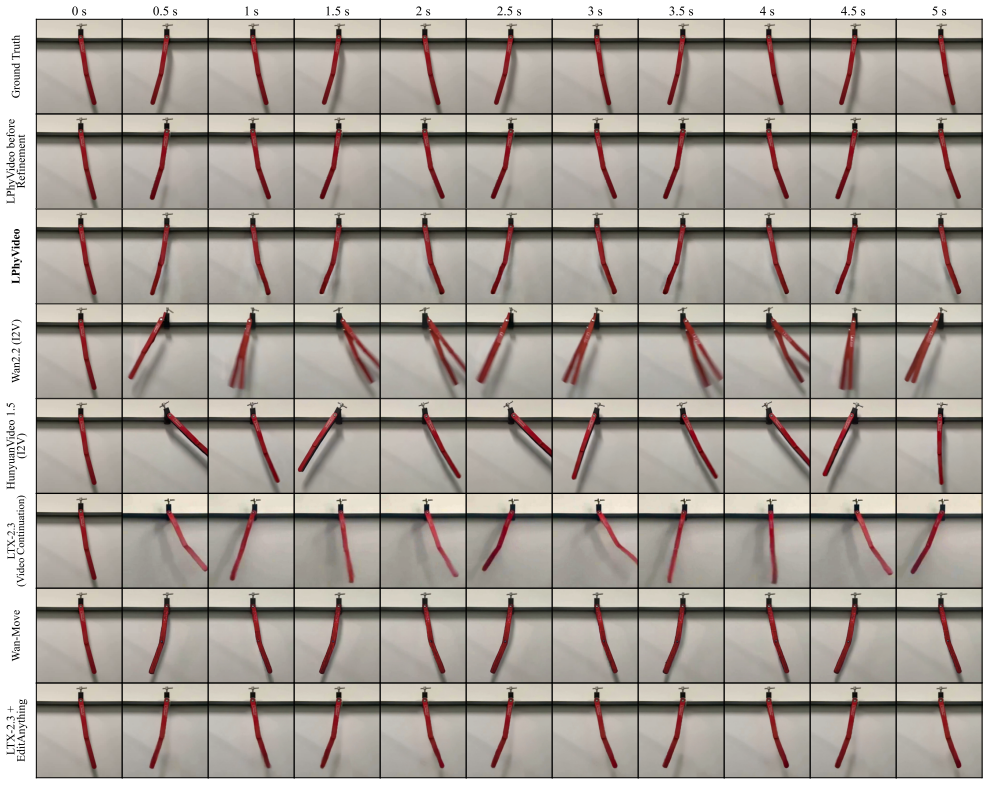

Figure 11: Qualitative comparison of Double Pendulum video predictions over the 5 s prediction interval. Frames are shown every 0.5 s for the ground truth, LPhyVideo before refinement, LPhyVideo, and all generation baselines.

A.5.3 ROLLING ORANGE RESULTS

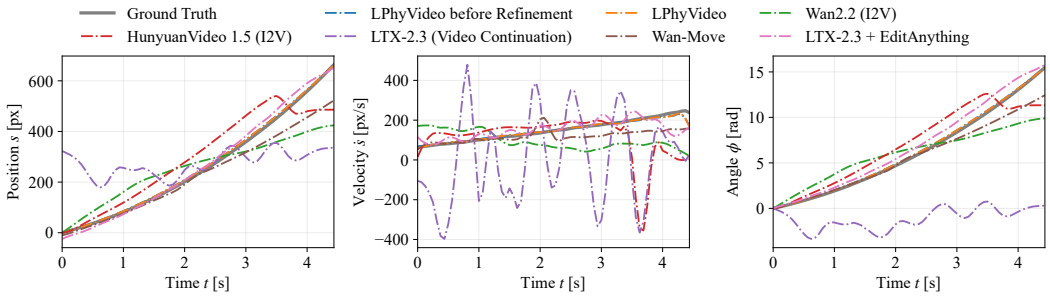

Figure 12: Rolling Orange state trajectories over the prediction interval. We compare the predicted slope-aligned position s, velocity $\dot{s}$, and rotation angle ϕ recovered from each generated video against the ground truth.



