PHYSV2V: SIMULATION-CONTROLLED GENERATIVE VIDEO EDITING OF OBJECT PHYSICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent video-to-video (V2V) generative models excel at coarse semantic editing, but their lack of explicit physical grounding causes them to hallucinate incorrect motions when tasked with editing object physics, such as altering object mass or velocity. Conversely, although recent physics-integrated image-to-video (I2V) methods provide such explicit grounding, their reliance on single-image inputs prevents them from accurately recovering the full dynamic state needed to preserve a source scene's physical properties. To address this gap, we introduce PhysV2V, a framework for editing object physics in videos that couples a differentiable physics simulator with a conditional video diffusion model to enable explicit control over physical parameters. Given an input video featuring object dynamics, PhysV2V first estimates 3D scene geometry using SAM-3D to extract object meshes. Second, it solves an inverse problem to optimize physical properties of both rigid and deformable objects (e.g., initial velocities, restitution, Young's Modulus), establishing a physically grounded proxy whose simulated trajectories faithfully replicate the observed video. Finally, after applying user-specified physics edits, PhysV2V projects points from the updated 3D simulation into 2D to condition a foundation video generator. Evaluations across simulated and real-world videos demonstrate that PhysV2V significantly improves physical realism, visual quality, and editing fidelity over state-of-the-art baselines.

1 INTRODUCTION

The ability to precisely manipulate the physical behavior of objects within a video is critical for immersive content creation, visual effects, and robotic simulation. While text-to-video and video-to-video (V2V) diffusion models (Blattmann et al., 2023; Ho et al., 2020; 2022; Jiang et al., 2025) excel at coarse semantic edits such as object replacement and style transfer, they struggle with exact edits of object physics. For instance, they lack the explicit mechanisms to reliably make a ball roll 15% faster or increase its mass by 20% to alter a collision while strictly preserving all other initial conditions. To bridge this gap, we propose a generative pipeline for physics-based, controllable editing of object physics in videos. As an initial step toward this broader goal, we focus on fundamental object dynamics, such as colliding balls or falling boxes, demonstrating how physical parameters can be precisely manipulated directly from an input video.

Executing controlled changes to a video's physical dynamics requires an accurate representation of the scene's underlying properties, which is inherently challenging without ground-truth data. Recent research tackles this problem from different angles, but none achieve high-fidelity, physics-based V2V editing (see Table 1). Foundation V2V models (Team, 2025; Wan et al., 2025; Yang et al., 2024) rely on an implicit, data-driven understanding of physics, leading to frequent hallucinations when simulating object motion under edited forces. Other approaches attempt to bridge this gap by integrating classical physical simulators with generative models, yet they also fall short for this specific task. They either focus on image-to-video (I2V) generation (Liu et al., 2024; 2026; Wang et al., 2025a), which lacks sufficient context to estimate latent scene parameters, or they rely on 3D Gaussian Splatting parameterizations (Liu et al., 2025; Xie et al., 2024; Zhang et al., 2024), which lack the explicit object boundaries necessary to accurately interface with rigid-body dynamics simulators.

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.

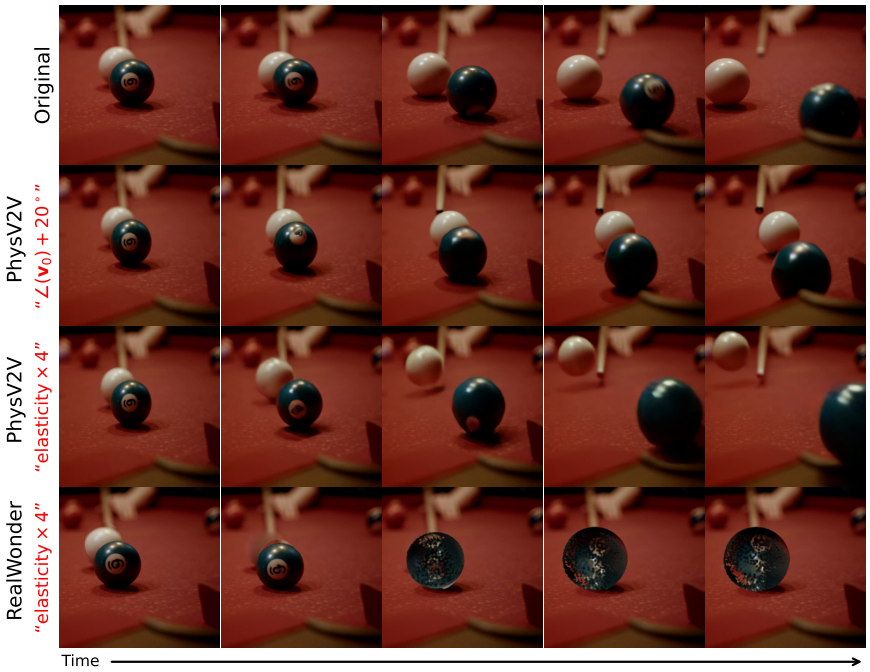


Figure 1: **Example results from PhysV2V and one I2V baseline (RealWonder (Liu et al., 2026)).** Given an input video (top row), PhysV2V allows users to apply physics edits to target objects, such as changing the initial velocity direction (second row), or increasing the elasticity of objects (causing the white ball to bounce in the third row). A recent I2V model RealWonder (fourth row) fails to model the scene.

Table 1: **Comparison of PhysV2V against related methods** (Yang et al., 2024; Team, 2025; Liu et al., 2024; 2026; Zhang et al., 2024) **across key capabilities.** PhysV2V is the only method that combines physics-in-the-loop editing with trajectory-matched parameter optimization directly from an input video. Unlike existing methods, PhysV2V enables precise editing of rigid-body parameters, including mass, elasticity, and initial velocity, while faithfully preserving the original scene dynamics.

	(ours)	V2V		I2V		3DGS
	PhysV2V	CogVideoX	Lucy Edit	PhysGen	RealWonder	PhysDreamer
General video editing	✗	✓	✓	✗	✗	✗
Physics-in-the-loop	✓	✗	✗	✓	✓	✓
Physics parameter editing	✓	✗	✗	✓	✓	✗
Real trajectory replicability	✓	✗	✗	✗	✗	✗
Photorealistic generative output	✓	✓	✓	✓	✓	✓

To address these limitations, we propose PhysV2V, a generative pipeline for object physics manipulation that couples a differentiable physics simulator with a conditional video diffusion model. PhysV2V operates in three stages. First, we estimate 3D scene geometry and camera parameters from the input video using SAM-3D (Chen et al., 2025b), extracting meshes for the objects and ground surface. Second, we formulate physical parameter estimation as an inverse problem: we optimize properties (e.g., initial velocities, mass, friction) so that the simulator's 2D-projected trajectories match point tracks from the source video. Although this inverse problem is ill-posed, our goal is to establish a physically grounded proxy sufficient for realistic editing rather than perfect parameter recovery. Finally, after applying user-specified edits in the 3D space, we render the updated motion as a 2D segmentation conditioning signal for a foundation video generator to synthesize the final output.

We evaluate PhysV2V against state-of-the-art V2V editing models on a set of 15 self-captured and web videos, as well as 250 simulated videos from the Kubric (Greff et al., 2022) engine. Results demonstrate significant improvements in physical realism, visual quality, and editing fidelity against various V2V and I2V baselines. We conclude by discussing limitations of this framework caused by inaccuracies in current 3D scene estimation and video generation foundation models.

2 Related Work

Physics estimation and differentiable simulation. Modeling physics from visual inputs is a long-standing computer vision challenge. Prior works estimate geometry (Gupta et al., 2010; Jia et al., 2014), mass, and friction (Denil et al., 2017; Wu et al., 2016), or predict dynamics in abstract spaces (Chang et al., 2017; Mottaghi et al., 2016b;a) and raw video (Finn et al., 2016; Wu et al., 2017). This naturally extends to differentiable simulation: DiffTaichi (Hu et al., 2019) propagates gradients through simulation steps for system identification, GradSim (Jatavallabhula et al., 2021) couples differentiable simulation and rendering to recover parameters and geometry, and PAC-NeRF (Li et al., 2023) identifies material properties inside a neural radiance field. While these methods focus on state prediction or parameter recovery, PhysV2V pairs explicit parameter estimation with video foundation models to render photorealistic, edited outputs.

Video generation and motion control. Current V2V models (Team, 2025; Wan et al., 2025; Yang et al., 2024) rely on an implicit understanding of physics and assume edits as text prompts. Recent benchmarks such as PhysWorldBench (Gu et al., 2026) and LikePhys (Yuan et al., 2026) confirm these models struggle with fundamental physical laws, necessitating explicit physics integration. To improve control, methods such as MotionCtrl (Wang et al., 2024) and DragNUWA (Yin et al., 2023) use user-specified trajectories, VideoComposer (Wang et al., 2023) uses mixed conditioning, and FLATTEN (Cong et al., 2023) recycles source optical flow. However, because these control signals are user-drawn or copied directly from the input rather than physically simulated, they cannot enforce consistent dynamics under parametric edits (e.g., halving mass). PhysV2V also uses structured control (segmentation masks), but derives it from an input-optimized simulator, providing an editable physical grounding absent in pure pixel-space flow.

