

PhysCodeGym: Benchmarking Coding Agents for Physically Verifiable 4D Reconstruction

Shuai Shao^{1,*,\dagger}, Jinchun He^{1,*}, Cunxin Fan^{1,*}, Zihan Guo^{2,3,*,\ddagger},
Shijian Wang⁴, Qihan Ren¹, Kangning Zhang¹, Zunnan Xu⁵, Haoyuan Li²,
Hao Wang⁶, Jianghao Lin^{1,\ddagger}, Weiwen Liu^{1,\ddagger}, Weinan Zhang^{1,\ddagger}

¹Shanghai Jiao Tong University ²Sun Yat-sen University ³Shanghai Innovation Institute

⁴Southeast University ⁵Nanyang Technological University ⁶Carnegie Mellon University



Figure 1. **PhysCodeGym** benchmarks coding agents on 156 video-to-Blender reconstruction tasks spanning rigid-body, liquid, deformable, and wind-driven motion. Examples pair reference frames (top) with agent reconstructions (bottom). Evaluation measures visual fidelity, dynamic behavior, and physical validity.

Abstract

Building physical simulations from video requires coding

*Equal contribution. All four co-first authors agree that each may list their own name first when presenting this work.

^{\dagger}Project lead. shaoshuai.ederson@sjtu.edu.cn.

^{\ddagger}Co-corresponding authors. Weinan Zhang: wnzhang@sjtu.edu.cn.

agents to translate observations into executable models and refine them through tool use. We introduce *PhysCodeGym*, a benchmark for physically verifiable 4D reconstruction with 156 tasks covering rigid-body, liquid, deformable, and wind-driven motion. Agents autonomously inspect videos, write and run Blender programs, and revise their scenes using simulation feedback. Six metrics combine rule-based

measurements with rubric-based judgments to assess visual and physical reconstruction quality. Program inspection, scene measurements, and counterfactual interventions test physical consistency under original and modified conditions. Experiments with frontier coding agents reveal a tradeoff between artifact quality and completion across reasoning-effort settings, while harness comparisons and agent traces link verification behavior with reconstruction performance. Case studies show that matching visible objects and events can coexist with physical defects. Counterfactual tests further expose latent contact and support failures in reconstructions that pass ordinary physical checks, highlighting the gap between reproducing observations and building reliable physical models. Code is available at <https://github.com/ShaoShuai0605/PhysCodeGym>.

1. Introduction

Physics-aware reconstruction enables the creation of simulation-ready objects and scenes for applications in animation [55], gaming [22], manufacturing [57], and robot learning [34]. Specialized methods estimate scene geometry and physical properties within predefined representations and simulation pipelines [9, 30, 52, 53]. More recently, coding agents have emerged as a promising path forward by generating executable scene programs and revising them through execution and visual feedback [49, 61].

Recent frontier models, such as GPT-6 Astra, demonstrate substantially improved capabilities in executable 4D scene construction, creating an urgent need to quantitatively assess their physical fidelity. We argue that a benchmark for this purpose should meet three requirements. First, **broad interaction coverage** should span diverse processes such as contact, fluid flow, and deformation [5]. Second, **hybrid assessments** should combine rule-based checks with rubric-based judgments [59]. Rule-based checks provide precise measurements but are limited to predefined criteria. Rubric-based judgments can assess broader task requirements [29, 43], though they may miss physical violations [32, 36]. Third, agreement with observed motion and appearance does not ensure correct behavior under altered physical conditions [6, 8, 45, 49, 60, 64], motivating **factual and counterfactual evaluation**. For example, hardcoded motion trajectories may make balls appear to collide even when contact is disabled. While prior benchmarks cover subsets of these design choices, we argue that considering them jointly provides a holistic view of the capabilities and limitations of frontier coding agents in 4D reconstruction.

To this end, we introduce PhysCodeGym, a benchmark for evaluating coding agents on physically verifiable 4D reconstruction (Fig. 1). Given a reference video, an agent writes a standalone Blender script to create an editable scene and render its animation. Our benchmark comprises 156

tasks spanning rigid-body, liquid, deformable, and wind-driven dynamics, with scenarios involving collisions, suspension, pouring, and cloth deformation.

To enable systematic evaluation across all 156 tasks (Sec. 4.3), we provide object checklists, 578 timestamped event annotations, and task-specific physical requirements. The evaluation protocol combine these annotations with LLM-based and rule-based judges to compute six metrics. Five assess the submitted video for structural and perceptual similarity, object accuracy, event completion, and motion overlap. To support event completion scoring, we specify failure and uncertainty criteria and aggregate three independent judgments per submission by majority vote. The sixth metric, physical validity, is assessed through a task-specific measurement that combines code and scene inspection with frame-by-frame analysis of native states, contacts, and forces. The scoring further incorporates counterfactual checks, in which copies of the scene are rerun under altered physical conditions. For example, our protocol redirects a billiard ball toward the table edge to check whether it falls after losing support. To verify that our protocol reliably detects physical violations, we perform manual checks by perturbing submitted scenes and injecting known defects.

We evaluate nine coding agents to characterize their strengths and failure modes in 4D physical reconstruction. We find that, while these agents can reconstruct recognizable objects, they often struggle to accurately capture motion and physical interactions (Tab. 2). Moreover, reconstructions with perfect object and event scores can violate physical constraints, including energy conservation (Sec. 5.2). Our harness-level comparison further reveals that, at high effort, Codex achieves higher visual and event scores, while Claude Code has slightly higher motion overlap (Fig. 5). Reasoning effort presents an additional tradeoff: low effort completes more tasks and leads overall under a fixed budget, whereas max effort scores higher among successful tasks (Sec. 5.2).

Our contributions are:

- **A benchmark for coding agents that building physical simulations.** PhysCodeGym uses 156 tasks to test coding agents’ ability to build physics-aware Blender scenes. The tasks cover gravity and support, contact and friction, fluids, deformation, and wind-driven motion.
- **A protocol for joint visual and physical evaluation.** We combine LLM-based and rule-based judges with factual and counterfactual checks. Our diagnostic cases include reconstructions that pass object and event checks but fail native-scene audits or targeted interventions.
- **An empirical study of coding agents for physical reconstruction.** We evaluate leading agents on PhysCodeGym and study how their results vary with harness choice and configured reasoning effort. We highlight their strengths, failure modes and shortcut behaviors, informing future development of physically aware coding agents.

Table 1. Comparison of programmatic scene generation and reconstruction benchmarks. **Task** distinguishes generation (Gen.) from reconstruction (Recon.); **Input** specifies the conditioning modality and source (Syn.: synthetic). **Visual Eval.** assesses rendered appearance or spatial layout. Under **Behavioral Fidelity**, **Event Eval.** checks target actions or outcomes, and **Motion Alignment** compares motion against a reference. Under **Physical Implementation**, **Internal Physics Checks** inspect scene structures, physical parameters, or simulation states, and **Mechanism Dependence** uses matched physical interventions to test whether the modeled mechanisms drive the observed behavior. **Harness** lists reported execution workflows or interfaces. ✓ included, ✓ limited scope, ✗ not evaluated under the column definition.

Benchmark	Task	Input	Visual Eval.	Behavioral Fidelity		Physical Implementation		Harness
				Event Eval.	Motion Alignment	Internal Physics Checks	Mechanism Dependence	
Code4D [67]	Gen.	Text	✓	✗	✗	✗	✗	Multi-agent code loop
4DBuildBench [32]	Gen.	Text	✓	✓	✗	✓	✗	Multi-agent MCP
PhysCodeBench [56]	Gen.	Text	✓	✓	✗	✓	✗	Direct codegen. / SMRF
SceneActBench [68]	Recon.	Syn. frames	✓	✗	✓	✗	✗	Shared MCP
VisPhyBench [31]	Recon.	Syn. frames	✓	✗	✓	✓	✗	Codegen. + repair
MPMWorlds [26]	Recon.	Syn. video	✓	✗	✓	✗	✗	Scaffolded codegen.
BVB [48]	Recon.	Real video	✓	✗	✓	✗	✗	Mini-BVB
4DCodeBench [45]	Recon.	Real + Syn. video	✓	✓	✓	✓	✗	Native CLIs / Stirrup
Ours	Recon.	Video	✓	✓	✓	✓	✓	Native CLIs (all models)

2. Related Work

Video-based Physical Reconstruction. Video-based physical reconstruction requires recovering scene geometry, physical properties, and dynamics from observations [44, 62]. Existing methods couple geometry and appearance representations with differentiable continuum simulation, mesh-free reduced-order simulation, or spring–mass dynamics to reconstruct simulatable scenes for subsequent prediction and re-simulation [9, 21, 30, 70]. Language models also support physical scene construction: Δ YNAMICS generates structured scene configurations, evaluates reconstruction through MuJoCo re-simulation, and supports editing gravity and initial velocities [24]. OVOW recovers instance meshes, poses, and deformations from monocular video, using ground-plane estimation and contact projection to assemble scenes for downstream simulation [10]. These methods study how to recover simulatable scenes from video. PhysCodeGym turns this reconstruction objective into an evaluation task for coding agents, testing whether their generated scene programs satisfy both reference observations and task-specific physical requirements.

Programmatic Scene Construction and Evaluation. Earlier inverse-graphics systems represented scenes with probabilistic programs in Picture and structured scene descriptions in Neural Scene De-rendering [28, 54]. Programmatic methods perform 3D generation and editing through code [13, 18, 19, 27, 47, 50, 58, 66], with benchmarks covering graphics editing, text- or image-conditioned 3D modeling, and image-to-scene inverse rendering [12, 14, 15, 33, 69]. Thinking in Blender reconstructs single images as editable Blender programs by progressively refining geometry, materials, composition, and lighting [17]. VIGA extends code iteration and visual feedback to dynamic scenes [61], while Code2Worlds

generates dynamic scenes from text and refines them through motion critique and code revision [67]. For physical implementation, SimWorlds’ 4DBuildBench separately evaluates visual output and internal physics, auditing solver configurations, constraints, and animation substitutes [32]; PhysCodeBench evaluates simulation behavior through physical assertions and conservation diagnostics read from engine states [56]. These dynamic generation benchmarks check scenes and physical implementations against textual requirements; video reconstruction additionally requires agreement with reference observations of appearance and behavior.

Video-to-Program Reconstruction Benchmarks. Video-to-program benchmarks evaluate agents’ ability to recover scene structure and dynamic behavior. BVB reconstructs real videos as Blender animations, evaluating spatiotemporal fact retention and representation-level similarity of layout and dynamics under a protocol that prohibits physics simulation [48]. SceneActBench’s dynamic task evaluates layout and keyframed trajectory recovery with supplied assets [68]. VisPhyWorld and MPMWorlds recover executable simulations from synthetic visual observations, studying physical-parameter and motion reconstruction, and future prediction of multi-material dynamics, respectively [26, 31]. The closely related 4DCodeBench reconstructs real and synthetic videos as executable dynamic scenes, evaluating appearance, geometry, and motion alongside state/event questions and mesh-integrity diagnostics, with direct 3D ground-truth comparisons on its synthetic subset. However, its task permits keyframes, prescribed motion, and physical simulation as alternative dynamic representations, primarily evaluating submissions by reconstruction fidelity [45]. Code as Worlds constructs, verifies, and refines executable worlds from text or real videos, supports re-execution un-

der changed physical conditions, and uses these worlds for physical-reasoning supervision [49]. Its emphasis is on discovering executable worlds and using them for physical reasoning; PhysCodeGym instead makes the physical implementation of core motion a requirement of video reconstruction and tests mechanism dependence through independent evaluation of submitted artifacts. Evaluation combines reference-event and motion alignment, scene and runtime checks, and matched interventions on contacts, support, or actuation, comparing behavior before and after intervention to determine whether reconstructed motion depends on task-required physical mechanisms.

Table 1 compares representative benchmarks by task, input, visual evaluation, behavioral fidelity, physical implementation evaluation, and execution harness.

3. PhysCodeGym: Dataset and Task

3.1. Dataset

The dataset contains 156 tasks covering rigid-body, liquid, deformable, and wind-driven motion; detailed category definitions and task statistics are provided in Appendix A.1.

Design principles. PhysCodeGym rests on four principles. (P1) Visible physical causation: the motion in every reference clip is caused by a visible physical process whose cause can be inferred from the footage, such as falling, rolling, collision, rotation, swinging, liquid pouring and splashing, deformable motion, and wind-driven motion. (P2) Simulator expressibility: the process can be expressed within the simulation capabilities of Blender; combustion, cutting, tearing and fracture, siphoning and absorption, liquid mixing, and magnetic forces are excluded. (P3) Mechanism-level taxonomy: every task has one mutually exclusive main category, and rigid-body and liquid tasks are further divided into subcategories by their dominant motion mechanism, so that results can be analyzed per mechanism. (P4) Observation-grounded annotation: annotations describe only what the reference video shows, with timestamped evidence; what it does not show, such as the path of an object while it briefly leaves the frame, is not required.

Collection pipeline. Following (P1) and (P2), we collect candidate videos from Pexels [42], Physics-IQ [37], BAAI-DataCube Physics-aware Videos [4], and IRIS [25]. We combine automated checks, visual agent review, and manual selection to identify clearly observable physical processes that can be reconstructed in Blender, prioritizing stable cameras, simple backgrounds, unobstructed events and diverse physical phenomena. From videos at least 8 seconds long, we select a continuous 8-second segment with pronounced motion and standardize it to 3840×2160 resolution, 30 FPS, and 240 frames while preserving the original playback speed.

After deduplication and final manual review, we retain 156 clips, each defining one task.

