PHYSIMCODE: A BENCHMARK AND EVALUATION METHOD FOR PHYSICS VIDEO TO CODE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) are good at video understanding and code generation, but it remains unclear whether they can recover the physical laws underlying videos, a capability essential for converting videos into executable simulations and for providing structured physical supervision in robotics. Captioning motion video does not imply that a model has inferred its governing dynamics, estimated quantitative parameters, or can regenerate the observed scene from code. Existing video–physics benchmarks evaluate plausibility, property estimation, or future-state prediction, but offer limited insight into whether models can translate visual dynamics into runnable simulation code. We introduce **PhySimCode**, a benchmark for physics video-to-code generation: given only a raw video, a model must infer the relevant physics, recover parameter values, and generate runnable Python that reproduces the input video. The dataset contains **160.6K procedurally generated samples** spanning **162 physics phenomena** across analytic SciPy simulations and PyBullet-rendered rigid-body, articulated, fluid, and contact-rich scenes; each sample bundles the source simulation code, sampled physical parameters, rendered video, and a structured chain-of-thought covering observations, laws, ODEs, numerical methods, and validation checks. We evaluate ten state-of-the-art closed- and open-source MLLMs along physical-law correctness, simulation equivalence, code executability, and parameter recovery. Models reliably state physics laws (4.27–4.93/5) yet score only 1.51–2.47/5 on inferring them from video, and the best system (Claude Sonnet 4.6) recovers just 7.7% of ground-truth parameters within $\pm 20\%$, exposing a substantial video-to-simulation gap and positioning PhySimCode as a testbed for multimodal systems that translate perception into precise, executable, and scientifically verifiable physics. Dataset and Code for generation and evaluation are released.

1 INTRODUCTION

State-of-the-art multimodal large language models (MLLMs) increasingly perform well in image/video understanding and coding tasks, but do they fully understand what they see (like humans)? Think of video captioning; these models' internal representations of visual inputs limit precision and verifiability: raw pixel embeddings are dense but opaque, and natural-language captions are readable but lossy. For *structured, information-dense, scientific* visuals: infographics and physics-simulation videos, captions fail to recover the dense numerical values or the spatiotemporal dynamics that a domain expert would immediately read off the input. Code, by contrast, is simultaneously human-readable, machine-executable, structured, and lossless with respect to such scientific visual input.

Recent work targets scientific chart-to-code generation (Tang et al., 2026; Zhao et al., 2025a;b; Wu et al., 2025). An infographic's plotting code has two parts: (i) the **template**: chart type, colors, labels, legends, and (ii) the **data values** displayed. In a pie chart of monthly tourist counts, the template is the chart design (slices, colors, labels) and the values are the per-month counts; many code variants can render the same chart, but the template specs and the values are unique to that infographic.

A structurally analogous problem exists in the video domain: physics-simulation video-to-code generation. A physics-simulation video is the video counterpart of a structured, information-dense scientific image, and the same template/value decomposition applies: different libraries, physics

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.

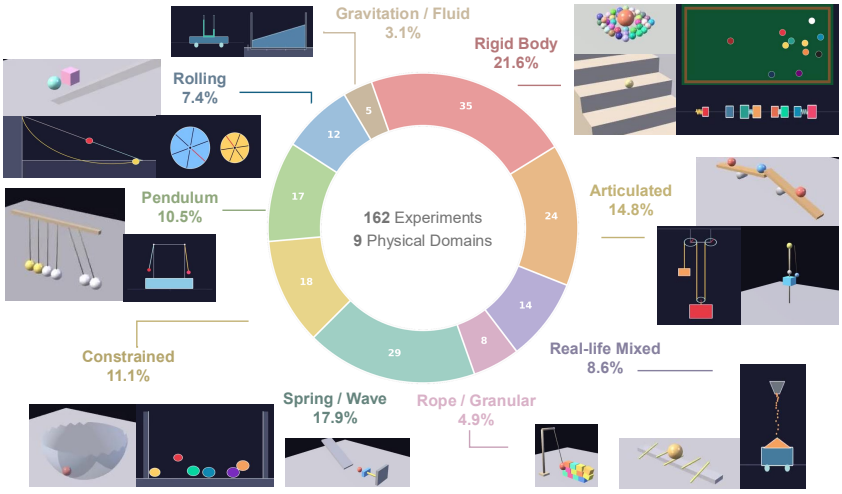


Figure 1: Categorical distribution of physical domains with some Scipy (2D) and Kubric (3D) examples from our dataset.

engines, or procedural languages (C#, Python) can produce identical dynamics, but the underlying physical laws and the values of the physical parameters (mass, length, elasticity, etc.) are unique. For example, a spinning disk shedding cubes from its rim involves rigid-body rotation, centripetal acceleration, the static-to-kinetic friction transition, and projectile motion under gravity. Because the same phenomenon can be expressed in different formulations (e.g. Newtonian or Lagrangian mechanics), we define the template at the level of the universal underlying physical concepts, such as conservation of energy or momentum.

Once the template and parameter values are defined, we can test whether SOTA MLLMs identify both directly from the video: a direct probe of their video understanding and information-extraction capability. A stronger probe of an MLLM's *true* physics understanding is to use those extracted concepts and values in a downstream task: generate simulation code that regenerates the exact same video. Because humans are naturally very sensitive to physical plausibility, executing the code and visually comparing the regenerated simulation to the original provides an intuitive and reliable evaluation: if the code is correct it serves as a lossless executable representation of the input video. Such physics-aware symbolic simulations of 2D/3D scenes are also valuable for teaching robots motion planning and the physics of the real world.

The task measures MLLM capability at three stages: (i) **video understanding**: extracting all visual features and tracking objects across video frames; (ii) **physics understanding and physics knowledge base**: reasoning on the extracted features (what it *sees*) and retrieving representative physics laws from pretraining (what it *knows*) to identify the underlying laws and estimate physical-parameter values; and (iii) **programming expertise**: synthesizing simulation code that handles boundary conditions and numerical-stability constraints, and regenerates physically and perceptually *identical*, *realistic* videos. Parameter estimation is a long-studied inverse problem traditionally addressed by physics-informed neural networks (Raissi et al., 2019); we test whether MLLMs can solve it zero-shot (Kojima et al., 2022). This is not a retrieval or memorization task: every sample in our dataset uses randomly sampled parameters drawn from physically plausible ranges, so the model must infer numerical values from visual dynamics.

How good are the best MLLMs at this challenging inverse problem? Our evaluation shows that only a fraction of the generated code is executable or produces a video, and even when a video is generated the object interactions are often physically implausible or differ significantly from the input. Because our method explicitly probes the model's reasoning, we further observe that only a small fraction of cases correctly identify the underlying physical law and an even smaller fraction estimate the parameters within reasonable tolerance. While MLLMs have been extensively evaluated on inverse code generation for scientific *images*, comparatively little work addresses scientific *videos*, and this lack of comprehensive benchmarks has hindered progress in physically plausible simulation-video-to-code generation. Motivated by this gap, we construct PhySimCode: a

Table 1: **Positioning against related physics and video benchmarks. Video**: evaluation grounded in video modality; **Text**: natural-language prompts; **CoT**: reasoning or chain-of-thought annotations; **Code**: executable simulation code as model output or dataset ground truth; **Physics Laws**: explicit governing laws in annotations or scoring; **Engine Agnostic**: the core benchmark pipeline spans multiple distinct simulation paradigms (such as analytic ODE-based and rigid-body-engine-based); **2D and 3D**: the benchmark spans both 2D and 3D visual content; **# Exp.**: number of phenomena or experiment types; **Size**: total count of entries or videos reported in the source paper.

	Grounding signals				Dataset properties			Dataset statistics	
Benchmark	**Video**	**Text**	**CoT**	**Code**	**Physics laws**	**Engine agnostic**	**2D+3D**	**# Exp.**	**Size**
Physion (Bear et al., 2021)	✓	✗	✗	✗	✗	✗	✗	8	∼24K
IntPhys (Riochet et al., 2018)	✓	✗	✗	✗	✗	✗	✗	–	15K
CLEVRER (Yi et al., 2019)	✓	✓	✗	✗	✗	✗	✗	–	20K
ComPhy (Chen et al., 2022)	✓	✓	✗	✗	✗	✗	✗	–	∼100K
ContPhy (Zheng et al., 2024)	✓	✓	✗	✗	✗	✗	✗	–	6.5K
PHYBench (Qiu et al., 2026)	✗	✓	✗	✗	✓	✗	✗	–	500
PhysBench (Chow et al., 2025)	✓	✓	✗	✗	✗	✗	✗	–	10,002
VideoPhy (Bansal et al., 2025)	✗	✓	✗	✗	✗	✗	✗	–	11,330
VBench (Huang et al., 2024)	✓	✓	✗	✗	✗	✗	✗	8	≥800
Morpheus (Zhang et al., 2025)	✓	✓	✗	✗	✓	✗	✗	9	130
PhysInOne (Zhou et al., 2026)	✓	✓	✗	✗	✗	✓	✗	71	2M
PhysCodeBench (Xie et al., 2026)	✗	✓	✗	✓	✓	✗	✗	–	700
SimuScene (Wang et al., 2026)	✗	✓	✓	✓	✓	✗	✗	52	7659
PhysVid (Zhan et al., 2026)	✓	✗	✗	✗	✗	✗	✗	–	12.1K
PhySimCode (Ours)	✓	✓	✓	✓	✓	✓	✓	162	160.7K

benchmark of 160.6K samples in which each sample is a tuple of a physics-simulation video, the corresponding simulation code, a list of physical parameters and their values, and a detailed chain-of-thought (CoT) explanation (Wei et al., 2022) describing the underlying physics using any valid formulation (including mixed Newtonian and Lagrangian approaches).

Our key contributions are:

(1) PhySimCode: the first comprehensive benchmark that combines physics-simulation video, executable code, parameter values, and physics explanations, making it possible to evaluate law identification, parameter recovery, and runnable simulation synthesis from video in a single framework.

(2) Benchmarking insights: extensive evaluation across a diverse set of closed- and open-source SOTA MLLMs, revealing their strengths and weaknesses across each aspect of the task.

(3) Comprehensive evaluation strategy: a holistic protocol combining verifiable metrics, LLM-based judges, and human assessment to measure different capabilities of MLLMs.

2 Related Works

We review the closest lines of work and identify the gap addressed by our video-to-simulation-code benchmark. Additional related-work details are provided in Appendix A.7.

Physical Law Recovery from Visual Dynamics: Prior work has studied physical property estimation, law discovery, and visual intuitive physics from images and videos. Early systems such as Galileo (Wu et al., 2015) and Physics 101 (Wu et al., 2016) combined physics engines with learning-based perception to infer object properties from video, while later work explored block-tower stability (Lerer et al., 2016), graph-based dynamics learning (Battaglia et al., 2016; Watters et al., 2017), and inverse-graphics formulations for recovering physical parameters (Jaques et al., 2019; Zhao et al., 2025c). Visual Physics (Chari et al., 2019) more directly targets physical law discovery from video trajectories. However, these approaches typically recover latent physical properties, equations, trajectories, or implicit per-particle dynamics, rather than producing a self-contained executable program from raw video. In contrast, our benchmark evaluates whether MLLMs can infer governing physical concepts and parameters from pixel-level video and also synthesize runnable simulation code that reproduces the observed dynamics.

Benchmarks and Simulators for Physical Reasoning: A complementary line of work benchmarks physical understanding in generative and reasoning models. PhysInOne (Zhou et al., 2026),

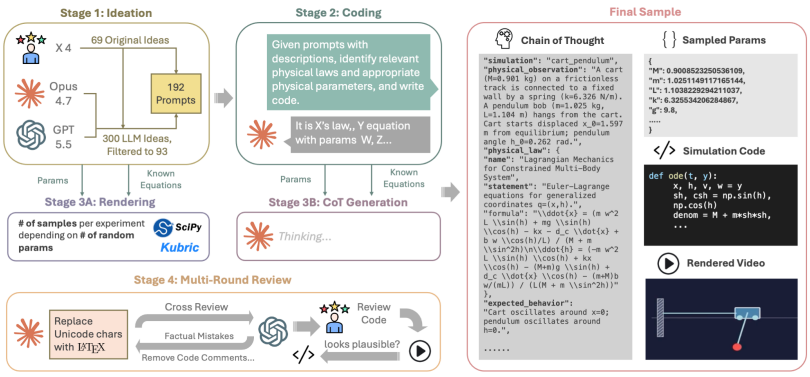


Figure 2: The dataset curation pipeline.

VideoPhy (Bansal et al., 2025), Morpheus (Zhang et al., 2025), VBench (Huang et al., 2024), PhysBench (Chow et al., 2025), PhyBench (Qiu et al., 2026), Physion (Bear et al., 2021), IntPhys (Riochet et al., 2018), CLEVRER (Yi et al., 2019), and ComPhy (Chen et al., 2022) evaluate physical plausibility, commonsense reasoning, invariant preservation, property estimation, or future-state prediction. These benchmarks probe whether models recognize, judge, or predict physical behavior, but they do not require recovering an executable simulation whose trajectory can be compared against the original video at the level of laws, parameters, and code. Related work also uses simulators for physics reasoning and learning, like simulator-generated QA for physics problem solving (Prabhudesai et al., 2026), and the popular simulators are MuJoCo (Todorov et al., 2012), DiffTaichi (Hu et al., 2019a), Brax (Freeman et al., 2021), Genesis (Authors, 2024), Isaac Gym (Makoviychuk et al., 2021), and NeuralSim (Heiden et al., 2021). The closest benchmark, PhysCodeBench (Xie et al., 2026), generates code for a single physics engine from text inputs, whereas ours spans multiple engines, uses standard Python with engine-specific starter code (testing physics understanding over engine familiarity), evaluates MLLMs' visual capabilities from video inputs, and is substantially larger (160.6K vs. 700 samples) with extensive regenerated-video-based physics evaluation that PhysCodeBench lacks. Our work thus complements prior benchmarks by focusing on scalable, visually grounded physics-understanding evaluation.