Figure 13: Qualitative comparison of Rolling Orange video predictions over the 5 s prediction interval. Frames are shown every 1 s for the ground truth, LPhyVideo before refinement, LPhyVideo, and all generation baselines.

A.5.4 SPRING MASS RESULTS

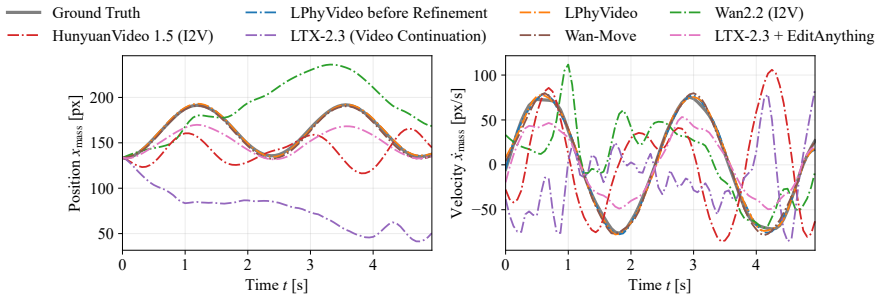

Figure 14: Spring Mass state trajectories over the prediction interval. We compare the predicted mass position x_{mass} and velocity $\dot{x}_{\text{mass}}$ recovered from each generated video against the ground truth.

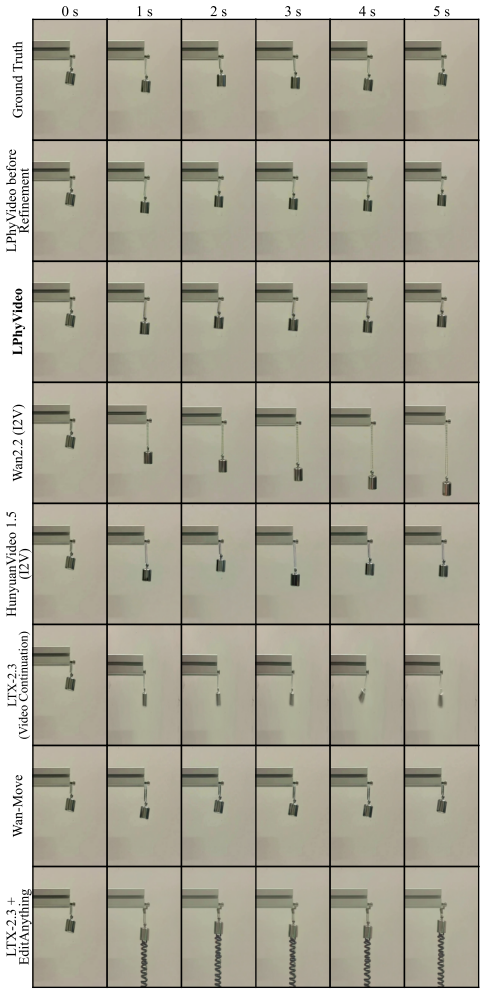

Figure 15: Qualitative comparison of Spring Mass video predictions. Frames are shown every 1 s for the ground truth, LPhyVideo before refinement, LPhyVideo, and all generation baselines.

A.5.5 TORRICELLI MOTION RESULTS

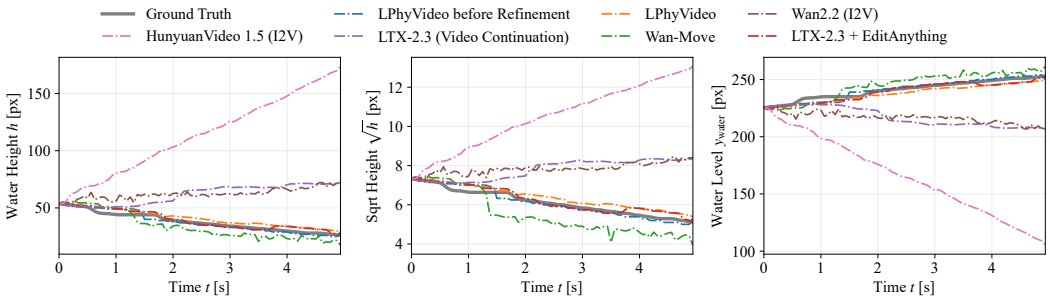

Figure 16: Torricelli Motion state trajectories over the prediction interval. We compare the predicted liquid height h, transformed state $\sqrt{h}$, and image space waterline position y_{water} recovered from each generated video against the ground truth.

