VERA: DECOUPLED SIMULATION FOR EFFICIENT AND PHYSICALLY-GROUNDED WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

World models have recently gained significant popularity, largely in the form of video-generation models which are able to generate high-quality diverse videos of various scenes. Due to these abilities, many have tried to apply them to the task of generating synthetic data or planning for robotics applications. However, these trials have so far been limited by the lack of physical accuracy in generated videos. Because these video models are trained purely to predict future video frames, they struggle to learn the underlying 3D dynamics and cues required to generate physically consistent and accurate videos. In this work, we propose Vera, a decoupled approach to world modeling. By splitting the task into two stages, simulation and rendering, and carefully designing the simulation representation and architecture, we are able to train a lightweight simulation model which can learn generalizable physics and can be used for robotics planning in real-time. This model outputs 3D trajectories, allowing us to leverage pre-trained models for high-quality video generation through motion conditioning. Across multiple benchmarks, we demonstrate Vera's ability to simulate complex dynamics, learn generalizable physics laws, and generate high-quality, physically-accurate videos.

1 INTRODUCTION

Building computational models that can simulate the physical world, often referred to as "world models" Ha & Schmidhuber (2018), is a foundational pursuit in artificial intelligence. An effective world model must not only generate plausible future scenarios but also adhere strictly to the fundamental laws of physics. However, modeling the complex dynamics of the real world directly from high-dimensional visual observations remains a formidable challenge.

Current research largely faces a dichotomy between visual expressiveness and physical accuracy. On one end, large-scale generative video models (e.g., Cosmos NVIDIA et al. (2025), Wan Wan et al. (2025)) have achieved unprecedented success in synthesizing high-fidelity, photorealistic video sequences. Yet, because these models operate by predicting pixels or compressed unstructured latents, they lack explicit representations of 3D geometry, object permanence, and classical mechanics. Consequently, they often suffer from structural hallucinations and physically implausible dynamics over long horizons Kang et al. (2024). Conversely, learned physical simulators such as Graph Network-based Simulators (GNS) Sanchez-Gonzalez et al. (2020) and Interaction Networks (IN) Battaglia et al. (2016) leverage explicit state spaces, excel at capturing the precise dynamics of interacting entities, but are limited by their strict data requirements and tend to be scene- or material-specific rather than general.

In this paper, we propose a novel framework, called *Vera*, which provides a principled approach to bridging this gap. In order to carefully design our architecture, we first construct a set of requirements that our model should meet. These are:

3D Native The model should have explicit 3D grounding. When working with 2D video, the location and size of an object cannot be uniquely determined, so there are many possible future paths. This uncertainty makes learning physics extremely difficult and leads to entangled representations that do not perform well or generalize.

Physics Inductive Bias The model should have physics inductive biases. While recent models have shown strong performance by trading off inductive bias for scale, prior work has shown that

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.

this same trade-off does not work as cleanly when considering physics learning. We would like our model to learn simple laws such as gravity from relatively few samples, and we find that physics inductive bias, such as having the model predict acceleration instead of position, helps improve learning efficiency.

Efficiency The model should be efficient. Video-based world models such as Cosmos or Wan can take 5-10 minutes to generate a single sample, making them prohibitively expensive for real-world use such as planning or synthetic data generation. We would like our model to be lightweight and efficient so it can be plugged into a variety of downstream applications.

Inspired by these requirements, we design Vera to be an object-centric framework that predicts acceleration, a grounded physics quantity, directly in 3D space, and that can run in real-time on consumer/datacenter GPUs. Our framework works in three steps: object tokenization, trajectory prediction, and rendering. Crucially, rather than predicting arbitrary latent transitions, our transformer-based dynamics model predicts the acceleration of each object. These predictions are then explicitly integrated to determine precise future states, enforcing a strict Newtonian prior on the network. Finally, we employ a pre-trained renderer to translate these updated physical states back into high-quality image space. Because our pipeline is trained using real-world RGB-D video, we can directly supervise the 3D positions of the objects, grounding the learned dynamics in observable reality. We train our model both on general physics understanding datasets and on robot learning benchmarks, and show that our model outperforms baselines strongly in both settings. In the robot learning setup, we are able to perform real-time in-the-loop planning with our model to perform tasks in a zero-shot fashion.

To summarize, our contributions are as follows:

1. We introduce a novel world model framework which decouples simulation from rendering through a 3D native, object-centric representation, a learned Trajectory Transformer for simulation, and pre-trained diffusion models.
2. We show that when trained on small-scale synthetic physics datasets, our model learns generalizable physics laws for gravity and friction; recovering the gravitational constant as $g = 9.91 \pm 0.08$ m/s^2 from learned predictions alone (ground truth: 9.81 m/s^2) and correctly capturing the inverse relationship between friction and stopping distance, even for out-of-distribution parameters.
3. We show that when applied to robot learning tasks across both the PushT Chi et al. (2023) and Language-Table Lynch et al. (2023) tasks, our world model can enable real-time planning which matches behavior cloning baselines and outperforms prior world modeling frameworks by up to 7x on planning success rates.

2 Related Work

2.1 Learned Physical Simulators

A set of work learns physical dynamics directly from simulator data. Graph-based methods like IN Battaglia et al. (2016) and GNS Sanchez-Gonzalez et al. (2020) pass messages between object nodes in an explicit state space, predicting accelerations or velocities that are integrated forward via Euler or Verlet. NeuralForceFields Li et al. (2025b) extends this by predicting forces with a graph network, but requires collision information as input and treats each timestep independently. Vera shares the acceleration + Euler inductive bias but operates on RGB-D images instead of ground-truth simulator states, replacing graph message passing with cross-object attention over object tokens and removing the need for external collision input.

A second set uses transformers but predicts positions or trajectories rather than higher-order derivatives. ParticleFormer Huang et al. (2025) and HD-VPD Whitney et al. (2024) operate over dense particle/point representations (often $>$1000 tokens; HD-VPD uses $>$100K), while PhysCtrl Wang et al. (2025a) uses a bidirectional diffusion transformer limited to a single object. We find direct position prediction yields noisy, choppy trajectories; Vera instead predicts accelerations and pairs sparse per-object tokens with a unidirectional transformer that scales to many interacting objects and supports fast in-the-loop planning. Closer to our object-centric formulation, Dyn-O Wang et al.

(2025b) learns disentangled static/dynamic features in a latent space, lacking explicit 3D physics or a Newtonian inductive bias; Vera's explicit 3D tokens enable direct physics evaluation and generalization to unseen parameters.

2.2 Video-Based World Models

Rapid scaling of generative models has led to breakthroughs in video generation and world modeling. Foundational models like Cosmos NVIDIA et al. (2025); Agarwal et al. (2026), Wan Wan et al. (2025), and CogVideoX Yang et al. (2024) leverage massive datasets to synthesize realistic videos from text or image prompts. However, they lack explicit 3D geometry and classical mechanics, meaning they suffer from structural hallucinations, object morphing, and physically implausible dynamics over long horizons.

Recent work injects physical or 3D awareness into video generators through fine-tuning. GEN3C Ren et al. (2025) adds 3D awareness by training a modified diffusion model on point-cloud-rendered cache inputs paired with ground-truth videos, but physics remains implicit in the latent generation process. PhyGDPO Cai et al. (2025) improves physics accuracy via RL fine-tuning on a curated set of videos, with gains largely limited to human motion. Vera makes physics first-class via Newtonian integration, so physical correctness is verifiable in 3D rather than approximated through fine-tuning.