3 Dataset Overview

3.1 Dataset Curation Process

Dataset construction proceeds through two complementary pipelines that together cover 162 distinct physics phenomena (Figure 2). Regardless of the pipeline, every sample comprises a Python simulation script, the rendered simulation video, a structured CoT annotation, and the underlying parameter values. We use Python as the common implementation language; it is widely represented in LLM training data, so models tend to generate it well, relying on solver libraries such as SciPy (Virtanen et al., 2020) or physics engines accessible through Python wrappers such as PyBullet (Coumans & Bai, 2016). The pipeline is not engine-specific: it can be extended to any physics engine with a Python wrapper and can generate an indefinite number of samples by randomizing all physical parameters.

Four human experts first curated 69 physics-simulation ideas from their own knowledge, then prompted Claude Opus 4.7 and GPT 5.5 to propose additional candidates; from the 300 generated scenes the experts filtered and refined 93 based on implementability and clarity of the intended physics laws, yielding the 162 experiments, accompanied by 3–4 line event descriptions written entirely by the experts. These descriptions were used to prompt Claude Opus 4.7 to generate the Python simulation code, extract the relevant physics laws, and identify the physical parameters appearing in the equations from its pretrained knowledge. The generated code was executed and the resulting videos iteratively refined through 2–10 rounds of human-in-the-loop correction until physically plausible. Finally, the physics knowledge and sampled parameters were recorded in a CoT follow-

ing the unified 13-key schema of Table 15, with parameter values drawn uniformly from physically plausible ranges to produce multiple samples per experiment.

Pipeline A produces 2D simulation data for 87 experiments entirely through standard Python libraries (SciPy, Math). Each module is structured around a parameterised dataclass, a numerical ODE solver, a renderer, and a code exporter, and is inspired by implementations from MyPhysicsLab[1], a widely used educational physics-simulation library.

Pipeline B (denoted "Kubric" throughout, after the 3D rigid-body domain it targets, but implemented with PyBullet + pyrender + trimesh because the Kubric[2] package itself requires Blender + Docker) covers 75 3D experiments needing contact-aware articulated and rigid-body dynamics beyond analytic ODE formulations. Each experiment is a self-contained module built on a shared infrastructure that exposes a uniform API for primitive meshes, joints, constraints, ropes, and granular media, and is structured around a parameterised dataclass, a parameter sampler that draws physically plausible values per seed, and a scene constructor.

Ground Truth Debiasing: Because the ground-truth code and CoT were initially generated by Claude and may carry Claude-specific stylistic biases or factual errors, we ran a three-round LLM-assisted debiasing pipeline followed by human verification. Round 1: Claude Opus 4.7 removed Unicode symbols from equations and replaced them with LaTeX and Python forms that are more suitable for MLLM training. Rounds 2-3: GPT-5.5 corrected factual and physics-related errors, fixed inconsistencies between LaTeX and Python equations, and stripped code comments to eliminate Claude-specific coding-style bias during code-similarity evaluation, making $14{,}700$ substitutions for a total Levenshtein distance of $693{,}600$ characters. Finally, four human experts manually verified the CoTs and applied additional corrections, yielding a ground truth that is both reliable and debiased. To further ensure fair evaluation (Section 4.1), we place limited emphasis on code similarity; many different implementations can produce the same simulation, and instead focus on simulation equivalence and the physical plausibility of the regenerated videos. We discuss more rigorous evaluations proving removal of Claude authorship bias in Section A.2.

3.2 Dataset Statistics

Scale. The 162 categories are unevenly split across the two pipelines because each pipeline targets a different physical regime, and the sample count per category scales with the number of independently randomized parameters. Pipeline A's 87 SciPy categories are analytically formulated and inexpensive to integrate, so we allocate $100 \times n_{\text{random_params}}$ samples per category, ranging from 200 (low-DOF systems such as `brachistochrone_race`) to 2,500 for seed-driven experiments that have high latent dimensionality (`complex_collision_plane`: 2–6 randomly-typed objects with per-object mass, size, and initial conditions). Pipeline B's 75 Kubric categories require GPU rendering and benefit from a wider visual diversity, so we apply the same $100 \times n_{\text{random_params}}$ samples per category, then $+50\%$ if the scene has a variable object count, then $+30\%$ if it has color/material randomization, floored at 400 (e.g. `billiard_break_3d`), capped at 5,000 (e.g. `multi_physics_arena`). There are **64,800** SciPy samples (mean 745, median 600 per category) and **95,855** Kubric samples (mean 1,278, median 900 per category) for a dataset total of **160,655** video–code–CoT–parameter tuples (more statistical details in Section A.12).

Distribution by Physical Domain and Law. We cover diverse experiments across rigid mechanics, waves and oscillations, and some fluid and magnetic phenomena. Figure 1 shows the counts of the 9 main categories of physical domains covered across the 162 experiments (more about these 9 physical domains in Section A.12) and some dataset examples from each category. Appendix Tables 18 and 19 list all experiments with their random-parameter and sample counts. Figure 3 shows that Kubric's 95,855 samples are fixed at 480×480 over 5.0s median duration (range 1.1–6.0s) with tight CoTs (median 428 words, range 295–848) and uniformly templated code (median 156 lines, range 135–232), whereas SciPy's 64,800 samples span wider 480×360 to 1560×1080 resolutions and longer 12.0s median durations (range 0.2–12.0s) with longer-tailed CoTs (median 452 words, up to 1,662) and broader code lengths (median 119, range 0–461).

[1] `https://www.myphysicslab.com/`

[2] `https://kubric.readthedocs.io/en/latest/`

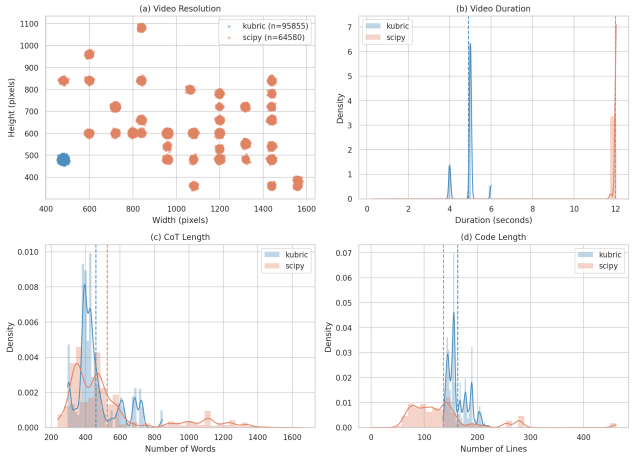


Figure 3: Per-sample distributions of (a) video resolution, (b) duration, (c) CoT length, and (d) code length across Kubric and SciPy. Dashed lines mark per-pipeline means.

4 METHODOLOGY

Task definition. The task is to watch a video of a physical process and infer the underlying physical law and equations, estimate key parameters, and recreate a working simulation. Because an MLLM may not be well pretrained on the variety of physics engines available, we provide the engine name and an experiment-agnostic code template as a starting point, ensuring we evaluate the model's understanding of physics rather than its familiarity with a specific engine. Formally, given only a rendered video, a model must produce (i) a structured chain-of-thought JSON object following the engine-conditional 4-key schema, a strict subset of the 13-key ground-truth schema (Table 15), and (ii) executable Python code that simulates the observed process.

4.1 EVALUATION METRICS

We want to measure the three capabilities introduced in Section 1: video understanding, physics understanding, and programming abilities of MLLMs. We explain the correspondence of our proposed evaluation metrics with the measured capabilities in Table 2. Video understanding is the prerequisite capability that every downstream metric passes through. It is observable through video appearance similarity; a very low video-similarity score implies the model failed to read the scene at all.

Physics understanding rests on (i) the model's prior knowledge of physical laws and (ii) its ability to infer the laws demonstrated in a given video; to disentangle the two, we (a) judge the **absolute** correctness of the stated laws regardless of the video to test the model's prior physics knowledge, and (b) compare the laws in each model's generated CoT against those in the dataset's reference CoT. Both are rated on a 1–5 Likert scale (5 = best match, 1 = lowest) averaged over three LLM judges: Gemini-2.5-flash, Grok-4.1-Fast, and GPT-5-Nano. Apart from evaluating the physics laws, we also verify parameter recovery. We extract the ground-truth physical parameters from the reference CoT (excluding scene metadata, geometry, color, and the gravitational constant) and the predicted parameters from each model's emitted CoT, pair predicted parameter names to ground-truth names by greedy bipartite matching over Jaccard similarity on alias sets that capture lexical and Greek-letter abbreviations, normalize units, and compute the per-pair relative error $|\hat{x} - x|/|x|$ (absolute error as fallback when $|x|$ is below numerical-zero tolerance); a parameter is counted as correctly estimated if its relative error is within $\pm 20\%$.

For coding and instruction following we compute verifiable metrics: code executability, syntax correctness, adherence to the CoT-then-code output format, and whether the code successfully writes the video MP4, together with CodeBLEU against the reference code. Finally, we measure end-to-end capabilities from the rendered videos by executing the generated code through (i) standalone physical plausibility, (ii) physics equivalence (same physical phenomenon as the ground truth). For each model, four human experts independently rate the same 100 random samples on (i) physical plausibility, (ii) physics equivalence, and (iii) video appearance similarity (same object shape, count,

Table 2: How different stages of capabilities of MLLMs are measured through all evaluation metrics.

Stage	Evaluation / Measurement
(i) Video understanding	Appearance equivalence (human), DINOv2/VideoCLIP (Table 4)
(ii) Physics understanding	Law correctness & equivalence (Table 3); parameter recovery (Tables 5, 9)
(iii) Programming	Compilation; execution; video generation; CodeBLEU (Table 5)
End-to-end	*Physical plausibility & equivalence (Table 4)*

color, etc.); we report the mean over the four raters and inter-rater Cohen's κ in Figure 4. Metric (iii) reveals whether models lean toward replicating appearance over physics. For scalability, we also report appearance similarity directly via the patch-wise video-similarity metrics DINOv2 and VideoCLIP. In Table 2, stages (i) and (iii) are largely solved; stage (ii) is the bottleneck, and that is the claim PhySimCode is built to isolate.

4.2 Benchmarking Details

Our generated dataset contains 160,655 samples spanning 162 distinct physics experiments (87 SciPy, 75 Kubric); within an experiment, samples differ only in random initial conditions and parameter values, while the underlying physical concept and required code structure remain fixed. We therefore treat each experiment as a natural unit of analysis and evaluate every model on $n = 15$ uniformly random samples per experiment under a single seed ($s = 42$), yielding $162 \times 15 = 2{,}430$ evaluations per model, a $\sim$66$\times$ reduction over the full corpus that we accept on statistical grounds (see Section A.3), so the relative rankings established on the 2,430-sample subset are statistically supported and very likely to hold on the full 160,655-sample corpus, with reduced API cost.

Models evaluated: On our 2,430-sample benchmarking dataset, we benchmark 10 MLLMs, five closed-source models: Claude Sonnet 4.6, GPT-5-mini, Grok-4-fast-reasoning, Gemini 2.5 Pro, and Amazon Nova 2 Lite, and five open-source models: InternVL3.5-30B-A3B, Qwen3-VL-30B-A3B, Pixtral-12B, Gemma3-12B-Instruct, and GLM-4.1V-9B-Thinking. All 10 models receive the *same* engine-conditioned prompt (Appendix Section A.4). More benchmarking results with GPT-6-Astra, Claude Fable-5.1, Kimi K3, and Grok-4.6 on 10 handpicked hard samples are in Section A.1.

Input format and decoding: For frame-based models (Claude, GPT-5-mini, Grok-4, and all five open-source models), we decode every frame at 10 FPS into a list of JPEG-encoded 360×180 (`scipy`) or 240×240 (`kubric`) RGB images and pass them as multi-image messages alongside the engine name. Gemini 2.5 Pro instead consumes the raw `.mp4` at the same FPS and resolution as others, since it natively supports video. We use a context length of $32{,}768$ tokens for the open-source models (49,152 for Pixtral, which has larger per-frame token cost), greedy decoding where supported, and per-model output-token budgets are 8K (Qwen3-VL, Pixtral, InternVl-3.5, Gemma-3), 9K (GPT, Claude, Nova), 12K (GLM-Thinking, Grok), and 16K (Gemini, to accommodate hidden-thinking tokens). Closed-source inference runs at concurrency 200 against the providers' endpoints, open-source inference runs as eight parallel vLLM shards on $8\times$NVIDIA H200 (144 GB HBM each) with `bfloat16` compute and prefix caching enabled.

5 Results and Discussions

We evaluate models along four axes: (i) extracted physical-law correctness and match with ground truth (Table 3); (ii) video-simulation equivalence by human judgment and verifiable patch-wise similarity (Table 4); (iii) parameter-estimation correctness; and (iv) code correctness and similarity (both Table 5). Claude leads only on parameter estimation (with GPT-5-mini close), suggesting our code-style debiasing pipeline removes the most direct Claude-specific advantage (coding).

Physical Law Understanding: Both open- and closed-source models struggle to infer physics from video (Table 3). Nearly every model scores >4.5/5 on stating physical laws and formulae, indicating decent *prior* physics knowledge, but this fails to translate into recognizing the laws actually demonstrated in a video, the best model, GPT-5 Mini, scores only 2.47/5, and closed-source systems average just 2.15 vs. 1.83 for open-source on this harder task.

Table 3: **Average Likert scores** (on a 1–5 scale) rated by 3 LLM judge models (Gemini-2.5-flash, Grok-4.1-Fast, GPT-5-Nano) for model outputs across physical law correctness and equivalence.