Annotation. Each task includes three structured annotations that specify the objects to reconstruct, the events to reproduce, and the physical constraints to satisfy. (i) The object checklist describes object categories, counts, appearance attributes, and initial spatial layout. (ii) Event annotations identify the participating objects and specify key actions, their temporal order, and the attainment and persistence of final states, supported by timestamped video evidence. (iii) Physical requirements are recorded in manually annotated rubrics that define permitted initial conditions, actuation, and equivalent representations, together with criteria for checking the physical mechanisms of the reconstructed scene. For selected tasks, counterfactual verification alters forces or contact conditions and re-simulates the scene to check whether its response is consistent with the task’s physical requirements. These annotations support object accuracy, event score, and physical validity, respectively (Sec. 4.3).

We draft object checklists with model assistance and event annotations from reference observations, then check both using reference videos and unrelated-video controls.

3.2. Task Definition

Every task in PhysCodeGym poses the same problem, reconstructing the physical process recorded in a video, and each submission is scored along two axes, each targeting a distinct capability: visual reconstruction (reproducing what the reference video shows) and physical reconstruction (reproducing the dynamic process through modeled physics). Figure 2 illustrates the reconstruction and verification pipeline.

Physical-process reconstruction. A video records only one presentation of a dynamic scene from one viewpoint. To understand why the process unfolds as it does, or how it would change under a different viewpoint or a modified condition, one needs not another similar video but a model of the physical process that can be run, inspected, and modified again. Executable worlds support re-simulation under changed conditions and rendering from new viewpoints [49]. PhysCodeGym therefore sets the reconstruction target to be a program. Given a reference video, a coding agent must write a scene program that starts from an empty Blender scene, builds geometry, materials, cameras, and lighting, and sets the physical properties, initial conditions, constraints, and forces of the objects; executing it yields a native scene that evolves over time, together with its rendered sequence. The submitted reconstruction is the program and the native scene it produces; the rendered sequence is only one presentation of this process from the reference viewpoint. We choose Blender because it provides modeling, rigid-body, cloth, fluid, and force-field simulation, and rendering in a

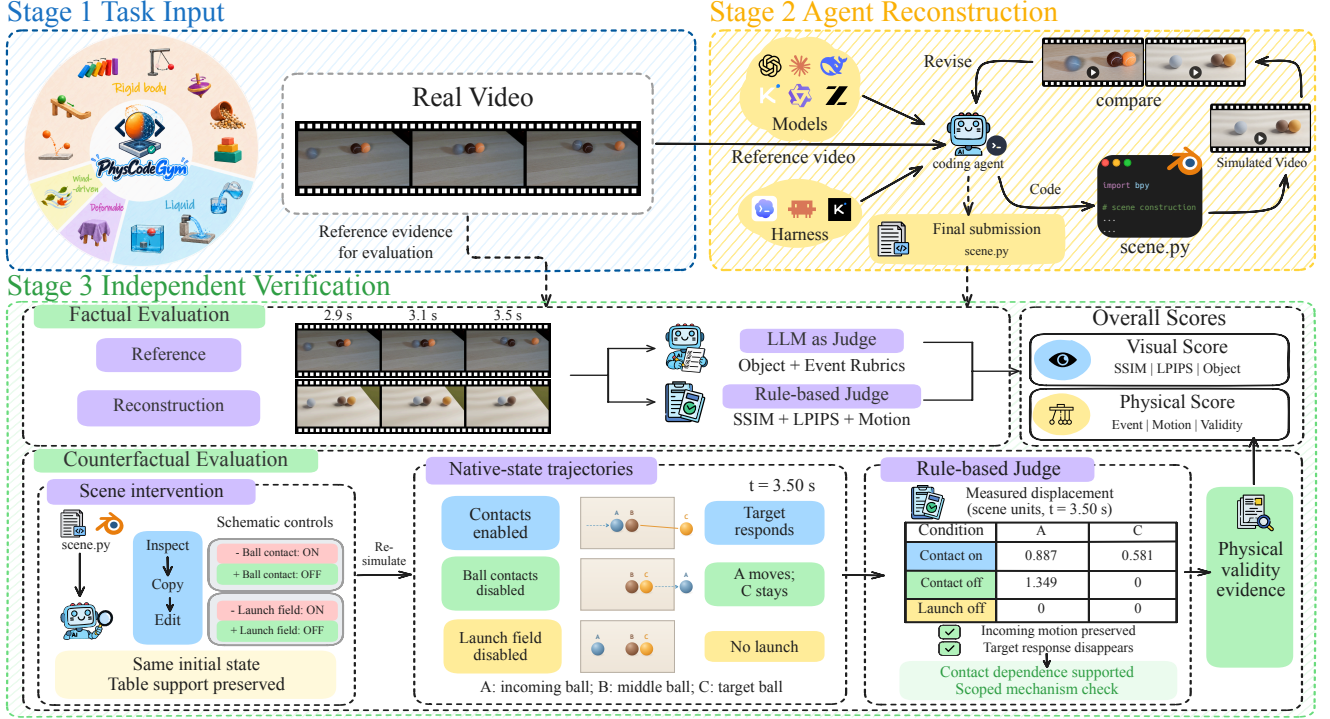


Figure 2. **Reconstruction and verification pipeline of PhysCodeGym.** **Stage 1:** A reference video defines the reconstruction task and supplies evidence for evaluation. **Stage 2:** A coding agent iterates between observing the video, writing a Blender scene program, running it, and comparing rendered previews, then submits the scene program. **Stage 3:** Independent evaluation combines rubric-based object and event judgments with image and motion metrics, while native-scene inspection and controlled re-simulation assess physical validity. In the illustrated contact test, disabling ball contacts preserves incoming motion but removes the target response; disabling the launch field removes the incoming motion as well. Visual and physical scores are reported separately.

single environment [7], and because both the scene structure and the simulation state are accessible through its scripting interface, so an agent’s decisions about objects, contacts, and the causes of motion can be inspected directly in the scene.

Valid reconstruction. A valid reconstruction must satisfy two conditions: (i) it reproduces the observations, matching the appearance, layout, and visible events of the reference video; and (ii) the observed motion is produced by the modeled physical process. The same footage can also be reproduced through keyframes, scripted poses, or hidden supports; such a scene may match the frames, but it does not explain the motion and does not respond correctly when conditions change. Prescribed trajectories can achieve high reconstruction scores while leaving physical mechanisms underconstrained [45]. The motion of core objects must therefore come from the physics solver, forces, constraints, and initial conditions. This excludes scripted trajectories and force fields with no counterpart in the video, such as an attractor that steers a ball along the observed path. Explicitly modeled external actuators are allowed, provided that the objects they act on respond through physical interaction. Physical parameters are not always identifiable from

the observed motion: in a modeled free-fall and inclined-sliding scenario, friction can be inferred while mass remains unobservable [23]. Monocular reconstruction also faces ambiguity in scale and unobserved geometry, for which physical dynamics can provide additional constraints [44]. PhysCodeGym therefore does not require recovering the true values of these parameters or the exact hidden structure, and accepts any implementation that reproduces the observations and satisfies the physical requirements of the task.

Observation and submission. The agent works in a terminal environment and uses command-line tools to observe the reference video and operate Blender. The observation tools return basic video metadata, specified frames, frame grids, and clips sampled at a given rate. Neither the source program nor the physical parameters of the reference scene are provided. The build tools run the scene program, inspect the saved scene structure, and render single-frame previews of the simulation state. The agent submits the scene program `scene.py`. The program must be self-contained, build the entire scene procedurally, and bake the simulation cache before rendering. It outputs the native scene file and a rendered sequence of 240 frames at 30 FPS and 640×360 , whose

duration matches that of the reference video, 8 seconds. The program may not decode media, access the network, spawn subprocesses, or import modules dynamically, which ensures that the scene is generated entirely by modeling statements. Before submission, a checking tool verifies the format and policy requirements and reports any problems, so that the agent can fix them before the run ends. The unified output specification allows every submission to be re-executed in an independent environment and compared with the reference video frame by frame.

Scoring. We report separate visual and physical scores, their component metrics, and the overall score defined in Sec. 4.3. The visual score measures how closely the rendered sequence resembles the reference video on screen, and averages three visual metrics: structural similarity (SSIM) and perceptual similarity (LPIPS) compare the frames at the pixel and perceptual levels, and object accuracy checks the rendered objects against the object checklist. The physical score measures the dynamic process of the scene, and averages three physical metrics: the event score checks whether the annotated events occur, the motion score compares where and for how long motion occurs, and physical validity checks the scene against the physical requirements. Except for physical validity, which reads the program and the native scene, all metrics use only the rendered sequence. Each metric is also reported individually, which separates failures that a single number would conflate [20], such as a scene that matches the frames through scripted motion and a physically valid scene that misses the observed events. Metric definitions are given in Sec. 4.3.

4. Experiments

4.1. Setup

We build the execution framework of PhysCodeGym on Harbor [16], integrating our Blender reconstruction environment and benchmark-specific evaluation pipeline. Appendix A.3 describes this integration and its relationship to the agent harnesses used in our experiments.

All agents work in the terminal environment of Sec. 3.2 and typically iterate between observing the reference, writing the program, running the simulation, comparing previews, and revising the scene. Each launch of a Blender process counts as one iteration, and the iteration limit constitutes the generation budget; observation commands do not count as iterations, but they consume running time and context, and the whole run is also bounded by a total time limit. Unless stated otherwise, all agents use high reasoning effort, with a run-time limit of 3 hours and at most 20 Blender launches per task. Each model is initially scheduled once per task. Audited serving-infrastructure failures are handled by separately recorded recovery runs, preserving the original trials;

verifier-side failures are repaired by re-executing archived programs. Errors caused by the agent’s own requests, such as exceeding the context window or sending a malformed image, count as failures of the agent. A task that is run more than once is scored by its latest run that produced a rendered sequence, not by its best-scoring run; a task for which no run produced a rendered sequence receives zero.

4.2. Models

We evaluate coding agents that autonomously write and run code in a terminal environment. The main comparison in Tab. 2 uses three harnesses: (i) Codex [38] for GPT-5.6-Sol and GPT-5.6-Luna [40], GPT-6-Luna [41], and GPT-6-Astra [39]; (ii) Claude Code [2] for Claude Sonnet 5 [3], GLM-5.3-Flash [63], DeepSeek-V4.1-Flash [11], and Qwen3.8-Flash [1]; and (iii) Kimi Code for Kimi K2.8 Preview, requested as `kimi-for-coding` [35]. Prior work explores sampling, revision, and verification strategies for language agents [71], while studies of LLM reasoning show that the effectiveness of test-time computation depends on inference strategy and problem difficulty [46]. We also compare DeepSeek-V4.1-Flash at low, high, and max configured reasoning effort in Claude Code, and at high effort in Codex.

4.3. Evaluation

Each submission is evaluated with three visual metrics, which compare the rendered sequence with the reference video, and three physical metrics, which assess the dynamic process of the scene (Sec. 3.2).

Visual metrics. Structural similarity is the per-frame SSIM [51] between the rendered sequence and the reference video. Perceptual similarity is $\text{clip}(1 - d, 0, 1)$, where d is the per-frame LPIPS [65] (AlexNet) distance. Object accuracy summarizes a vision-language model’s judgments against the object checklist as an aggregate pass rate. Structural and perceptual similarity measure closeness at the pixel and perceptual levels, while object accuracy focuses on object-level correctness and is insensitive to subtle differences in lighting and texture.

Event score. A vision-language model judges the candidate video against the event annotation item by item and cites time intervals in the candidate video as evidence. When judging, the model sees still images of each role from the reference video, to confirm object identities, together with the candidate video and supplementary temporal frames; it does not receive the action descriptions of the reference video, and decides whether each event occurs based on the candidate video alone. Each candidate is judged independently three times, and the majority verdict is taken per item. The event score is the weighted fraction of passed items; if a key event that defines the task fails, the score is capped at zero. For

Table 2. Visual and physical reconstruction scores on PhysCodeGym (higher is better). Metric definitions and aggregation follow Sec. 4.3. Best and second-best scores are **bold** and underlined.

Harness	Model	Visual				Physical				$S_{\text{tot}} \uparrow$
		$S_{\text{SSIM}} \uparrow$	$S_{\text{LPIPS}} \uparrow$	$S_{\text{obj}} \uparrow$	$S_{\text{vis}} \uparrow$	$S_{\text{evt}} \uparrow$	$S_{\text{mot}} \uparrow$	$S_{\text{val}} \uparrow$	$S_{\text{phys}} \uparrow$	
Codex	GPT-6-Astra	0.610	0.561	0.909	0.693	0.805	0.324	0.768	0.633	0.663
	GPT-5.6-Sol	<u>0.550</u>	<u>0.448</u>	<u>0.835</u>	<u>0.611</u>	<u>0.687</u>	<u>0.210</u>	<u>0.690</u>	<u>0.529</u>	<u>0.570</u>
	GPT-6-Luna	0.518	0.363	0.633	0.505	0.393	0.128	0.555	0.359	0.432
	GPT-5.6-Luna	0.506	0.355	0.639	0.500	0.448	0.157	0.564	0.391	0.446
Claude Code	DeepSeek-V4.1-Flash	0.514	0.398	0.615	0.509	0.446	0.200	0.522	0.391	0.450
	Claude Sonnet 5	0.513	0.315	0.481	0.436	0.368	0.111	0.559	0.347	0.392
	Qwen3.8-Flash	0.354	0.239	0.410	0.334	0.307	0.090	0.350	0.252	0.293
	GLM-5.3-Flash	0.286	0.226	0.362	0.291	0.240	0.103	0.295	0.213	0.252
Kimi Code	Kimi K2.8	0.435	0.304	0.477	0.405	0.349	0.110	0.448	0.302	0.354

example, if the ball does not roll around the tray, no credit is given even if the tray layout is correct.