Physics-integrated image-to-video (I2V) generation. I2V methods (Chen et al., 2025a; Liu et al., 2024; 2026; Wang et al., 2025a) couple simulators with diffusion models by using VLMs to coarsely estimate object properties from a single image. While sufficient for I2V, where any plausible motion is acceptable, this coarse estimation fails the strict scene-matching constraints required for V2V, as our experiments confirm. Similarly, works animating still images with plausible motion via learned spectral representations (Li et al., 2024) or large diffusion priors (Girdhar et al., 2024; Zhang et al., 2023) lack an editable physical parameter space, rendering precise relative edits (e.g., $+50\%$ elasticity) impossible. PhysV2V overcomes this by ensuring every parametric edit yields a physically consistent and visually faithful rendering of a full input video.

Neural rendering and physics simulation. Recent works integrate soft 3D Gaussian Splatting representations (Kerbl et al., 2023) with physics simulators for interactive scenes (Liu et al., 2025; Xie et al., 2024; Zhang et al., 2024). For example, PhysDreamer (Zhang et al., 2024) optimizes physical properties through a differentiable Material Point Method (MPM) simulator using video diffusion priors. However, MPM struggles with rigid collisions, excelling mostly with fluids and deformable solids. Conversely, rigid-body simulators (Freeman et al., 2021; Todorov et al., 2012) require explicit object boundaries (e.g., meshes) absent in standard Gaussian representations. PhysV2V directly addresses this problem by designing a per-scene property optimization framework that is compatible with both rigid-body and continuum dynamics.

3 Method

We assume a static-camera input video $\mathbf{V}^{\text{in}} \in \mathbb{R}^{H\times W\times T}$ depicting a scene with objects and a flat surface, driven by initial conditions and passive forces (e.g., gravity, friction). A user specifies edits modifying physical properties (e.g., mass, elasticity, or doubling initial velocity $|\mathbf{v}_0|$). The goal of PhysV2V is to generate a realistically edited video $\mathbf{V}^{\text{out}} \in \mathbb{R}^{H\times W\times T}$ that transforms the motion and interactions of each object according to the edits, while faithfully preserving the scene's visual appearance. The core challenge lies in translating the observed 2D pixel-space dynamics of $\mathbf{V}^{\text{in}}$ into a 3D representation that supports grounded physical editing.

PhysV2V operates in three stages (Fig. 2): (1) 3D Scene Reconstruction, (2) Physical Property Optimization, and (3) Simulation-Controlled Video Generation. First, we detect the ground plane and object geometries using a 3D foundation model to build a 3D simulation environment. Second, we bridge the observed 2D dynamics and 3D simulation by tracking 2D point trajectories from $\mathbf{V}^{\text{in}}$ and optimizing the simulator's physical parameters so the projected 3D trajectories match the observed

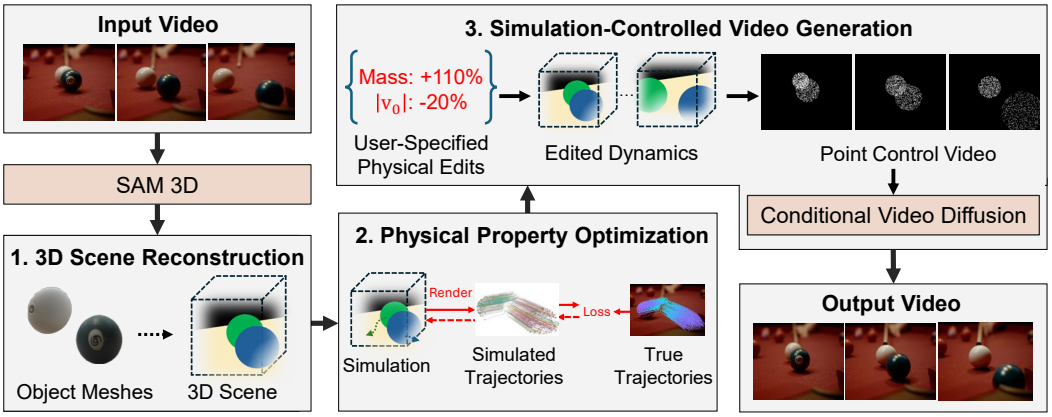


Figure 2: **Overview of PhysV2V.** Given an input video and target physical edits, our PhysV2V operates in three stages. First, we reconstruct 3D object meshes and a ground surface. Second, we use a differentiable simulator to optimize physical properties to match the observed motions in the video. Third, after applying user edits to the simulation space, we render a control video that conditions a diffusion model to synthesize the final output.

2D motion. Third, we apply user edits to the optimized simulation, render a 2D point control video $\mathbf{S}^{\text{out}}$, and synthesize the final video $\mathbf{V}^{\text{out}}$ using a conditional foundation video diffusion model. We now describe the stages in more detail.

3.1 3D Scene Reconstruction

To isolate the dynamic elements within $\mathbf{V}^{\text{in}}$, we first apply Grounded-SAM-2 (Ren et al., 2024) to the initial frame for zero-shot detection and segmentation of all objects and the ground surface. We propagate these masks throughout $\mathbf{V}^{\text{in}}$ using the SAM-2 video predictor. By thresholding total pixel displacements, we filter for objects that exhibit significant motion; these *dynamic* object instances (denoted D in total), serve as the subjects for physical property optimization in Stage 2.

To prepare the detected objects for simulation, we reconstruct their 3D geometries using the SAM-3D-Objects (Chen et al., 2025b) feed-forward model, which predicts a complete mesh for each mask. We then register these meshes to the camera reference frame via pose estimation and scale alignment. Next, we apply CoACD (Wei et al., 2022) convex decomposition to the meshes to improve the stability of simulator collision dynamics. We fit a plane to the ground surface using RANSAC.

3.2 Physical Property Optimization

Next, we leverage a differentiable physics simulator to estimate the physical properties of each dynamic object via trajectory matching.

Object Parameter Initialization. We first apply a Vision-Language Model (VLM) to the first frame of $\mathbf{V}^{\text{in}}$ to classify each detected object as either rigid or deformable and initialize its intrinsic physical properties. For rigid objects, we estimate mass m_i, elasticity e_i, and friction μ_i. For deformable objects, we estimate mass m_i, Young's modulus E_i, and friction μ_i.

For the D dynamic objects, we additionally parameterize their initial kinematic state via linear and angular velocities, $\mathbf{v}_{0,i} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_{0,i} \in \mathbb{R}^3$. While the VLM provides reasonable starting estimates, it empirically fails to accurately reproduce observed collision dynamics and deformations. Thus, we refine the parameters of all dynamic objects through optimization.

Ground-Truth Trajectory Computation. To obtain target trajectories, we uniformly sample N points inside the 3D mesh of each dynamic object i: $\{\mathbf{P}_{i,n}\}_{n=1}^{N} \subset \mathbb{R}^3$. Projecting these into the 2D image plane yields initial 3D-2D correspondences. We then track these 2D points across all frames using CoTracker (Karaev et al., 2024), producing ground-truth 2D trajectories $\mathbf{u}_{i,n}(t)$, with $t = [1, T]$, representing the observed motion.

Optimization via Trajectory Matching. Let $\hat{\mathbf{P}}_{i,n}(t)$ denote the simulated 3D position of the n-th sampled point on object i at time t. For rigid objects, the simulator maintains a 3D kinematic state consisting of center-of-mass $\mathbf{x}_i(t)$, linear velocity $\mathbf{v}_i(t)$, orientation quaternion $\mathbf{q}_i(t)$, and angular velocity $\boldsymbol{\omega}_i(t)$, together with physical parameters $\{m_i, e_i, \mu_i\}$. A differentiable rigid-body simulation step $\mathcal{S}_{\text{rigid}}$ advances this state by Δt:

$$\{\Delta\mathbf{x}_i, \Delta\mathbf{q}_i\} = \mathcal{S}_{\text{rigid}}\left(\{\mathbf{x}_i, \mathbf{v}_i, \mathbf{q}_i, \boldsymbol{\omega}_i, m_i, e_i, \mu_i\}\right). \tag{1}$$

For deformable objects, the points sampled from the reconstructed mesh for tracking are directly initialized as material points in a differentiable Material Point Method (MPM) simulator. Their positions and velocities are evolved according to the object's physical parameters $\{m_i, E_i, \mu_i\}$, where E_i denotes Young's modulus:

$$\{\hat{\mathbf{P}}_{i,n}(t+\Delta t)\}_{n=1}^{N} = \mathcal{S}_{\text{def}}\left(\{\hat{\mathbf{P}}_{i,n}(t)\}_{n=1}^{N}, m_i, E_i, \mu_i\right). \tag{2}$$

We defer details of both simulation formulations to the Supplementary.

For either object type, we apply the corresponding simulator for T steps to generate full 3D point trajectories. Projecting these points into the camera plane produces simulated 2D trajectories $\mathbf{u}_{i,n}(t)$. We optimize the physical parameters via gradient descent, constraining $m_i > 0$, $e_i \in [0, 1]$, and $E_i > 0$, by minimizing the mean absolute trajectory error across all objects, points, and time steps:

$$\mathcal{L}(\mathbf{u}, \hat{\mathbf{u}}) = \frac{1}{DNT} \sum_{i=1}^{D} \sum_{n=1}^{N} \sum_{t=1}^{T} \left\|\mathbf{u}_{i,n}(t) - \hat{\mathbf{u}}_{i,n}(t)\right\|_1. \tag{3}$$

3.3 Simulation-Controlled Video Generation

After finding the optimal parameters, we apply the user-specified edits to them and re-simulate the scene. We assume the edits are given in a form readily applicable to the parameters, e.g., initial velocity or elasticity deltas. We then project the edited 3D point trajectories from the simulator into the camera plane, producing a 2D point control video $\mathbf{C}^{\text{out}}$.