Another family explores object-centric latent representations: SlotAttention Locatello et al. (2020) learns to decompose scenes into a fixed number of slots and is extended by SlotFormer Wu et al. (2022) with a transformer for prediction in slot space, while SAVI++ Elsayed et al. (2022) jointly decomposes video and predicts future states. These latent representations prevent direct physics evaluation and require larger models to implicitly capture dynamics that explicit 3D representations encode by construction. Vera's explicit 3D tokens support smaller models and verifiable physics.

2.3 Prescribed Physics with Neural Rendering

A growing body of work decouples physics simulation from rendering by pairing explicit physics engines with neural renderers. gradSim Jatavallabhula et al. (2021) combines a differentiable multi-physics engine (rigid bodies, deformables, cloth) with a differentiable rasterizer for system identification from pixels, while PhysGen Liu et al. (2024) uses a similar three-stage pipeline (perception, rigid-body simulation, rendering) but in 2D image space. Newer methods connect simulators with 3D representations Xie et al. (2024); Zhang et al. (2024); Li et al. (2025c); Xu et al. (2025); Wang et al. (2026), but require either a pre-built simulator or explicit input of the governing physics equations. Vera retains the decoupled structure but *learns* dynamics from data rather than relying on hand-coded equations, operating natively in 3D.

These three lines of work have complementary limitations: learned simulators cannot ingest raw video or render photorealistic output; video-based models conflate physics with rendering and produce implausible dynamics; and prescribed-physics approaches require known equations and are often bound to scene-specific 3DGS reconstructions. Vera uniquely combines decoupled simulation/rendering, explicit 3D object-centric tokens with a Newtonian inductive bias, and learned (rather than prescribed) dynamics.

3 Physically Grounded World Model: Architecture

This section describes the details of our proposed model architecture. Inspired by the requirements outlined in the introduction, Vera consists of three stages: object tokenization, dynamics prediction, and rendering. For rendering, we rely on pre-trained diffusion models which have been adapted to accept optical flow, trajectories, or other motion cues as input, and focus the majority of our novelty and effort in the tokenization and prediction stages. An overview of our model architecture can be found in Fig. 1. Code for this model architecture and training will be released.

At a high level, the object tokenizer takes an input image plus optional supplementary information and reconstructs an object-centric 3D representation that includes information about object position, physics properties, and shape. The dynamics prediction stage then takes in this representation and predicts future states by predicting just the acceleration of each object. The renderer finally takes in

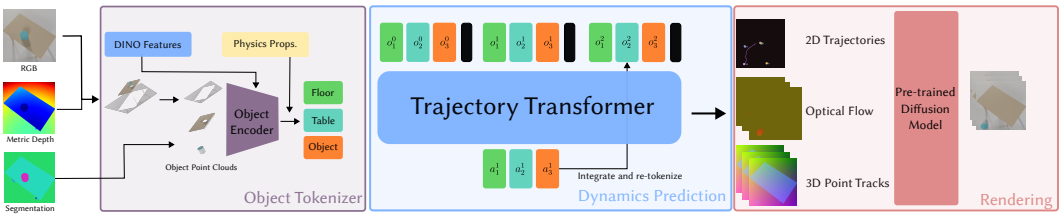


Figure 1: **Architecture Overview.** Our framework consists of three components: *Left.* The Object Tokenizer either takes in or estimates RGB, Depth, Segmentation, and physics features, and uses them encoded the image into object tokens. *Middle.* The trajectory transformer takes in object tokens for prior steps and predicts the acceleration of each object at the next timestep. This acceleration is used to integrate the object position forward and the object is re-tokenized to get the next set of inputs. *Right.* We can turn the predicted trajectories into a variety of output formats and pass them to pre-trained diffusion models for rendering.

full trajectories in various forms and turns it into a video. We now describe each of these sections in further detail:

Object Tokenizer The goal of the object tokenizer is to take in various 2D and physics-based inputs, including an RGB image, depth, segmentation, and physics properties such as friction and mass, and turn them into object-centric tokens, where each token contains the information needed to simulate the scene. If depth and segmentation are not provided, they are estimated using DepthAnythingv3 Lin et al. (2025) and SAM3 Carion et al. (2025). In addition, image features are computed using DINOv3 Siméoni et al. (2025). From the depth and RGB images, we compute a back-projected colored point cloud for the initial scene. The segmentation mask then partitions this into per-object point clouds, which are converted to a relative point cloud by subtracting each centroid. We then encode the object by concatenating the object position, rotation, relative point differences, and DINO features and applying a 2-layer MLP. A separate 2-layer MLP maps the input physics properties to FiLM parameters Perez et al. (2018), which scale each object token. We found that FiLM conditioning improves physics adherence and mitigates *"object drift"*, where the model predicts non-zero movement for objects that should be at rest. The result is a single token per object containing all information necessary for simulation: physics properties, current state, and shape. We denote these tokens o_i^t for the ith object at timestep t. Since the object position/state is included in the token, the object token varies as it moves over time, even though the DINO, physics, and shape features stay the same.

Dynamics Prediction After tokenization, we have N object tokens $\{o_1^0, o_2^0, ..., o_N^0\}$ for N objects in our scene. We then pass these to our trajectory transformer $\mathcal{T}$. This transformer is designed to take a sequence of past object tokens (across multiple timesteps potentially) and then predict *the acceleration of each object at the next timestep*. This is a key architectural choice which we found had a significant impact on the ability of our model to learn physics and also allowed us to train a significantly smaller model than would otherwise be needed (only 5.5M params). Additionally, by having the model predict accelerations, we guarantee that the object positions will be smooth and differentiable. Another key architectural decision was to adjust the attention formulation of our model. Traditional attention Vaswani et al. (2017) uses a softmax operation to normalize attention values, which forces every token to distribute attention across the full sequence. While this is well suited for text and video, we find it is often not optimal for physical simulation where objects may not interact (e.g. an object in free-fall or at rest). Therefore, we replace softmax with a sigmoid operation which allows each value to be between 0 and 1, enabling an object to attend to all, some, or no other objects.

Given object tokens across prior timesteps, the trajectory transformer predicts per-object accelerations. Mathematically,

$$\{a_1^{t+1}, \ldots, a_N^{t+1}\} = \mathcal{T}(o_1^0, \ldots, o_N^0, o_1^1, \ldots, o_N^1, \ldots, o_1^t, \ldots, o_N^t) \tag{1}$$

A 2-layer MLP maps the transformer output features to acceleration predictions, which are then integrated forward to obtain the position, rotation and linear/angular velocities of each object using semi-implicit Euler as in Sanchez-Gonzalez et al. (2020) so that the acceleration at time t directly

affects the position at time t. Predicting accelerations rather than positions is a deliberate design choice: when trained to directly predict future positions, we found the outputs were often noisy and choppy, whereas acceleration prediction yields smooth, physically consistent trajectories. Additionally, because our formulation operates over a variable-length set of object tokens, it naturally extends to any number of objects, even beyond how many were seen during training. We evaluate this generalization ability in Table 8. After computing the next state, we then re-tokenize our object using cached state, DINO, and physics features to compute o_i^{t+1} which is then concatenated to the full token list. In between timesteps, we add a learned *"time separator"* token to help the model differentiate between timesteps and objects. The time separator token is a learned embedding initialized randomly. The trajectory transformer is built of six traditional transformer encoder layers with a hidden size of 256, resulting in a total of 5.5M parameters including the object encoder. Our model is fully differentiable end-to-end, including tokenization, integration, and re-tokenization. This allows us to back-propagate through the model if needed for system identification or planning.