Motion score. The motion score adopts the motion mask of Physics-IQ [37]: grayscale frames are Gaussian-blurred and differenced against a running-average background, then thresholded and morphologically processed to obtain the motion region of each frame. We compare the reference and candidate masks with two IoU measures. Spatial IoU compares the regions where motion occurs at any point in the video. Weighted IoU also reflects how long motion lasts: with w_p and $\hat{w}_p$ the fractions of frames in which pixel p is moving in the reference and the candidate, it equals $\sum_p \min(w_p, \hat{w}_p) / \sum_p \max(w_p, \hat{w}_p)$. The motion score is the average of the two. This metric does not require recognizing objects, and complements motion details not covered by the event annotation, such as the decay of swing amplitude or the extent of a liquid stream.

Physical validity. An agent verifies physical validity against manually annotated task rubrics. It inspects the submitted program and native scene, executes Blender/Python measurements, and supports each judgment with evidence. The main-table score S_{val} evaluates the submitted program and its Blender scene, counting each task once per configuration. For selected tasks, *counterfactual verification* intervenes on forces, contacts, or initial conditions and re-simulates the scene to test the physical dependencies specified by the rubric. Counterfactual checks are scored as criteria of the physical-validity rubric. The verification procedure and intervention example appear in Appendix B.

Score aggregation. We report visual, physical, and overall scores:

$$\begin{aligned}
 S_{\text{vis}} &= \frac{1}{3} (S_{\text{SSIM}} + S_{\text{LPIPS}} + S_{\text{obj}}), \\
 S_{\text{phys}} &= \frac{1}{3} (S_{\text{evt}} + S_{\text{mot}} + S_{\text{val}}), \\
 S_{\text{tot}} &= \frac{1}{2} (S_{\text{vis}} + S_{\text{phys}}).
 \end{aligned} \tag{1}$$

where S_{SSIM} , S_{LPIPS} , and S_{obj} are structural similarity, perceptual similarity, and object accuracy, and S_{evt} , S_{mot} , and S_{val} are the event score, the motion score, and physical validity. All three summaries are reported together with the component metrics. For a fully measured task, S_{tot} is also the arithmetic mean of the six component metrics. In model comparisons, a task for which no valid artifact is submitted receives zero on every metric and is counted in the denominator. For the provisional validity results, scores use the lower endpoints of the rubric-score intervals; existing-artifact technical gaps are omitted while registered missing-artifact zeros remain included. The physical aggregate is first computed per task and then averaged on exactly the same available-case subset as validity. It therefore does not average the differently covered model-level columns of Tab. 2.

For model-aggregate tables, S_{tot} directly averages the displayed visual and physical score columns. The equivalence to a six-component mean over tasks requires all six metrics to share the same averaging set; it does not hold for differently covered columns merely by averaging their means.

5. Results & Analysis

We first present visual and physical reconstruction scores and resource use, then examine reasoning effort, harness settings, and cases exposing gaps between visual fidelity and physical correctness.

5.1. Benchmark Results

Table 2 reports six metrics and aggregate scores for nine model-harness configurations at default high effort, follow-

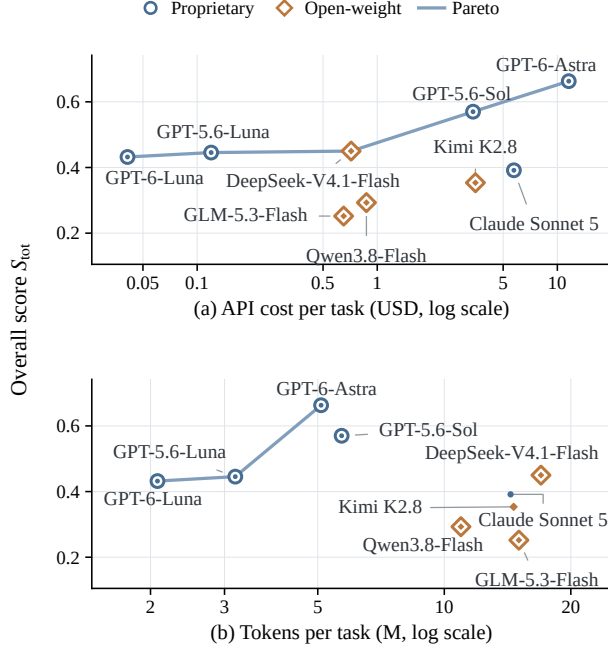


Figure 3. Overall score S_{tot} versus (a) estimated API cost and (b) recorded token use per task. Greater cost or token use does not consistently yield better reconstruction across models.

ing Sec. 4.3.

Overall performance. GPT-6-Astra leads in overall, visual, and physical scores (0.663, 0.693, and 0.633); GPT-5.6-Sol ranks second overall at 0.570.

Performance across dimensions. All nine configurations score higher on objects than events; GPT-5.6-Sol scores 0.835 and 0.687, respectively. Motion remains weak, with GPT-6-Astra reaching only 0.324. Rankings also vary: GPT-6-Luna has a slightly higher visual score but lower event and motion scores than GPT-5.6-Luna.

Resource use and quality. Figure 3 compares quality against estimated API cost and recorded token use. GPT-5.6-Sol and GPT-6-Astra lie on the estimated-cost Pareto frontier: Sol scores 0.570 at approximately \$3.40 per task, while Astra reaches 0.663 at \$11.57, or 3.4 times the cost.

Greater token use need not improve reconstruction. DeepSeek-V4.1-Flash records 16.98M cumulative tokens per task, nearly 3 times GPT-5.6-Sol’s 5.70M, but scores 0.450; its estimated cost is lower (\$0.72 per task). Cumulative tokens include repeated context across interactions and cannot directly measure effective reasoning. We next examine how effort and harness settings relate to budget use.

5.2. Analysis

We analyze how artifact quality and completion shape aggregate scores through DeepSeek-V4.1-Flash effort and harness comparisons, followed by physical failure cases.

Reasoning effort and task completion. Figure 4 compares low, high, and max with the same limits of 3 hours and 20 Blender launches. Low leads overall (0.476) when missing artifacts receive zero. Among produced artifacts, max scores 0.547 versus low’s 0.527, and also has the highest mean on the shared subset. Gap decomposition links low’s aggregate advantage mainly to completion, while max has longer runtimes and more recorded output tokens. Appendix C.3 details this completion–quality tradeoff under limited budgets.

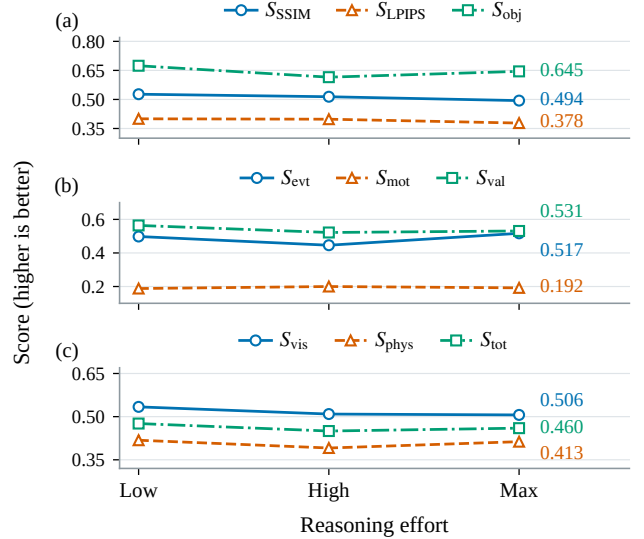


Figure 4. Comparison of reasoning effort levels. We evaluate DeepSeek-V4.1-Flash on (a) visual metrics, (b) physical metrics, and (c) aggregate scores. Under the same budget of 3 hours and 20 Blender launches, low effort achieves the highest overall score, largely because it completes more tasks than max effort.

Harness and verification. Figure 5 compares DeepSeek-V4.1-Flash at high effort in Claude Code and Codex. Codex scores 0.479 overall versus 0.450, and 0.550 on events versus 0.446, with slightly lower motion overlap. In the rolling-wood and wine-pouring traces, Codex identifies problems, revises, reruns, and inspects outputs; Claude Code’s final revisions lack successful execution verification within the launch budget. This pattern may relate to proactive verification guidance, check timing, and budget allocation (Appendix D.4).

Physical failures beyond visual fidelity. Physical checks reveal flaws even when object and event judgments pass. All

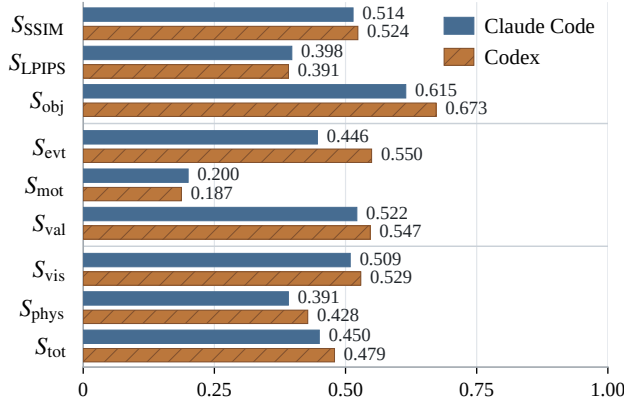


Figure 5. Claude Code and Codex harness comparison. Visual, physical, and aggregate scores of DeepSeek-V4.1-Flash at high reasoning effort. In this comparison, Codex achieves a higher overall score, with the largest gain in event reconstruction despite slightly lower perceptual similarity and motion overlap.

six reconstructions in Appendix D.2 score $S_{obj} = S_{evt} = 1$, yet program inspection and scene measurements expose failures such as unexplained energy increases and interpenetration. Changing support, contact, or initial conditions further exposes latent defects beyond the original trajectories (Appendix D.3).

6. Conclusion

We introduced PhysCodeGym, a 156-task benchmark for evaluating how coding agents reconstruct physical processes from video as executable Blender programs. Visual and dynamic metrics assess rendered outcomes; verification agents inspect scene programs and perform measurements against task rubrics, including counterfactual rubric checks for selected tasks. Our results highlight a gap between matching visible behavior and constructing physically valid scenes: object and event checks can pass while native-scene diagnostics identify physical inconsistencies. The current benchmark is limited to processes that Blender can express and the physical checks specified by its rubrics. Future work can extend PhysCodeGym to more simulation engines and cover more physical interactions.

References

- [1] Alibaba Cloud. Qwen3.8-Flash - Alibaba Cloud. <https://www.alibabacloud.com/help/en/model-studio/qwen3-8-flash>. 6
- [2] Anthropic. Claude Code by Anthropic | AI Coding Agent, Terminal, IDE. <https://claude.com/product/claude-code>. 6
- [3] Anthropic. Introducing Claude Sonnet 5. <https://www.anthropic.com/news/claude-sonnet-5>, 2026. 6
- [4] BAAI-DataCube. BAAI-DataCube/Physics-Aware-Videos · Datasets at Hugging Face. <https://huggingface.co/datasets/BAAI-DataCube/Physics-aware-videos>, 2026. 4, 14
- [5] 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 *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. 2
- [6] Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. CoPhy: Counterfactual Learning of Physical Dynamics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2
- [7] Blender Foundation. Blender Python API. <https://docs.blender.org/api/current/>. 5
- [8] Kunlin Cai, Rui Song, Jinghui Zhang, Kaiyuan Zhang, Pranav Bodapati, Alicia Yu, Fnu Suya, Mohammad Rostami, Jiaqi Ma, and Yuan Tian. What-If World: A Causal Benchmark for General World Models in Embodied Scenarios. *arXiv preprint arXiv:2605.27589*, 2026. 2
- [9] Chuhan Chen, Zhiyang Dou, Chen Wang, Yiming Huang, Anjun Chen, Qiao Feng, Jiatao Gu, and Lingjie Liu. Vid2Sim: Generalizable, Video-Based Reconstruction of Appearance, Geometry and Physics for Mesh-Free Simulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 26545–26555. Computer Vision Foundation / IEEE, 2025. 2, 3
- [10] Junhao Chen, Boran Zhang, Mingjin Chen, Henghaofan Zhang, Saining Zhang, Congcong Zhu, Hao Zhao, Ruqi Huang, Zhihao Li, and Yufei Wang. One Video, One World: Turning Monocular Video into Physical 4D Scenes. In *Computer Vision - ECCV 2026 - 19th European Conference, Malmö, Sweden, September 8-12, 2026, Proceedings, Part X*, pages 271–293. Springer, 2026. 3
- [11] DeepSeek. Introducing DeepSeek-V4.1-Flash: Smarter, Faster, More Efficient. <https://www.deepseek.com/en/news/deepseek-v4-1-flash/>, 2026. 6
- [12] Anna C. Doris, Jacob Thomas Sony, Ghadi Nehme, Era Syla, Amin Heyrani Nobari, and Faez Ahmed. CADBench: A Multimodal Benchmark for AI-Assisted CAD Program Generation. *arXiv preprint arXiv:2605.10873*, 2026. 3
- [13] Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, and Benyou Wang. BlenderLLM: Training Large Language Models for Computer-Aided Design with Self-Improvement. *arXiv preprint arXiv:2412.14203*, 2024. 3
- [14] Yipeng Gao, Lei Shu, Genzhi Ye, Xi Xiong, Ameesh Makadia, Meiqi Guo, Laurent Itti, and Jindong Chen. 3DCodeBench: Benchmarking Agentic Procedural 3D Modeling via Code. *arXiv preprint arXiv:2606.01057*, 2026. 3
- [15] Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas J. Guibas. BlenderGym: Benchmarking Foundational Model Systems for Graphics Editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 18574–18583. Computer Vision Foundation / IEEE, 2025. 3
- [16] Harbor Framework Team. Harbor: A Framework for Evaluating and Optimizing Agents and Models in Container Environments, 2026. doi:10.5281/zenodo.20953922. 6, 19
- [17] Guangzhao He, Rundong Luo, Wei-Chiu Ma, and Hadar Averbuch-Elor. Thinking in Blender: Staged Executable Inverse Graphics with Vision-Language Models. *arXiv preprint arXiv:2606.02580*, 2026. 3
- [18] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong

- Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. SceneCraft: An LLM Agent for Synthesizing 3D Scenes as Blender Code. In *Forty-First International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, pages 19252–19282. PMLR / OpenReview.net, 2024. 3
- [19] Ian Huang, Guandao Yang, and Leonidas J. Guibas. Blender-Alchemy: Editing 3D Graphics with Vision-Language Models. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXIX*, pages 297–314. Springer, 2024. 3
- [20] Swarnim Jain and Shangzhe Wu. RigidBench: Evaluating Rigid-Body Physics in Video Generation Models. *arXiv preprint arXiv:2608.15555*, 2026. 6
- [21] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pages 7219–7230. IEEE, 2025. 3
- [22] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, and Chenfanfu Jiang. VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality. In *ACM SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024, <https://arxiv.org/abs/2401.16663>. 2
- [23] Rama Krishna Kandukuri, Jan Achterhold, Michael Möller, and Joerg Stueckler. Learning to Identify Physical Parameters from Video Using Differentiable Physics. In *Pattern Recognition - 42nd DAGM German Conference, DAGM GPCR 2020, Tübingen, Germany, September 28 - October 1, 2020, Proceedings*, pages 44–57. Springer, 2021. 5
- [24] Chia-Hsiang Kao, Cong Phuoc Huynh, Chien-Yi Wang, Noranart Vesdapunt, Stefan Stojanov, Bharath Hariharan, Oleksandr Obiednikov, and Ning Zhou. Dynamics: Language-Based Representation for Inferring Rigid-Body Dynamics from Videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pages 42364–42374. Computer Vision Foundation, 2026. 3
- [25] Rasul Khanbayov, Mohamed Rayan Barhdadi, Erchin Serpedin, and Hasan Kurban. IRIS: A Real-World Benchmark for Inverse Recovery and Identification of Physical Dynamic Systems from Monocular Video. In *Computer Vision - ECCV 2026 - 19th European Conference, Malmö, Sweden, September 8-12, 2026, Proceedings, Part LVI*, pages 366–383. Springer, 2026. 4, 14
- [26] Žiga Kovačič and Kevin Ellis. MPMWorlds: Material-Point-Method Simulations for Inferring and Extrapolating Physical Dynamics. *arXiv preprint arXiv:2606.01538*, 2026. 3
- [27] Peter Kulits, Haiwen Feng, Weiyang Liu, Victoria Fernandez Abrevaya, and Michael J. Black. Re-Thinking Inverse Graphics with Large Language Models. *Transactions on Machine Learning Research*, 2024. 3
- [28] Tejas D. Kulkarni, Pushmeet Kohli, Joshua B. Tenenbaum, and Vikash Mansinghka. Picture: A Probabilistic Programming Language for Scene Perception. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4390–4399. IEEE Computer Society, 2015. 3
- [29] Seongyun Lee, Seungone Kim, Sue Hyun Park, Geewook Kim, and Minjoon Seo. Prometheus-Vision: Vision-Language Model as a Judge for Fine-Grained Evaluation. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and Virtual Meeting, August 11-16, 2024*, pages 11286–11315. Association for Computational Linguistics, 2024. 2
- [30] 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. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 2, 3
- [31] Jiarong Liang, Max Ku, Ka-Hei Hui, Ping Nie, and Wenhu Chen. VisPhyWorld: Probing Physical Reasoning via Code-Driven Video Reconstruction. *arXiv preprint arXiv:2602.13294*, 2026. 3
- [32] Chunjiang Liu, Xiaoyuan Wang, Haoyu Chen, Yizhou Zhao, Ming-Hsuan Yang, and László A. Jeni. SimWorlds: A Multi-Agent System for Dynamic 3D Scene Creation. *arXiv preprint arXiv:2607.01766*, 2026. 2, 3
- [33] Hengyu Liu, Chenxin Li, Zhengxin Li, Yipeng Wu, Wuyang Li, Zhiqin Yang, Zhenyuan Zhang, Yunlong Lin, Sirui Han, and Brandon Y. Feng. IR3D-Bench: Evaluating Vision-Language Model Scene Understanding as Agentic Inverse Rendering. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. 3
- [34] Haozhe Lou, Mingtong Zhang, Haoran Geng, Hanyang Zhou, Sicheng He, Zhiyuan Gao, Siheng Zhao, Jiageng Mao, Pieter Abbeel, Jitendra Malik, Daniel Seita, and Yue Wang. D-REX: Differentiable Real-to-Sim-to-Real Engine for Learning Dexterous Grasping. In *The Fourteenth International Conference on Learning Representations*, 2026, <https://openreview.net/forum?id=13jshGCK9i>. 2
- [35] Moonshot AI. What's New | Kimi Code Docs. <https://www.kimi.com/code/docs/en/kimi-code/whats-new.html>. 6
- [36] Saman Motamed, Minghao Chen, Luc Van Gool, and Iro Laina. TRAVL: A Recipe for Making Video-Language Models Better Judges of Physics Implausibility. *arXiv preprint arXiv:2510.07550*, 2025. 2
- [37] Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do Generative Video Models Understand Physical Principles? In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2026, Tucson, AZ, USA, March 6-10, 2026*, pages 948–958. IEEE, 2026. 4, 7, 14
- [38] OpenAI. Codex in ChatGPT | AI Coding Agents for Software Engineering. <https://chatgpt.com/codex/>. 6
- [39] OpenAI. GPT-6 Astra: A New Generation of Intelligence. <https://openai.com/index/gpt-6-astra/>. 6
- [40] OpenAI. GPT-5.6: Frontier Intelligence That Scales with Your Ambition. <https://openai.com/index/gpt-5-6/>, 2026. 6
- [41] OpenAI. Introducing GPT-6 Sol and Luna. <https://openai.com/index/introducing-gpt-6-sol-and-luna/>, 2026. 6
- [42] Pexels. Free 4K Stock Videos & Full HD Video Clips to Download. <https://www.pexels.com/videos/>. 4, 14
- [43] Atin Pothiraj, Jaemin Cho, Yue Zhang, Elias Stengel-Eskin, and Mohit Bansal. Physics Question Scene Graph: Fine-Grained Evaluation of Physical Plausibility in Text-to-Video Generation. In *Computer Vision - ECCV 2026 - 19th Eu-*

- ropean Conference, Malmö, Sweden, September 8-12, 2026, *Proceedings, Part IX*, pages 301–318. Springer, 2026. 2
- [44] Daniel Rho, Jun Myeong Choi, Matthew Thornton, Biswadip Dey, and Roni Sengupta. MonoPhysics: Estimating Geometry, Appearance, and Physical Parameters from Monocular Videos. *arXiv preprint arXiv:2605.30320*, 2026. 3, 5
- [45] Ruihong Shen, Žiga Kovačič, Peter Kulits, Xingrui Wang, Zizhang Li, Joshua B. Tenenbaum, Alan Yuille, Jieneng Chen, and Jiajun Wu. 4DCodeBench: Benchmarking Agents on Inverse Graphics of Dynamic Scenes. *arXiv preprint arXiv:2610.03715*, 2026. 2, 3, 5
- [46] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. 6
- [47] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3D-GPT: Procedural 3D Modeling with Large Language Models. In *International Conference on 3D Vision, 3DV 2025, Singapore, March 25-28, 2025*, pages 1253–1263. IEEE, 2025. 3
- [48] Yolo Y. Tang, Daiki Shimada, Jiayue Meng, Jing Bi, Pinxin Liu, Yicheng Wang, Yunzhong Xiao, Zhangyun Tan, Zeliang Zhang, Chao Huang, Susan Liang, Qianxiang Shen, Luchuan Song, Ali Vosoughi, Mingqian Feng, Melika Filvantorkaman, and Chenliang Xu. BVB: Benchmarking Agentic Video Understanding via Programmatic Reconstruction in Blender. *arXiv preprint arXiv:2609.15478*, 2026. 3
- [49] Hanyang Wang, Yimo Cai, Weiliang Chen, Jiawei Chi, Haowen Sun, Qiyu Dai, Yi-Hsin Hung, Xingzhao Guo, Jinshan Ren, Runmao Yao, Ziwei Liu, Mingsheng Long, Yueqi Duan, Jun Gao, Jiangran Lyu, Fangfu Liu, and Jialong Wu. Code as Worlds: Agentic Discovery of Executable World Representations for Physical Reasoning. *arXiv preprint arXiv:2608.27549*, 2026. 2, 4
- [50] Hao Wang, Wenhui Zhu, Shao Tang, Zhipeng Wang, Xuanzhao Dong, Xin Li, Xiwen Chen, Ashish Bastola, Xinhao Huang, Yalin Wang, and Abolfazl Razi. EzBlender: Efficient 3D Editing with Plan-and-ReAct Agent. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2026 - Workshops, Tucson, AZ, USA, March 6-10, 2026*, pages 1343–1352. IEEE, 2026. 3
- [51] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. 6
- [52] Jiajun Wu, Ilker Yildirim, Joseph J. Lim, Bill Freeman, and Joshua B. Tenenbaum. Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 127–135, 2015. 2
- [53] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to See Physics via Visual De-Animation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 153–164, 2017. 2
- [54] Jiajun Wu, Joshua B. Tenenbaum, and Pushmeet Kohli. Neural Scene De-Rendering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7035–7043. IEEE Computer Society, 2017. 3
- [55] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 4389–4398. IEEE, 2024. 2
- [56] Tianyidan Xie, Peiyu Wang, Hu Jiabin, Yuyi Qian, Yuxuan Wang, Shenyi Wang, Rui Ma, Yanlun Peng, Lanjun Wang, Ying Tai, Jian Yang, and Zili Yi. PhysCodeBench: Benchmarking Physics-Aware Symbolic Simulation of 3D Scenes via Self-Corrective Multi-Agent Refinement. *arXiv preprint arXiv:2604.23580*, 2026. 3
- [57] Xiaoyu Xiong, Changyu Hu, Chunru Lin, Pingchuan Ma, Chuang Gan, and Tao Du. TopoGaussian: Inferring Internal Topology Structures from Visual Clues. In *The Thirteenth International Conference on Learning Representations, 2025*, https://proceedings.iclr.cc/paper_files/paper/2025/hash/f41b6e5af73421e46ceed9cb036e72e7-Abstract-Conference.html. 2
- [58] Yutaro Yamada, Khyathi Raghavi Chandu, Bill Yuchen Lin, Jack Hessel, Ilker Yildirim, and Yejin Choi. L3GO: Language Agents with Chain-of-3D-Thoughts for Generating Unconventional Objects. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - System Demonstrations, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 456–469. Association for Computational Linguistics, 2025. 3
- [59] Runmao Yao, Kairui Hu, Yukang Cao, Ruisi Wang, Shulin Tian, Ziang Cao, Weichen Fan, Ziqi Huang, Yuhao Dong, Hao Li, Zhaoxi Chen, Zhongang Cai, Lei Yang, and Ziwei Liu. Apple- π : Benchmarking Thinking with Video towards Law-Grounded Physical Intelligence. *arXiv preprint arXiv:2607.16401*, 2026. 2
- [60] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. CLEVRER: Collision Events for Video Representation and Reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2
- [61] Shaofeng Yin, Jiabin Ge, Zora Zhiruo Wang, Chenyang Wang, Xiuyu Li, Michael J. Black, Trevor Darrell, Angjoo Kanazawa, and Haiwen Feng. Vision-as-Inverse-Graphics Agent via Interleaved Multimodal Reasoning. In *Computer Vision - ECCV 2026 - 19th European Conference, Malmö, Sweden, September 8-12, 2026, Proceedings, Part XXXV*, pages 557–575. Springer, 2026. 2, 3
- [62] Conghan Yue, Yuanjie Chen, Yue Han, Ya Gao, Yunyan Xiao, Wei Yao Zhang, and Zhineng Chen. VideoPhysEdit: Physical Counterfactual Video Editing via Rigid-Body Physical Scene Reconstruction. *arXiv preprint arXiv:2609.35134*, 2026. 3
- [63] Z.ai. GLM-5.3-Flash: Frontier Intelligence, Flash Cost. <https://z.ai/blog/glm-5.3-flash>. 6
- [64] Jiahao Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal M. Patel, Paul Pu Liang, Daniel Khashabi, Cheng Peng, Rama Chellappa, Tianmin Shu, Alan L. Yuille, Yilun Du, and Jieneng Chen. World-in-World: World Models in a Closed-Loop World. In *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. 2

- [65] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 586–595. Computer Vision Foundation / IEEE Computer Society, 2018. 6
- [66] Yunzhi Zhang, Zizhang Li, Matt Zhou, Shangzhe Wu, and Jiajun Wu. The Scene Language: Representing Scenes with Programs, Words, and Embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 24625–24634. Computer Vision Foundation / IEEE, 2025. 3
- [67] Yi Zhang, Yunshuang Wang, Zeyu Zhang, and Hao Tang. Code2Worlds: Empowering Coding LLMs for 4D World Generation. *arXiv preprint arXiv:2602.11757*, 2026. 3
- [68] Yifei Zhao, Xiangxin Zhou, Wenhao Yang, Jiaqi Tang, Pu Jian, Huanjin Yao, Jiarui Yao, Haowei Lin, Chunchao Guo, Zhuo Chen, Wenkai Lyu, Jianzhu Ma, Xueqian Wang, and Wenxi Zhu. SceneActBench: Can Agents Act on the 3D Scenes They See? *arXiv preprint arXiv:2607.22393*, 2026. 3
- [69] Yan Zheng and Florian Bordes. VoxelCodeBench: Benchmarking 3D World Modeling through Code Generation. *arXiv preprint arXiv:2604.02580*, 2026. 3
- [70] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part II*, pages 407–423. Springer, 2024. 3
- [71] King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling Test-Time Compute for LLM Agents. *arXiv preprint arXiv:2506.12928*, 2025. 6