Finally, we synthesize $\mathbf{V}^{\text{out}}$ using a pretrained latent video diffusion model conditioned on $\mathbf{C}^{\text{out}}$ and the first frame of $\mathbf{V}^{\text{in}}$. We keep the base diffusion model frozen and train In-Context LoRA (IC-LoRA) adapters (Huang et al., 2024) to process the point-based control on a separate dataset of natural videos. To construct the training controls, for each video, we uniformly sample points within the target object's segmentation mask in the first frame and track these points throughout the video. The resulting 2D point trajectories form the point control video used to condition the diffusion model.

We train the IC-LoRA model using a masked flow matching objective (HaCohen et al., 2024) to reconstruct a target video from corrupted latents:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t,\mathbf{z}_0,\epsilon}\left[\frac{1}{|\mathcal{M}|}\sum_{k\in\mathcal{M}} \left\|v_\theta((1-\sigma_t)\mathbf{z}_0 + \sigma_t\epsilon, t, k) - (\epsilon - \mathbf{z}_0)\right\|^2\right], \tag{4}$$

where $\mathbf{z}_0$ are target latents, σ_t is the noise schedule, and v_θ is the velocity prediction field. The mask $\mathcal{M}$ indexes tokens to be predicted, excluding conditioning elements. At inference, this model translates the edited point control $\mathbf{C}^{\text{out}}$ into the photorealistic output $\mathbf{V}^{\text{out}}$.

4 Experiments and Results

4.1 Datasets

Real: For qualitative evaluation, we collected 15 real-world videos (sourced online or captured via smartphone) depicting object collision dynamics of both rigid and elastic deformable objects. To ensure motion is driven strictly by initial velocities, gravity, and friction, all videos feature a flat surface and a static camera. **Synthetic:** For quantitative evaluation, we generated simulated videos using Kubric (Greff et al., 2022) and the PyBullet (Coumans & Bai, 2016) physics simulator. From a base reference video, we generated 50 videos per physical edit type using uniformly spaced parameter variations: velocity magnitude (50% to 200% of reference), velocity direction (-30° to 30° offset), elasticity (50% to 150%), mass (50% to 150%), and friction (50% to 150%).

Additionally, we trained IC-LoRA adapters for conditional video diffusion (see Sec. 3.3) on 100 videos from the DAVIS-2017 (Pont-Tuset et al., 2017) dataset, which provides high-quality masks for diverse real-world scenes. We use the CoTracker model to track points and generate the control videos for training.

4.2 Implementation Details

3D Scene Reconstruction: We reconstructed object meshes via SAM-3D-Objects (Chen et al., 2025b) and registered to the camera frame using MoGE-2 (Wang et al., 2025b) intrinsics alongside SAM-3D alignments. **Physical Simulation:** For rigid body dynamics, we used Brax (Freeman et al., 2021), a differentiable JAX-based rigid-body simulator. For deformable objects, we use the differentiable MPM simulator from PAC-NeRF (Li et al., 2023). We initialized physical parameters via GPT-4V (Achiam et al., 2023) and optimized them for 1,000 iterations using Adam (lr = 10^{-4}), with v_0 fixed for the latter 500 iterations. **Point Tracking:** We used CoTracker (Karaev et al., 2024) to track $N = 100$ points uniformly sampled within each dynamic object's 3D mesh and projected into 2D. To mitigate unreliable boundary trajectories, we discarded points within 5% of the 2D mask edge. **Video Diffusion:** We used LTX-Video (HaCohen et al., 2024) as our frozen backbone. To enable point video conditioning, we trained rank-128 IC-LoRA adapters for 2,000 iterations using AdamW (lr = 2×10^{-4}).

4.3 Baselines and Quantitative Evaluation

V2V Methods: We compared against state-of-the-art text-based V2V models CogVideoX (Yang et al., 2024) and Lucy Edit Dev (Team, 2025), using ChatGPT-4o to translate target physical edits (e.g., doubling initial velocity) into the required natural language prompts. **I2V Methods:** We evaluated physics-based I2V models PhysGen (Liu et al., 2024) and RealWonder (Liu et al., 2026), giving the reference video's first frame as input. Because these VLM-based methods cannot estimate initial velocities, we ensured a fair comparison by providing them with PhysV2V's optimized velocity estimates for qualitative assessments, and ground-truth velocities for quantitative assessments. **Quantitative Evaluation:** We assessed editing fidelity using a synthetic two-ball collision simulated in Kubric. For each method and edit type, we generated outputs matching the 50 ground-truth parameter variations and computed average PSNR, SSIM, and LPIPS scores.

4.4 Results

Fig. 3 demonstrates PhysV2V successfully performing physical edits on real-world videos.

Quantitative Results. Table 2 summarizes quantitative performance on the Kubric dataset, where PhysV2V achieves the best scores across all edit types (velocity magnitude/direction, elasticity, friction, mass). Text-based V2V baselines frequently fail to apply the edits, yielding similar outputs regardless of the prompt. I2V baselines, even when provided with ground-truth initial velocities, underperform due to the aforementioned geometry and material estimation errors.

Qualitative Results. Fig. 4 presents a side-by-side comparison with V2V and I2V baselines. For a fair evaluation, I2V methods receive the optimized initial velocities computed by PhysV2V alongside the first video frame. As seen in the bottom two rows of Fig. 4, the V2V methods (CogVideoX and Lucy Edit Dev) fail to implement the requested physics edits, instead simply reproducing the original video and ignoring the text prompts. Meanwhile, the I2V methods (PhysGen and RealWonder) exhibit rendering artifacts and motion inconsistencies caused by inaccurate geometry and material property estimation.

To isolate the impact of geometry estimation and physical property optimization, we conducted two ablation studies (Table 3). For geometry, we compared PhysV2V against an oracle variant that uses ground-truth scene geometry (object poses, sizes, and ground plane). The oracle achieves substantially higher scores across all edit types, demonstrating that 3D reconstruction errors currently bottleneck PhysV2V's performance.

For optimization, we compared standard PhysV2V against two variants: one relying entirely on VLM-based estimates (with initial velocities $v_0 = 0$), and one using VLM estimates combined with known ground-truth initial velocities. PhysV2V outperforms both ablated versions, showing that even



Figure 3: **Qualitative Results.** For each case, the top row shows the input video, while subsequent rows demonstrate physical edits: angular velocity of the brown box (first case) and Young's Modulus of the dough ball (second case). These results show that PhysV2V successfully synthesizes high-quality visualizations of physical modifications applied directly to the input video. Please see Supplementary for full videos.

with known initial velocities, unoptimized intrinsic parameters substantially degrade performance. Qualitative results of this ablation on real-world videos with negligible initial motion (Fig. 5) confirm this: while PhysV2V faithfully reproduces the input dynamics, the VLM-only estimates fail entirely. We further analyze the optimization by tracking the evolution of physical property estimates (Fig. 7) and evaluating sensitivity to their initialization (Table 5).

5 Discussion and Conclusion

Results demonstrate that PhysV2V successfully synthesizes realistic physical edits from reference videos while maintaining fine-grained parameter control. In contrast, text-based V2V baselines (CogVideoX (Yang et al., 2024), Lucy Edit Dev (Team, 2025)) largely ignore requested physics edits. Notably, Lucy Edit Dev achieves competitive perceptual scores (Table 2) simply by reproducing the unedited reference video. Meanwhile, physics-based I2V baselines (PhysGen (Liu et al., 2024), RealWonder (Liu et al., 2026)) lack the temporal context required to preserve original dynamics. They suffer from inaccurate geometry estimation, rendering artifacts, and VLM-based material property errors. Specifically, errors in VLM-estimated elasticity prove especially detrimental; because elasticity dictates collision responses, its miscalculation drastically alters post-collision trajectories. Through ablation studies, we confirmed that our proposed per-video property optimization is critical to

overcoming these issues. Relying on VLM estimates alone, even with known initial velocities, fails to reproduce basic scene dynamics.