Supervision Our model is supervised directly from RGB-D video. We first run CoTracker Karaev et al. (2024), a 2D point tracking model on input videos to obtain 2D tracks with visibility. We then use these point tracks to tie 3D points from different frames together into a single trajectory which can be used directly to supervise the integrated positions from the model. Put together, for each object i, we have a sequence of positions p and rotations q: $\{p_i^0, q_i^0, p_i^1, q_i^1, ...p_i^T, q_i^T\}$, and our loss function can be formulated as: $\mathcal{L} = \frac{1}{T}\frac{1}{N}\sum_i \sum_t ||\hat{p}_i^t - p_i^t|| + ||\hat{q}_i^t - q_i^t||$

3.1 Rendering

Another key aspect is rendering, turning the predicted trajectories into high-quality videos. Because our simulation stage outputs explicit 3D trajectories, it can interface with any renderer that accepts motion cues as conditioning, such as optical flow, point tracks, or projected 3D positions. In this work, we focus on two renderers with complementary trade-offs, which are also visualized in Fig. 7.

Point Splatting Renderer. This is a fast point cloud splatting renderer Alexa et al. (2004). We take our back-projected point cloud, translate and rotate each object's point cloud per timestep according to the predicted trajectory, and render with depth-aware splatting. This renderer is deterministic, fast, and faithfully follows the predicted trajectories, making it well suited for planning and evaluation tasks where physical accuracy matters more than visual fidelity. However, it is limited by the information in the initial point cloud: moving objects leave holes in the rendering, and large rotations may expose surfaces not visible in the first frame. To mitigate the first issue, we mask out all objects in the first frame and use Stable Diffusion Inpainting Lugmayr et al. (2022) to fill in the background.

DiffusionAsShader. Our second renderer is DiffusionAsShader Gu et al. (2025), a diffusion model conditioned on 3D point tracks. We compute these tracks by sampling points on each object in the first frame and propagating them through the predicted trajectory, encoded as an RGB video with x, y, and depth as the three channels. Because it is trained on a large and diverse video dataset, DiffusionAsShader generates photorealistic videos across a wide range of objects, lighting conditions, and settings, handling novel viewpoints and occluded surfaces that PointSplat cannot. This makes it the better choice when visual quality is the primary objective. However, as a generative model it may occasionally disregard the input motion cues. It is also significantly slower, taking up to 400 seconds on a single AMD Instinct™ MI300X GPU.

4 World Model Evaluation

A key goal of our work is to deliver a model which has both strong physical accuracy and high-quality videos. In order to study the ability of our model to learn and predict physics, we train and test it on a synthetic dataset consisting of scenes that test different physics laws and scenarios. We gather data from the Physion dataset Bear et al. (2021), specifically taking from the Drop, Collide, Contain, and Support sets which cover collision, gravity, support, and more. In addition, we generate new synthetic samples which cover gravity, through falling objects, and friction, through videos of objects sliding on a table. The new samples are all collected using the Kubric simulator Greff et al. (2022). All samples in this dataset include both mass and friction which are provided as input to the model. In order to test both accuracy and generalization, we limit the values we input to a specific

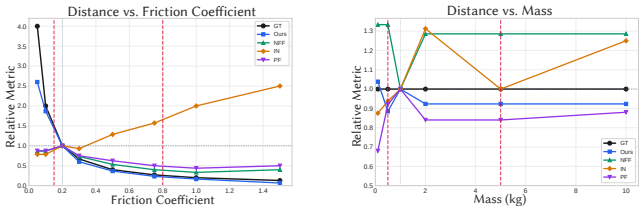


Figure 2: **Evaluation of friction understanding.** In order to evaluate the friction understanding of our model, we inference it with the same starting image/conditions but varying friction coefficients and masses for the sliding object. According to the laws of friction, the distance the object travels should be inversely proportional to friction coefficient and independent of the objects mass. We see that while most models learn the inverse correlation somewhat, only our model is able to generalize to unknown friction parameters (outside the red bars). On mass, while all models learn a relatively strong degree of independence, we see that our models shows less deviation.

range: 0.15-0.8 for friction coefficients and 1-5 kg for mass. This ensures there are values the model has never seen and can be used to test out-of-distribution (OOD) generalization.

Baselines We compare our model against both Particle-Based Models and Video World Models. In particular we compare against Interaction Networks Battaglia et al. (2016), NeuralForceFields Li et al. (2025b), ParticleFormer Huang et al. (2025), Cosmos 3 Nano Agarwal et al. (2026), Wan 2.2 Wan et al. (2025). Cosmos and Wan are video-based models, so we compare on video metrics such as PSNR and LPIPS. These models do not take in physics properties, but Cosmos 3 takes in 5 frames of motion which provides clues for future frames. The other methods all explicitly take in physics properties as inputs, allowing us to directly compare physics learning.

4.1 Results

We use a combination of three datasets/benchmarks to evaluate the physics understanding of our model. First, we evaluate on the validation set of the synthetic dataset described earlier. We show these results in Table 1a. Our model significantly outperforms the baselines on per-step prediction error across most scenarios, but sometimes suffers on endpoint error due to its auto-regressive nature and the large sequence lengths. It is able to achieve low error across almost all scenarios, demonstrating it is able to learn a variety of dynamics. To compare against video-based models such as Cosmos 3 and Wan 2.2, we evaluate the generated videos using PSNR and LPIPs, and show Vera to be superior in both metrics in Table 1b. Qualitative results from our model are shown in Fig. 5, Fig. 6 and Fig. 7.

Physics Generalization Evaluation In order to evaluate how well these models have learned specific physics laws, we perform a controlled test on the sliding setup. Specifically, we use the same starting image but vary the friction coefficient of the object and the mass of the object with the goal of understanding what relationship the models have learned between the physics parameters and object motion. The stopping distance of an object with initial velocity v_0, friction coefficient μ is $d = \frac{v_0^2}{2\mu g}$. This tells us that mass does not matter at all, and friction is inversely proportional to distance. In Fig. 2, we plot the results from the baselines and our model across a range of friction values going from 0.05 to 1.5 and mass values going from 0.1 to 10 kg and compare against the ground truth (GT). The red lines indicate the in-distribution region of values the models saw during training. While NeuralForceFields and ParticleFormer both learn a good friction relation in-distribution, they perform poorly out-of-distribution on both ends, while InteractionNetworks learns an entirely incorrect relationship. Visual results from our model with varying friction coefficients are shown in Fig. 5. For mass, we see that all models have learned that distance is independent of mass, though other models show higher variance than ours.

PISA-Bench We further evaluate our model on PISA-Bench Li et al. (2025a), a benchmark which tests the ability of video-generation models to simulate falling objects. It consists of both a real and simulated evaluation set of falling objects with 2-10 objects per scene. We do not fine-tune our model on this set, and apply it zero shot. We show quantitative results in the Appendix in Table 4: our model is able to match or outperform the previous best models, both zero-shot and fine-tuned, on PISA-Bench, showing our model's ability to simulate gravity and collision dynamics on new data.