PhysCodeGym: Benchmarking Coding Agents for Physically Verifiable 4D Reconstruction

Appendix

Contents	Page
A Dataset and Execution Framework	14
A.1 Composition and Sources	14
A.2 Reference Video Examples	15
A.3 Harbor Integration	19
B Evaluation Protocol	20
B.1 Task Rubrics and Executable Verification	20
B.2 Counterfactual Tests	20
C Additional Results and Analysis	21
C.1 Scores on Aligned Task Subsets	21
C.2 Scores on Produced Artifacts	21
C.3 Reasoning Effort, Completion, and Reconstruction Quality	22
D Case Studies	23
D.1 Metrics and Reconstruction Quality	23
D.1.1 Two Pendulums	24
D.1.2 Rolling Quail Egg	25
D.1.3 Wine Pouring	26
D.1.4 Avocado Halves Entering Water	27
D.1.5 Thai Flag	28
D.2 Physical Validity Failures	29
D.3 Counterfactual Mechanism Checks	32
D.3.1 A Nearby Obstacle Has No Contact Response	32
D.3.2 Support Extends Beyond the Physical Table	33
D.3.3 A Bearing Hides a Missing Table Collision	34
D.3.4 A Different Reconstruction Omits Contact with a Chess Piece	35
D.3.5 A Solid Token Behaves as if It Were Absent	36
D.4 Cross-Harness Verification	37
D.4.1 Checking Rolling After Contact Changes	37
D.4.2 Checking Accumulation Across a Pour	38
D.4.3 Harness Instructions and Completed Verification	39

A. Dataset and Execution Framework

A.1. Composition and Sources

The 156 tasks contain 578 event items in total (Tab. 3). By main category, there are 95 rigid-body, 40 liquid, 12 deformable, and 9 wind-driven tasks.

Table 3. Composition of PhysCodeGym: number of tasks and event items per main category and subcategory.

Category	Subcategory	Tasks	Event items
Rigid body	Falling and bouncing	16	71
	Rolling and tracks	18	68
	Collision and chain reactions	14	56
	Swinging and suspension	16	51
	Rotation	20	63
	Granular and multi-object piling	7	22
	Stacking and static equilibrium	4	13
Liquid	Pouring and container filling	25	109
	Jets and surface flow	8	25
	Liquid-object interaction	7	30
Deformable	–	12	41
Wind-driven	–	9	29
Total		156	578

Data sources. The reference videos come from Pexels [42], Physics-IQ [37], BAAI-DataCube Physics-aware Videos [4], and IRIS [25]. Table 4 reports the number of retained tasks from each source by main category.

Table 4. Data sources of PhysCodeGym. Each retained reference clip contributes one task.

Source	Rigid body	Liquid	Deformable	Wind-driven	Total
Pexels	69	35	9	9	122
Physics-IQ	19	5	3	0	27
BAAI-DataCube Physics-aware Videos	5	0	0	0	5
IRIS	2	0	0	0	2
Total	95	40	12	9	156

Rigid-body tasks. Rigid-body tasks fall into seven subcategories. *Falling and bouncing* tests gravity, collision restitution, and coming to rest after landing. *Rolling and tracks* covers objects rolling on planes, slopes, trays, or tracks, with paths determined by the track geometry. *Collision and chain reactions* focuses on momentum transfer between objects, as in billiards, bowling, dominoes, and Newton’s cradles. *Swinging and suspension* involves swings, pendulums, metronomes, and hanging objects, whose motion is organized by hinges or ropes. *Rotation* involves turntables, windmills, spinning tops, and rotating platforms. *Granular and multi-object piling* involves pouring and piling of many objects, such as rice grains, flakes, and paper clips. *Stacking and static equilibrium* tests whether blocks or stones remain stable or topple under loading.

Liquid tasks. Liquid tasks fall into three subcategories. *Pouring and container filling* is the most common case: liquid flows from a bottle mouth, spout, or pipe into a container, and the liquid level rises accordingly. *Jets and surface flow* covers jets from hoses, nozzles, and fountains, as well as the spreading and splashing of liquid after it lands on a surface. *Liquid-object interaction* covers objects falling into liquid, sinking or floating in it, and liquid streams striking objects.

Deformable and wind-driven tasks. The deformable and wind-driven categories contain few tasks and are not subdivided.

A.2. Reference Video Examples

Figures 6 to 9 show 32 reference clips selected to illustrate the categories and subcategories of PhysCodeGym. Each row shows four frames at approximately 0%, 33%, 67%, and 100% of the clip. Full frames are resized without cropping or color adjustment. This selection illustrates the range of phenomena; the task frequencies are given in Appendix A.1.

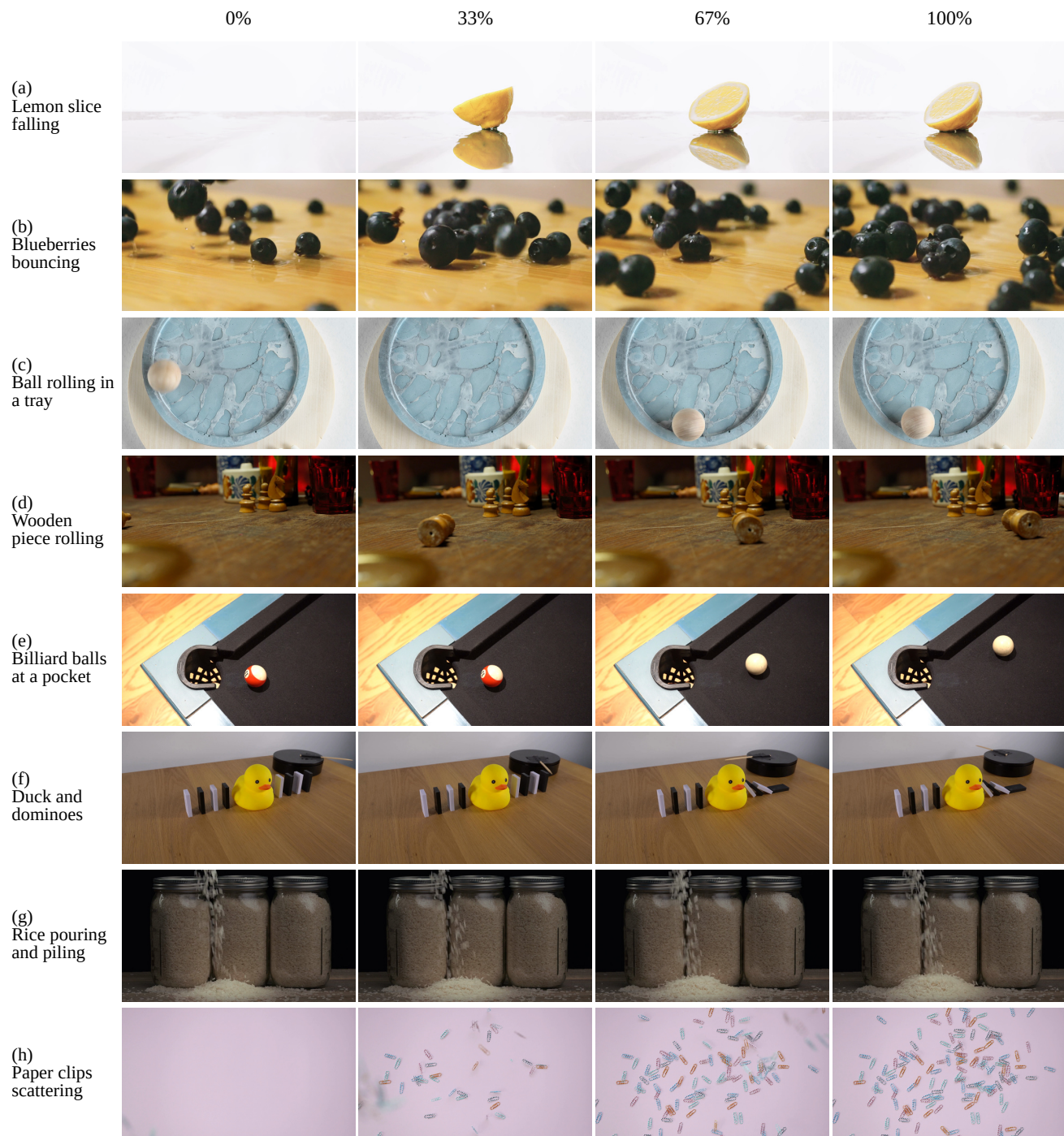


Figure 6. Rigid-body reference examples (1/2): falling and bouncing, rolling and tracks, collision and chain reactions, and granular or multi-object piling. Each row contains four uniformly sampled frames from one reference video.

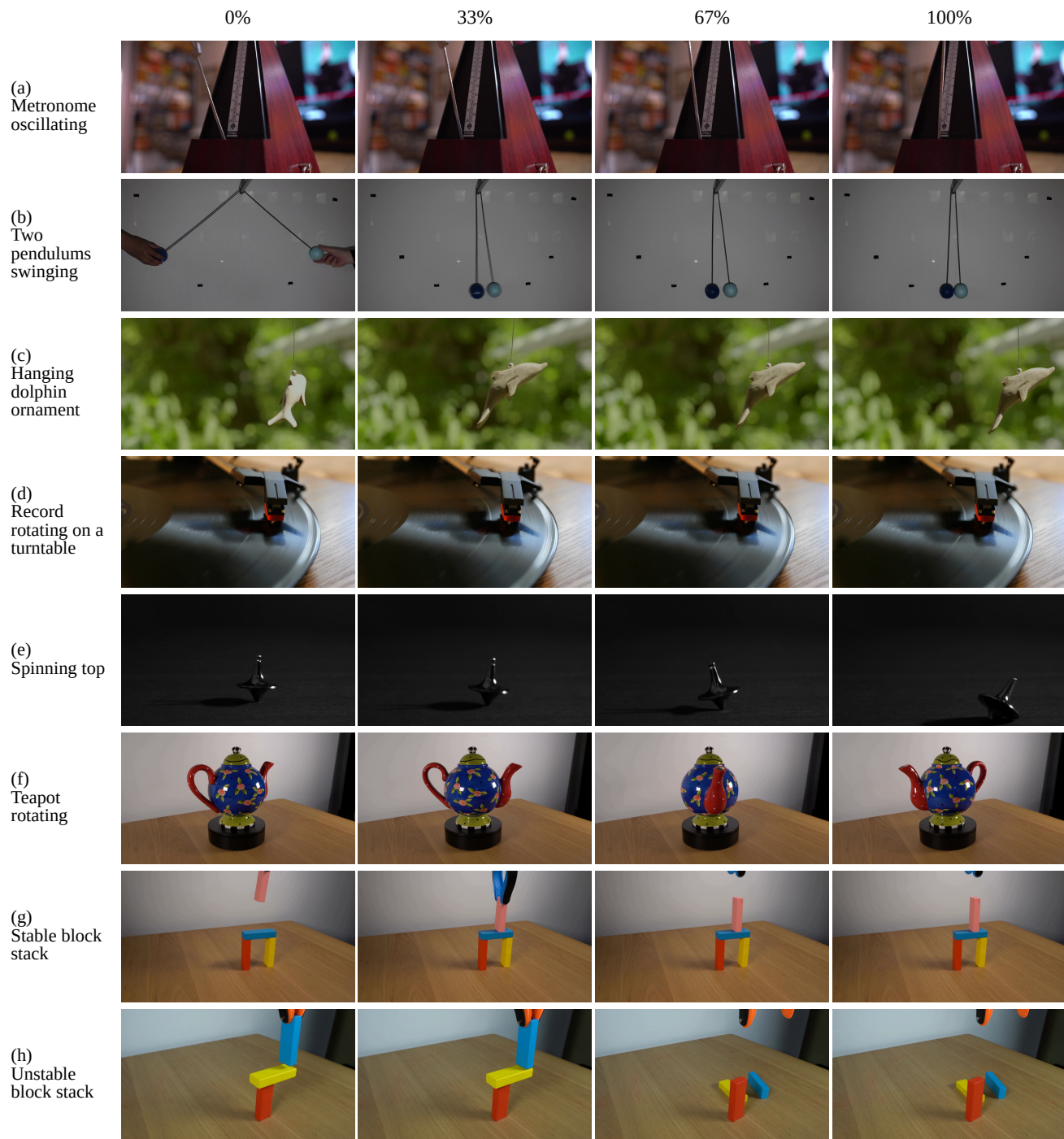


Figure 7. Rigid-body reference examples (2/2): swinging and suspension, rotation, and stacking or static equilibrium. Each row contains four uniformly sampled frames from one reference video.