Model	Physical Law Correctness		Physical Law Equivalence			
	Formula	Overall	Name	Statement	Formula	Overall
Open-source models						
QWEN3-VL-30B-A3B-INSTRUCT	4.58	4.68	**2.18**	**2.09**	**1.92**	**2.02**
INTERNVL3.5-30B-A3B	4.68	4.64	1.98	1.96	1.75	1.82
GEMMA-3-12B-IT	**4.90**	**4.93**	2.15	1.95	1.72	1.90
GLM-4.1V-9B-THINKING	4.46	4.43	2.14	2.03	1.80	1.91
PIXTRAL-12B-2409	4.43	4.50	1.70	1.57	1.45	1.51
Closed-source models						
GPT-5 MINI	4.75	**4.70**	**2.71**	**2.69**	**2.24**	**2.47**
GEMINI 2.5 PRO	**4.76**	4.66	2.49	2.48	2.16	2.34
CLAUDE SONNET 4.6	4.53	4.33	2.58	2.45	2.09	2.26
GROK 4 FAST REASONING	4.65	4.58	2.16	2.16	1.85	1.98
NOVA 2 LITE	4.29	4.27	1.80	1.87	1.59	1.69

Table 4: **Video simulation equivalence table** containing human judgment of Physical Plausibility, Video's Physics Equivalence, Appearance Equivalence, and verifiable video frame similarity scores using DINOv2 and VideoCLIP. Human scores on 1–5 Likert (on per model 100 random samples).

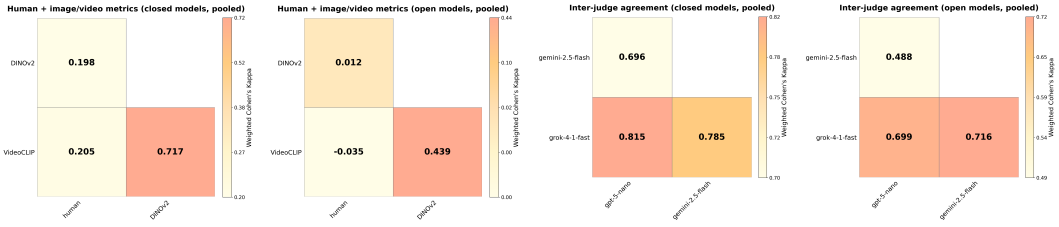
Model	Human Evaluation			Patchwise Video Similarity	
	Physical Plausibility	Physics Equivalence	Appearance Equivalence	DINOv2 sim.	VideoCLIP sim.
Open-source models					
QWEN3-VL-30B-A3B-INSTRUCT	**3.16**	**1.88**	2.31	**0.557**	0.713
INTERNVL3.5-30B-A3B	1.43	1.23	1.59	0.482	0.667
GEMMA-3-12B-IT	2.64	1.80	**2.56**	0.447	0.643
GLM-4.1V-9B-THINKING	2.43	1.32	2.04	0.551	**0.732**
PIXTRAL-12B-2409	1.48	1.13	1.26	0.448	0.638
Closed-source models					
GPT-5 MINI	3.40	2.40	2.66	0.672	0.785
GEMINI 2.5 PRO	**3.96**	**2.86**	**3.45**	**0.774**	**0.838**
CLAUDE SONNET 4.6	3.30	2.30	3.14	0.767	0.837
GROK 4 FAST REASONING	1.90	1.65	2.28	0.629	0.750
NOVA 2 LITE	2.62	1.14	1.63	0.480	0.687

Simulation Equivalence: Table 4 reveals a consistent open-vs.-closed gap on video-simulation equivalence (human rating guidelines in Section A.5), with closed models stronger on both human judgment and embedding-based similarity. GEMINI 2.5 PRO ranks highest overall, leading on human-rated Physical Plausibility, Physics Equivalence, and Appearance Equivalence as well as DINOv2 and VideoCLIP similarity; despite this, Table 5 shows Gemini lagging other closed models on code executability, which we attribute to its long thinking trace consuming most output tokens and truncating the code. We observe that across all models the patchwise video similarity metrics are far more lenient than human evaluations of physics equivalence or appearance equivalence of videos since these patchwise metrics are driven by pixel-level semantics, while humans are very sensitive to any deviation from reality. Across all models, multimodal systems are considerably better at preserving visual appearance than at modeling or verifying physical behavior: Figure 4 shows that DINOv2 and VideoCLIP correlate highly with each other but not with human raters, who judge physics equivalence (whether the regenerated video exhibits the same physics laws as the GT); the figure also shows the maximum pairwise correlation between any two judges is < 0.85, so our judge set is not highly correlated and is reliable (model-specific bias is eliminated).

Parameter Estimation Correctness: Table 5 reports the fraction of all ground-truth physical parameters each model both names correctly *and* estimates within $\pm 20\%$, symmetrically penalizing naming failures and value errors. Appendix Table 9 decomposes this into naming recall and value accuracy: across the 10 MLLMs, CLAUDE-SONNET-4.6 and GPT-5-MINI are strongest at naming the right parameters, while GLM-THINKING achieves the highest per-pair accuracy on the (small) set of parameters it actually names. Claude names 70% more parameters per sample at near-equal per-pair accuracy (18.5%), yet recovers only 7.7% of all ground-truth parameters within tolerance, indicating that quantitative physics-from-video estimation is substantially harder than qualitative scene description.

Table 5: **Evaluation of parameter estimation accuracy and generated simulation code quality**. Percentages computed as a fraction of all N=2430 inputs that successfully reached each stage (CoT extracted, Code extracted, Compiles, Runs without error, Saves video).

Model	Param. Estimation Acc. (±20%)	Code Generation and Execution					
		CoT	Code	Compile	No Errors	Video Save	CodeBLEU
Open-source models							
QWEN3-VL-30B-A3B-INSTRUCT	**6.16%**	99.8%	96.8%	96.6%	73.8%	77.9%	0.206
INTERNVL3.5-30B-A3B	3.52%	99.8%	99.7%	94.0%	48.0%	62.5%	0.195
GEMMA-3-12B-IT	3.98%	99.0%	95.4%	95.2%	39.6%	47.7%	0.200
GLM-4.1V-9B-THINKING	4.84%	96.6%	87.6%	87.4%	46.6%	48.0%	0.198
PIXTRAL-12B-2409	3.31%	99.8%	99.7%	99.0%	42.6%	60.7%	0.187
Closed-source models							
GPT-5 MINI	7.13%	83.7%	98.6%	98.2%	92.8%	93.3%	0.241
GEMINI 2.5 PRO	6.06%	96.1%	66.2%	66.1%	58.7%	59.2%	0.223
CLAUDE SONNET 4.6	**7.69%**	100.0%	99.9%	99.8%	98.8%	99.0%	0.232
GROK 4 FAST REASONING	5.69%	98.9%	100.0%	99.6%	89.4%	90.4%	0.209
NOVA 2 LITE	3.76%	97.0%	99.0%	98.5%	75.9%	78.7%	0.204



(a) (b) (c) (d)

Figure 4: Inter-judge Cohen's κ plots averaged over (a) closed and (b) open-source models, and Cohen's κ plots for alignment between human-rated physics equivalence of videos and verifiable patchwise similarity metrics (with DinoV2 and VideoCLIP) averaged over (c) closed and (d) open-source models.

Code Correctness: Table 5 shows that almost every model generates the CoT and syntactically correct code for almost all samples, the only exception being Gemini, which fails to emit code in $>30\%$ of samples. However, open-source code often does not execute: averaged across the five open models only 50.1% runs to completion without errors and 59.4% writes a video file, versus an average closed-source success rate of 83.1%, with CLAUDE SONNET 4.6 the most reliable overall (98.8% error-free execution, 99.0% video-write). CodeBLEU is similar across models and we caution it is partially unreliable: the same physics phenomenon admits different code.

6 CONCLUSION

We presented PhySimCode, a procedurally generated benchmark of 160.6K (video, simulation code, chain-of-thought, parameters) tuples spanning 162 experiments across two simulation engines, paired with an LLM-judge, human-rater, and verifiable scoring suite for holistic MLLM physics-understanding evaluation. Evaluating ten frontier MLLMs reveals four consistent gaps: models can state physics laws ($>$4.5/5 Likert) but struggle to infer them from video (best 2.47/5, GPT-5 Mini); closed-source models dominate qualitative simulation equivalence, with Gemini 2.5 Pro leading visual-physics ratings and DINOv2/VideoCLIP similarity and Claude Sonnet 4.6 leading code-execution reliability (98.8%). End-to-end parameter recovery is severely limited, with the best model (Claude Sonnet 4.6) recovering only $\sim$7.7% of ground-truth parameters within $\pm20\%$; and video appearance reproduction is far easier than physics reproduction. Two limitations remain: (i) rigid-body dynamics dominate the benchmark, leaving only a small fraction of fluid/deformable scenarios that an MPM-based engine such as Genesis would naturally extend and (ii) these benchmark videos are simulator-generated, so success on this benchmark doesn't prove success in real-world noisy videos. However, as discussed in A.16, we believe that our benchmark reports failure in the absence of nuisance factors isolating the cause to MLLM capabilities. We hope the released dataset, generators, prompts, inference scripts, and evaluation pipeline accelerate community progress on physics-grounded video understanding.

AI USE STATEMENT

In this work, we used generative AI tools for synthetically generating the dataset with human supervision as discussed in Section 3, finding a few more related work to ensure we did not miss any similar recently released work from the literature, and polishing the writing. We have not used generative AI tools for proposing research ideas and executing them, and draft sections of the paper from scratch entirely, and proving mathematical claim is not applicable to this work. Additionally, we used generative AI tools for benchmarking them on the dataset. We have reviewed all AI-assisted work. For example, we checked LLM-generated dataset for any error in Physics concepts or equations in the CoT through human verification, LLM-generated code in the dataset was verified and tested for correctness in terms of physical plausibility by 4 experts. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All human rating for evaluation were IRB-approved and consented, there is no discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues with the dataset as the dataset contains LLM-generated and human reviewed physics simulation videos.

REPRODUCIBILITY STATEMENT

We have open-sourced the generator code to replicate synthetic procedural generation of unlimited size of dataset, the code for this new suite of evaluation, and the benchmark dataset itself at the end of the Abstract. The reproducible dataset curation process is mentioned in Section 3 and more details about inference, dataset statistics, prompts, rigorous proof of removal of authorship bias, evaluation on the latest frontier models, etc. are in the Appendix, properly referred throughout the main paper.

AUTHOR CONTRIBUTIONS

Will acknowledge author contributions after acceptance.

ACKNOWLEDGMENTS

Will acknowledge funding agencies after acceptance.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL `https://github.com/Genesis-Embodied-AI/genesis-world`.

Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. Videophy: Evaluating physical commonsense for video generation. In *International Conference on Learning Representations*, volume 2025, pp. 102075–102121, 2025.

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.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Leo Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gic: Gaussian-informed continuum for physical property identification and simulation. *Advances in Neural Information Processing Systems*, 37:75035–75063, 2024.

Junyi Cao, Shanyan Guan, Yanhao Ge, Wei Li, Xiaokang Yang, and Chao Ma. Neuma: Neural material adaptor for visual grounding of intrinsic dynamics. *Advances in Neural Information Processing Systems*, 37:65643–65669, 2024.

Pradyumna Chari, Chinmay Talegaonkar, Yunhao Ba, and Achuta Kadambi. Visual physics: Discovering physical laws from videos. *arXiv preprint arXiv:1911.11893*, 2019.

Zhenfang Chen, Kexin Yi, Yunzhu Li, Mingyu Ding, Antonio Torralba, Joshua B Tenenbaum, and Chuang Gan. Comphy: Compositional physical reasoning of objects and events from videos. *arXiv preprint arXiv:2205.01089*, 2022.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Campagnolo Guizilini, and Yue Wang. Physbench: Benchmarking and enhancing vision-language models for physical world understanding. In *International Conference on Learning Representations*, volume 2025, pp. 97959–98108, 2025.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning (2016–2021). *URL http://pybullet. org*, 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.

Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9474–9481. IEEE, 2021.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

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*, 2019a.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019b.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. *arXiv preprint arXiv:1905.11169*, 2019.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, pp. 1–52. 2016.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *International conference on machine learning*, pp. 430–438. PMLR, 2016.

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. *arXiv preprint arXiv:2303.05512*, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Joe J Monaghan. Smoothed particle hydrodynamics. *In: Annual review of astronomy and astrophysics. Vol. 30 (A93-25826 09-90), p. 543-574.*, 30:543–574, 1992.

Mihir Prabhudesai, Aryan Satpathy, Yangmin Li, Zheyang Qin, Nikash Bhardwaj, Amir Zadeh, Chuan Li, Katerina Fragkiadaki, and Deepak Pathak. Solving physics olympiad via reinforcement learning on physics simulators. *arXiv preprint arXiv:2604.11805*, 2026.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Zhang Haoxu, Yi Hu, et al. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *Advances in Neural Information Processing Systems*, 38, 2026.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Véronique Izard, and Emmanuel Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning. *arXiv preprint arXiv:1803.07616*, 2018.

Zitian Tang, Xu Zhang, Jianbo Yuan, Yang Zou, Varad Gunjal, Songyao Jiang, and Davide Modolo. Mm-recoder: Advancing chart-to-code generation with reinforcement learning and self-correction. *arXiv preprint arXiv:2604.01600*, 2026.

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.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

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.

Yanan Wang, Renxi Wang, Yongxin Wang, Xuezhi Liang, Fajri Koto, Timothy Baldwin, Xiaodan Liang, and Haonan Li. Training and benchmarking code generation for physics-inspired animations, 2026. URL https://arxiv.org/abs/2602.10840.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3006–3028, 2025.

Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. *Advances in neural information processing systems*, 28, 2015.

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.

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

Tianyidan Xie, Peiyu Wang, Hu Jiaxin, Yuyi Qian, Yuxuan Wang, Shenyi Wang, Rui Ma, Yanlun Peng, Lanjun Wang, Ying Tai, et al. Physcodebench: Benchmarking physics-aware symbolic simulation of 3d scenes via self-corrective multi-agent refinement. *arXiv preprint arXiv:2604.23580*, 2026.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

Guanqi Zhan, Xianzheng Ma, Weidi Xie, and Andrew Zisserman. Inferring dynamic physical properties from video foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5214–5224, 2026.

Chenyu Zhang, Daniil Cherniavskii, Antonios Tragoudaras, Antonios Vozikis, Thijmen Nijdam, Derck WE Prinzhorn, Mark Bodracska, Nicu Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Morpheus: Benchmarking physical reasoning of video generative models with real physical experiments. 2025.

Xuanle Zhao, Deyang Jiang, Zhixiong Zeng, Lei Chen, Haibo Qiu, Jing Huang, Yufeng Zhong, Liming Zheng, Yilin Cao, and Lin Ma. Vincicoder: Unifying multimodal code generation via coarse-to-fine visual reinforcement learning. *arXiv preprint arXiv:2511.00391*, 2025a.

Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Zhiyuan Liu, and Maosong Sun. Chartcoder: Advancing multimodal large language model for chart-to-code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7333–7348, 2025b.

Yizhou Zhao, Haoyu Chen, Chunjiang Liu, Zhenyang Li, Charles Herrmann, Junhwa Hur, Yinxiao Li, Ming-Hsuan Yang, Bhiksha Raj, and Min Xu. Toward material-agnostic system identification from videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5944–5956. IEEE, 2025c.

Zhicheng Zheng, Xin Yan, Zhenfang Chen, Jingzhou Wang, Qin Zhi Eddie Lim, Joshua B Tenenbaum, and Chuang Gan. Contphy: Continuum physical concept learning and reasoning from videos. *arXiv preprint arXiv:2402.06119*, 2024.

Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, pp. 407–423. Springer, 2024.

Siyuan Zhou, Hejun Wang, Hu Cheng, Jinxi Li, Dongsheng Wang, Junwei Jiang, Yixiao Jin, Jiayue Huang, Shiwei Mao, Shangjia Liu, et al. Physinone: Visual physics learning and reasoning in one suite. *arXiv preprint arXiv:2604.09415*, 2026.

A Appendix

A.1 Evaluating Latest Frontier Models

As the latest LLMs are solving previously unprecedented multimodal tasks better, we wanted to test our benchmark (which turned out to be hard for the models evaluated in the main paper) on the latest frontier models. Due to cost concerns, we evaluate four frontier models on a handpicked set of 10 hard samples from 5 SciPy experiments and 5 Kubric experiments to get a sense of how the latest frontier models stand against the 10 models evaluated in our benchmark. We test three closed-source

frontier models: Claude Fable 5.1, GPT-6 Astra, Grok 4.6, and an open-source frontier model: Kimi-K3. The huge boost in Tables 6, 7, 8 show that these frontier models have surpassed the models from the main paper. Kimi K3 does a really good job in coding and visually similar video generation, which is a win for open-source models, however with much more thinking token generation than the others. GPT-6-Astra is also good at visually similar video generation, however, Claude-Fable-5.1 seems to have the most physics understanding among the evaluated models.

Table 6: **Average Likert scores** (on a 1–5 scale) rated by 3 LLM judge models (Gemini-2.5-flash, Grok-4.1-Fast, GPT-5-Nano) for model outputs across physical law correctness and equivalence dimensions, on the 10-sample hard subset.

	Physical Law Correctness		**Physical Law Equivalence**			
Model	Formula	Overall	Name	Statement	Formula	Overall
New frontier models						
CLAUDE FABLE 5.1	4.40	4.33	3.47	**3.13**	**2.90**	**3.00**
GPT-6 ASTRA	**4.73**	**4.50**	3.27	2.87	2.63	2.80
GROK 4.6	4.40	4.23	2.87	2.53	2.30	2.40
KIMI K3	4.40	4.37	**3.57**	2.63	2.40	2.67
Open-source models						
QWEN3-VL-30B-A3B-INSTRUCT	4.73	4.87	1.83	1.67	1.50	1.60
INTERNVL3.5-30B-A3B	4.53	4.53	1.47	1.73	1.17	1.40
GEMMA-3-12B-IT	**5.00**	**5.00**	**2.10**	1.87	1.43	1.73
GLM-4.1V-9B-THINKING	4.50	4.60	1.90	**1.90**	1.70	**1.80**
PIXTRAL-12B-2409	4.63	4.63	1.80	1.50	**1.83**	1.53
Closed-source models						
GPT-5 MINI	**4.87**	4.60	1.87	2.07	1.53	1.60
GEMINI 2.5 PRO	4.19	4.30	2.07	1.85	**1.74**	**1.96**
CLAUDE SONNET 4.6	3.73	3.50	**2.17**	**2.10**	1.43	1.77
GROK 4 FAST REASONING	4.77	4.53	2.03	**2.10**	1.70	1.93
NOVA 2 LITE	4.80	**4.63**	2.10	1.93	1.70	1.80

A.2 RIGOROUS EVALUATION TO MEASURE CLAUDE BIAS

We request to observe a few facts: (i) Claude Sonnet 4.6 leads 0 of the 11 physics/similarity metrics in Tables 3 and 4. (ii) No Claude model was on the judge panel deliberately, so self-bias was not there during evaluation. (iii) In Table 5, Claude leads in some code generation and execution reliability metrics, but it can't be due to a GT-authorship bias; it's just Claude being a strong coding model. And Claude leads in the parameter/ estimation accuracy; actually, it comes from naming recall (0.460, best of 10) rather than per-parameter value accuracy, where it ranks only 3rd (0.185, behind GLM-4.1V 0.199 and Pixtral 0.188) (Table 9), so breadth of coverage, not better physics, and not authorship.

To validate the debiasing, we rebuilt a matched subset with GPT-5.5: 15 SciPy experiments × 5 samples = 75 samples. Everything is GPT-generated this time, including parameter choices and sampling ranges, through the identical curation pipeline with the same 2-10 rounds of human physical-plausibility iteration. This GPT arm is not debiased (only mandatory preprocessing: comment stripping, CoT normalisation to LaTeX/Python), while the Claude arm was already debiased. We then ran 8 MLLMs (3 closed + 5 open) on both arms.

For each of Claude and GPT (as inference models), we measure how much its score moves when the ground truth switches from Claude-authored to GPT-authored, then subtract the average move of the 6 neutral models (neither Claude- nor GPT-family). Subtracting the neutral models cancels out the fact that the two arms simply differ in difficulty, so what remains is the family-specific effect only. We call this the bias score (formally, difference-in-differences). A positive bias score means the model does relatively better on GPT-authored ground truth; a negative score means it does better on Claude-authored ground truth. So if Claude bias existed, Claude's bias score would be clearly negative (it loses its home advantage) and GPT-5 Mini's clearly positive. Neither happens (Table 10).

Claude's bias score on parameter estimation is positive (+0.76 pp); it does better on GPT-authored ground truth. GPT-5 Mini's is negative (-1.93 pp); it does worse on its own family's ground truth.

Table 7: **Video simulation equivalence table** containing verifiable video frame similarity scores using DINOv2 and VideoCLIP, on the 10-sample hard subset. Cosine similarities on $[0, 1]$, averaged over successfully rendered videos (scipy and kubric pooled). Human-judgment columns are not available for this subset.

	Patchwise Video Similarity	
Model	DINOv2 sim.	VideoCLIP sim.
New frontier models		
CLAUDE FABLE 5.1	0.855	0.881
GPT-6 ASTRA	**0.902**	0.896
GROK 4.6	0.781	0.869
KIMI K3	0.863	**0.903**
Open-source models		
QWEN3-VL-30B-A3B-INSTRUCT	0.506	0.716
INTERNVL3.5-30B-A3B	0.580	0.733
GEMMA-3-12B-IT	**0.595**	0.609
GLM-4.1V-9B-THINKING	0.576	**0.740**
PIXTRAL-12B-2409	0.339	0.669
Closed-source models		
GPT-5 MINI	0.714	0.812
GEMINI 2.5 PRO	0.788	**0.843**
CLAUDE SONNET 4.6	**0.802**	0.841
GROK 4 FAST REASONING	0.625	0.768
NOVA 2 LITE	0.428	0.666

Table 8: **Evaluation of parameter estimation accuracy and generated simulation code quality** on the 10-sample hard subset (77 ground-truth parameters). Percentages computed as a fraction of all $N = 10$ inputs that successfully reached each stage (CoT extracted, Code extracted, Compiles, Runs without error, Saves video). CodeBLEU averaged over scored samples.

	Param. Estimation	Code Generation and Execution					
Model	Acc. ($\pm$20%)	CoT	Code	Compile	No Errors	Video Save	CodeBLEU
New frontier models							
CLAUDE FABLE 5.1	**18.18%**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	0.240
GPT-6 ASTRA	5.19%	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	0.240
GROK 4.6	9.09%	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	0.237
KIMI K3	12.99%	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**0.244**
Open-source models							
QWEN3-VL-30B-A3B-INSTRUCT	**3.90%**	**100.0**	**100.0**	**100.0**	**80.0**	**80.0**	**0.202**
INTERNVL3.5-30B-A3B	**3.90%**	**100.0**	90.0	60.0	10.0	10.0	0.184
GEMMA-3-12B-IT	**3.90%**	**100.0**	**100.0**	**100.0**	10.0	10.0	0.193
GLM-4.1V-9B-THINKING	**3.90%**	**100.0**	70.0	70.0	30.0	30.0	0.189
PIXTRAL-12B-2409	**3.90%**	**100.0**	**100.0**	**100.0**	50.0	60.0	0.179
Closed-source models							
GPT-5 MINI	3.90%	50.0	90.0	90.0	80.0	80.0	**0.245**
GEMINI 2.5 PRO	5.19%	90.0	60.0	60.0	60.0	60.0	0.199
CLAUDE SONNET 4.6	5.19%	**100.0**	90.0	90.0	**90.0**	**90.0**	0.232
GROK 4 FAST REASONING	**6.49%**	**100.0**	**100.0**	**100.0**	**90.0**	**90.0**	0.196
NOVA 2 LITE	2.60%	**100.0**	**100.0**	**100.0**	70.0	80.0	0.195

On CodeBLEU, the one metric that literally string-compares model output against the ground-truth code and is therefore most exposed to authorship style, Claude's bias score is exactly 0.000, and GPT-5 Mini beats Claude on both arms (0.203 / 0.219 vs 0.189 / 0.201). On the Claude-authored arm, GPT-5 Mini wins parameter recovery (7.81%) and Claude places only 3rd (6.30%); on the GPT-authored arm, Claude leads (7.52%). A benchmark that favored its creator model could not place that model 3rd on its own ground truth.

Finally, the two arms rank the 8 models almost identically. In Table 13, $\rho = +1$ means the two arms produce the same ranking of the 8 models, and ρ = 0 means the rankings are unrelated. ρ is positive on all 13 metrics (mean +0.775). p is the standard significance value for each correlation: the probability of getting a ρ that large purely by luck, if the two arms actually ranked the 8 models in unrelated orders. Small p = the agreement is real; large p = it could easily be coincidence. We

Table 9: Parameter-estimation evaluation across 10 MLLMs on the 2,430 benchmark samples (162 experiments × 15 samples). **Parse fails**: number of samples in which the model's response could not be parsed into a valid CoT block (no parameters recovered for those samples). **Avg pred params**: mean number of physical parameters the model emits per sample. **Avg matched**: mean number of model parameters that were successfully aligned to a ground-truth parameter, by greedy bipartite matching over alias sets that capture deabbreviation and Greek/Latin synonyms. **Mean frac reported**: mean over rows of $n_{\text{matched}}/n_{\text{gt}}$, i.e. recall on *naming* the correct parameters. **Mean frac within 20%**: mean over rows of $n_{\text{within tol}}/n_{\text{matched}}$, i.e. *value*-accuracy among matched parameters at $\pm 20\%$ relative-error tolerance. **Global accuracy**: $\sum n_{\text{within tol}} / \sum n_{\text{matched}}$ pooled across all 2,430 rows. **End-to-end recovery**: $\sum n_{\text{within tol}} / \sum n_{\text{gt}}$, the fraction of *all* ground-truth parameters that the model both named correctly and estimated within $\pm 20\%$. Best per column in **bold**.

Model	Parse fails	Avg pred params	Avg matched	Mean frac reported	Mean frac within 20%	Global accuracy	End-to-end recovery
claude_sonnet_4_6	**0**	7.7	**3.29**	**0.460**	0.171	0.185	**0.0769**
gemini_2_5_pro	96	5.5	2.83	0.399	0.150	0.170	0.0606
gemma_3_12b_it	330	4.8	1.94	0.281	0.130	0.163	0.0398
glm_4_1v_9b_thinking	420	4.0	1.93	0.273	0.147	**0.199**	0.0484
gpt_5_mini	465	**8.4**	3.17	0.426	0.139	0.178	0.0713
grok_4_fast_reasoning	32	5.9	2.74	0.389	0.155	0.165	0.0569
internvl3_5_30b_a3b	22	3.7	1.77	0.259	0.141	0.158	0.0352
nova_2_lite	203	4.0	1.78	0.258	0.136	0.167	0.0376
pixtral_12b	365	3.5	1.39	0.198	0.134	0.188	0.0331
qwen3vl_30b_a3b	29	6.2	2.80	0.407	**0.158**	0.174	0.0616

Table 10: Bias scores across evaluation metrics for Claude Sonnet 4.6 and GPT-5 Mini. The gap is computed as GPT-5 Mini minus Claude Sonnet 4.6.