Model	Gravity	Table Slide	Drop	Collide	Contain	Support	Overall
IN Battaglia et al. (2016)	0.1492	0.0765	0.1257	0.2502	0.2111	0.2309	0.1332
NFF Li et al. (2025b)	0.1044	0.2804	0.0910	0.3515	**0.1435**	0.2532	0.2253
PF Huang et al. (2025)	0.1681	0.1513	0.4207	0.3460	0.9770	0.7471	0.4101
Ours	**0.0317**	**0.0143**	**0.0225**	**0.1065**	0.1470	**0.0675**	**0.0849**
IN Battaglia et al. (2016)	0.5073	0.3276	0.4193	**0.7939**	**0.5955**	0.7151	0.4635
NFF Li et al. (2025b)	1.1266	1.3086	0.7348	1.7319	0.8013	1.6662	1.1382
PF Huang et al. (2025)	0.6220	0.5845	1.243	1.125	3.265	2.680	1.398
Ours	**0.159**	**0.0805**	**0.1812**	0.9750	0.8551	**0.6342**	**0.4108**

(a) **Prediction Results across all scenarios.** *Top.* MAE of position per timestep. *Bottom.* Mean endpoint error at the end of the rollout. Lower is better.

Model	PSNR ↑	LPIPS ↓	FVD ↓
Cosmos 3	17.23	0.191	176.1
Wan 2.2	14.12	0.229	223.7
Ours (Splat)	15.98	0.208	203.6
Ours	**18.13**	**0.187**	**141.5**

(b) **Video Quality on the Synthetic Physics Dataset**.

Table 1: **Performance of physical prediction and video generation baselines on the synthetic physics validation set.**

WorldBench We also evaluate our model on WorldBench Upadhyay et al. (2026), specifically the PerfectPhysics subset. This benchmark estimates physical constants such as g, making it a relevant benchmark for a 3D model such as ours. We show quantitative results in the Appendix Table. 5. While Cosmos 2.5, Wan 2.2, and CogVideoX produce inaccurate results, our model is consistently able to replicate a gravitational constant near g (Avg: 9.91), showing that our model is accurate at modeling physics such as gravity. This result validates that our Newtonian inductive bias of predicting acceleration enables accurate learning and recovery of physical constants.

5 Robotics Evaluation

To demonstrate the ability of our model to simulate real robotics tasks, we test it on two datasets related to robot learning: PushT Chi et al. (2023) and Language-Table Lynch et al. (2023).

5.1 PushT

PushT is a robot learning dataset where the agent, represented by a circle, must push a T-block into a *"goal"* position Chi et al. (2023). The goal position is denoted by a green T. We train our model to predict the acceleration of the T-block given the *"action"* of the agent. The T-block state is represented by a 3-dimensional vector representing x,y position and angle. As in the earlier datasets, our model predicts the linear and angular accelerations. For planning, we follow prior work and leverage the stableworldmodel Maes et al. (2026) library for planning and evaluation. The model is trained on a dataset of 20,000 rollouts, with 2,000 unique starting configurations and 100 noise variants of each configuration. We compare against four baselines: DINO-WM Zhou et al. (2024) and PLDM Sobal et al. (2025) are latent world models while Goal-Conditioned Behavior Cloning (GCBC) and Goal-Conditioned Implicit Value Learning (GCIVL) are traditional RL policies. Quantitative success rates are in Table 3a. Our model outperforms both prior latent world models, such as DINO-WM and PLDM, and behavior cloning baselines GCBC and GCIVL. Qualitative results can be seen in Fig. 4. The agent trajectory was computed using CEM planning with our model as the cost function. We used a 25-step horizon with 200 sampled trajectories per step. This figure shows both the ability of our model to plan successful trajectories and the fidelity of its simulations compared to the real model.

5.2 Language-Table

Language-Table Lynch et al. (2023) is a dataset that tests various robot pushing tasks on a table top with multiple small objects. We test our model on this benchmark to further evaluate its ability to simulate robotic scenes and model complex physics such as multi-object collision, pushing, and friction. We compare against both particle-based baselines (InteractionNetworks, NeuralForceFields, and ParticleFormer), and a latent world model (DINO-WM). Since DINO-WM is entirely latent, we cannot compare it directly on 3D accuracy, but we show planning results for all models in Table 2. All results were obtained using model predictive path integral (MPPI), 12-step horizon, and 200-step rollouts. We additionally include a visual planning mode for our model in which the cost function is computed by prompting a VLM to judge rendered rollouts. Our model outperforms

Model	Dataset Size	Oracle Data	Block2Block ↑	Block2AbsolutePos ↑	Validation Sampled ↑	Parameters
BC Baseline	200K	✓	70 ± 4.2	72 ± 4.6	-	
DINO-WM	12K	✗	8 ± 0.4	2 ± 0.2	26 ± 2.1	1.1B
NFF	12K	✗	22 ± 5.1	23 ± 3.7	34 ± 2.4	1.1M
IN	12K	✗	10 ± 2.2	6 ± 0.8	14 ± 1.5	1.1M
ParticleFormer	12K	✗	24 ± 1.1	27 ± 2.5	38 ± 3.8	5.5M
Ours (vision)	12K	✗	42 ± 2.3	38 ± 7.2	50 ± 6.1	5.5M
Ours	12K	✗	$\mathbf{57 \pm 3.5}$	$\mathbf{48 \pm 2.9}$	$\mathbf{76 \pm 4.3}$	5.5M

Table 2: **Planning Results on Language-Table** Our model outperforms all prior world model approaches, including DINO-WM which has 20x the parameters. Validation Sampled tasks are generated by choosing a trajectory sample from the validation set and setting the final state as the goal.

Model	Success Rate ↑
Ours	**80**
DINO-WM	74
PLDM	78
GCBC	75
GCIVL	33

(a) **Planning Results on the PushT dataset**. Our model outperforms both prior world model based approaches and RL baselines, showing our model is highly accurate and allows for robust planning.

Model	MAE (Moving) ↓	MAE (All) ↓	Endpoint ↓
IN	3.83	**0.85**	5.49
NFF	3.41	1.18	4.91
PF	4.22	2.81	6.93
Ours	**1.51**	0.90	**1.73**

(b) **Prediction Results on Language-Table**. All units are in centimeters. The moving metric is computed only on blocks which move during the episode, while "All" computes it over all blocks.

Table 3: **Results on PushT and Language-Table benchmarks verses baselines.**

all baselines in terms of planning ability, with success rates up to 7x higher in non-visual mode. Compared to the 3D models, we also achieve stronger prediction accuracy across both per-step error and endpoint error. Results can be seen in Table 3b. Rollouts with our model are shown in Fig. 3.

6 Further Evaluation

Generalization We evaluate how well our model generalizes to unseen amounts of objects or sequence lengths. We utilize the Language-Table data for this test to evaluate the performance in a real task. We test both more blocks than the model has ever seen, with ten blocks (four or eight are seen during training), and with longer sequence lengths than the model saw during training. The results can be seen in Table 7 and Table 8. Despite never seeing six or ten blocks during training, our model is able to generalize and simulate those settings close to its own performance at eight blocks.

Ablation To isolate the contribution of each choice, we ablate three key components: sigmoid attention, FiLM-based physics conditioning, and acceleration prediction. Results are shown in Table 10. Removing acceleration prediction causes the largest degradation, confirming that predicting accelerations and integrating via Euler is critical for learning physically-grounded dynamics. Removing sigmoid attention increases endpoint error to 2.85 while removing the FiLM encoder has a more modest effect (1.93 error). We additionally evaluate our model on degraded or estimated physical parameters to evaluate its robustness. The results are shown in Tab. 6 and show our model degrades gracefully and is relatively robust to noise and estimation error.

Efficiency Another key aspect of our model is efficiency. The physics inductive biases allow us to train a model with fewer parameters than competing world models while achieving similar performance. Both parameter and runtime comparisons are shown in Table 9. Among sim-only models with comparable parameter counts and inference speed, our model demonstrates substantially stronger physics adherence. For planning applications that do not require rendered video, our simulation module alone is orders of magnitude faster and smaller than video-based world models, and when full rendering is included, the complete pipeline remains competitive in both speed and visual quality.