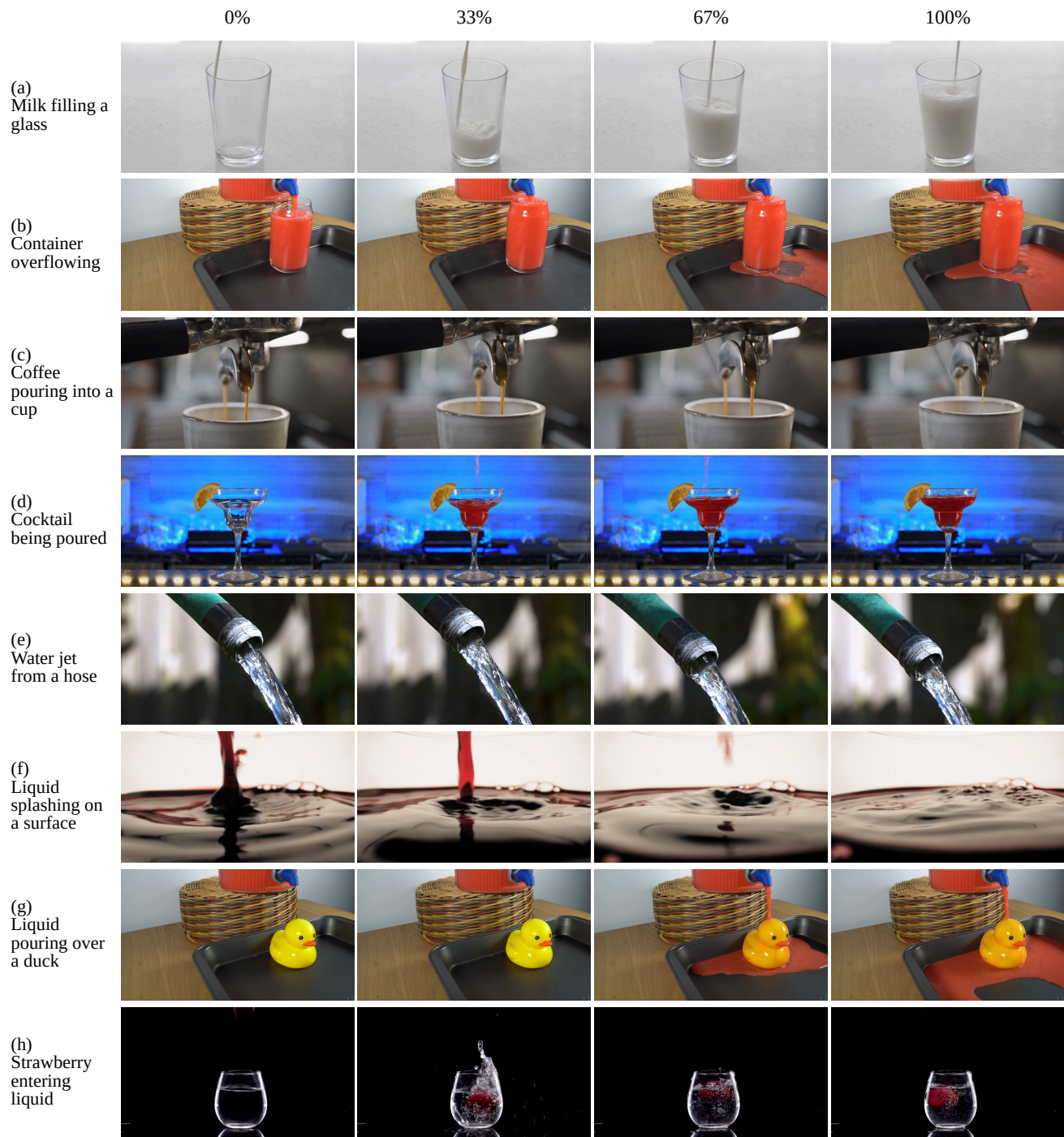


Figure 8. Liquid reference examples: pouring and container filling, jets and surface flow, and liquid–object interaction. Each row contains four uniformly sampled frames from one reference video.

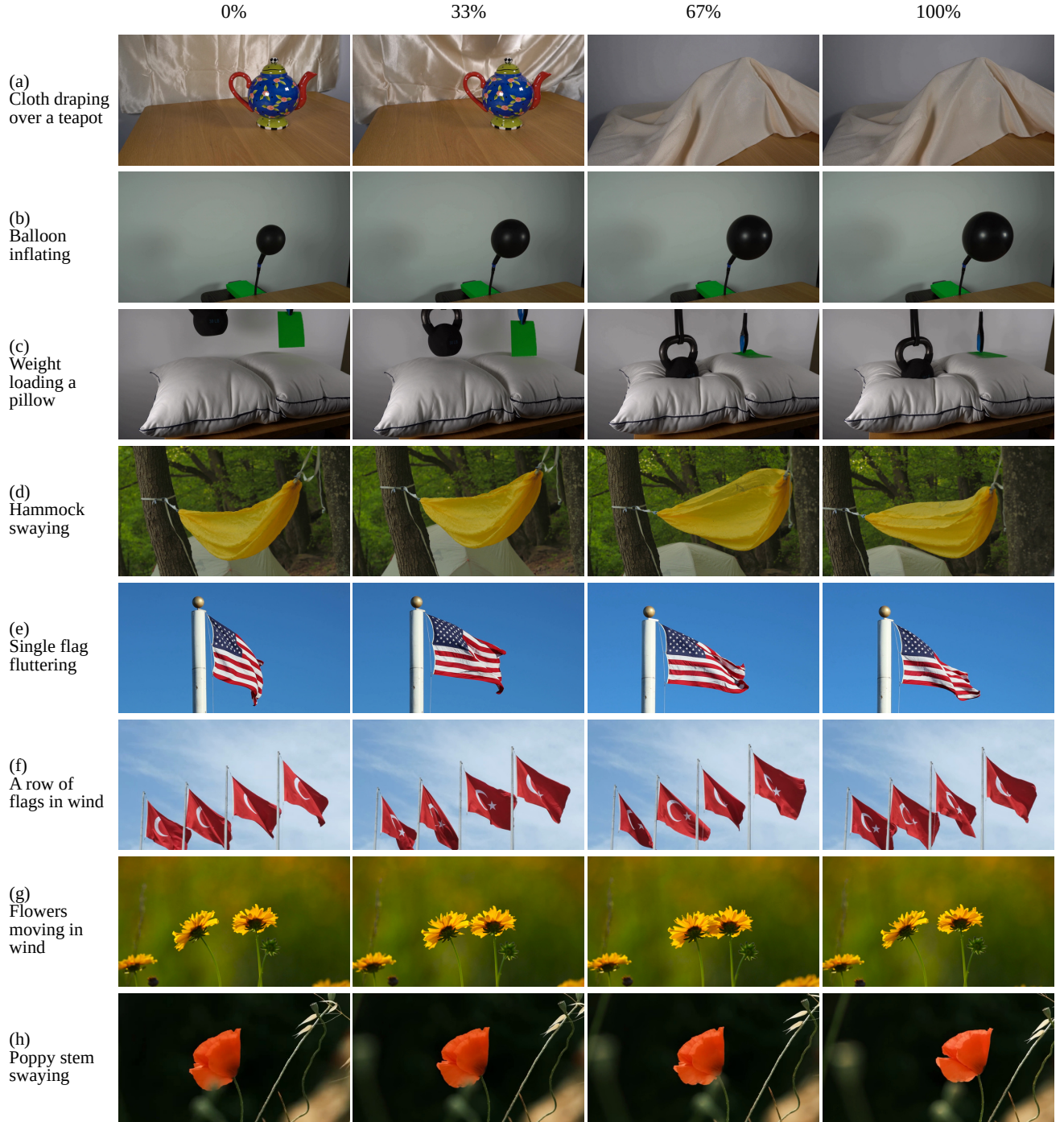


Figure 9. Deformable-category and wind-driven reference examples. Rows (a)–(d) show draping, membrane inflation, pillow deformation, and hammock motion; rows (e)–(h) show flag and plant motion in wind. Each row contains four uniformly sampled frames from one reference video.

A.3. Harbor Integration

Framework organization. PhysCodeGym builds on Harbor [16] v0.22.0, with local compatibility patches for CLI setup and container execution. We use Harbor’s Docker backend with Ubuntu 24.04 containers and the official Blender 4.0.2 distribution. Harbor’s installed-agent adapters launch the native Codex, Claude Code, and Kimi Code CLIs inside these containers (Sec. 4.2). The CLIs are installed in the image or mounted as preinstalled binaries; our Kimi adapter overrides only installation and inherits Harbor’s Kimi Code execution path. Harbor manages container and agent lifecycles, while PhysCodeGym supplies the reconstruction tasks, Blender tools, and evaluation protocol.

Reconstruction workflow. Each reconstruction task is packaged as a separate Harbor task containing task instructions, a reference video, and environment configuration. The task image provides the Blender runtime and command-line tools for video observation, scene execution and inspection, and preview rendering. The agent observes, programs, and checks its reconstruction inside the container (Sec. 3.2).

Unless otherwise specified, Harbor limits agent execution to 3 hours per run, and the task environment’s launch interface permits at most 20 Blender process launches (Sec. 4.1). Scene execution, scene inspection, and preview rendering share this quota. Video observation commands do not consume Blender launches but count toward execution time. Once the launch quota is exhausted, new Blender launch requests are rejected; when the execution time limit is reached, Harbor ends the agent run.

Evaluation integration. After agent execution, Harbor collects the submitted program and invokes the benchmark’s checking script in a separate verification container. This stage executes the program, checks the scene and rendered outputs, and records basic image and motion measurements. Subsequent evaluation jobs assess objects and events in the archived video. A verification agent checks the program and scene against manually annotated physical criteria, including counterfactual tests for selected tasks (Appendix B). The component scores are aggregated according to the evaluation protocol in Sec. 4.3.

B. Evaluation Protocol

This section describes task-level physical verification and counterfactual tests.

B.1. Task Rubrics and Executable Verification

Task rubrics. We assess physical validity with an agent acting as a verifier, guided by manually annotated physical rubrics for each task. Each rubric specifies the permitted initial setup and actuation, the mechanisms to check, and the evidence required for a judgment. The criteria cover numerical stability, gravity, mass, and actuation, together with task-specific requirements on contact, support, and structural integrity. The same rubric applies to every candidate for a task.

Executable verification. The verification agent reads the submitted program and native scene, maps the rubric’s semantic roles to scene objects, and traces how their motion is implemented. It formulates a measurement plan and writes probes adapted to that implementation. The agent executes these probes in Blender to inspect geometry and measure object states, contact or support relations, and motion over time; Python tools analyze the resulting measurements. It inspects the outputs and refines its measurements when further evidence is needed. The final report gives a judgment for each criterion with supporting source references and measured evidence, and the evaluation harness applies the rubric’s aggregation rules. A missed reference event is assessed by the event score and is not by itself a physical violation.

B.2. Counterfactual Tests

Counterfactual verification evaluates rubric criteria that require controlled interventions. The rubric specifies the physical dependence to test and the expected response under a controlled intervention. The agent locates the corresponding mechanism in the program, creates an independent scene copy, implements the prescribed intervention, and re-simulates it for comparison with the unchanged baseline. Interventions may alter a force, remove a contact, or change an initial condition while holding other relevant conditions fixed. Before interpreting the response, the agent verifies that the intervention takes effect and uses control scenes to check the measurement. For example, when a ball moves a target through collision, removing the ball–target contact before impact while preserving the ball’s incoming motion should eliminate the target’s impact-induced response (Fig. 2). The agent compares the measured response with the rubric’s expectation to determine whether the reconstructed motion depends on the required physical interaction.

C. Additional Results and Analysis

We compare scores on aligned task subsets, report quality conditional on producing an artifact, and examine the relationship between reasoning effort and reconstruction quality.

C.1. Scores on Aligned Task Subsets

The main table’s video metrics cover 156 tasks, whereas physical scores exclude technical gaps. Table 5 recomputes all metrics on the task set used for each configuration’s physical score, aligning the averaging set within each row. Event and motion remain video measurements; validity uses the corresponding reports. Task sets still differ across configurations, rather than forming a common intersection for all models.

Table 5. Scores recomputed on each configuration’s physical-evaluation task subset.

Model	Harness	$S_{\text{vis}} \uparrow$	$S_{\text{evt}} \uparrow$	$S_{\text{mot}} \uparrow$	$S_{\text{val}} \uparrow$	$S_{\text{phys}} \uparrow$	$S_{\text{tot}} \uparrow$
GPT-6-Astra	Codex	0.694	0.806	0.325	0.768	0.633	0.664
GPT-5.6-Sol	Codex	<u>0.611</u>	<u>0.687</u>	<u>0.210</u>	<u>0.690</u>	<u>0.529</u>	<u>0.570</u>
GPT-6-Luna	Codex	0.507	0.392	0.129	0.555	0.359	0.433
GPT-5.6-Luna	Codex	0.503	0.450	0.157	0.564	0.391	0.447
DeepSeek-V4.1-Flash	Claude Code	0.511	0.452	0.200	0.522	0.391	0.451
Claude Sonnet 5	Claude Code	0.436	0.370	0.111	0.559	0.347	0.392
Qwen3.8-Flash	Claude Code	0.335	0.316	0.090	0.350	0.252	0.294
GLM-5.3-Flash	Claude Code	0.291	0.241	0.104	0.295	0.213	0.252
Kimi K2.8	Kimi Code	0.404	0.348	0.110	0.448	0.302	0.353

All scores within a row use the same task subset. Best and second-best scores are **bold** and underlined, respectively.

For GPT-6-Astra, restricting the visual mean from 156 to 155 tasks changes S_{vis} from 0.693 to 0.694 and S_{tot} from 0.663 to 0.664, without changing individual task scores.

C.2. Scores on Produced Artifacts

Table 6 averages the same selected artifacts as Tab. 2, conditional on each configuration producing a rendered sequence. Tasks without an output are excluded; zero scores on existing outputs are retained. The five observed metrics use all rendered outputs, while validity and the physical score use those with an available physical judgment from either evidence scope, following Sec. 4.3. S_{tot} averages the reported visual and physical scores, retaining their respective coverage. These output-conditioned means describe reconstruction quality on produced artifacts; the full-task results also account for missing outputs.

Table 6. Mean scores on tasks with rendered outputs for the configurations in Tab. 2. Higher is better. Best and second-best scores are **bold** and underlined.