Metric	**Claude Sonnet 4.6**	**GPT-5 Mini**	**Gap (GPT − Claude)**
Law equivalence – Overall	+0.27	+0.30	+0.03
Law validity – Overall	+0.02	−0.04	−0.06
CodeBLEU	**+0.000**	−0.003	−0.003
Runs without error	−3.2 pp	−3.9 pp	−0.7 pp
Video save	−2.2 pp	−1.6 pp	+0.6 pp
Param. Est. Acc. ($\pm 20\%$)	**+0.76 pp**	−1.93 **pp**	−2.69 **pp**
If the benchmark favored Claude	*large negative*	*large positive*	*large positive*

restricted the counterfactual arm to SciPy because SciPy GT code is free-form, so generator style has maximum room to leave a signature, whereas Kubric code is written against a fixed 5-block skeleton, which suppresses authorship style; a Kubric arm would bound the bias lower, not higher. Table 14 shows the parameter estimation evaluation comparison; Table 11 shows the physical law correctness and equivalence scores (scored by the same LLM judge panel); and Table 12 shows the $\pm 20\%$ parameter estimation accuracy along with code evaluation metric values of 5 open-source and 3 closed-source benchmarking models on the smaller 75-sample subset authored by Claude and GPT.

A.3 Statistical Justification of choosing Subset for Benchmarking

Within an experiment, samples are i.i.d. in parameter values with the physics held constant, so the per-experiment success rate is a binomial proportion whose standard error at $n = 15$ is bounded by $\sqrt{0.25/15} \approx 0.129$ assuming the maximum possible variance for a Bernoulli. Averaging across 162 independent experiments bounds the per-model aggregate standard error to $0.129/\sqrt{162} \approx 0.010$, i.e. a 95% confidence interval of $\pm 2\%$ on every metric reported in Tables 3–5. This 95% CI bound on every reported metric (±0.010) statistically supports every between-model gap we report (e.g., 7.7% vs. 3.3% on parameter recovery), so the relative rankings established on the 2,430-sample subset are statistically supported and very likely to hold on the full 160,655-sample corpus, while API cost remains tractable.

Table 11: **Average Likert scores** (on a 1–5 scale) rated by 3 LLM judge models (Gemini-2.5-flash, Grok-4.1-Fast, GPT-5-Nano) for model outputs on a matched subset of 75 samples per arm — 15 Scipy experiments with 5 samples each, generated by **GPT-5.5 (left) vs. Claude Opus 4.7 (right)** within each cell as "GPT arm / Claude arm".

Model	Physical Law Correctness		Physical Law Equivalence			
	Formula	Overall	Name	Statement	Formula	Overall
Open-source models						
QWEN3-VL-30B-A3B-INSTRUCT	4.76 / 4.71	4.75 / 4.77	2.08 / 2.00	2.10 / 1.95	1.58 / 1.56	1.81 / 1.80
INTERNVL3.5-30B-A3B	4.67 / 4.67	4.45 / 4.56	1.75 / 1.83	1.84 / 2.04	1.35 / 1.53	1.53 / 1.73
GEMMA-3-12B-IT	4.57 / 4.74	4.69 / 4.81	1.77 / 2.49	1.70 / 2.22	1.29 / 1.59	1.48 / 1.99
GLM-4.1V-9B-THINKING	4.29 / 4.44	4.08 / 4.32	1.99 / 1.98	2.00 / 1.97	1.45 / 1.54	1.69 / 1.74
PIXTRAL-12B-2409	4.57 / 4.61	4.46 / 4.59	1.88 / 2.12	1.61 / 1.85	1.32 / 1.50	1.49 / 1.73
Closed-source models						
GPT-5 MINI	4.67 / 4.72	4.50 / 4.68	2.69 / 2.46	2.65 / 2.46	2.19 / 1.96	2.35 / 2.19
GEMINI 2.5 PRO	4.51 / 4.71	4.34 / 4.59	2.13 / 2.10	2.28 / 2.09	1.90 / 1.75	2.02 / 1.89
CLAUDE SONNET 4.6	4.43 / 4.56	4.28 / 4.40	2.55 / 2.43	2.52 / 2.38	1.95 / 1.86	2.22 / 2.09

Table 12: **Evaluation of parameter estimation accuracy and generated simulation code quality on the matched 75-sample subset**. Each cell reports **GPT arm / Claude arm** on 15 Scipy experiments with 5 samples each. Percentages computed as a fraction of $N = 75$ inputs per arm that successfully reached each stage. CodeBLEU averaged over scored samples. Param. Estimation Acc. uses alias matching (deabbreviation + Greek/Latin synonyms) with $\pm 20\%$ tolerance.

Model	Param. Estimation	Code Generation and Execution					
	Acc. ($\pm 20\%$)	CoT	Code	Compile	No Errors	Video Save	CodeBLEU
Open-source models							
QWEN3-VL-30B-A3B-INSTRUCT	7.33% / 4.52%	100.0% / 100.0%	96.0% / 100.0%	96.0% / 98.7%	60.0% / 58.7%	70.7% / 64.0%	0.142 / 0.156
INTERNVL3.5-30B-A3B	4.36% / 4.25%	100.0% / 100.0%	98.7% / 100.0%	98.7% / 100.0%	42.7% / 42.7%	60.0% / 65.3%	0.125 / 0.137
GEMMA-3-12B-IT	2.57% / 2.88%	98.7% / 100.0%	98.7% / 94.7%	98.7% / 94.7%	22.7% / 20.0%	33.3% / 34.7%	0.137 / 0.145
GLM-4.1V-9B-THINKING	6.73% / 6.99%	100.0% / 97.3%	100.0% / 97.3%	100.0% / 96.0%	57.3% / 48.0%	58.7% / 50.7%	0.142 / 0.152
PIXTRAL-12B-2409	3.17% / 4.25%	100.0% / 100.0%	100.0% / 100.0%	100.0% / 100.0%	40.0% / 33.3%	73.3% / 72.0%	0.118 / 0.131
Closed-source models							
GPT-5 MINI	6.34% / 7.81%	90.7% / 89.3%	97.3% / 93.3%	97.3% / 93.3%	84.0% / 80.0%	84.0% / 80.0%	0.203 / 0.219
GEMINI 2.5 PRO	5.74% / 4.25%	96.0% / 97.3%	73.3% / 81.3%	73.3% / 81.3%	70.7% / 73.3%	70.7% / 76.0%	0.165 / 0.181
CLAUDE SONNET 4.6	7.52% / 6.30%	100.0% / 100.0%	98.7% / 98.7%	98.7% / 98.7%	94.7% / 93.3%	94.7% / 94.7%	0.189 / 0.201

A.4 INFERENCE PROMPT

Every model under evaluation receives the engine-conditional prompt template shown in Listing 1. The placeholders {`engine_label`}, {`schema`}, and {`code_rules`} are filled with the values of Listings 2–5 when the engine is `scipy` or `kubric`, respectively. The two delimited output blocks (`@@COT_BEGIN@@...@@COT_END@@` and `@@CODE_BEGIN@@...@@CODE_END@@`) are parsed deterministically; parse failures count toward the "parse fails" column of Table 9.

Listing 1: Engine-conditional prompt template (Python f-string source). The same template is sent to every model; {`engine_label`}, {`schema`}, and {`code_rules`} are interpolated from Listings 2–5.

```
You are a physics simulation expert. You will be shown a video
(10 FPS, duration-preserving) of a physical scene. The simulation
engine is: {engine_label}.

Your task: from the video alone, infer the physics and output BOTH:
  (1) a Chain-of-Thought analysis as a JSON object (schema below), and
  (2) a self-contained pure-Python script (see CODE REQUIREMENTS for the
      exact libraries) that, when executed, reproduces the dynamics
      and writes the rendered video to video.mp4.

== OUTPUT PROTOCOL (strict) ==
Emit exactly two blocks, each delimited by the markers below. Do NOT
wrap them in code fences. Do NOT add any text outside the markers.

@@COT_BEGIN@@
<one valid JSON object matching the schema below>
```

Table 13: Model ranking similarity score measured by Spearman's rank correlation coefficient (ρ) and corresponding significance value (p) for all the evaluation metrics

Metric	ρ	p
CodeBLEU	+1.000	$\sim$0
Law validity - Statement	+0.976	$\sim$0
Runs without error	+0.976	$\sim$0
Law validity - Name	+0.952	$\sim$0
Law validity - Overall	+0.905	0.002
Video save	+0.905	0.002
Param. Est. Acc. ($\pm$20%)	+0.786	0.021
Law equivalence - Formula	+0.762	0.028
Law equivalence - Statement	+0.667	0.071
Law equivalence - Overall	+0.643	0.086
Syntax OK	+0.643	0.086
Law validity - Formula	+0.500	0.207
Law equivalence - Name	+0.357	0.385
All 13 metrics	**Mean +0.775**	**13/13 positive**

Table 14: **Parameter-estimation evaluation split by GT authorship**, on the matched 75-sample subset (15 Scipy experiments $\times$ 5 samples). Each cell reports **GPT arm / Claude arm** using the same alias-matcher as Table 9. Column definitions unchanged.

Model	Parse fails	Avg pred params	Avg matched	Mean frac reported	Mean frac within 20%	Global accuracy	End-to-end recovery
claude_sonnet_4_6	0 / 0	7.7 / 7.3	3.07 / 3.45	0.459 / 0.346	0.176 / 0.164	0.165 / 0.178	0.0752 / 0.0630
gemini_2_5_pro	3 / 2	5.9 / 5.3	2.51 / 2.68	0.378 / 0.284	0.137 / 0.175	0.154 / 0.154	0.0574 / 0.0425
gemma_3_12b_it	21 / 34	4.6 / 3.6	1.75 / 1.67	0.268 / 0.181	0.063 / 0.114	0.099 / 0.168	0.0257 / 0.0288
glm_4_1v_9b_thinking	6 / 11	5.2 / 4.7	2.59 / 2.59	0.385 / 0.272	0.141 / 0.228	0.175 / 0.263	0.0673 / 0.0699
gpt_5_mini	7 / 10	10.5 / 9.4	3.63 / 3.87	0.544 / 0.422	0.116 / 0.168	0.118 / 0.197	0.0634 / 0.0781
internvl3_5_30b_a3b	0 / 0	4.0 / 3.9	1.57 / 1.81	0.235 / 0.195	0.131 / 0.223	0.186 / 0.228	0.0436 / 0.0425
pixtral_12b	9 / 0	4.0 / 4.4	1.25 / 1.99	0.183 / 0.204	0.104 / 0.206	0.170 / 0.208	0.0317 / 0.0425
qwen3vl_30b_a3b	0 / 1	6.1 / 5.8	2.61 / 2.76	0.413 / 0.297	0.156 / 0.145	0.189 / 0.159	0.0733 / 0.0452

```
@@COT_END@@

@@CODE_BEGIN@@
<full Python source code, raw (no triple backticks)>
@@CODE_END@@

== CHAIN-OF-THOUGHT JSON SCHEMA ==
```

Table 15: **Chain-of-thought schema.** Every sample's CoT uses this fixed key ordering; sub-schemas use 10–13 of the keys depending on the system's complexity.

Key	**Content**
simulation	Experiment-type identifier.
category	Physics domain (e.g., articulated, rigid-body collision).
physical_observation	Natural-language description of the visual dynamics, with parameter values.
physical_law	Law name, statement, and governing equation in LaTeX.
modeling_assumptions	Idealisations baked in (point masses, frictionless surfaces, etc.).
ode_system	State variables and governing ODE in both LaTeX and Python form.
event_schedule	Discrete events for collisions / contacts (only when applicable).
derivation_steps	Ordered derivation from first principles to the implemented update.
physical_constants	Computed quantities (moments of inertia, equilibrium positions, etc.).
numerical_method	Solver, tolerances, and closed-form availability.
code_validation	Analytical checks, limiting behaviours, and plausibility flags.
expected_behavior	Qualitative predictions about the resulting trajectory.
parameters	All sampled values with units and human-readable descriptions.

```
The JSON object MUST use these exact top-level keys (and nested keys
where shown). Use null where not applicable.
{schema}

== CODE REQUIREMENTS ==
{code_rules}

Begin now. Output only the two delimited blocks, nothing else.
```

Listing 2: Chain-of-thought JSON schema for the `scipy` engine.

```
{
  simulation: name, eg: falling objects,
  physical_observation: <one paragraph: what you see, geometry, key
    ↪ numerical scales>,
  physical_law: {
    name: <law / theorem name>,
    statement: <one-sentence statement>,
    formula: <LaTeX formula>
  },
  parameters: {
    <name>: {value: <number>, unit: <SI unit or empty>, description: <
    ↪ short>},
    <...>: <...>
  }
}
```

Listing 3: Chain-of-thought JSON schema for the `kubric` engine. Identical structure to the `scipy` schema; only the `simulation` description changes to reflect 3-D rigid-body scenes.

```
{
  simulation: name, eg: falling object,
  physical_observation: <one paragraph: objects, geometry, key parameter
    ↪ values>,
  physical_law: {
    name: <law name>,
    statement: <one sentence>,
    formula: <LaTeX>
  },
  parameters: {
    <name>: {value: <number>, unit: <units>, description: <short>},
    <...>: <...>
  }
}
```