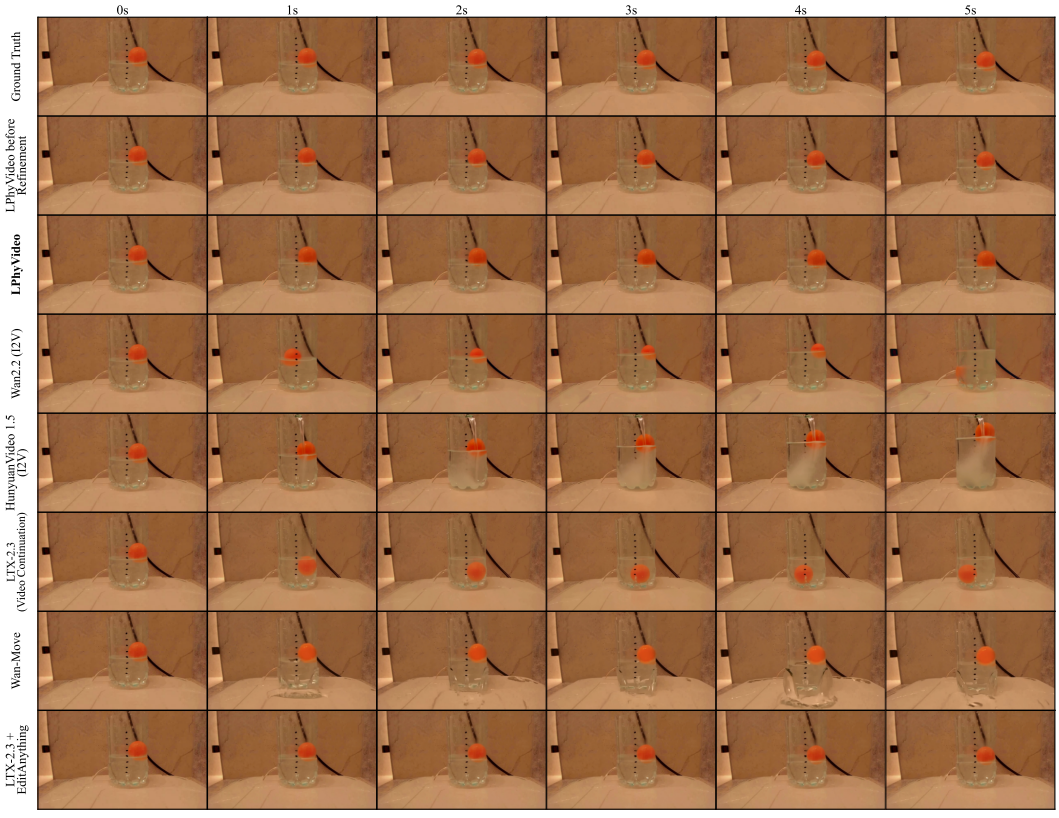

Figure 17: Qualitative comparison of Torricelli Motion video predictions over the 5 s prediction interval. Frames are shown every 1 s for the ground truth, LPhyVideo before refinement, LPhyVideo, and all generation baselines.