Harness	Model	Visual				Physical				$S_{\text{tot}} \uparrow$
		$S_{\text{SSIM}} \uparrow$	$S_{\text{LPIS}} \uparrow$	$S_{\text{obj}} \uparrow$	$S_{\text{vis}} \uparrow$	$S_{\text{evt}} \uparrow$	$S_{\text{mot}} \uparrow$	$S_{\text{val}} \uparrow$	$S_{\text{phys}} \uparrow$	
Codex	GPT-6-Astra	0.610	0.561	0.909	0.693	0.805	0.324	0.768	0.633	0.663
	GPT-5.6-Sol	0.553	0.451	<u>0.840</u>	<u>0.615</u>	<u>0.692</u>	0.211	<u>0.694</u>	<u>0.532</u>	<u>0.574</u>
	GPT-6-Luna	0.554	0.387	0.677	0.539	0.420	0.136	0.593	0.384	0.462
	GPT-5.6-Luna	0.529	0.371	0.669	0.523	0.469	0.165	0.591	0.409	0.466
Claude Code	DeepSeek-V4.1-Flash	<u>0.599</u>	<u>0.463</u>	0.716	0.593	0.520	<u>0.233</u>	0.609	0.456	0.525
	Claude Sonnet 5	0.568	0.349	0.532	0.483	0.407	0.122	0.619	0.384	0.434
	Qwen3.8-Flash	0.564	0.381	0.653	0.532	0.489	0.142	0.566	0.407	0.470
	GLM-5.3-Flash	0.551	0.436	0.696	0.561	0.462	0.199	0.572	0.414	0.488
Kimi Code	Kimi K2.8	0.570	0.398	0.626	0.531	0.458	0.145	0.592	0.399	0.465

C.3. Reasoning Effort, Completion, and Reconstruction Quality

We separate the benchmark aggregate, which retains missing-output zeros, from reconstruction quality conditional on producing an artifact. All comparisons use the same frozen scores and validity composition as the main results. Table 7 follows the score-aggregation convention in Sec. 4.3.

Table 7. Overall scores under different averaging sets. Best and second-best scores are **bold** and underlined.

Averaging set	Low	High	Max
Benchmark aggregate	0.476	0.450	<u>0.460</u>
Each setting’s produced artifacts	<u>0.527</u>	0.525	0.547
Shared tasks with outputs and physical judgments	0.533	<u>0.535</u>	0.564

Quality conditional on an output. Figure 10 plots the nine scores using each setting’s produced artifacts. Video metrics use those artifacts, while validity and physical scores use the available physical judgments within that subset. These subsets can differ between settings. The overall ordering changes: max scores 0.547, compared with 0.527 for low and 0.525 for high. Max also has the highest conditional event and physical scores (0.615 and 0.492). Thus, the benchmark-level advantage of low does not imply better reconstruction quality whenever an output is produced.

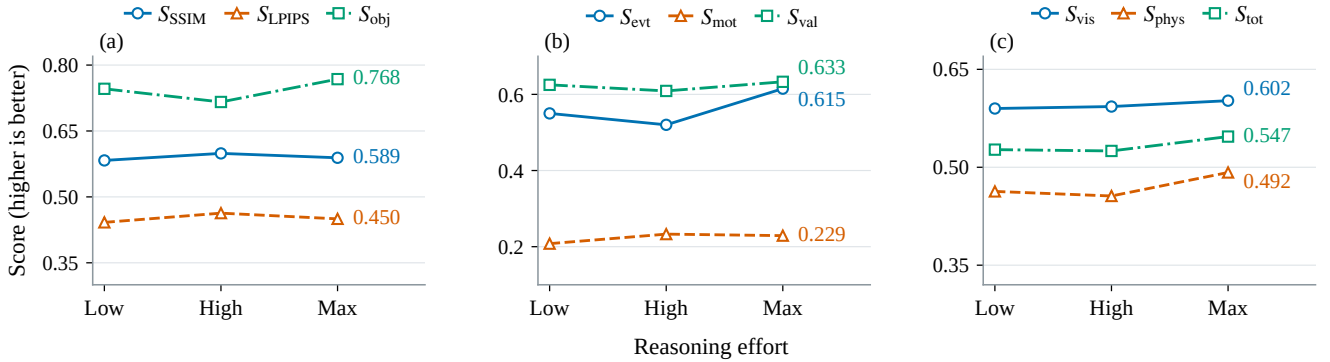


Figure 10. DeepSeek-V4.1-Flash scores on tasks with rendered outputs (axis ranges differ).

Comparison on shared tasks. We also restrict all three settings to the intersection of tasks with both a submitted video and an available physical judgment for every setting, retaining zero scores on existing artifacts. Max again has the highest visual, physical, and overall means on this shared subset. Its event score is 0.628, compared with 0.551 for low and 0.535 for high. The mean paired, unrounded per-task overall-score difference between max and low is 0.030; its paired task-bootstrap 95% interval is $[-0.001, 0.061]$. The bootstrap measures sensitivity to task sampling; repeated-run variability is not estimated here. This interval includes zero, so the observed mean ordering should not be interpreted as a universal effort advantage.

Where the aggregate gap arises. On the aligned task set with defined scores for both low and max, asymmetric missing outputs contribute $+0.036$ to the low-minus-max mean per-task score, while tasks with outputs from both settings contribute -0.020 . Their net contribution is $+0.016$. This diagnostic uses the same eligible tasks and unrounded case scores, rather than recombining differently covered headline columns. It identifies completion outcomes as the main source of low’s aggregate lead. Using original-condition-only validity reduces the low–max gap by approximately 0.003, making the CF component a secondary contributor.

Execution overhead. Max uses $2.46\times$ the recorded output tokens of low on average, and mean agent execution time rises from 93.3 to 110.7 minutes under the fixed three-hour budget. The additional missing outputs are concentrated in timeouts and final submissions that fail execution or lack required submission files. These results highlight a tradeoff between task completion and reconstruction quality under a fixed execution budget.

D. Case Studies

These case studies connect evaluation scores with reconstructed behavior and executable evidence. They cover metric interpretation, physical failures in unchanged scenes, counterfactual mechanism checks, and verification across harnesses.

D.1. Metrics and Reconstruction Quality

Different metrics assess reconstruction appearance, event progression, and physical constraints. The five cases in Figs. 11 to 15 examine differences in visible behavior and physical constraints that remain when individual metric scores are high or equal. Each case pairs synchronized frames from a reference video and six reconstructions with component and summary scores.

Table 8. Overview of the five case studies and their diagnostic questions.

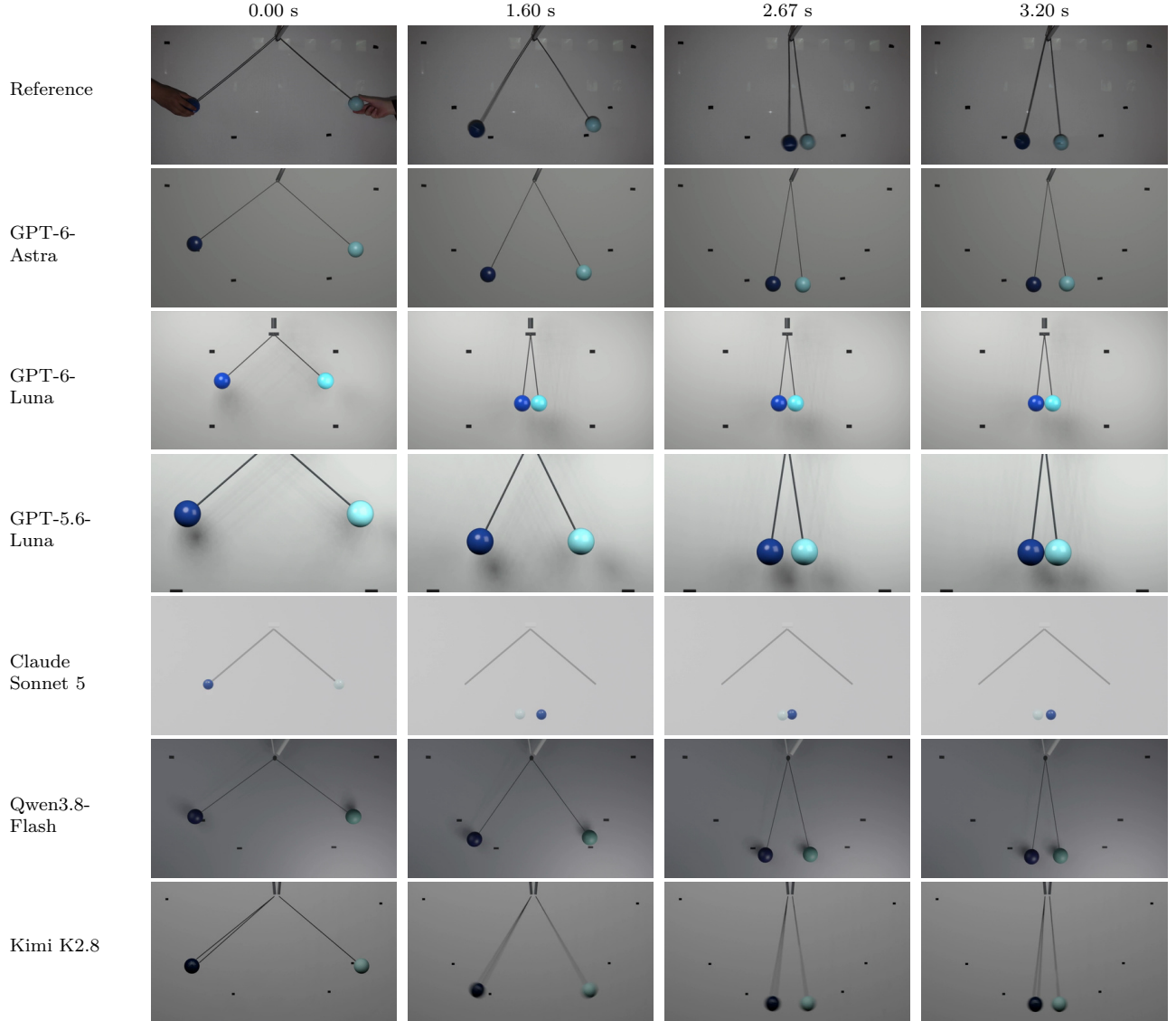
Question	Cases	Main observation
What differs when one metric is high?	Two pendulums; rolling quail egg	High motion overlap can coexist with incomplete events or physical violations.
What differs when one metric is equal?	Wine pouring; avocado halves	Equal event or validity scores coexist with differences in appearance and event progression.
What does scene inspection reveal?	Thai flag	An event score of 1 coexists with cloth self-intersections.

Reading the figures and scores. Display times follow the reference event stages and are shared by the reference and all candidates; frames are shown without cropping. Table rows follow the image-row order, identify each execution harness, and report six components together with the visual, physical, and overall scores. The highest and second-highest distinct scores in each column are **bold** and underlined, respectively; ties are determined before rounding. Full videos and denser temporal samples accompany the cases.

Structural similarity, perceptual similarity, object accuracy, event score, and motion score evaluate submitted videos. Physical validity evaluates the corresponding submitted program and its Blender scene. All table metrics are higher-is-better ($\uparrow$); S_{LPIPS} is perceptual similarity, $\text{clip}(1 - d, 0, 1)$, rather than raw LPIPS distance d . Score definitions and aggregation follow Sec. 4.3. Component scores and their evidence identify the properties on which candidates differ. All displayed score intervals are resolved, while validity judgments remain provisional pending semantic and evidence review.

Case and model selection. The archived 7 October 2026 pool contains 22 eligible tasks, each with a GPT-6-Astra video and at least five other configurations with complete observed metrics and resolved physical-validity intervals. Each task retains Astra and five other configurations selected by report availability, with configuration ID breaking ties; score values do not enter selection. We inspect five non-flag candidates under randomized model labels and select four for distinct phenomena and visible differences, retaining one flag example. All configurations use high effort, a three-hour budget, and at most twenty Blender launches. Model sets can differ by task. These targeted cases explain evaluation dimensions; they do not estimate benchmark-wide ranking accuracy or constitute a human-preference experiment.