Listing 4: Code-generation rules for the `scipy` engine. The model must emit a self-contained Python script that uses only stdlib, NumPy, SciPy, and Matplotlib's `FuncAnimation` to render the predicted dynamics into `video.mp4`.

```
- Write PURE PYTHON that runs in a standard Python environment
  (no Docker, no external physics engines).
- Use ONLY stdlib + numpy + scipy + matplotlib. Do NOT import any 3D
    ↪ engine.
- Use this exact import block (you may omit a line you don't need;
  do not add other heavy deps):
    import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.animation as animation
    from matplotlib.patches import Polygon, Rectangle, Circle
    from matplotlib.collections import LineCollection
    from scipy.integrate import solve_ivp
- Solve dynamics analytically when a closed form exists (kinematics,
  damped oscillators); otherwise call scipy.integrate.solve_ivp with
```

```
  RK45 and tight tolerances (rtol=1e-9, atol=1e-11).
- Render with matplotlib.animation.FuncAnimation: build patches/lines
  once, then update them via set_xy / set_data / set_segments inside
  the update(frame) function. Use blit=False if state is complex.
- IMPORTANT: assign FuncAnimation to a top-level variable (e.g.
  ani = animation.FuncAnimation(...)) and then call ani.save(...).
  Do NOT chain inline like animation.FuncAnimation(...).save(...) -
  matplotlib garbage-collects the unreferenced animation and silently
  writes an empty / corrupt video.
- Save with: ani.save('video.mp4',
      writer=animation.FFMpegWriter(fps=30, bitrate=1800), dpi=120).
  Output file MUST be exactly 'video.mp4' in the current working
  directory.
- Render at fps=30. Match the duration of the observed video as best
  you can infer.
- The script must be standalone (no command-line args, no relative
  file deps) and runnable as `python simulation_code.py`.
- DO NOT wrap the code in ```python ... ``` triple-backtick fences.
  Emit raw Python ONLY between the @@CODE_BEGIN@@ and @@CODE_END@@
  markers, no fences anywhere.
```

Listing 5: Code-generation rules for the `kubric` engine. The model must emit a self-contained PyBullet + pyrender + trimesh + imageio Python script (the literal `kubric` package is out of scope because it requires Blender + Docker). The rules list hard environment bans, an exact 5-block skeleton, the keyword signatures of every PyBullet API the script may call, and a self-check the model is asked to perform before emitting. For brevity we reproduce the abridged version; the full text is in `prompts.py` of the released code.

```
Write PURE PYTHON. Do NOT use Docker. Do NOT `import kubric` - that
package requires Blender + Docker and is OUT OF SCOPE. 'kubric' here
is the 3D-rigid-body category; implement with PyBullet (physics) +
pyrender (offscreen rendering) + trimesh + imageio.

== HARD ENVIRONMENT BANS ==
- trimesh.creation.sphere(...) -> use trimesh.creation.icosphere(
  subdivisions=3, radius=R). Allowed shapes: box, icosphere,
  cylinder, capsule. Anything else is forbidden.
- trimesh.visual.<anything> -> all forbidden. Color is set
  EXCLUSIVELY via pyrender.MetallicRoughnessMaterial(...).
- loadURDF(<anything other than 'plane.urdf'>) is forbidden. All
  shapes built with p.createCollisionShape(GEOM_BOX|SPHERE|CYLINDER|
  CAPSULE) + p.createMultiBody(...).
- p.createConstraint(...) is forbidden in this build.
- p.GEOM_MESH is forbidden.

== EXACT 5-BLOCK SKELETON ==
1. IMPORTS (do not add heavy deps beyond these):
   import os, math, random
   import numpy as np
   import pybullet as p
   import pybullet_data
   import trimesh, pyrender, imageio
2. PHYSICS SETUP (setAdditionalSearchPath BEFORE any loadURDF).
3. RENDERER + SCENE: at least one DirectionalLight AND exactly one
   PerspectiveCamera, both added before the frame loop.
4. WORLD: pybullet bodies + matching pyrender nodes (kept paired).
5. SIM LOOP -> SAVE: append to frames each iteration; write
   'video.mp4' at fps=30, 480x480.

== EXACT API SIGNATURES (use EXACT keyword names) ==
- p.createCollisionShape(GEOM_BOX, halfExtents=[hx,hy,hz],
                         physicsClientId=c)
- p.createCollisionShape(GEOM_SPHERE, radius=R, physicsClientId=c)
- p.createCollisionShape(GEOM_CYLINDER, radius=R, height=H,
```

```
                          physicsClientId=c)         # NOT length=
- p.createMultiBody(baseMass, baseCollisionShapeIndex,
                    baseVisualShapeIndex=-1,
                    basePosition=[x,y,z],
                    baseOrientation=[qx,qy,qz,qw],
                    physicsClientId=c)
- p.changeDynamics(body_id, -1, lateralFriction=...,
                   restitution=..., physicsClientId=c)
- p.resetBaseVelocity(body_id, linearVelocity=[...],
                      angularVelocity=[...], physicsClientId=c)
- p.applyExternalForce(body_id, -1, forceObj=[...],
                       posObj=[0,0,0], flags=p.LINK_FRAME,
                       physicsClientId=c)

== SELF-CHECK BEFORE EMITTING ==
(a) setAdditionalSearchPath precedes every loadURDF;
(b) no createJoint / createConstraint call exists;
(c) every PyBullet call passes physicsClientId=c;
(d) the frame loop appends to frames exactly once per iteration;
(e) the script ends with p.disconnect(c); renderer.delete().

Output file MUST be exactly `video.mp4` in the current working
directory; resolution 480x480, fps=30; runnable as
`python simulation_code.py` with no args.
```

A.5 Human Rating Guidelines

Raters were given coarse band definitions and asked to apply holistic judgment: for physical plausibility, 1-2 = physically impossible, 3 = broadly plausible with visible defects, 4 = minor defects, 5 = fully plausible; for equivalence (physics and appearance), 1 = unrelated, 2 = mostly different with a few matching elements, 3 = about half matching, 4 = mostly matching, 5 = closely matching. We deliberately kept the bands coarse because the failure modes vary widely across 162 phenomena. E.g. one regenerated video reproduced a correctly-colored ball bouncing at the right rhythm but with no floor present, which we scored 4 on appearance similarity and 2 on physical plausibility.

A.6 Unobservable parameters and sensitivity stratification

Different physical parameters have different extent of sensitivity. Sometimes, perturbing a parameter doesn't result into significant dynamics change, sometimes the video doesn't have enough information to estimate certain parameters (like experiments which have mass-independent dynamics). We ran a perturbation pilot study to categorize the sensitivity of the parameters.

For 15 experiments covering 108 physical parameters, we shifted each parameter by $\pm 10\%$, re-rendered the video, and measured how much the video changed (mean per-frame pixel MSE against the unperturbed video; $N = 1,080$ measurements). We heuristically define a parameter's sensitivity as that video change divided by the average video change of all parameters in the same experiment. So sensitivity = 1 means the parameter moves the video as much as a typical parameter in its own scene, 2 means twice as much, and 0.5 means half. Normalizing within an experiment is essential, because raw pixel change is dominated by how visually busy a scene is rather than by physics: it ranges $142\times$ across experiments (0.000144 for `ball_wrapping_pole`, a single ball against a dark background, up to 0.0205 for `slab_springs_block_drop`, which has many moving parts). After normalization that $142\times$ spread collapses to $4.4\times$.

The result in Table 16 is that sensitivity is far more uniform than expected: half of all parameters sit within $\pm 5\%$ of their scene's average, and only 7 of 108 fall outside [0.6, 1.5]. In other words, at a fixed relative perturbation, most physical parameters in most scenes produce comparably-sized visual consequences. This speaks directly to why we chose a single fixed $\pm 20\%$ tolerance: because our threshold is also relative, and sensitivity at fixed relative perturbation is near-uniform, a single tolerance is much less crude than it first appears. We also found no parameter that is unobservable in the strict sense. The minimum normalized sensitivity is 0.48, roughly half the scene average, not zero; there is a sensitivity spectrum, but no degenerate cases like the stationary-object mass in

Table 16: Sensitivity across the 108 parameters

Statistical parameters	Value
Median	0.98
Middle 50% (IQR)	**[0.93, 1.05]**
90th / 95th percentile	1.17 / 1.22
Full range	[0.48, 2.12]
Parameters above 1.5 / below 0.6	3 / 4 (of 108)

our sampled scenes. This is because our generators sample initial conditions so that every body participates in the dynamics, and Sec. 4.1 already discussed removing non-physical entries (scene metadata, geometry, color, and g) from scoring. We note one more honest limitation: pixel-space MSE is a proxy for trajectory deviation and conflates visual salience with dynamical consequence, so the stratification should be read as ordinal rather than as a calibrated sensitivity coefficient.

A.7 Additional Related Work Details

This appendix provides additional context for the related work summarized in Section 2. The main paper focuses on the closest conceptual gaps, while this section gives more detailed descriptions of prior benchmarks, evaluation protocols, and simulator-based approaches.

A.7.1 Physical Property Estimation and Law Discovery from Video

Physics 101 (Wu et al., 2016) builds on the earlier Galileo framework (Wu et al., 2015), which couples a physics engine with deep networks to infer object properties from video. Physics 101 is an early benchmark for learning physical properties from unlabeled video, compiling 17,408 clips across 101 objects that vary in material, shape, color, and size, and undergo four types of physical interaction: sliding, collision, spring attachment, and free fall. Related work has explored block-tower stability prediction (Lerer et al., 2016), graph-based dynamics learners that propagate object interactions (Battaglia et al., 2016; Watters et al., 2017), and inverse-graphics formulations that recover physical parameters by re-rendering (Jaques et al., 2019).

Visual Physics (Chari et al., 2019) takes a more direct approach to physics extraction from video: given bounding-box trajectories automatically extracted from video sequences, the system jointly discovers both the functional form of governing equations and their associated parameter values, without prior knowledge of which physical laws are operative. A complementary line of recent work jointly recovers 3D geometry and physical parameters from video by combining radiance fields or Gaussian splats with differentiable continuum mechanics (Li et al., 2023; Xie et al., 2024; Zhong et al., 2024; Cao et al., 2024; Cai et al., 2024). These methods are closely related in spirit to our goal of recovering physics from visual observations, but the recovered representations are typically equations, parameters, trajectories, or implicit per-particle dynamics models rather than self-contained, re-runnable simulation programs.

A.7.2 Benchmarks for Physical Understanding

PhysInOne (Zhou et al., 2026) is a large-scale benchmark for visual physics understanding, constructing 153,810 dynamic 3D scenes and 2 million videos across 71 physical phenomena, including rigid-body mechanics, fluid dynamics, optics, and electromagnetism. It uses multiple physics engines: Unreal Engine 5's Chaos Physics for rigid bodies, the Material Point Method (Jiang et al., 2016; Hu et al., 2018) implemented with Taichi (Hu et al., 2019b) for granular and deformable objects, and Smoothed Particle Hydrodynamics (Monaghan, 1992) for liquids. Despite its scale, PhysInOne is a perception and estimation benchmark: models are evaluated on their ability to classify phenomena, estimate physical properties such as mass or elasticity, or predict future states.

Morpheus (Zhang et al., 2025) evaluates whether contemporary video generative models, including SORA (Brooks et al., 2024), COSMOS (Agarwal et al., 2025), and WAN-2.1 (Wan et al., 2025), preserve physical invariants. Rather than relying only on trajectory matching or subjective human ratings, Morpheus grounds evaluation in conservation laws: a Dynamical Score uses physics-informed

neural networks (Raissi et al., 2019) to assess trajectory consistency with governing differential equations, while a Physical Invariance Score measures conservation of total energy and momentum across generated videos. VideoPhy (Bansal et al., 2025) instead focuses on physical commonsense in text-to-video generation, emphasizing interactions between materials of different types and using both human annotation and an automatic VideoCon-Physics evaluator, a VLM fine-tuned to assess physical plausibility from text-video pairs.

Beyond these closely related efforts, a broader benchmarking ecosystem evaluates physical understanding from complementary angles. VBench (Huang et al., 2024) evaluates video-generation quality; PhysBench (Chow et al., 2025), PhyBench (Qiu et al., 2026), and ComPhy (Chen et al., 2022) evaluate physical-commonsense reasoning in VLMs or LLMs; and Physion (Bear et al., 2021), IntPhys (Riochet et al., 2018), CLEVRER (Yi et al., 2019), and ComPhy (Chen et al., 2022) study visual intuitive physics. These benchmarks are complementary to ours, but they generally evaluate physical understanding through classification, multiple-choice reasoning, plausibility judgment, or future-state prediction rather than executable program recovery.

A.7.3 Physics Reasoning via Simulation

Solving Physics Olympiad via RL on Physics Simulators (Prabhudesai et al., 2026) addresses the scarcity of physics-reasoning training data relative to mathematics. The central idea is that physics simulators such as MuJoCo (Todorov et al., 2012) can serve as procedurally scalable sources of supervision by randomizing physically valid scene graphs along a domain-specific language and querying numerical, inverse, and symbolic relationships between physical quantities. This line of work demonstrates the value of simulator-generated supervision for physics reasoning.

A parallel line of work develops differentiable simulators that propagate gradients through dynamics, including DiffTaichi (Hu et al., 2019a), Brax (Freeman et al., 2021), and Genesis (Authors, 2024), with related infrastructure such as Isaac Gym (Makoviychuk et al., 2021) and NeuralSim (Heiden et al., 2021) used for robotics learning and system identification. Our work is related in that we also use physics simulators to generate ground-truth data, but differs in focusing on the video-conditioned inverse problem: given only a visual observation of a physical process, a model must recover explicit laws, parameters, and executable simulation code.

A.8 CodeBLEU Specification

Computed via the codebleu==0.7.0 PyPI package, weights=(0.25, 0.25, 0.25, 0.25), equal weights across the four components (n-gram match, weighted n-gram match with keyword upweighting, AST syntax match, dataflow match), default 4-gram BLEU with default Python keyword list, and the package's default Python tokenizer (Python's tokenize module).