Limitations. While PhysV2V demonstrates high editing fidelity, we identify three primary limitations. **First,** our geometry ablation indicates that replacing the 3D scene reconstruction module with ground-

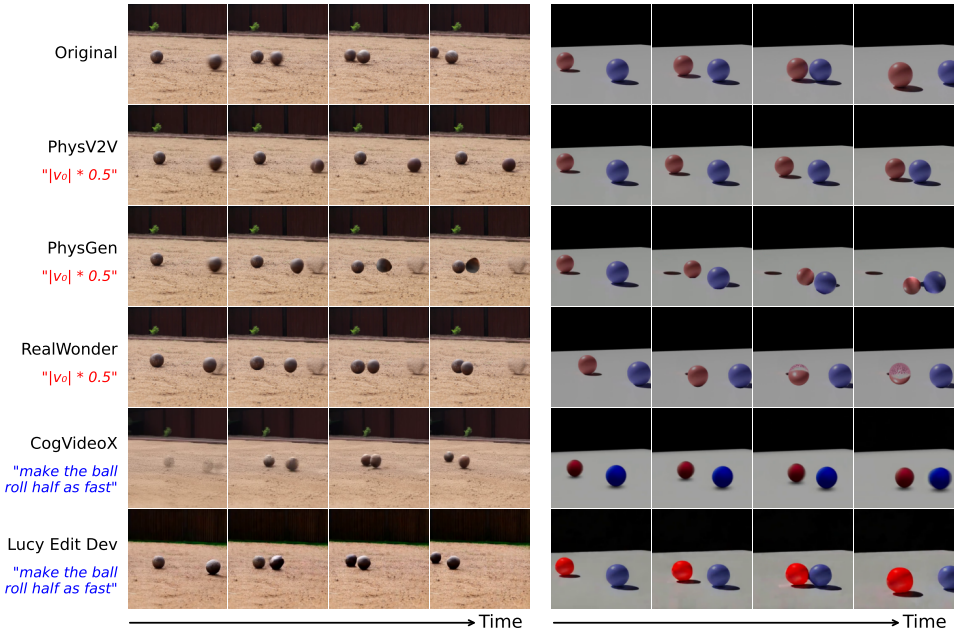


Figure 4: **Qualitative comparison against baselines.** Given an input video (top row), we compare the editing results of PhysV2V (second row) with physics-based I2V methods PhysGen (Liu et al., 2024) (third row) and RealWonder (Liu et al., 2026) (fourth row), alongside text-based V2V methods CogVideoX (Yang et al., 2024) (fifth row) and Lucy Edit Dev (Team, 2025) (sixth row). Please see Supplementary for full videos.

Table 2: **Quantitative evaluation of all methods on the Kubric synthetic dataset.** For each physical edit type (velocity magnitude, velocity direction, elasticity, friction, mass), we report PSNR (dB), SSIM, and LPIPS averaged across 50 generated videos per edit. PhysV2V consistently achieves the best metric scores.

Method	Metric	$\lvert v_0 \rvert$	$\angle v_0$	e	μ	m
PhysV2V	PSNR	**27.13**	**25.78**	**25.81**	**26.61**	**26.44**
	SSIM	**0.95**	**0.93**	**0.94**	**0.94**	**0.94**
	LPIPS	**0.11**	0.08	0.09	0.09	0.08
V2V methods						
CogVideoX (Yang et al., 2024)	PSNR	20.43	20.41	20.18	20.93	20.21
	SSIM	0.89	0.89	0.89	0.89	0.89
	LPIPS	0.13	0.15	0.13	0.14	0.12
Lucy Edit Dev (Team, 2025)	PSNR	21.49	21.31	21.12	21.15	21.66
	SSIM	0.91	0.90	0.90	0.92	0.91
	LPIPS	0.13	**0.08**	**0.08**	**0.08**	**0.08**
I2V methods (with v_0 given)						
PhysGen (Liu et al., 2024)	PSNR	16.63	16.71	16.64	16.61	16.84
	SSIM	0.71	0.71	0.71	0.71	0.71
	LPIPS	0.22	0.24	0.24	0.23	0.24
RealWonder (Liu et al., 2026)	PSNR	20.52	20.03	18.18	20.11	17.38
	SSIM	0.90	0.89	0.87	0.89	0.86
	LPIPS	0.16	0.15	0.18	0.19	0.21

Table 3: **Ablation results on the Kubric synthetic dataset.** We ablate SAM-3D scene reconstruction (Stage 1) by comparing PhysV2V (first row) with an oracle variant using ground-truth scene geometry (second row). We ablate physical property estimation (Stage 2) with two variants: one using only VLM-estimated material properties with zero initial velocity (third row), and one using VLM-estimated properties but with ground-truth initial velocity v_0 provided (fourth row).

Method	$\lvert v_0 \rvert$	$\angle v_0$	e	μ	m
PhysV2V	26.1/0.9/0.1	27.7/0.9/0.1	25.8/0.9/0.1	26.6/0.9/0.2	26.4/0.9/0.1
w/ GT geom.	**30.3/0.9/0.1**	**30.1/0.9/0.1**	**29.7/0.9/0.1**	**29.7/0.9/0.2**	**29.8/0.9/0.1**
w/o optim.	20.5/0.8/0.1	20.1/0.8/0.1	21.4/0.8/0.1	20.9/0.8/0.2	21.0/0.8/0.1
w/o optim., w/ GT v_0	22.8/0.8/0.1	23.1/0.8/0.1	23.2/0.8/0.1	21.9/0.8/0.2	22.1/0.8/0.1

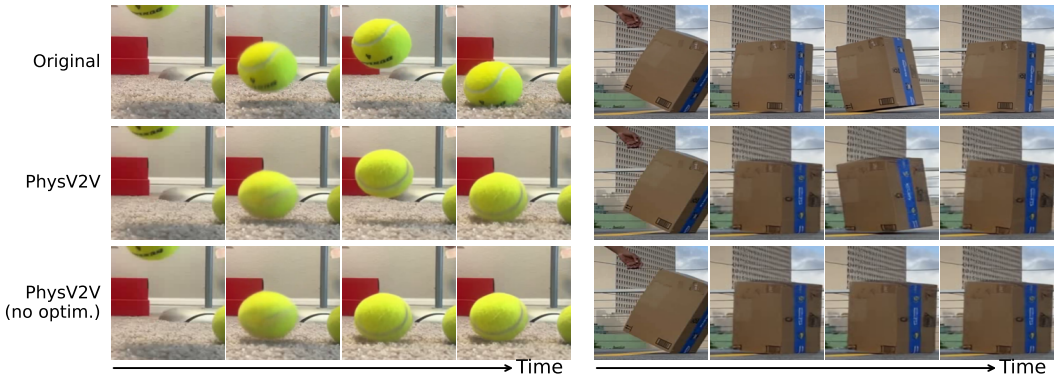


Figure 5: **Property optimization ablation.** For a given input video (top row), PhysV2V optimizes physical properties based on observed motion, producing a zero-edit reconstruction (second row). Bypassing this optimization and relying solely on VLM-estimated properties (third row) fails to reproduce the original dynamics, such as the bounce of the tennis ball or the wobble of the box.

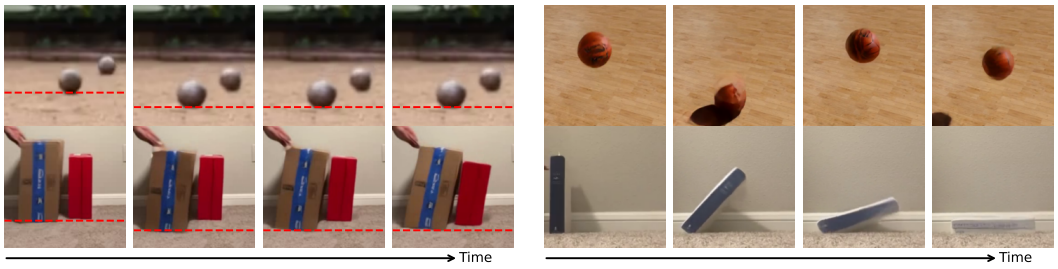


Figure 6: **Failure Cases.** Left: Objects sometimes "fall" onto the simulated surface (visualized as a red dashed line) in early frames due to surface geometry estimation errors. Right: Rendering artifacts produced by the underlying video diffusion model, including flickering textures on a bouncing basketball and color shifts on a falling book.

truth geometry yields substantial performance improvements. We attribute current reconstruction inaccuracies to SAM-3D's per-object alignment outputs, which can lack global scene consistency. These spatial misalignments cause trajectory mismatches that downstream simulation cannot recover, leading to failure cases like objects unnaturally settling onto poorly estimated surface planes (Fig. 6, left). Integrating stronger single-image-to-3D scene reconstruction models would directly improve PhysV2V's fidelity. **Second,** PhysV2V occasionally inherits rendering artifacts from the underlying video diffusion model (Fig. 6, right). The diffusion model's training distribution fundamentally constrains the framework's overall generalization capabilities and visual consistency. **Third,** unlike purely data-driven V2V models, our explicit physical formulation is currently restricted to a specific domain: static-camera videos of objects interacting on a flat surface. Extending this explicit scene modeling to accommodate dynamic cameras and broader causal changes represents a critical next step to expand PhysV2V's scope to casual, in-the-wild videos.

AI USE STATEMENT

In this work, we used generative AI tools for *implementing methods (code assist) and reformatting dataset*. We have not used generative AI tools for *generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, support qualitative and thematic data analysis, interpret results* are not applicable to this work. Additionally, we used generative AI tools for *create or modify scientific figures or images, create or edit software code, edit a research paper to improve readability*. We have reviewed all AI-assisted work, by doing a thorough code review of LLM-generated code, and checking the text for any LLM-caused hallucinations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCABILITY STATEMENT

In the Supplement, we include details about the physical simulator implementation in Subsec. A.3 and A.4, which are both open-source implementations. We include details about the training of the conditional video diffusion model, and details about the physical property optimization in Sec. 4. We also provide an anonymized link to the code in the Supplement, and will make our synthetic dataset available.