A.5.6 FORCED 3D ROBOT ARM RESULTS

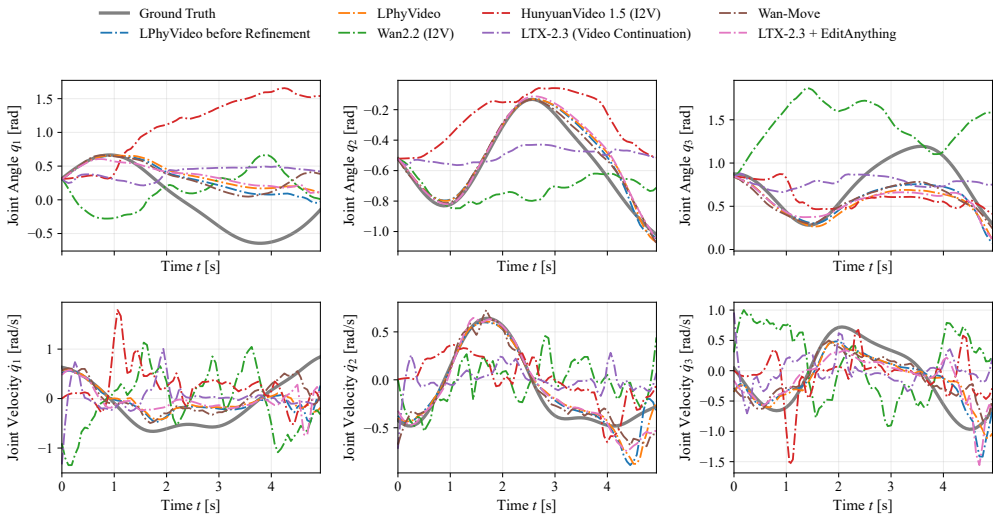

Figure 18: Forced 3D Robot Arm state trajectories over the 5 s prediction interval. We compare the predicted joint angles q_1, q_2, q_3 and angular velocities $\dot{q}_1, \dot{q}_2, \dot{q}_3$ against the ground truth.



Figure 19: Qualitative comparison of Forced 3D Robot Arm video predictions over the 5 s prediction interval. Frames are shown every 0.5 s for the ground truth, LPhyVideo before refinement, LPhyVideo, and all generation baselines.

A.5.7 Moving Camera 3D Robot Arm Results

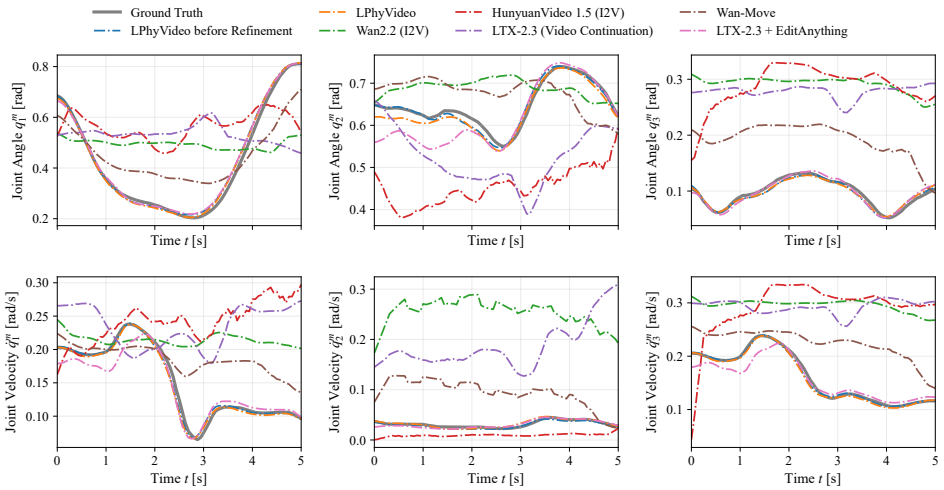

Figure 20: Moving Camera 3D Robot Arm state trajectories over the 5 s prediction interval. We compare the predicted joint angles q_1^m, q_2^m, q_3^m and angular velocities $\dot{q}_1^m, \dot{q}_2^m, \dot{q}_3^m$ against the ground truth.



Figure 21: Qualitative comparison of Moving Camera 3D Robot Arm video predictions over the 5 s prediction interval. Frames are shown every 0.5 s for the ground truth, LPhyVideo before refinement, LPhyVideo, and all generation baselines.

A.5.8 PER-SYSTEM ABLATION RESULTS

Table 7: Per-system component ablations of LPhyVideo. Each variant changes one component while keeping the remaining pipeline, VACE configuration, and evaluation protocol fixed. For variants evaluated over five random seeds, results are reported as mean ± standard deviation.