A.9 LLM Inter-rater Reliability

Analyzing the stored justification fields of LLM judgment, we find 86% of judgments agree within 1 unit and only 1.9% show a spread ≥ 3; the substantive disagreements cluster on coupled multi-body problems (e.g. `wedge_block`, `wedge_sphere`, `spring_atwood`, `double_pendulum`, `differential_pulley_chain_fall`) where the ground truth uses one valid formulation (typically Lagrangian) but the candidate reaches for another (typically Newtonian). GPT-5-Nano tends to treat these as "same physics, different formalism" (higher scores), while Gemini-2.5-Flash treats them as distinct theoretical frameworks (lower scores). Judge disagreement is also higher when scoring models whose outputs are vague-but-partially-correct (Gemma-3-12B mean spread ≈ 1.0) than models whose outputs are unambiguously wrong (Pixtral-12B mean spread ≈ 0.6).

A.10 Models Excluded From Evaluation

- **LLaVA-OneVision-1.5-8B-Instruct** (`lmms-lab`). vLLM 0.11.0 has no registered architecture for it; the bundled HF code requires a `transformers 5.x` build that is incompatible with vLLM in our environment. Cannot be loaded.
- **GLM-4.5V** (`zai-org`). Listed nominally as a 9B-class model; the actual repository is a $>$ 100B-parameter MoE that exceeds both our disk budget and a single H200's memory. Excluded as out of scope.

Table 17: **Open-source model parse rates on the 2,430-sample benchmark. CoT** = fraction of outputs whose `@@COT_BEGIN@@` block contains parseable JSON; **Code** = fraction whose `@@CODE_BEGIN@@` block contains a non-empty Python program. Code metric is a structural check, not an executability check. † Tiled into a single 4-column composite image (vLLM single-image limit).

Model	scipy		kubric		Frames
	CoT	Code	CoT	Code	per video
GLM-4V-9B	67.1%	94.0%	47.1%	70.5%	16†
Pixtral-12B-2409	99.6%	99.8%	82.1%	87.1%	24
InternVL3.5-30B-A3B (MoE)	99.2%	99.8%	**100.0%**	93.2%	32
Qwen3-VL-30B-A3B (MoE)	**99.9%**	94.2%	99.9%	85.3%	60
GLM-4.1V-9B-Thinking	88.2%	96.6%	95.6%	85.5%	60
Gemma-3-12B-IT	60.5%	**99.9%**	79.7%	**100.0%**	32

- **DeepSeek-VL2-Small** (`deepseek-ai`). Pipeline runs, but the model parrots prompt placeholders verbatim and never closes the `@@COT_END@@` block before exhausting its output budget. Throughput is also ∼213 s/sample (∼144 hours per benchmark on one H200).
- **VideoLLaMA3-7B** (`DAMO-NLP-SG`). 100% refusal rate. Every output is the string *"I'm sorry, but I can't assist with that."* The safety post-training rejects long structured reasoning-plus-code prompts. The refusal itself is a reportable finding.
- **LLaVA-OneVision-7B** (`lmms-lab`, `llava-hf`). Pipeline runs but every output degenerates into a 4096-token loop of `import matplotlib.pyplot as plt` repeated; the structured CoT block is never produced.

A.11 Issues Affecting Included Models (Mitigated)

- **Per-model frame budgets are unequal.** Every video is decoded at 10 FPS, duration-preserving, and resized to 360×180 (SciPy) or 240×240 (Kubric). Each model then receives the largest number of frames its context window allows, sampled uniformly across the full clip. Gemini 2.5 Pro consumes the raw .mp4 at the same FPS and resolution. The effective sampling rate depends on clip duration, for a 12 s SciPy clip, budgets of 8/16/32/60/80 frames correspond to about 0.7/1.3/2.7/5.0/6.7 FPS; for a 5 s Kubric clip the 10 FPS decode caps the pool at 50 frames. We treat the closed-vs-open comparison as confounded by frame budget; within-OSS comparisons are cleanly shown in Table 17.
- **GLM-4V-9B is single-image-only in vLLM and capped at 8K positional embeddings.** We tile the sampled frames into a single 4-column composite image; we also capped `max_tokens` at 4096 (with ∼2K-token prompts), which contributes truncation-not-malformation to its CoT-parse failures, particularly on the deeper kubric schema.

A.12 More Dataset Statistics

For Scipy experiments that deal with ordinary differential equations (ODEs), the Dormand–Prince RK45 scheme of SciPy is used with tight tolerances (`rtol` = 10^{-9}, `atol` = 10^{-11}) consistently across all simulation types. The resulting trajectories are rendered into animations using Matplotlib's `FuncAnimation` and encoded as MP4 videos at 30 FPS over a 12-second duration using FFmpeg. The physics of the Kubric experiments is simulated by PyBullet in headless mode at 240 Hz with 8 substeps per render frame, and rendering uses `pyrender` and `trimesh` under EGL-accelerated OpenGL with PBR materials. The resulting trajectories are rendered as MP4 videos.

The CoT field `equations_latex` contains a median of 3 LaTeX equations per sample (range 1–22) and `derivation_steps` contains a median of 4 ordered steps from first principles to the implemented update (range 3–47). The total annotation corpus comprises **1,112,171 derivation steps** and **588,023 LaTeX equations** across 160,655 video–code–CoT samples, providing dense supervision for both equation-recovery and step-by-step reasoning evaluations. The annotation set covers **154 distinct named physical laws** (86 in Pipeline A, 68 in Pipeline B), spanning Newton's second law,

Objects per category	≤ 2	2–5	5–10	> 10
SciPy	61	9	1	0
Kubric	38	14	15	7

Lagrangian mechanics, rigid-body Newton-Euler dynamics, Coulomb friction, restitution-impulse contact, wave equations, constrained dynamics, and rotational kinematics, among others. For experiments whose internal randomization is governed by a single integer seed leading to many choices (categorical object types, variable object counts $N \in [2, 10]$), we manually elevate the sample count proportionally to the effective hidden dimensionality.

Across all 160,614 samples we ship **21.4 million lines of executable Python** (mean 134 LOC, median 121 LOC per sample), each of which is a self-contained, runnable simulation script. The `parameters` field contains a structured numerical record with a median of 9 and up to 21 typed parameters per sample, totaling **1.54M parameter values** suitable for parameter-estimation benchmarks. On disk, the dataset occupies 51.7 GB (49.8 GB SciPy video + 1.96 GB Kubric video + 0.9 GB code + 0.9 GB CoT JSON).

The 154 distinct physical-law annotations span the standard undergraduate to early-graduate physics curriculum: classical mechanics (Newton's laws, Lagrangian/Hamiltonian formulations), rigid-body dynamics with contact (Newton-Euler equations, Coulomb friction, restitution-impulse models), oscillatory and wave systems (linear and nonlinear pendula, coupled oscillators, 1-D wave equations with damping), constrained dynamics (Atwood machines, pulleys, ropes), and fluid-coupled systems (granular flow, sloshing, sphere-basin oscillations). Figure 5 shows the categorical counts as per physical laws and physical domain. Each law name is paired with a formal statement and LaTeX formula, providing a labeled, grounded ontology for evaluating whether a model identifies the *correct* physics regime in addition to extracting the correct numerical parameters.

A.13 Object Count per Experiment Category

Object count per experiment is an indicator of the complexity of that experiment. We computed this from the released params.json of all 162 categories heuristically (because sometimes defining an object is very difficult, read below). A script counted independently simulated dynamical entities (rigid bodies and point masses; ropes/chains by their links), excluding static scenery and the ground plane.

Per sample we read an explicit count field (like number of balls, pit, dominoes, etc) where one exists, else the length of the per-object arrays (objects, masses, object shapes), else the number of distinct mass parameters; we then average over a category's samples. This table counts categories, not samples.

For 17 categories an object count is genuinely undefined, which is itself informative: in `brachistochrone_race` and `hemisphere_roll_off` the dynamics are mass-independent, so no mass is ever sampled; `accelerating_beaker_water`, `u_tube_fluid_oscillation` and `wave_interference` are continua (fluid columns, strings with linear density μ) rather than discrete bodies; and in `three_gear_chain`, $N_A = 30$ counts gear teeth, not objects. We would caution, though, that object count is a weak proxy for the complexity a rater must track. A double pendulum has only two bobs but is chaotic and requires coupled nonlinear equations, which is complex, whereas two blocks colliding on a plane involves the same object count and a single impulse, which is much simpler. Object count also cannot be equated with the number of masses, because several of our categories depend on massless elements that are nonetheless dynamically essential, springs, strings, tethers and pulley constraints that transmit forces without contributing inertia. This is why the complexity statistic we already report is `n_rand`, the number of independently randomized physical parameters per category (Tables 19 and 18), which drives our per-category sample budget; and for categories where a single integer seed determines many latent choices at once (object type, object count, material), `n_rand` understates the true dimensionality, which is why we manually elevated their sample counts (Sec. 3.2).

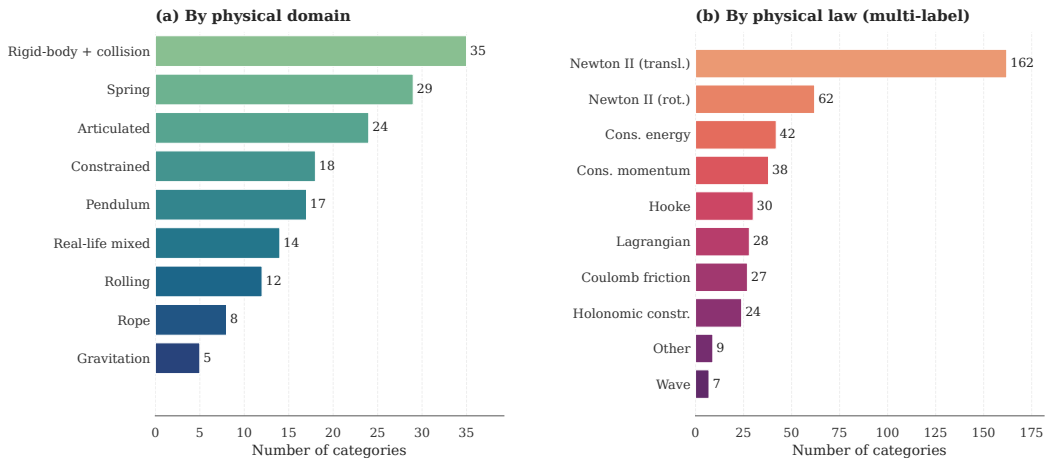


Figure 5: Distribution of the 162 categories. **(a)** Nine top-level physical domains. **(b)** Physical laws invoked across the dataset (multi-label, so counts sum to more than 162); Newton's 2nd law appears in every category by construction, while the long tail captures laws like the 1-D damped wave equation, gravitation, hydrostatic equilibrium, and the Lorentz force.

Table 18: Pipeline A (SciPy): per-experiment statistics. ⋆ marks seed-based experiments where a single integer seed drives non-factorizable hidden choices, requiring manually-elevated sample counts.

Experiment	n_{rand}	Samples	Seeded
complex_collision_plane	2	2500	⋆
accelerating_cart_sliding_mass	4	2000	⋆
balls_in_basin	4	2000	⋆
disk_spinoff	7	1500	⋆
block_on_sphere_basin	13	1300	
molecule_2	12	1200	
three_body_gravity	11	1100	
hammer_projectile_bounce	10	1000	
rolling_race_hoop_vs_disk_vs_sphere	4	1000	⋆
springs_two_slabs_compound	10	1000	
toppling_cylinder_wedge	10	1000	
two_conveyor_drop	10	1000	
wedge_block	10	1000	
wheel_down_stairs	10	1000	
cart_two_metronomes	9	900	
double_2d_spring	9	900	
falling_spring	9	900	
moveable_double_pendulum	9	900	
rod_ball_strike	9	900	
spinning_disc_block_spring	9	900	
wheel_belt_drop	9	900	
cart_pendulum	8	800	
dangle_stick	8	800	
differential_pulley_chain_fall	8	800	
discs_friction_coupling	8	800	
pendulum_block_strike	8	800	
slab_springs_block_drop	8	800	
sliding_block_step_topple	8	800	
spinning_ball_bounce	8	800	

(continued on next page)

Experiment	n_{rand}	Samples	Seeded
spring_2d	8	800	
tilting_wedge_cylinder	8	800	
wheel_wall_bounce	8	800	
block_strikes_spring_pair	7	700	
bouncing_rod	7	700	
chain_of_springs	7	700	
cylinder_piston_ball	7	700	
double_spring	7	700	
gear_rack_spring_oscillator	7	700	
half_cylinder_drop	7	700	
magnet_wheel	7	700	
ring_sticky_collision	7	700	
spinning_ball_on_moving_belt	7	700	
spring_mass_string_fixed	7	700	
spring_mass_string_ring	7	700	
stacked_balls_drop	7	700	
string_ring_wave	7	700	
three_gear_chain	7	700	
three_gear_middle_drive	7	700	
wedge_sphere	7	700	
ballistic_pendulum	6	600	
belt_gear_rack_train	3	600	⋆
billiards_break	6	600	
block_chain_collision	6	600	
colliding_blocks	6	600	
double_pendulum	6	600	
driven_string_fixed	6	600	
driven_string_ring	6	600	
hemisphere_roll_off	6	600	
moveable_pendulum	6	600	
projectile_bounce	6	600	
rigid_double_pendulum	6	600	
roller_single	6	600	
spool_pull_direction_2d	6	600	
string_block_wave	6	600	
two_gear_belt	6	600	
u_tube_fluid_oscillation	6	600	
watt_straight_line_linkage	6	600	
wave_interference	6	600	
accelerating_beaker_water	5	500	
block_belt_spring_stickslip	5	500	
conveyor_belt_drop	5	500	
robot_speed	5	500	
semicircular_track	5	500	
simple_pendulum	5	500	
single_spring	5	500	
two_carts_slack_string	5	500	
atwood_on_table	4	400	
ball_wrapping_pole	4	400	
brachistochrone	4	400	
lagrange_roller	4	400	
ring_pendulum	4	400	
spring_atwood	4	400	
two_body_gravity	4	400	
compound_atwood	3	300	
sand_onto_moving_cart	3	300	

(continued on next page)

Experiment	n_{rand}	Samples	Seeded
brachistochrone_race	2	200	
pendulum_slack_revolution	2	200	
Total (87 experiments)		**64,800**	

Table 19: Pipeline B (Kubric): per-experiment statistics. • in *Var* marks experiments with variable object counts (boost $\times 1.5$); • in *Col* marks experiments with random color/material choice (boost $\times 1.3$). Sample counts are floored at 400 and capped at 5,000.