5.1 ETHICS STATEMENT

Societal Impacts. Our work has the potential to positively impact content creation, visual effects, and robotic simulation by enabling precise, physics-based video editing that preserves realistic dynamics. This capability could lower barriers for creators and researchers who need to manipulate physical behavior in videos without expensive re-shooting or complex manual animation. However, like any generative video editing technology, PhysV2V could also be misused to create misleading or fake media content, such as altering the dynamics of real-world events to deceive viewers. We encourage responsible use of this technology and advocate for the development of detection methods to identify synthetically edited videos.

Model Safeguards. Our method builds upon pretrained foundation models (SAM-3D, CoTracker, LTX-Video) and a differentiable physics simulator, rather than releasing a new pretrained model from scratch. The primary component we release is the optimization and conditioning pipeline, which does not independently generate misleading content without access to a video diffusion backbone. Nevertheless, to support responsible use, we will release our code under a research-only license with clear usage guidelines, and we encourage practitioners to apply safety filters on the underlying diffusion model to detect and block potentially harmful video generations.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Michael Chang, Tomer Ullman, Antonio Torralba, and Joshua Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations*, 2017.

Boyuan Chen, Hanxiao Jiang, Shaowei Liu, Saurabh Gupta, Yunzhu Li, Hao Zhao, and Shenlong Wang. Physgen3d: Crafting a miniature interactive world from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6178–6189, 2025a.

Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025b.

Yuren Cong, Mengmeng Xu, Christian Simon, Shoufa Chen, Jiawei Ren, Yanping Xie, Juan-Manuel Perez-Rua, Bodo Rosenhahn, Tao Xiang, and Sen He. Flatten: optical flow-guided attention for consistent text-to-video editing. *arXiv preprint arXiv:2310.05922*, 2023.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.

Misha Denil, Pulkit Agrawal, Tejas D Kulkarni, Tom Erez, Peter Battaglia, and Nando de Freitas. Learning to perform physics experiments via deep reinforcement learning. In *International Conference on Learning Representations*, 2017.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Factorizing text-to-video generation by explicit image conditioning. In *European Conference on Computer Vision*, pp. 205–224. Springer, 2024.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

Jing Gu, Xian Liu, Yu Zeng, Ashwin Nagarajan, Fangrui Zhu, Daniel Hong, Yue Fan, Qianqi Yan, Kaiwen Zhou, Ming-Yu Liu, and Xin Eric Wang. $phyworldbench$: A comprehensive evaluation of physical realism in text-to-video models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Abhinav Gupta, Alexei A Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*, pp. 482–496. Springer, 2010.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

Lianghua Huang, Wei Wang, Zhi-Fan Wu, Yupeng Shi, Huanzhang Dou, Chen Liang, Yutong Feng, Yu Liu, and Jingren Zhou. In-context lora for diffusion transformers. *arXiv preprint arXiv:2410.23775*, 2024.

Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=c_E8kFWfhp0.

Zhaoyin Jia, Andrew C Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d reasoning from blocks to stability. *IEEE transactions on pattern analysis and machine intelligence*, 37(5):905–918, 2014.

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

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *European conference on computer vision*, pp. 18–35. Springer, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *International Conference on Learning Representations (ICLR)*, 2023.

Zhengqi Li, Richard Tucker, Noah Snavely, and Aleksander Holynski. Generative image dynamics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24142–24153, 2024.

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.

Wei Liu, Ziyu Chen, Zizhang Li, Yue Wang, Hong-Xing Yu, and Jiajun Wu. Realwonder: Real-time physical action-conditioned video generation. *arXiv preprint arXiv:2603.05449*, 2026.

Zhuoman Liu, Weicai Ye, Yan Luximon, Pengfei Wan, and Di Zhang. Unleashing the potential of multi-modal foundation models and video diffusion for 4d dynamic physical scene simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11016–11025, 2025.

Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3521–3529, 2016a.

Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. "what happens if..." learning to predict the effect of forces in images. In *European conference on computer vision*, pp. 269–285. Springer, 2016b.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

DecartAI Team. Lucy edit: Open-weight text-guided video editing, 2025.

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.

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.

Chen Wang, Chuhao Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. *arXiv preprint arXiv:2509.20358*, 2025a.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025b.

Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *Advances in Neural Information Processing Systems*, 36:7594–7611, 2023.

Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pp. 1–11, 2024.

Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41 (4):1–18, 2022.

Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, pp. 7, 2016.

Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. *Advances in neural information processing systems*, 30, 2017.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4389–4398, 2024.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory. *arXiv preprint arXiv:2308.08089*, 2023.

Jianhao Yuan, Fabio Pizzati, Francesco Pinto, Lars Kunze, Ivan Laptev, Paul Newman, Philip Torr, and Daniele De Martini. Likephys: Evaluating intuitive physics understanding in video diffusion models via likelihood preference. In *The Fourteenth International Conference on Learning Representations*, 2026.

Shiwei Zhang, Jiayu Wang, Yingya Zhang, Kang Zhao, Hangjie Yuan, Zhiwu Qin, Xiang Wang, Deli Zhao, and Jingren Zhou. I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models. *arXiv preprint arXiv:2311.04145*, 2023.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pp. 388–406. Springer, 2024.

A SUPPLEMENTARY MATERIAL

All experiments were done on a single NVIDIA H100 GPU with 80GB VRAM. Our code is available at: `https://anonymous.4open.science/r/PhysV2V-7FAF/`. Full output videos are available in the file attachment.

A.1 ABLATION ON POINT SAMPLING

Table 4: **Ablation of point sampling on the Kubric synthetic dataset.** We study the effect of the number of sampled point trajectories used per-object in the physical property optimization of PhysV2V. Performance (reported as PSNR/SSIM/LPIPS) degrades when using fewer than 20 points, but remains relatively stable beyond this threshold.

Points	$\lvert v_0 \rvert$	$\angle v_0$	e	μ	m
5	21.6/0.9/0.1	21.4/0.9/0.1	21.0/0.9/0.1	21.8/0.9/0.2	21.6/0.9/0.1
10	24.5/0.9/0.1	26.0/0.9/0.1	23.8/0.9/0.1	24.4/0.9/0.2	23.9/0.9/0.1
20	25.8/0.9/0.1	**27.8/0.9/0.1**	25.6/0.9/0.1	25.8/0.9/0.2	24.9/0.9/0.1
50	25.9/0.9/0.1	27.3/0.9/0.1	**26.1/0.9/0.1**	26.4/0.9/0.2	26.3/0.9/0.1
100	**26.1/0.9/0.1**	27.7/0.9/0.1	25.8/0.9/0.1	**26.6/0.9/0.2**	**26.4/0.9/0.1**

We perform an ablation to study the effect of the number of sampled points per object in the physical property optimization (Stage 2). By default, PhysV2V samples $N = 100$ points within each object mesh, projects these points into 2D, and tracks them using CoTracker (Karaev et al., 2024). As shown in Table 4, using fewer than 20 points per object causes noticeable degradation in performance across all edit types. Beyond 20 points, performance remains relatively stable. We attribute this behavior to noise and errors in the tracked point trajectories, which arise from occlusions and fast motion in the input video. Using a larger number of trajectories dilutes the influence of outlier error trajectories, leading to more robust optimization. The optimal number of sampled points likely depends on object geometry, occlusions, and motion complexity of each scene.

A.2 Analysis of Physical Property Optimization

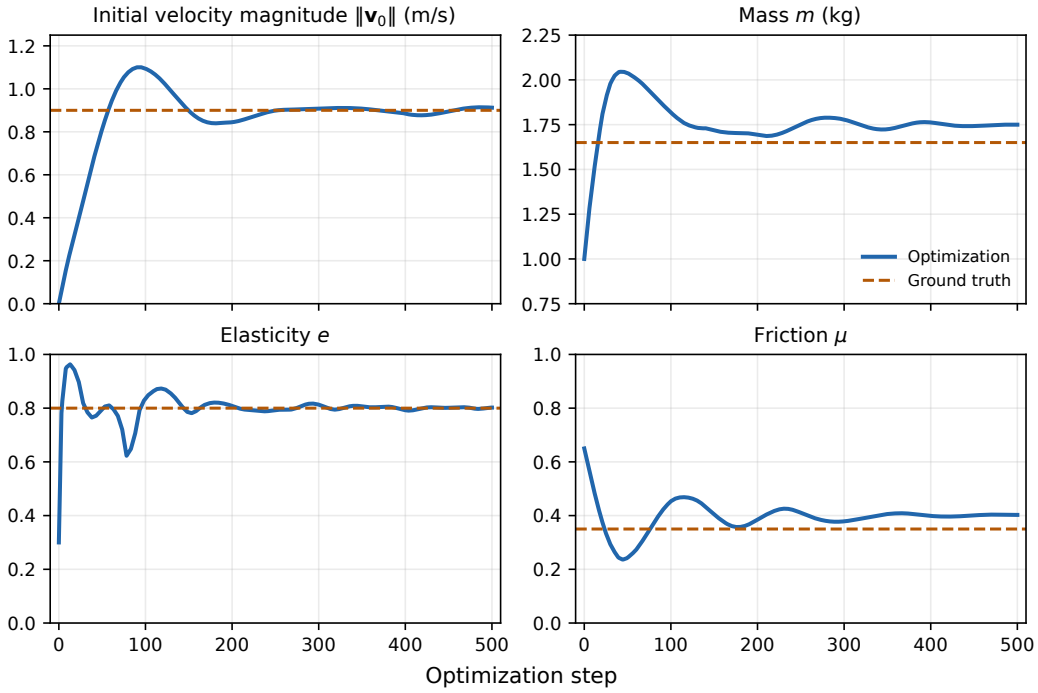


Figure 7: **Property values vs. optimization step.** We show the evolution of physical property values during the trajectory optimization for the synthetic Kubric ball collision. For mass, elasticity, and friction, the step-0 value is the VLM's initialization. We observe that despite errors in the initial estimate, the properties approach the ground-truth values as optimization progresses.