System	Variant	Spatial IoU ↑	Spatiotemporal IoU ↑	Weighted Spatial IoU ↑	MSE ↓	Trajectory Error ↓
Double Pendulum	w/o Dynamics Rollout	0.15213 ± 0.00308	0.05623 ± 0.00152	0.08040 ± 0.00221	0.01051 ± 0.00026	0.19713 ± 0.00010
	w/o Preliminary Video	0.59305 ± 0.25825	0.18751 ± 0.06569	0.37935 ± 0.21185	0.01234 ± 0.00304	0.06412 ± 0.00015
	w/o Edit Mask	0.74056 ± 0.02183	0.47364 ± 0.00272	0.62896 ± 0.00146	0.00479 ± 0.00068	0.01091 ± 0.00006
	LPhyVideo	**0.80072** ± 0.02430	**0.51232** ± 0.01273	**0.65995** ± 0.00933	**0.00238** ± 0.00006	**0.00905** ± 0.00010
Rolling Orange	w/o Dynamics Rollout	0.26596 ± 0.00702	0.19034 ± 0.01660	0.21327 ± 0.02042	0.00166 ± 0.00002	0.09812 ± 0.00016
	w/o Preliminary Video	0.01489 ± 0.00915	0.06944 ± 0.02622	0.24987 ± 0.00464	0.00213 ± 0.00006	0.19691 ± 0.00004
	w/o Edit Mask	0.85113 ± 0.00933	0.65083 ± 0.01699	0.68448 ± 0.02054	0.00094 ± 0.00056	0.00536 ± 0.00004
	LPhyVideo	**0.91114** ± 0.00491	**0.71355** ± 0.01513	**0.80580** ± 0.02070	**0.00042** ± 0.00003	**0.00478** ± 0.00008
Spring Mass	w/o Dynamics Rollout	0.19370 ± 0.07631	0.30771 ± 0.03975	0.03993 ± 0.01767	0.00592 ± 0.00016	0.07845 ± 0.00842
	w/o Preliminary Video	0.13152 ± 0.05415	0.41897 ± 0.03342	0.03339 ± 0.01838	0.00572 ± 0.00065	0.01112 ± 0.00019
	w/o Edit Mask	0.73159 ± 0.00997	0.60209 ± 0.02525	0.55583 ± 0.01057	0.00291 ± 0.00021	0.01109 ± 0.00011
	LPhyVideo	**0.74493** ± 0.01979	**0.61915** ± 0.01815	**0.56886** ± 0.01144	**0.00218** ± 0.00008	**0.01097** ± 0.00005
Torricelli Motion	w/o Dynamics Rollout	0.55316 ± 0.02671	0.46832 ± 0.05407	0.48392 ± 0.04242	0.00036 ± 0.00008	0.00899 ± 0.00170
	w/o Preliminary Video	0.00622 ± 0.02856	0.11120 ± 0.05210	0.00104 ± 0.08329	0.00040 ± 0.00003	0.03182 ± 0.00356
	w/o Edit Mask	0.58643 ± 0.02253	0.46744 ± 0.02821	0.51284 ± 0.04914	0.00115 ± 0.00032	0.00967 ± 0.00111
	LPhyVideo	**0.59500** ± 0.02596	**0.48186** ± 0.04111	**0.52755** ± 0.04573	**0.00027** ± 0.00004	**0.00750** ± 0.00154
Forced 3D Robot Arm	w/o Dynamics Rollout	0.08710 ± 0.02225	0.08257 ± 0.02569	0.03896 ± 0.03085	0.00694 ± 0.00018	0.10035 ± 0.00052
	w/o Preliminary Video	0.06894 ± 0.03489	0.38750 ± 0.02187	0.06769 ± 0.03183	0.00523 ± 0.00041	0.03251 ± 0.00042
	w/o Edit Mask	0.48972 ± 0.01963	0.39487 ± 0.02195	0.50874 ± 0.02410	0.00348 ± 0.00042	0.02498 ± 0.00012
	LPhyVideo	**0.56544** ± 0.01777	**0.49939** ± 0.00446	**0.58478** ± 0.00796	**0.00218** ± 0.00004	**0.01376** ± 0.00010
Moving Camera 3D Robot Arm	w/o Dynamics Rollout	0.47613 ± 0.00548	0.12507 ± 0.01009	0.34496 ± 0.00446	0.02396 ± 0.00026	0.22864 ± 0.00164
	w/o Preliminary Video	0.48135 ± 0.00475	0.28905 ± 0.01413	0.32427 ± 0.00495	0.03302 ± 0.00011	0.31013 ± 0.01960
	w/o Edit Mask	0.83673 ± 0.00478	0.74041 ± 0.00971	0.85674 ± 0.00400	0.00850 ± 0.00119	0.01824 ± 0.00150
	LPhyVideo	**0.93237** ± 0.00233	**0.75003** ± 0.00373	**0.87872** ± 0.00248	**0.00420** ± 0.00005	**0.01372** ± 0.00050