Limitations Our current formulation models each object as a single rigid body, which does not capture deformable objects, fluids, or articulated structures. Extending the object tokenizer and tra-

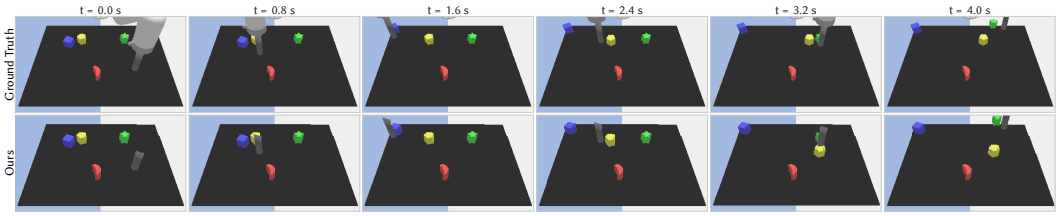


Figure 3: **Rendered Rollout on the Language-Table set**. Our model is able to both predict the robot/block collisions and render them with high-fidelity. The model accurately predicts the robot pushing the blue block into the corner and then the green to the top edge. Since the language-table dataset contains small objects and minimal rotation, we use the PointSplat renderer for speed. A limitation of our framework is that since the initial segmentation only includes the cylinder on the robot arm, our renders are also missing the actual arm body. This could be improved with better segmentation or by using a different rendering technique to improve the videos.

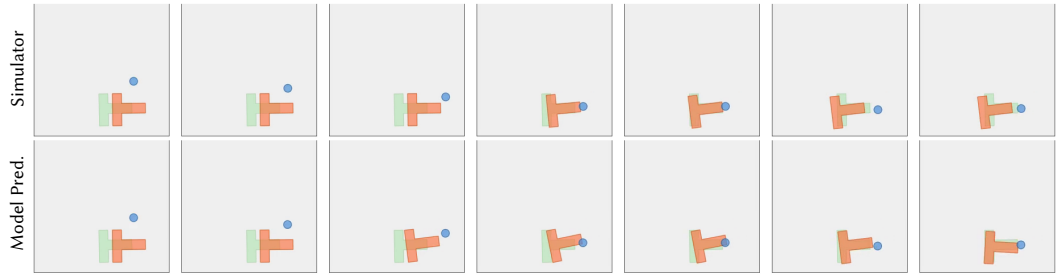


Figure 4: **Sample Model Rollouts vs. Simulation Rollout for the same action sequence on PushT.** The action sequence was obtained through CEM planning with red as the starting point and green as the goal. Planning with our model allows the agent to learn to go around the T to the right side before pushing towards the goal. Simulator denotes the block motion when the same actions are run in the simulator.

jectory transformer to predict per-point or mesh-level deformations is a natural direction for future work. Second, because we do not fine-tune the rendering stage jointly with the trajectory transformer, our pipeline is currently compatible only with renderers that accept motion cues such as point tracks or optical flow as conditioning. Finally, our model requires per-object physical parameters as input. In robotics settings, these can be estimated through brief exploratory interactions, but for single-frame video generation they are generally unavailable and must be assumed or estimated from visual appearance. Although results (Tab. 6) show our model is robust to noise and estimation error, extending the framework to infer physical parameters directly from visual observations is an important direction for future work.

7 Conclusion

In this work, we introduce Vera, a world model framework which decouples simulation from rendering, and introduces deliberate physics bias in order to more accurately and efficiently learn and simulate physics. The dynamics prediction alone can be done in real-time with just a 5.5M parameter model, while high-quality videos can also be rendered from the model outputs using pre-trained diffusion models. When applied to various synthetic physics examples, we find that our model is able to learn generalizable physics laws such as gravity and friction. This allows our model to simulate complex robot tasks such as PushT and Language-Table, delivering state of the art planning results and strong prediction accuracy while still being performant enough for use in in-the-loop planning. The strong results show that Vera's physics inspired architecture opens up many future directions including scaling up the model to real-world internet scale data, non-rigid bodies, and more complicated robotics modalities.

8 AI DISCLOSURE

In this work, we used generative AI tools for writing the code for the methods and evaluations described above. This includes writing the code to generate synthetic datasets via simulators and existing (non-generative) renderers. We have not used generative AI tools for generating the synthetic data, any aspect of the writing or editing of this paper, figure creation, summarizing existing literature, interpreting results, or ideation/brainstorming. All AI work was evaluated by humans through examination of the generated code as well as automated sanity checks on outputs.

9 REPRODUCABILITY STATEMENT

In order to make our work reproducible, we have extensively described our model architecture and training practices in Section 3 above and Section D of the Appendix. Additionally, we have included code for the model architecture in the supplementary material. We have also included details on the evaluation methodology and hyperparameters in Section 5 above. For the synthetic dataset, we have provided details on how samples were generated or selected from Physion in Section C of the Appendix and will also release the code for this generation.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Marc Alexa, Markus Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker. Point-based computer graphics. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, pp. 7–es, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 9781450378017. doi: 10.1145/1103900.1103907. URL https://doi.org/10.1145/1103900.1103907.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

Daniel M Bear, Elias Wang, Damian Mrowca, Felix J Binder, Hsiao-Yu Fish Tung, RT Pramod, Cameron Holdaway, Sirui Tao, Kevin Smith, Fan-Yun Sun, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021.

Yuanhao Cai, Kunpeng Li, Menglin Jia, Jialiang Wang, Junzhe Sun, Feng Liang, Weifeng Chen, Felix Juefei-Xu, Chu Wang, Ali Thabet, et al. Phygdpo: Physics-aware groupwise direct preference optimization for physically consistent text-to-video generation. *arXiv preprint arXiv:2512.24551*, 2025.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Blender Online Community. Blender - a 3d modelling and rendering package, 2026. URL http://www.blender.org.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

Gamaleldin Elsayed, Aravindh Mahendran, Sjoerd Van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. *Advances in Neural Information Processing Systems*, 35:28940–28954, 2022.

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.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, et al. Diffusion as shader: 3d-aware video diffusion for versatile video generation control. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–12, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. Particleformer: A 3d point cloud world model for multi-object, multi-material robotic manipulation. *arXiv preprint arXiv:2506.23126*, 2025.

Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*, 2021.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How far is video generation from world model: A physical law perspective. *arXiv preprint arXiv:2411.02385*, 2024.

Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proc. arXiv:2410.11831*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Chenyu Li, Oscar Michel, Xichen Pan, Sainan Liu, Mike Roberts, and Saining Xie. Pisa experiments: Exploring physics post-training for video diffusion models by watching stuff drop. *arXiv preprint arXiv:2503.09595*, 2025a.

Shiqian Li, Ruihong Shen, Yaoyu Tao, Chi Zhang, and Yixin Zhu. Neural force field: Few-shot learning of generalized physical reasoning. *arXiv preprint arXiv:2502.08987*, 2025b.

Wenxuan Li, Hang Zhao, Zhiyuan Yu, Yu Du, Qin Zou, Ruizhen Hu, and Kai Xu. Pinwm: Learning physics-informed world models for non-prehensile manipulation. *arXiv preprint arXiv:2504.16693*, 2025c.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

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.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11461–11471, 2022.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

Lucas Maes, Quentin Le Lidec, Dan Haramati, Nassim Massaudi, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel-v1: Reproducible world modeling research and evaluation. *arXiv preprint arXiv:2602.08968*, 2026.