To show how VLM physical-property estimates change during optimization, Fig. 7 reports an optimization trajectory on the synthetic Kubric two-ball collision example, together with the corresponding ground-truth properties. Initial velocity and elasticity change substantially from their VLM initializations and converge near their ground-truth values, demonstrating that their final estimates are primarily determined by trajectory matching rather than the initial VLM prediction. Mass and friction remain closer to their initializations because they are less strongly identifiable from the image-space trajectories in the collision scenarios considered.

We additionally test sensitivity to initialization by initializing all properties from several scaled versions of their ground-truth values. We report the final relative error for each property, defined as the absolute prediction error divided by its ground-truth value. The results are shown in Table 5.

Table 5: **Relative property estimation error vs. initialization.**

Initialization	$\|v_0\|$	m	e	μ
$0.5\times$ GT	0.05	0.23	0.10	0.17
$0.75\times$ GT	0.04	0.15	0.08	0.14
$1.25\times$ GT	0.05	0.18	0.08	0.18
$1.5\times$ GT	0.05	0.29	0.11	0.21

Velocity and elasticity converge consistently across different initializations, indicating stable recovery for the parameters that most strongly affect the observed motion. Mass is more initialization-dependent, consistent with its weaker identifiability. The VLM's initial mass estimate provides a useful prior that prevents convergence to implausible values.

A.3 Rigid Body Dynamics Simulation

We model the motion of objects using rigid body dynamics, where each object is characterized by a set of physical properties: mass m, friction coefficient μ, restitution (elasticity) e, and initial conditions including linear velocity $\mathbf{v}_0$ and angular velocity $\boldsymbol{\omega}_0$. The state of an object at time t

is represented by the vector $\mathbf{s}(t) = [\mathbf{x}(t), \mathbf{q}(t), \mathbf{v}(t), \boldsymbol{\omega}(t)]^\top$, where $\mathbf{x} \in \mathbb{R}^3$ is the center of mass position, $\mathbf{q}$ is the orientation quaternion, $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity, and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity. All positions, velocities, and forces are modeled in full 3D space. The dynamics are governed by Newton's second law for linear motion and Euler's equation for rotational motion. We express the time derivative of the state as:

$$\frac{d}{dt}\mathbf{s}(t) = \frac{d}{dt}\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{q}(t)\\ \mathbf{v}(t)\\ \boldsymbol{\omega}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{v}(t)\\ \frac{1}{2}\boldsymbol{\omega}(t) \otimes \mathbf{q}(t)\\ \frac{1}{m}\mathbf{F}_{\text{total}}(t;\mu)\\ \mathbf{I}(t)^{-1}\left(\boldsymbol{\tau}_{\text{total}}(t;\mu) - \boldsymbol{\omega}(t) \times \mathbf{I}(t)\boldsymbol{\omega}(t)\right)\end{bmatrix},$$

where $\otimes$ denotes quaternion multiplication and $\mathbf{I}(t) \in \mathbb{R}^{3\times3}$ is the inertia tensor in the world frame. The total force $\mathbf{F}_{\text{total}} \in \mathbb{R}^3$ and torque $\boldsymbol{\tau}_{\text{total}} \in \mathbb{R}^3$ depend on the friction coefficient μ through Coulomb's friction model.

Given a time step Δt, we integrate this system using a semi-implicit Euler scheme. The velocity update from time t to $t + \Delta t$ is:

$$\begin{bmatrix}\mathbf{v}(t+\Delta t)\\ \boldsymbol{\omega}(t+\Delta t)\end{bmatrix} = \begin{bmatrix}\mathbf{v}(t)\\ \boldsymbol{\omega}(t)\end{bmatrix} + \Delta t \begin{bmatrix}\frac{1}{m}\mathbf{F}_{\text{total}}(t;\mu)\\ \mathbf{I}(t)^{-1}\left(\boldsymbol{\tau}_{\text{total}}(t;\mu) - \boldsymbol{\omega}(t) \times \mathbf{I}(t)\boldsymbol{\omega}(t)\right)\end{bmatrix}.$$

The position and orientation are then updated using the new velocities:

$$\begin{bmatrix}\mathbf{x}(t+\Delta t)\\ \mathbf{q}(t+\Delta t)\end{bmatrix} = \begin{bmatrix}\mathbf{x}(t)\\ \mathbf{q}(t)\end{bmatrix} + \Delta t \begin{bmatrix}\mathbf{v}(t+\Delta t)\\ \frac{1}{2}\boldsymbol{\omega}(t+\Delta t) \otimes \mathbf{q}(t)\end{bmatrix},$$

followed by renormalizing the quaternion $\mathbf{q}(t + \Delta t)$ to maintain unit length.

To compute the contact contributions to $\mathbf{F}_{\text{total}}$ and $\boldsymbol{\tau}_{\text{total}}$, we detect collisions between objects represented by their convex hulls. For each contact between two bodies, the effective restitution is computed as the average of their individual elasticity parameters: $e = (e_1 + e_2)/2$. This value determines post-collision velocities, with $e = 0$ giving perfectly inelastic collisions (no bounce) and $e = 1$ giving perfectly elastic collisions. For each contact, we compute normal and friction forces based on the penetration depth and relative motion at the contact point. These contact forces are determined by solving a convex optimization problem that enforces physical constraints such as non-penetration. The resulting forces are then projected into generalized coordinates using a constraint Jacobian J, contributing to the total force $\mathbf{F}_{\text{total}}$ and torque $\boldsymbol{\tau}_{\text{total}}$ in the dynamics equations before updating the object state through the integration steps above.

Overall, this formulation simulates rigid-body collisions and interactions of objects, computing per-object simulation states $\{\mathbf{s}_i\}_{i=1}^N$ for N simulation steps. Importantly, the simulation is fully differentiable, enabling gradient-based optimization of physical properties $\theta = \{m, \mu, e, \mathbf{v}_0, \boldsymbol{\omega}_0\}$.

A.4 Deformable Body Dynamics Simulation

We model deformable objects using the differentiable Material Point Method (MPM) implementation from PAC-NeRF (Li et al., 2023). We consider only elastic materials, modeled using a Neo-Hookean constitutive model. Each deformable object is represented by a collection of Lagrangian material points. For each material point p, the simulator maintains its position $\mathbf{x}_p(t) \in \mathbb{R}^3$, velocity $\mathbf{v}_p(t) \in \mathbb{R}^3$, affine velocity field $\mathbf{C}_p(t) \in \mathbb{R}^{3\times3}$, and deformation gradient $\mathbf{F}_p(t) \in \mathbb{R}^{3\times3}$. The deformation gradient measures the local deformation of the material relative to its initial configuration, with $\mathbf{F}_p(0) = \mathbf{I}$.

The object's elastic behavior is parameterized by Young's modulus E, which controls its material stiffness. Given a fixed Poisson's ratio ν, we convert E to the Lamé parameters

$$\mu_{\text{L}} = \frac{E}{2(1+\nu)}, \qquad \lambda_{\text{L}} = \frac{\nu E}{(1+\nu)(1-2\nu)}.$$

For a material point with deformation gradient $\mathbf{F}_p$, let $J_p = \det(\mathbf{F}_p)$. Under the Neo-Hookean model, the Cauchy stress $\mathbf{T}_p$ is given by

$$J_p \mathbf{T}_p = \mu_{\mathrm{L}} \mathbf{F}_p \mathbf{F}_p^{\top} + (\lambda_{\mathrm{L}} \log J_p - \mu_{\mathrm{L}})\, \mathbf{I}.$$

This stress determines the internal elastic forces used by MPM to evolve the object according to the continuum momentum equation

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \mathbf{T} + \rho \mathbf{g},$$

where ρ is the material density and $\mathbf{g}$ is gravity.

Given a simulation time step Δt, each MPM substep alternates between the material points and a temporary Eulerian background grid. We first update the deformation gradient of each material point using its local affine velocity field:

$$\mathbf{F}_p(t + \Delta t) = (\mathbf{I} + \Delta t\, \mathbf{C}_p(t))\, \mathbf{F}_p(t).$$

The corresponding Neo-Hookean stress is then computed and, together with particle mass and momentum, transferred from the material points to nearby grid nodes using MPM interpolation weights. On the grid, the transferred momentum is converted to velocity and updated under gravity, internal elastic forces, and collision boundary conditions. The updated grid velocities are then interpolated back to the material points:

$$\mathbf{v}_p(t + \Delta t) = \sum_i w_{ip}\, \mathbf{v}_i(t + \Delta t),$$

where w_{ip} denotes the interpolation weight between material point p and grid node i. The affine velocity field $\mathbf{C}_p(t + \Delta t)$ is similarly reconstructed from the local grid velocity field, and the material point position is advected using the updated velocity:

$$\mathbf{x}_p(t + \Delta t) = \mathbf{x}_p(t) + \Delta t\, \mathbf{v}_p(t + \Delta t).$$

The background grid is cleared after each substep, while the material-point states $\{\mathbf{x}_p, \mathbf{v}_p, \mathbf{C}_p, \mathbf{F}_p\}$ persist throughout the simulation.