Experiment	n_{rand}	Var	Col	Samples
multi_physics_arena	47	•		5000
tumbling_dice	19	•	•	3705
bouncing_balls_arena	16	•	•	3120
balloon_tether_wall_pull	15	•	•	2925
turntable_mirror_blocks	12	•	•	2340
rope_pendulum_swing	14	•		2100
ball_pit_drop	10	•	•	1950
block_tower_projectile	13	•		1950
block_wall_hit_by_tethered_mass	10	•	•	1950
rod_spin_hits_balloon	14		•	1820
jenga_pull	12	•		1800
accelerating_panel_spins_objects	9	•	•	1755
avalanche_slope	9	•	•	1755
granular_hourglass_3d	9	•	•	1755
concave_bowl_spin_orbit	12		•	1560
fan_push_stationary_block	8	•	•	1560
moving_cart_hits_moving_cart	12		•	1560
spring_compress_vs_stretch	12		•	1560
scissor_collapse	10	•		1500
stacked_discs_topple	10	•		1500
staircase_bounce	10	•		1500
carried_pendulum_inertia	11		•	1430
crank_push_friction_stop	11		•	1430
lever_catapult	14			1400
trebuchet_throw	14			1400
domino_chain	9	•		1350
funnel_sorting	9	•		1350
wrecking_ball_pendulum	9	•		1350
moving_cart_hits_stationary_cart	10		•	1300
ramp_to_spring_projectile	10		•	1300
marble_run_ramps	8	•		1200
robot_arm_collapse	8	•		1200
mass_spring_collision_chain	6	•	•	1170
spinning_top_precession	9		•	1170
stacked_blocks_earthquake	6	•	•	1170
toppling_bookshelf_row	6	•	•	1170
wind_balanced_falling_panel	6	•	•	1170
gyroscope_on_stand	8		•	1040
spinning_rod_balloon_tether	8		•	1040
two_car_collision_3d	8		•	1040
object_collision_mid_air	10			1000
arch_collapse_keystone	5	•	•	975
bridge_truss_load_failure	5	•	•	975
coin_toss_spin_land	7		•	910
seesaw_with_falling_mass	7		•	910

(continued on next page)

Experiment	n_{rand}	Var	Col	Samples
springboard_rebound_ball	7		•	910
vertical_fall_spinning_panel	7		•	910
newton_cradle_3d	6	•		900
plasticine_ball_inelastic_drop	9			900
projectile_parabola	9			900
ragdoll_free_fall	9			900
ramp_slide_friction	9			900
rolling_cylinder_slope	9			900
seesaw_imbalance	9			900
wedge_collision	9			900
bowling_strike	8			800
double_pendulum_3d_chaotic	8			800
ramp_jump_landing	8			800
sphere_bowl_oscillation	8			800
conical_pendulum	6		•	780
oblique_projectile_hits_slope	6		•	780
pendulum_3d_single	6		•	780
rolling_vs_sliding_blocks	6		•	780
coin_spin_topple	7			700
elastic_coupled_cart_collision	7			700
stick_support_catapult_release	7			700
compound_pulley_lift	5		•	650
offset_load_turntable	5		•	650
rolling_ball_loop_track	5		•	650
rolling_ball_up_slope	5		•	650
atwood_machine	6			600
double_pendulum_impact_transfer	6			600
hinged_door_slam	6			600
center_vs_offset_seesaw	5			500
billiard_break_3d	4			400
Total (75 experiments)				**95,855**

A.14 Frame-count Ablation

We wanted to see how changing the number of frames affects the model performance for the open-source models. So, we re-ran the full pipeline over 162 videos (1 from each experiment) × 5 frame budgets for Qwen3-VL-30B and 4 budgets for InternVL3.5-30B, and re-scored both the Claude and GPT arms with the parameter matcher. In Table 20, * denotes the frame budget used in the paper. (N=162) per row, with 1,284 ground-truth parameters and zero inference errors.

We see that rankings are mostly frame-budget invariant. Qwen3-VL at its worst budget (5.76%) still beats InternVL at its best (4.28%). CodeBLEU is essentially constant. For Qwen, naming recall keeps rising with frames while value accuracy falls after 32. More frames enable the model to name more of the right parameters, but not better quantitative estimation. This supports the paper's central claim that the bottleneck is quantitative physical inference, not perception bandwidth.

A.15 Additional Cohen-κ Plots

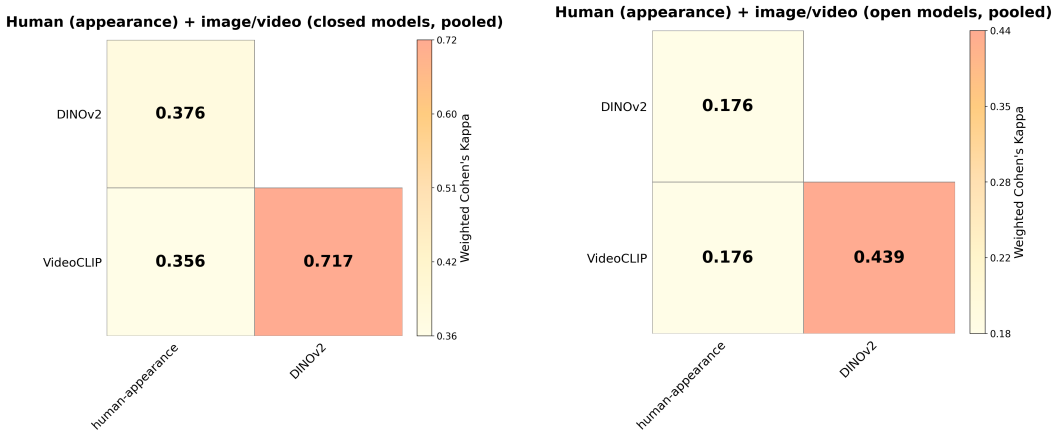
Figure 6 shows that the human-rated video appearance similarity score is less correlated with patch-wise similarity, but the correlation is definitely much higher than Figure 4's human-rated physics similarity. The reason is clear. Both these patchwise similarity metrics and human (when asked to rate based on visual similarity) focus on the pixel level match rather than physics understanding.

A.16 Transfer to real world

We scoped the paper to single-scene, short clips. But we think the synthetic setting is the right place to start. Our videos are the easy case: no sensor noise, no occlusion, clean lighting, and we even

Table 20: How the parameter estimation and coding performance of the two open-source models vary with different number of input frames for Qwen3-VL-30B and InternVL3.5-30B.

Model	Frames	Param. Acc.	Naming Recall	Value Acc.	No Errors	Video Save	CodeBLEU	Prompt Tok.
Qwen	8	5.84%	0.384	0.155	78.4%	82.7%	0.203	2,327
Qwen	16	6.23%	0.387	0.165	75.3%	80.9%	0.206	2,863
Qwen	32	**6.31%**	0.399	**0.168**	71.0%	72.8%	0.206	3,937
Qwen	60*	6.07%	**0.409**	0.157	75.3%	80.9%	0.206	5,468
Qwen	80	5.76%	0.408	0.143	77.8%	80.2%	0.208	6,190
InternVL	8	3.12%	0.247	0.126	48.1%	65.4%	0.194	3,905
InternVL	16	3.43%	0.256	0.132	49.4%	64.2%	0.196	6,016
InternVL	32*	4.05%	0.265	0.163	**50.6%**	**66.7%**	0.196	10,240
InternVL	60	**4.28%**	**0.273**	**0.168**	46.3%	60.5%	0.195	16,247



(a) (b)

Figure 6: Cohen's-κ plot of human-rated video appearance similarity score with the verifiable patch-wise similarity metrics (DinoV2 and VideoCLIP) for (a) closed and (b) open-source models.

hand the model the engine name and an experiment-agnostic code template so it is not penalized for engine unfamiliarity. Models still fail badly: 2.24/5 on inferring the demonstrated law, 7.7% parameter recovery within ±20%. A model that cannot recover friction from a clean render of a block on a ramp will not recover it from a noisy, partially occluded one. Reporting the failure in the absence of nuisance factors is more informative than reporting it in their presence, because it isolates the cause. Also, this measurement is possible synthetically. It's hard to annotate real videos with ground-truth mass, stiffness or restitution, etc., so a ±20% parameter-recovery metric cannot be defined on it easily.

We also observed two failures that are genuinely about physics rather than about guessing a simulator's generative parameters. (i) Models state physics laws more correctly than infer them from video. So the missing capability is binding perception to physics. (ii) Appearance is reproduced far better than dynamics; Figure 4 shows the patchwise metrics correlate strongly with each other but not with human physics judgment. Models match pixels and miss the dynamics, a physics gap.

A.17 Benchmark Novelty

The individual ingredients exist in literature, but no prior benchmark joins them. Prior work splits cleanly into two lines that have not met: property estimation from video (Zhan et al. 2025, ContPhy, Physion, ComPhy, PhysInOne) recovers a handful of latent values but never a program; physics code generation (PhysCodeBench, SimuScene) produces executable simulations but is text-conditioned and single-engine, so nothing is inferred from pixels. PhySimCode closes the loop: video to laws + parameters + runnable code to regenerated video scored back against the input. Table 1 shows

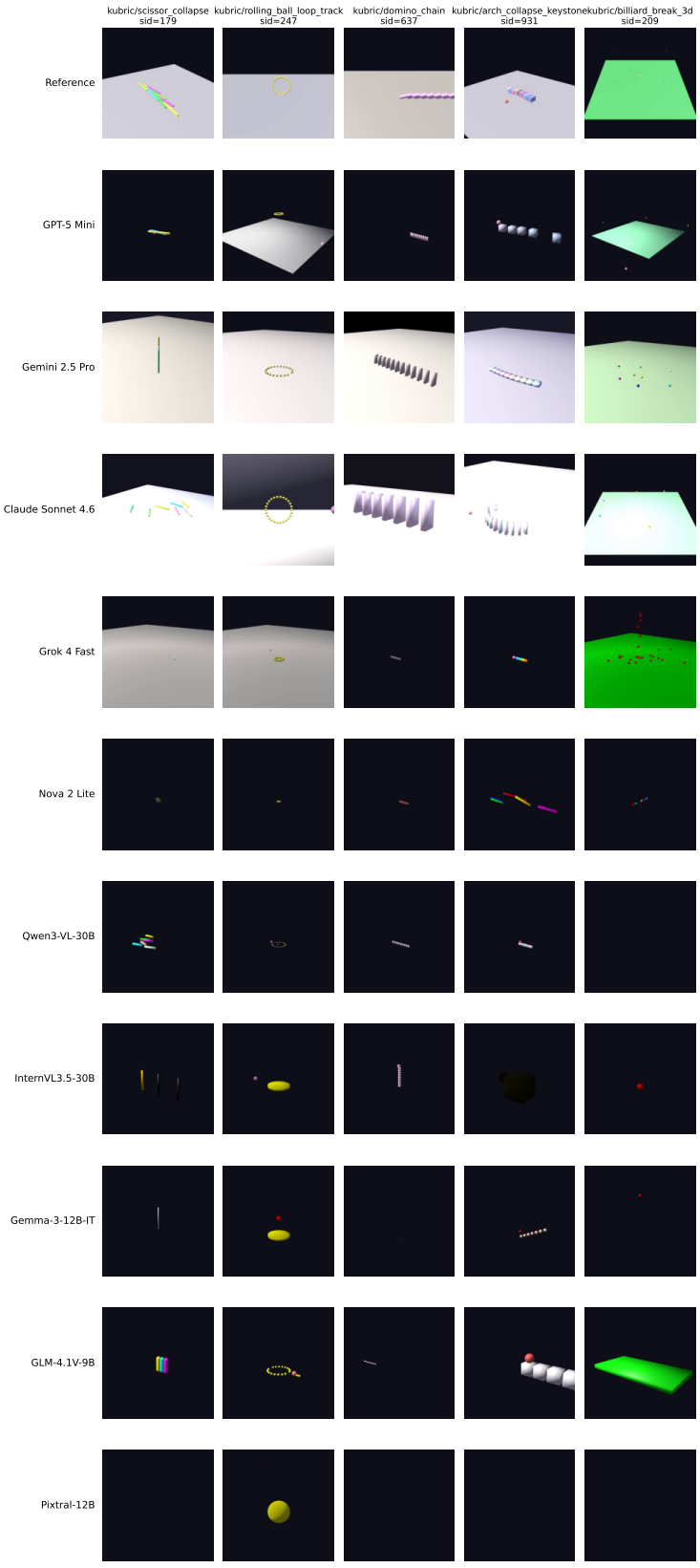


Figure 7: Videos rendered by generated code of various models

we are the only row with Video, CoT, Code, Physics laws, Engine-agnostic, and 2D+3D all present, at 162 phenomena vs. 4–71 in prior benchmarks. We annotate the governing law and ground it in video. We can separate what a model knows from what it can infer: retrieval vs. grounding. We treat code as a lossless, executable representation of scientific visual media. So PhySimCode gives some novel and useful insights about MLLM's physics understanding.