NVIDIA, Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, Prithvijit Chattopadhyay, Mike Chen, Yongxin Chen, Yu Chen, Shuai Cheng, Yin Cui, Jenna Diamond, Yifan Ding, Jiaojiao Fan, Linxi Fan, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Ruiyuan Gao, Yunhao Ge, Jinwei Gu, Aryaman Gupta, Siddharth Gururani, Imad El Hanafi, Ali Hassani, Zekun Hao, Jacob Huffman, Joel Jang, Pooya Jannaty, Jan Kautz, Grace Lam, Xuan Li, Zhaoshuo Li, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Yen-Chen Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Yifan Lu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Seungjun Nah, Yashraj Narang, Abhijeet Panaskar, Lindsey Pavao, Trung Pham, Morteza Ramezanali, Fitsum Reda, Scott Reed, Xuanchi Ren, Haonan Shao, Yue Shen, Stella Shi, Shuran Song, Bartosz Stefaniak, Shangkun Sun, Shitao Tang, Sameena Tasmeen, Lyne Tchapmi, Wei-Cheng Tseng, Jibin Varghese, Andrew Z. Wang, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Jiashu Xu, Dinghao Yang, Xiaodong Yang, Haotian Ye, Seonghyeon Ye, Xiaohui Zeng, Jing Zhang, Qinsheng Zhang, Kaiwen Zheng, Andrew Zhu, and Yuke Zhu. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6121–6132, 2025.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

Rishi Upadhyay, Howard Zhang, Jim Solomon, Ayush Agrawal, Pranay Boreddy, Shruti Satya Narayana, Yunhao Ba, Alex Wong, Celso M de Melo, and Achuta Kadambi. Worldbench: Disambiguating physics for diagnostic evaluation of world models. *arXiv preprint arXiv:2601.21282*, 2026.

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

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

Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

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.

Meizhong Wang, Wanxin Jin, Kun Cao, Lihua Xie, and Yiguang Hong. Contactgaussian-wm: Learning physics-grounded world model from videos. *arXiv preprint arXiv:2602.11021*, 2026.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations. *arXiv preprint arXiv:2507.03298*, 2025b.

William F Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. *arXiv preprint arXiv:2406.19800*, 2024.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

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.

Yijun Xu, Jingrui Zhang, Hongyi Liu, Yuhan Chen, Yuanyang Wang, Qingyao Guo, Dingwen Wang, Lei Yu, and Chu He. Pegs: Physics-event enhanced large spatiotemporal motion reconstruction via 3d gaussian splatting. *arXiv preprint arXiv:2511.17116*, 2025.

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.

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.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Additional Quantitative Results

In this section, we provide quantitative results for the PISA-Bench Li et al. (2025a) and World-Bench Upadhyay et al. (2026) benchmarks, along with detailed results for generalization and ablation studies. Table 4 shows that Vera strongly out-performs the prior best out of distribution model (CogVideoX) and even performs better or comparable to the best in-distribution model across the real and synthetic sets, showing the generalizability of our approach. In Tab. 5 our model consistently outputs physically accurate videos, with the mean very close 9.81 and a spread which is magnitudes smaller than most prior video models.

Model	Best OOD CogVideoX	Best ID OpenSora+ORO	Ours	Best OOD CogVideoX	Best ID OpenSora+ORO	Ours
L2 ↓	0.138	0.067	**0.054**	0.112	**0.026**	0.033
IoU ↑	0.080	0.136	**0.214**	0.020	0.170	**0.224**
Chamfer ↓	0.366	**0.164**	0.173	0.315	**0.062**	0.091

Table 4: **Results on the PISA-Bench Real set (top) and Synthetic set (bottom).** Our model matches or out-performs prior models, including those which are trained on in-distribution data, despite not being trained or fine-tuned on any PISA-Bench specific data. Our model performs worse on synthetic data, likely due to those samples being more difficult for DepthAnything.

Model	Cosmos 2.5	CogVideoX	Wan 2.2	Ours	Ground Truth
Free-Fall	4.778 ± 4.474	-0.039 ± 0.136	0.378 ± 0.784	$\mathbf{9.91 \pm 0.081}$	9.81
Parabolic	5.375 ± 2.763	0.181 ± 0.239	1.682 ± 3.065	$\mathbf{9.76 \pm 0.131}$	9.81

Table 5: **Results on the WorldBench PerfectPhysics Subset**. Our model is the only one to consistently predict videos that closely follow the gravitational constant g in both free-fall and parabolic scenarios.

Degradation	Per Step Error	Endpoint Error
0%	0.0849	0.4108
5%	0.0985	0.4792
10%	0.1015	0.4876
20%	0.1212	0.5939
VLM Estimate	0.1092	0.4793

Table 6: **Performance of our model when noise is added to input physical parameters on the synthetic physics set**. Our model shows graceful degradation when noise is added and does not catastrophically fail. In addition, when we use parameters estimated by Gemini 3.8 Flash, our model shows only 16% increase in endpoint error (on par with 5% added noise), showing that our model can be used in-the-wild.

Tab. 10 shows the results of an ablation study which evaluates how key architectural decisions affect performance. We find that all components improve performance, with the largest factor being predicting acceleration instead of position. Tab. 9 shows the runtime comparison for our models and prior work. Our model is competitive on simulation time with all models and is significantly faster than all video-generation models.

Tab. 8 and Tab. 7 show generalization experiments, where we test the model on scenarios it has not seen before: Tab. 8 tests whether a model trained on either 4 or 8 blocks can generalize to 6 or 10 while Tab. 7 tests whether our model can handle sequence lengths up to 4x longer than those seen in training. Across both, our model shows strong generalization, although performance does degrade in OOD scenarios, specifically in ultra-long sequence lengths (96 frames).

A.1 Robustness

A key property of our model architecture is that it takes the mass and friction of objects as input. For real world scenes, this may not always be available, so we evaluate how well our model adapts to degraded physical parameters and to parameters estimated by Vision-Language Models. We find that our model's performance degrades gracefully under artificial noise added to the input parameters or when Gemini 3.8 Flash estimates the parameters (VLM Estimate row) as shown in Tab. 6.

B Additional Qualitative Results

We also show additional qualitative results below. Fig. 5 shows the results of rolling out our model with the same starting point but varying friction coefficients for the object. As the friction coefficient increases, the object travels less and less distance, going from falling off the edge to coming to a stop almost immediately. Fig. 7 shows a comparison of the two rendering methods we evaluate: PointSplat and DiffusionAsShader. While PointSplat is fast, it results in artifacts from the inpainting

Metric	24	48	72	96
Pos MAE	1.51	1.64	1.91	2.20
Endpoint Error	1.73	1.83	2.71	4.89

Table 7: **Generalization Tests: Sequence Length**. All units are in centimeters. Our model is natively trained on sequences up to 24 frames long (4.8 s at dt=0.2 s) and evaluated at extended sequences. Although performance degrades, it is still competitive at long sequence lengths.

Metric	6	8	10
Pos MAE	1.34	1.51	1.88
Endpoint Error	1.61	1.73	2.18

Table 8: **Generalization Test: Number of Blocks**. All units are cm. Our model is trained with 4 or 8 blocks, but generalizes well to 6 and 10.

Model	IN	NFF	PF	DINO-WM	Cosmos 3	Wan 2.2	Ours
Parameters	1.1M	1.1M	5.5M	1.1B	16B	5B	5.5M
Sim Time (s)	0.546	0.581	2.39	-	-	-	1.09
E2E Time (s)	-	-	-	4.51	475	530	401

Table 9: **Runtime and Parameter Counts for 48 frames of inference**. Sim Time refers to the time to predict trajectories/complete the simulation only. E2E time refers to the full process of both simulation and rendering. All models were benchmarked on a single AMD Instinct™ MI300.

and is missing secondary effects such as shadows. On the other hand, DiffusionAsShader is able to generate a high quality video while sacrificing speed.