Table 8: Per-system generative backbone comparison. VACE-1.3B and VACE-14B use the same physical rollout, preliminary video, input resolution, generation settings, and 50 denoising steps. Results over five random seeds are reported as mean ± standard deviation.

System	Model & Parameters	GPU	Time	Spatial IoU ↑	Spatiotemporal IoU ↑	Weighted Spatial IoU ↑	MSE ↓	Trajectory Error ↓
Double Pendulum	LPhyVideo (VACE-1.3B)	1 × A40	12′10″	0.80072 ± 0.02430	0.51232 ± 0.01273	0.65995 ± 0.00933	0.00238 ± 0.00006	0.00905 ± 0.00010
	LPhyVideo (VACE-14B)	8 × A40	10′11″	0.81177 ± 0.01550	0.50884 ± 0.01106	0.63160 ± 0.01008	0.00231 ± 0.00026	0.00992 ± 0.00026
Rolling Orange	LPhyVideo (VACE-1.3B)	1 × A40	12′01″	0.91114 ± 0.00491	0.71355 ± 0.01513	0.80580 ± 0.02070	0.00042 ± 0.00003	0.00478 ± 0.00008
	LPhyVideo (VACE-14B)	8 × A40	10′08″	0.91070 ± 0.01634	0.75283 ± 0.01261	0.81121 ± 0.01624	0.00031 ± 0.00005	0.00462 ± 0.00005
Spring Mass	LPhyVideo (VACE-1.3B)	1 × A40	11′56″	0.74493 ± 0.01979	0.61915 ± 0.01815	0.56886 ± 0.01144	0.00218 ± 0.00008	0.01097 ± 0.00005
	LPhyVideo (VACE-14B)	8 × A40	10′09″	0.72151 ± 0.03163	0.60391 ± 0.01013	0.56480 ± 0.03469	0.00213 ± 0.00013	0.01087 ± 0.00003
Torricelli Motion	LPhyVideo (VACE-1.3B)	1 × A40	12′09″	0.59500 ± 0.02596	0.48186 ± 0.04111	0.52755 ± 0.04573	0.00027 ± 0.00004	0.00750 ± 0.00154
	LPhyVideo (VACE-14B)	8 × A40	10′09″	0.55956 ± 0.01986	0.43160 ± 0.06161	0.51675 ± 0.04057	0.00023 ± 0.00008	0.00914 ± 0.00074
Forced 3D Robot Arm	LPhyVideo (VACE-1.3B)	1 × A40	12′22″	0.56544 ± 0.01777	0.49939 ± 0.00446	0.58478 ± 0.00796	0.00218 ± 0.00004	0.01376 ± 0.00010
	LPhyVideo (VACE-14B)	8 × A40	10′12″	0.58347 ± 0.01650	0.47261 ± 0.02369	0.59143 ± 0.00588	0.00211 ± 0.00014	0.01436 ± 0.00013
Moving Camera 3D Robot Arm	LPhyVideo (VACE-1.3B)	1 × A40	12′26″	0.93237 ± 0.00233	0.75003 ± 0.00373	0.87872 ± 0.00248	0.00420 ± 0.00005	0.01372 ± 0.00050
	LPhyVideo (VACE-14B)	8 × A40	10′13″	0.93895 ± 0.00203	0.72281 ± 0.00826	0.84748 ± 0.00390	0.00474 ± 0.00015	0.01432 ± 0.00043