For our trajectory-matching formulation, the 3D points sampled from the reconstructed object geometry are directly initialized as MPM material points. Their simulated positions therefore provide the 3D trajectories that are projected into the camera view and compared with the corresponding observed 2D point tracks. As in the rigid-body case, the simulation is fully differentiable, allowing gradients of the trajectory loss to be propagated through all MPM time steps to optimize the material parameters.

A.5 Qualitative Results on Synthetic Dataset

We present qualitative results on our synthetic Kubric dataset in Fig. 8, focusing on edits to the initial velocity direction of colliding objects. PhysV2V successfully simulates realistic collision dynamics across different edit settings, closely matching the ground-truth edited videos. We observe that the fidelity of PhysV2V to the ground-truth edits decreases toward the end of videos, as small differences in collision timing and response between the simulated and ground-truth trajectories magnify over time.

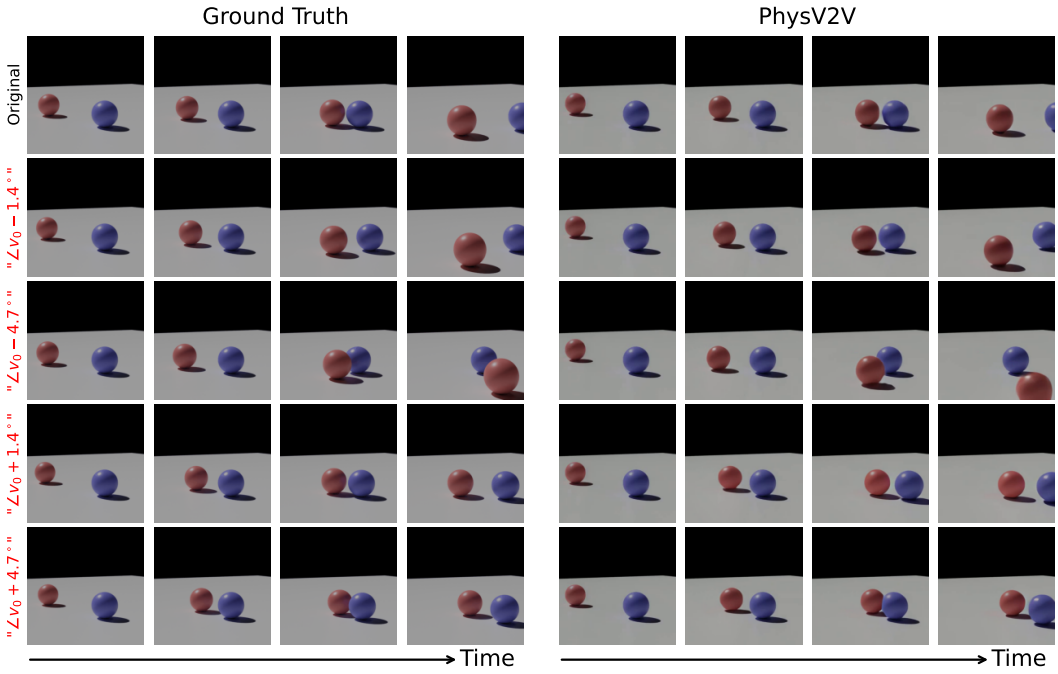


Figure 8: **Qualitative Results on Kubric Dataset.** We compare PhysV2V's editing results (right) against ground-truth edited videos (left) on our synthetic dataset, in velocity direction edits. The first row shows the original input video alongside the PhysV2V "no-edit" output, demonstrating that PhysV2V faithfully reproduces the original motion. PhysV2V generally matches the ground-truth edited dynamics.

A.6 Qualitative Results on Deformable Objects

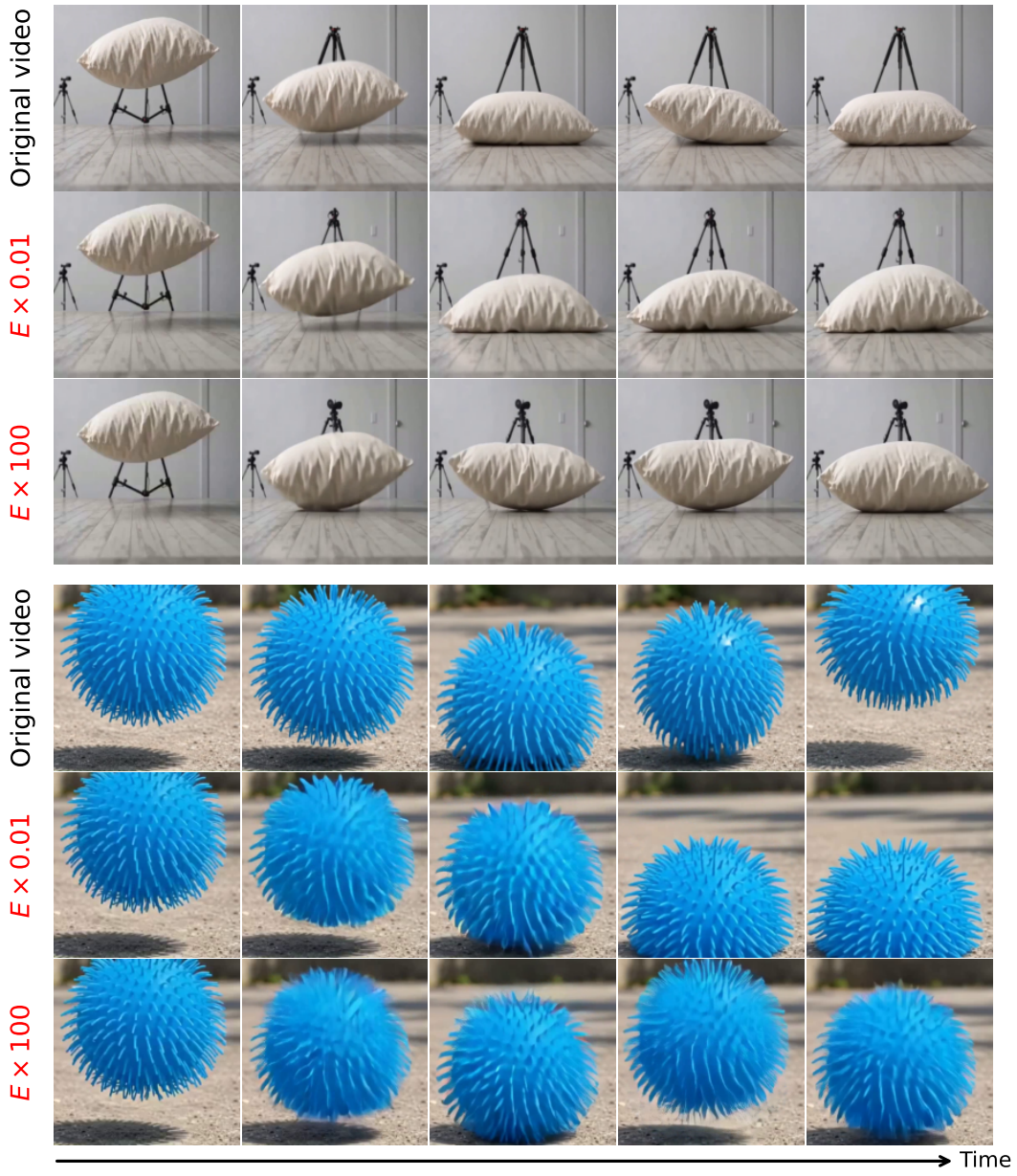


Figure 9: **Young's modulus edits on deformable objects.** For each example, the top row shows the original video, and the following rows show PhysV2V outputs after decreasing Young's modulus to $E \times 0.01$ and increasing it to $E \times 100$, respectively. A lower Young's modulus produces greater deformation and expansion on impact, whereas a higher Young's modulus better preserves object shape and produces more rigid bouncing in the output video.