We also visualize some rendering limitations in Fig. 9. Although our model predicts a plausible trajectory and the tracking video matches it, DiffusionAsShader generates a video which makes the armadillo into a non-rigid object and does not accurately model the armadillo sliding off the bunny. DiffusionAsShader occasionally relies on its extensive priors rather than the motion cues provided as input.

C Dataset Construction

In this section, we provide details on the process used to generate the synthetic data for model training and evaluation in Section 4.

C.1 Generated Data

All data was generated using the Kubric framework Greff et al. (2022) which internally uses PyBullet Coumans & Bai (2016–2019) for simulation and Blender Community (2026) for rendering. We generated data in four different scenarios: Gravity, Table Drop, and Table Slide. Gravity consisted of a single object falling onto a ground plane. The object location, size, shape, and material was randomized. Table Drop consisted of a single object dropped from free-fall along with a table. The object location was randomized so it would sometimes fall onto the table and sometimes directly onto the ground. The location and size of both the object and table were randomized and the object shape and material were also varied. Table Slide consisted of one or two objects sliding on a table. The initial velocity of the object, the friction coefficient of the object, the shape of the object, and the size of the table was randomized. For the friction coefficient, we used only values in between 0.15 and 0.8. For all simulations, the camera position was also randomized within a hemisphere so that all objects were visible in the first frame. Additionally, the mass of the object was randomized

Metric	MAE (Moving) ↓	MAE (All) ↓	Endpoint Err. ↓
Ours	1.51	0.90	1.73
w/o Sigmoid Attn.	1.77	1.23	2.85
w/o FiLM Encoder	1.58	1.39	1.93
w/o Acc. Pred	2.51	0.72	3.71

Table 10: **Ablation Study** All units are in centimeters. We evaluate removing sigmoid attention, the FiLM encoding, and using position prediction directly instead of acceleration. Removing the FiLM encoding has the least effect on moving objects, but has a large effect on static objects. Removing sigmoid attention increases error across both types of objects, but not as much as predicting positions directly.

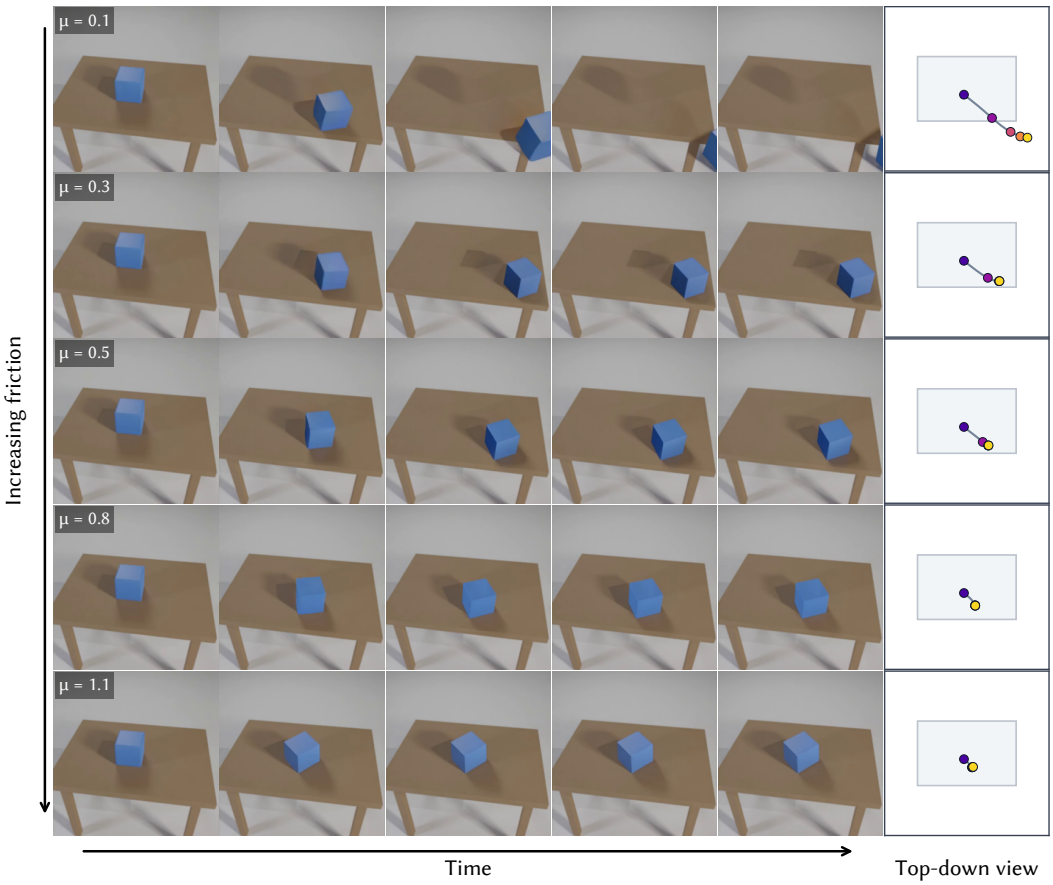


Figure 5: **Generated videos with the same starting image but varying friction coefficients for the object.** The figure on the right shows the trajectory of the object as it slides from a top-down viewpoint. The rectangle denotes the edges of the table. As the friction coefficient increases, the block travels slower and stops earlier. At $\mu = 0.1$, the block slides all the way to the edge and falls off the table, while at $\mu = 1.1$ it barely moves. All videos are rendered with DiffusionAsShader.

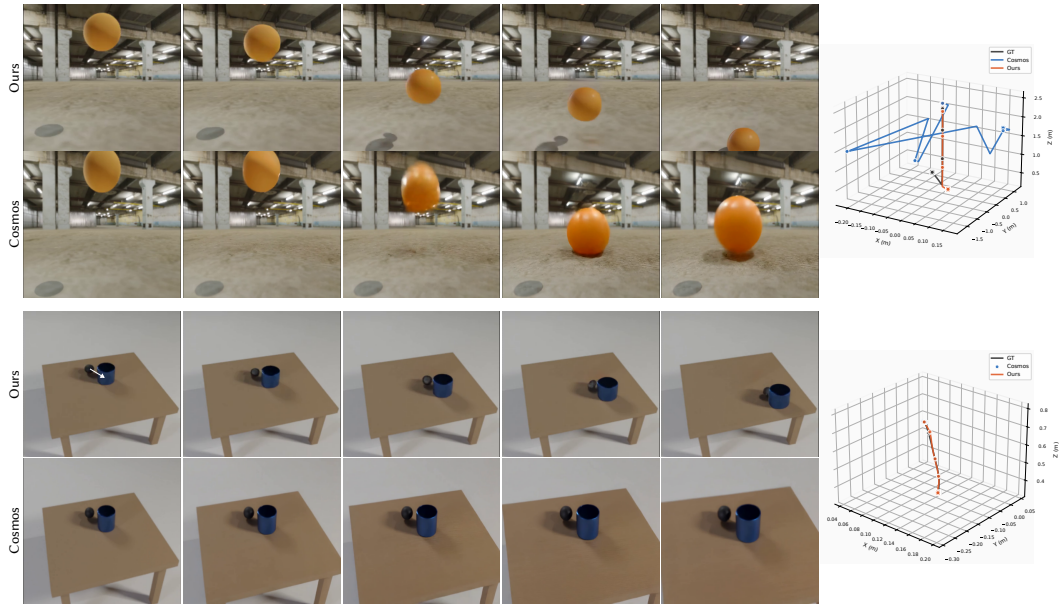


Figure 6: **Our model accurately models scenarios that prior video models cannot.** Even on simple scenes such as a ball falling under gravity, large video models like Cosmos generate unrealistic videos (ball teleports in depth half way through the video). Similarly, in the pushing scenario where an object is pushed by a ball, Cosmos generates almost no motion and instead moves the camera viewpoint. Vera more accurately matches the motion of the ball through the fall and collision. Our model received only the first frame as input along with user input on initial velocity of objects. Cosmos receives the first 5 frames.

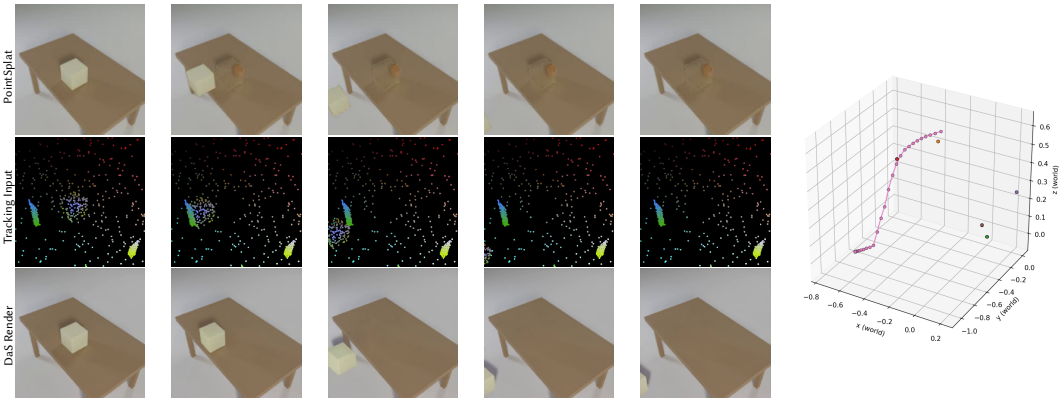


Figure 7: **PointSplat Renderer vs. DiffusionAsShader Renderer.** We render a scene of an object sliding off a table with both renderers. *Top* The PointSplat renderer displays artifacts in the region initially occluded by the object (from inpainting) and is missing artifacts such as the shadow of the cube as it falls. *Middle* The tracking video input to DiffusionAsShader shows the points on the cube moving while the table stays stationary. *Bottom* The DiffusionAsShader renderer is able to more accurately render the cube and its show as it slide off the table and to the ground, but occasionally is unfaithful to the tracking video. *Right* The 3D trajectory predicted by our model for the cube. The model accurately models the cube sliding, falling, then coming to a rest on the ground.

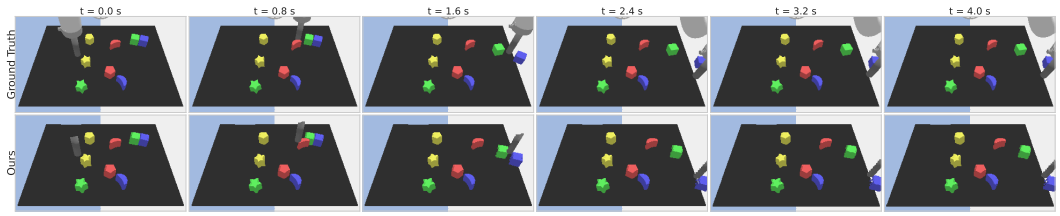


Figure 8: **Additional Language-Table qualitative result.** Even in the presence 8 blocks, our model accurately models motion: the robot arm passes closely between two yellow blocks without moving them, then pushed the green and blue ones.

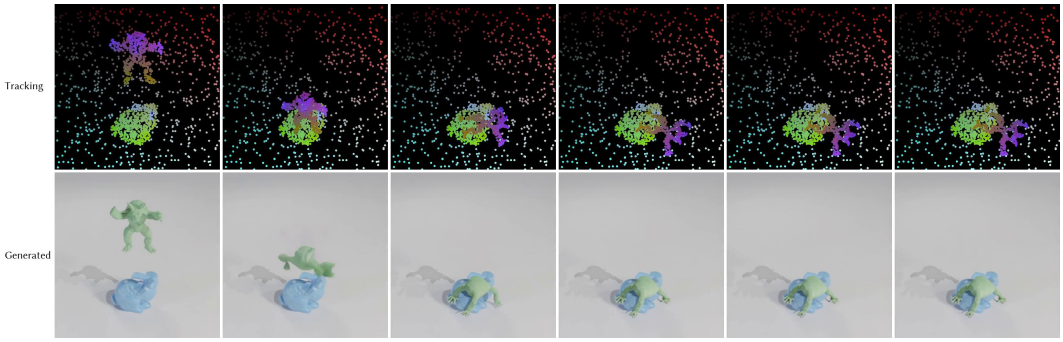


Figure 9: **Rendering Limitations**. In this scene with the Stanford Bunny and Stanford Armadillo, also our model predicts a physically plausible trajectory for the collision (as seen in the tracking video where the armadillo collides with the bunny and slides off to the side), DiffusionAsShader Gu et al. (2025) turns the armadillo into a soft object and keeps it directly on top of the bunny despite the prompt detailing that both are rigid objects.

in all simulations between 1-5 kg. All data was collected at 24 FPS and 512x512 resolution. We generated 1,500 samples for each scenario, resulting in 6,000 total generated samples. These were split 80/20 into training and test sets. Code to setup these scenes and generate data will be released.

C.2 Physion Data

We additionally selected four Physion scenarios which we included in our dataset. We chose Support, Contain, Drop, and Collide since they all: only use rigid bodies, include gravity, collision, and friction, and contain multiple objects. We take 1,000 samples from each category, for 4,000 total from Physion. We split the samples into train/val sets with an 80/20 split. Depth and segmentation were provided by the Physion simulator, as were the physics properties.

D Implementation Details

In this section, we provide details of our training process and additional details of our model architecture.

D.1 Model Architecture

As described in the main paper, the Trajectory Transformer is a six-layer transformer with hidden size 256. During the forward pass, we alternate the attention masking scheme at each layer: even layers perform causal attention across objects, allowing for attention within each timestep group but masking out across timesteps, while odd layers allow each object token to attend only to the same objects past tokens and time separators.

We also provide a detailed description of the sigmoid attention layer. Traditional attention can be written as follows:

$$\text{Attention(Q, K, V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

Our sigmoid attention block can instead be written as:

$$\text{SigAttention(Q, K, V)} = (\sigma(\frac{QK^T}{\sqrt{d_k}}) \odot G)V \tag{3}$$

The sigmoid operation replaces the softmax, and the output of the attention weights is also element-wise multiplied by G. G helps represent contact forces and is calculated pair-wise using an MLP on relative distance and rotation between pairs of objects.

D.2 TRAINING

All models were implemented using PyTorch Paszke et al. (2019) and trained on AMD Instinct™ MI210 or AMD Instinct™ MI300 GPUs. We used the Adam optimizer Kingma & Ba (2014) with a learning rate starting at 1e-4 with cosine decay to 1e-6 over the course of training. We used a batch size of 32 with 1000 warmup steps and trained for 200 epochs total. We additionally randomly translated/rotated our scene to increase the variation in data and help the model learn invariance to absolute positions/rotations.