A.7 Additional Qualitative Results

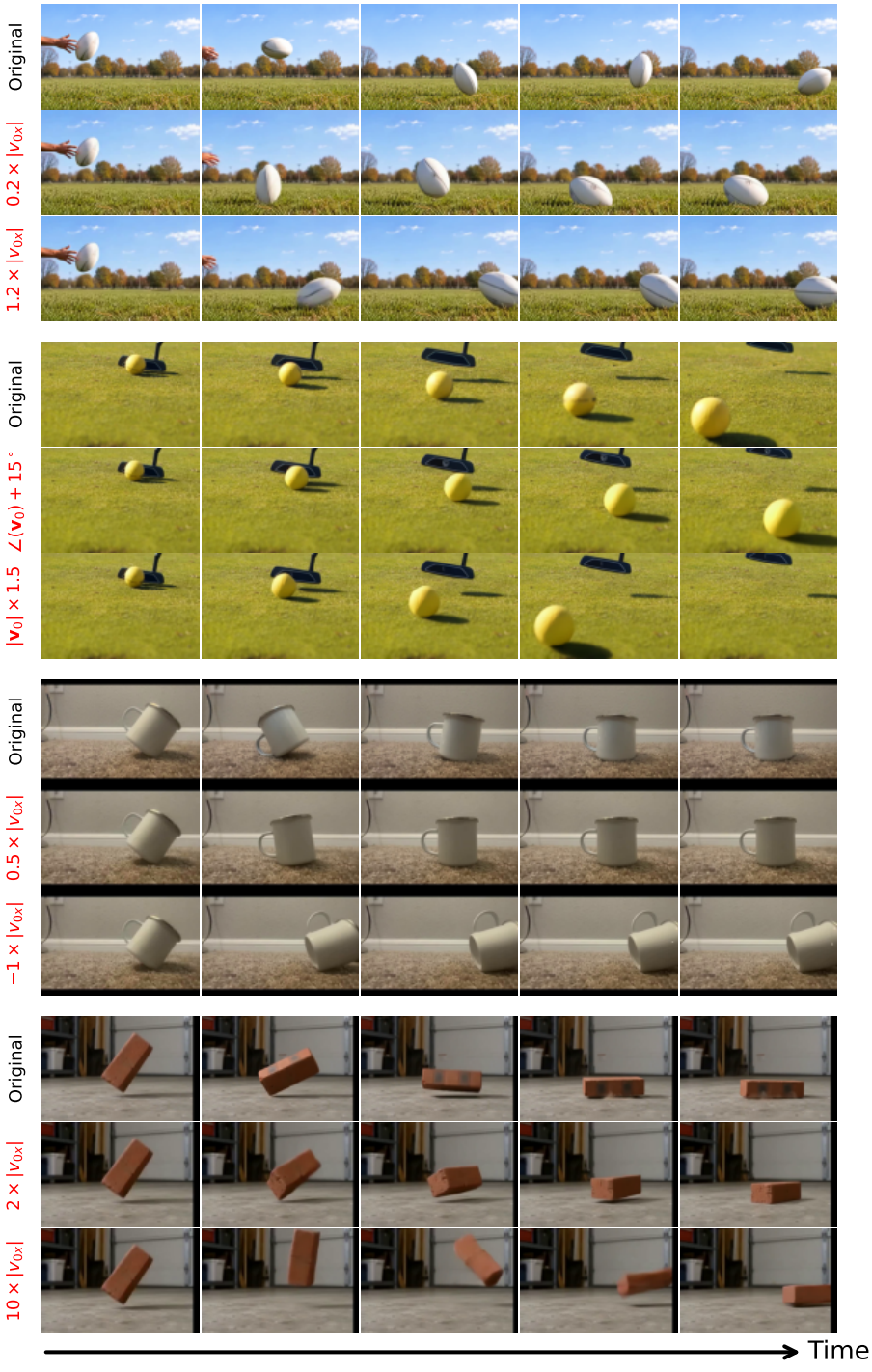


Figure 10: **Additional qualitative results of velocity edits.** For each example, the top row shows the original video and the following rows show PhysV2V outputs after modifying the initial velocity. The modified velocities are reflected in the generated videos' object trajectories and final positions.

A.8 Qualitative Comparison Against Baselines on Real-World Examples

In the following figures, we show qualitative comparisons of PhysV2V against physics-based I2V baselines for two real-world videos. We give the reference video's first frame as input for the I2V baselines. We also provide these methods with the optimized initial velocity from PhysV2V.



Figure 11: **Qualitative Results for edit "elasticity $\times$ 2".** The top row shows the input video, and the second row shows the output of PhysV2V with the elasticity edit applied. The third and fourth rows show the results of I2V methods PhysGen and RealWonder respectively. We observe that PhysV2V's edit results in the desired behavior of the ball bouncing higher.

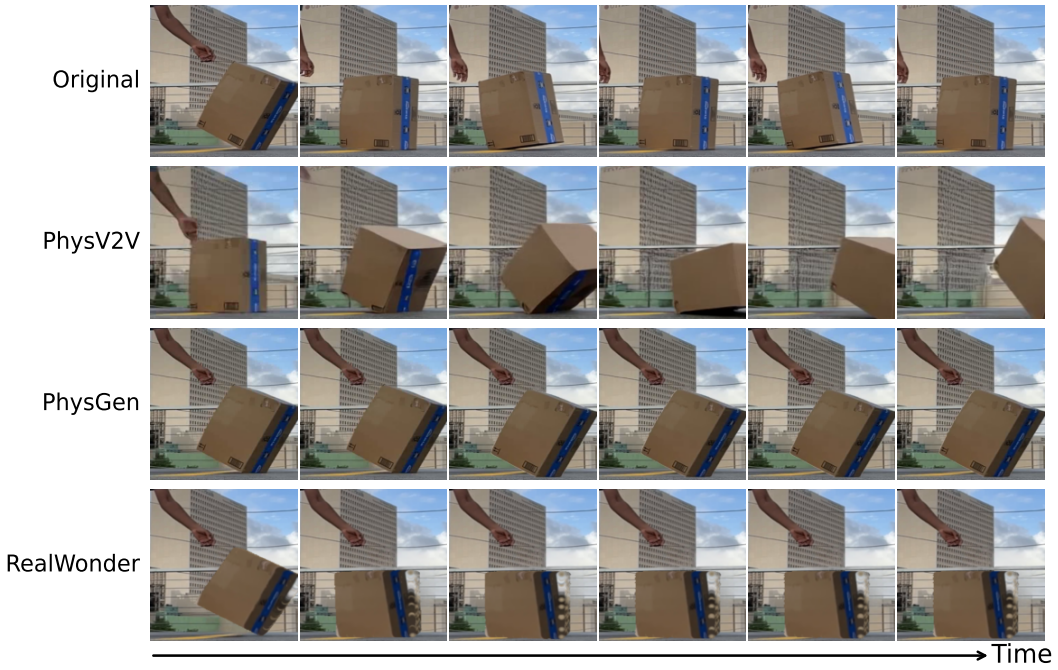


Figure 12: **Qualitative Results for edit "angular velocity magnitude $\times$ 2".** The top row shows the input video, and the second row shows the output of PhysV2V with the angular velocity magnitude (tipping speed) edit applied. The third and fourth rows show the results of I2V methods PhysGen and RealWonder respectively. We observe that PhysV2V's edit results in the desired behavior of the box tipping forward with increased motion.

A.9 Quantative Evaluation with Standard Deviation Intervals

Table 6: **Quantitative evaluation of all methods on the Kubric synthetic dataset.** For each physical edit type (velocity magnitude, velocity direction, elasticity, friction, mass), we report PSNR (dB), SSIM, and LPIPS averaged across 50 generated videos per edit. PhysV2V consistently achieves the best metric scores. Values shown as mean ± std.

Edit type	Method	PSNR (dB)	SSIM	LPIPS
$\|v_0\|$	PhysV2V	**27.13 ± 0.81**	**0.95 ± 0.01**	**0.11 ± 0.02**
	CogVideoX	20.43 ± 0.14	0.89 ± 0.01	0.13 ± 0.01
	Lucy Edit Dev	21.49 ± 0.08	0.91 ± 0.01	0.13 ± 0.02
	PhysGen	16.63 ± 0.24	0.71 ± 0.01	0.22 ± 0.02
	RealWonder	20.52 ± 0.77	0.90 ± 0.01	0.16 ± 0.02
$\angle v_0$	PhysV2V	**25.78 ± 0.93**	**0.93 ± 0.01**	0.08 ± 0.01
	CogVideoX	20.41 ± 0.15	0.89 ± 0.01	0.15 ± 0.01
	Lucy Edit Dev	21.31 ± 0.08	0.90 ± 0.01	**0.08 ± 0.01**
	PhysGen	16.71 ± 0.24	0.71 ± 0.01	0.24 ± 0.01
	RealWonder	20.03 ± 0.77	0.89 ± 0.01	0.15 ± 0.02
e	PhysV2V	**25.81 ± 1.01**	**0.94 ± 0.01**	0.09 ± 0.02
	CogVideoX	20.18 ± 0.14	0.89 ± 0.01	0.13 ± 0.01
	Lucy Edit Dev	21.12 ± 0.08	0.90 ± 0.01	**0.08 ± 0.01**
	PhysGen	16.64 ± 0.21	0.71 ± 0.01	0.24 ± 0.01
	RealWonder	18.18 ± 0.83	0.87 ± 0.01	0.18 ± 0.01
μ	PhysV2V	**26.61 ± 0.80**	**0.94 ± 0.01**	0.09 ± 0.02
	CogVideoX	20.93 ± 0.14	0.89 ± 0.01	0.14 ± 0.01
	Lucy Edit Dev	21.15 ± 0.08	0.92 ± 0.01	**0.08 ± 0.01**
	PhysGen	16.61 ± 0.21	0.71 ± 0.01	0.23 ± 0.01
	RealWonder	20.11 ± 1.03	0.89 ± 0.01	0.19 ± 0.02
m	PhysV2V	**26.44 ± 0.80**	**0.94 ± 0.01**	0.08 ± 0.02
	CogVideoX	20.21 ± 0.14	0.89 ± 0.01	0.12 ± 0.01
	Lucy Edit Dev	21.66 ± 0.08	0.91 ± 0.01	**0.08 ± 0.01**
	PhysGen	16.84 ± 0.22	0.71 ± 0.01	0.24 ± 0.01
	RealWonder	17.38 ± 0.81	0.86 ± 0.01	0.21 ± 0.02