D.1.1. Two Pendulums

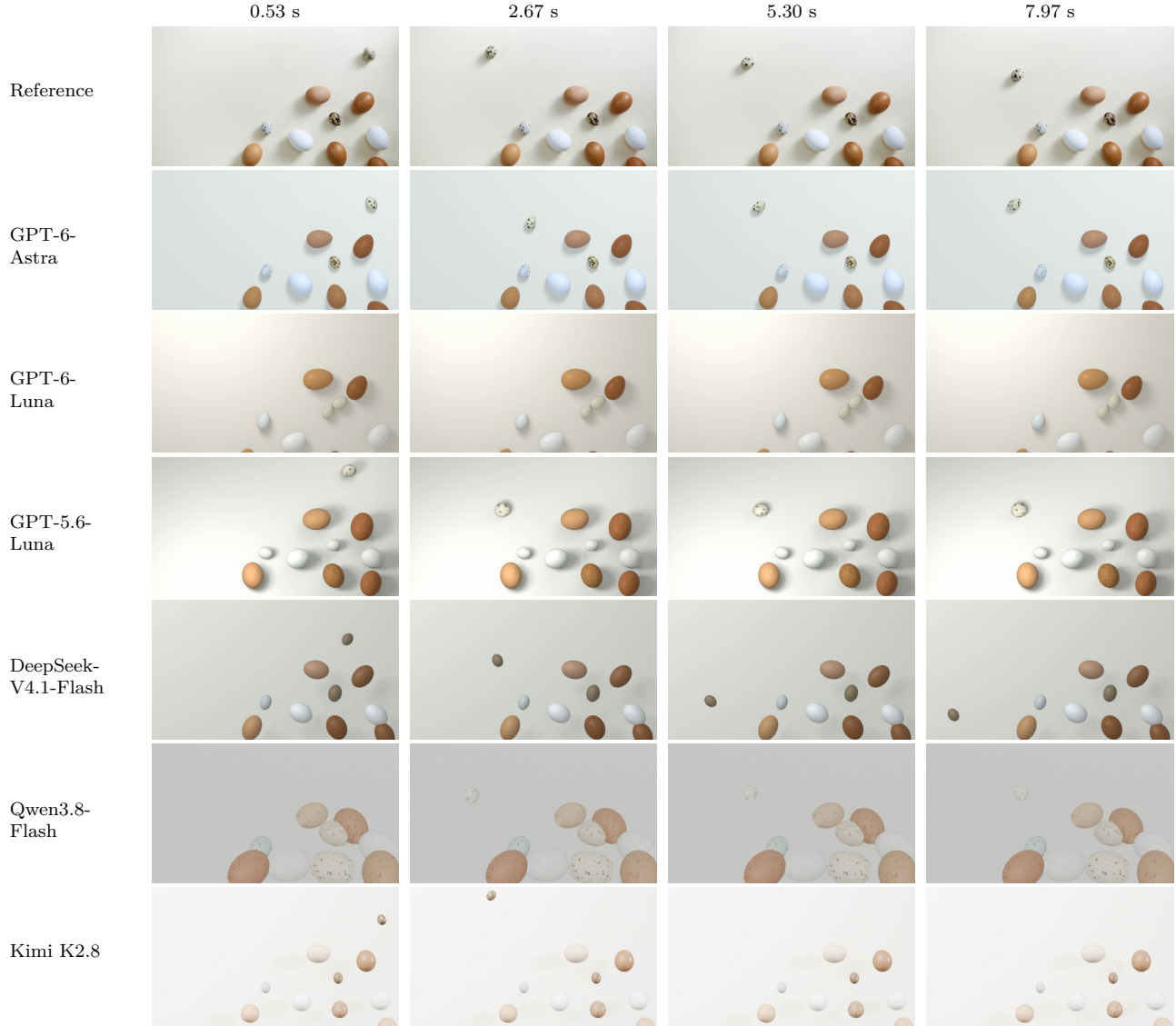


Component and summary scores. Best and second-best scores are **bold** and underlined, respectively.

Model	Harness	Visual metrics			Physical metrics			Summary scores		
		$S_{SSIM} \uparrow$	$S_{LPIPS} \uparrow$	$S_{obj} \uparrow$	$S_{evt} \uparrow$	$S_{mot} \uparrow$	$S_{val} \uparrow$	$S_{vis} \uparrow$	$S_{phys} \uparrow$	$S_{tot} \uparrow$
GPT-6-Astra	Codex	<u>0.905</u>	0.801	1.000	1.000	0.242	1.000	0.902	<u>0.747</u>	0.825
GPT-6-Luna	Codex	0.758	0.458	1.000	0.000	0.141	1.000	0.739	0.380	0.559
GPT-5.6-Luna	Codex	0.736	0.454	<u>0.857</u>	0.000	0.318	<u>0.500</u>	0.682	0.273	0.478
Claude Sonnet 5	Claude Code	0.743	0.423	<u>0.857</u>	<u>0.294</u>	0.128	1.000	0.674	0.474	0.574
Qwen3.8-Flash	Claude Code	0.906	0.729	1.000	1.000	<u>0.306</u>	<u>0.500</u>	0.878	0.602	0.740
Kimi K2.8	Kimi Code	0.891	<u>0.780</u>	1.000	1.000	0.254	1.000	<u>0.891</u>	0.751	<u>0.821</u>

Figure 11. **Motion overlap and event completion.** GPT-5.6-Luna has the highest motion IoU but an event score of 0; GPT-6-Astra ranks fourth on motion IoU and scores 1 on events. Luna’s validity report identifies connector-solid overlap outside the shared pivot throughout frames 55–240, yielding validity 0.5 versus Astra’s 1. Displayed frames cover early motion; scores cover all 8 seconds.

D.1.2. Rolling Quail Egg

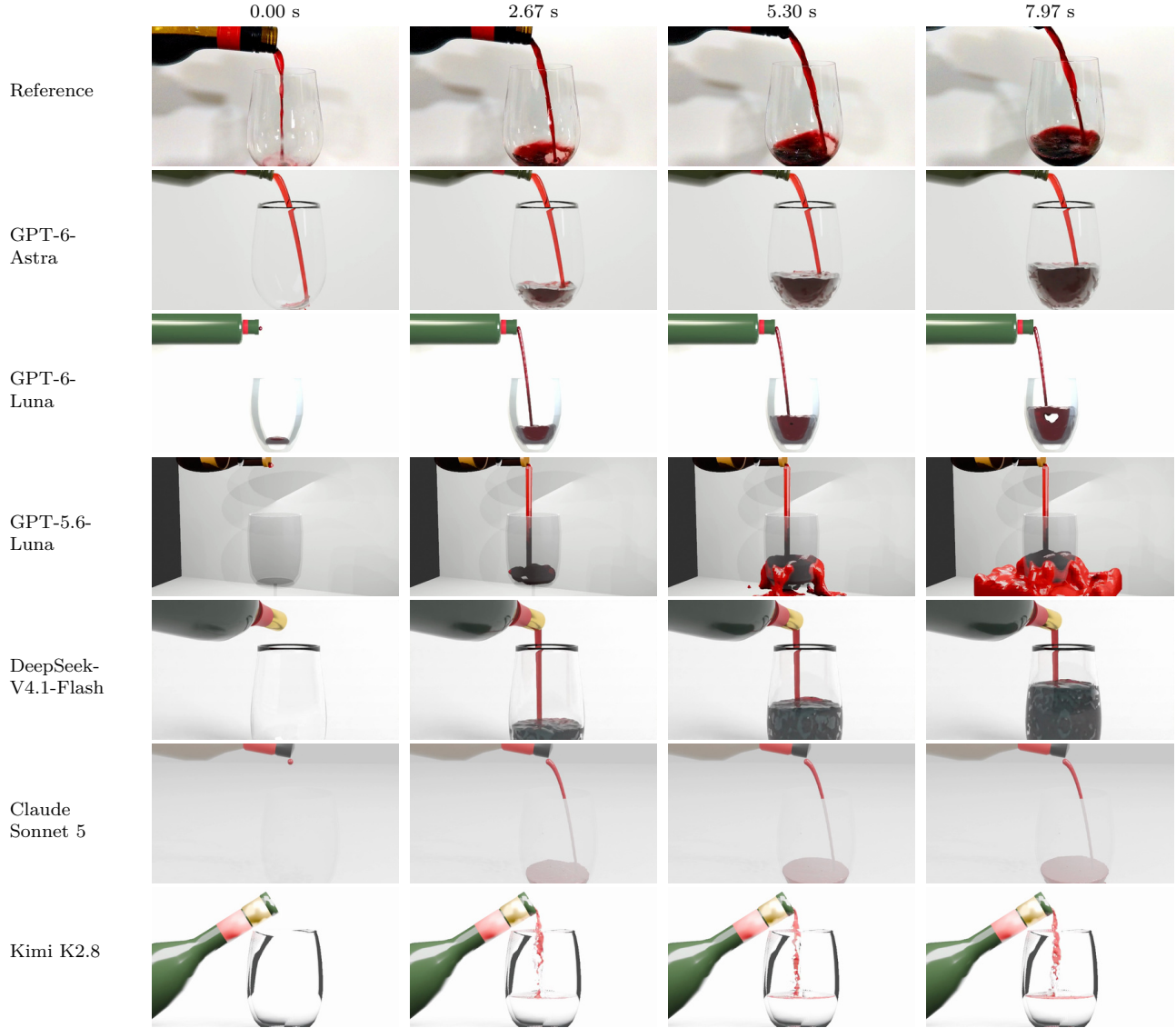


Component and summary scores. Best and second-best scores are **bold** and underlined, respectively.

Model	Harness	Visual metrics			Physical metrics			Summary scores		
		$S_{SSIM} \uparrow$	$S_{LPIPS} \uparrow$	$S_{obj} \uparrow$	$S_{evt} \uparrow$	$S_{mot} \uparrow$	$S_{val} \uparrow$	$S_{vis} \uparrow$	$S_{phys} \uparrow$	$S_{tot} \uparrow$
GPT-6-Astra	Codex	0.919	0.864	1.000	1.000	<u>0.225</u>	1.000	0.927	0.742	0.835
GPT-6-Luna	Codex	0.895	0.753	0.500	0.000	0.000	0.450	0.716	0.150	0.433
GPT-5.6-Luna	Codex	0.851	0.708	1.000	1.000	0.083	0.450	<u>0.853</u>	0.511	0.682
DeepSeek-V4.1-Flash	Claude Code	<u>0.907</u>	<u>0.822</u>	0.500	1.000	0.235	<u>0.500</u>	0.743	0.578	0.660
Qwen3.8-Flash	Claude Code	0.873	0.705	0.333	1.000	0.001	0.450	0.637	0.484	0.560
Kimi K2.8	Kimi Code	0.896	0.796	<u>0.833</u>	<u>0.625</u>	0.124	1.000	0.842	<u>0.583</u>	<u>0.712</u>

Figure 12. **Motion overlap and scene constraints.** DeepSeek-V4.1-Flash exceeds GPT-6-Astra on motion IoU (0.235 versus 0.225), but its validity is capped at 0.5, versus Astra’s 1. The validity report identifies persistent interpenetration between two stationary eggs across all 240 frames; this violation concerns the resting cluster rather than the moving egg. Astra retains the cluster and speckled appearance, with a timing mismatch.

D.1.3. Wine Pouring

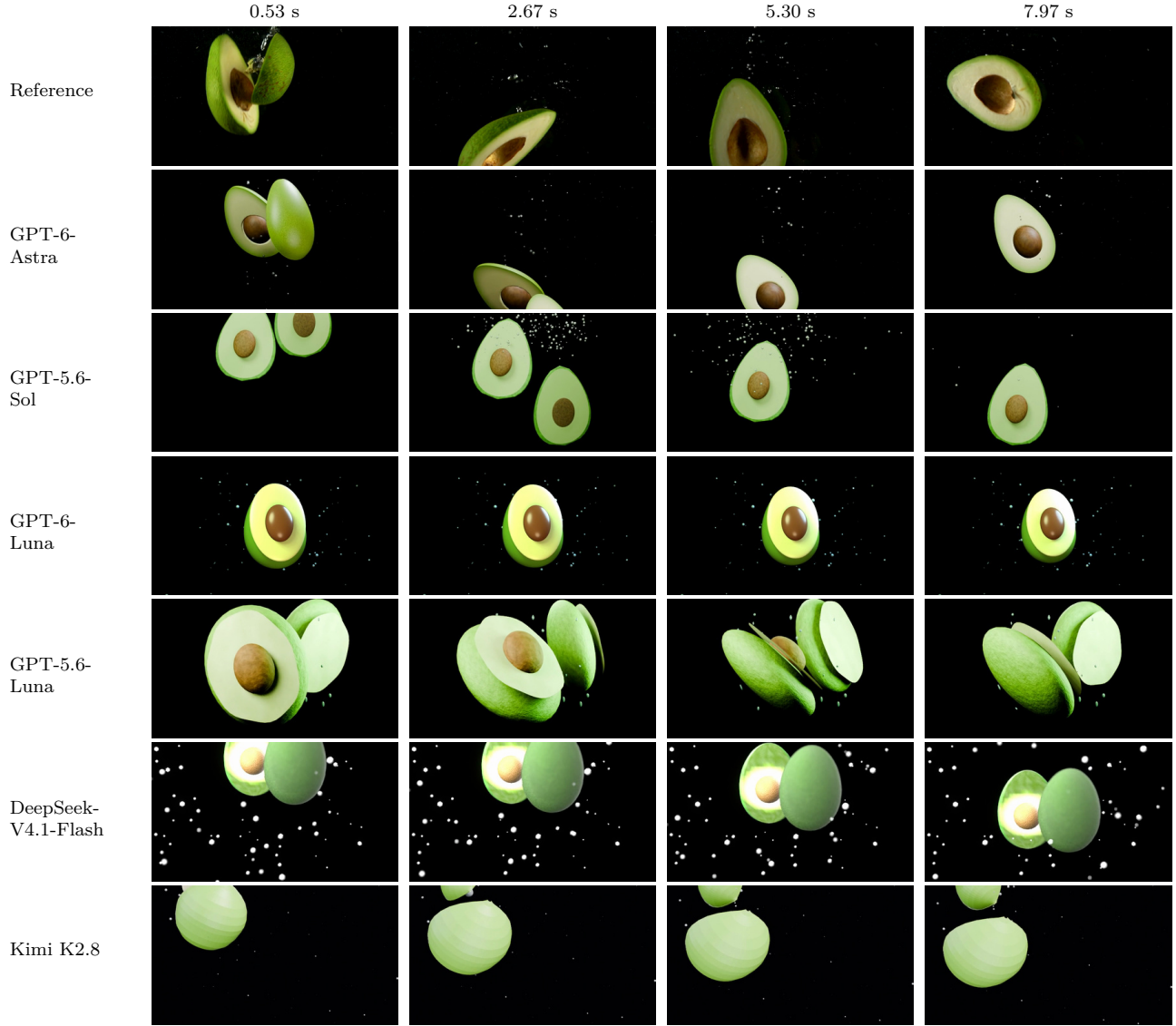


Component and summary scores. Best and second-best scores are **bold** and underlined, respectively.

Model	Harness	Visual metrics			Physical metrics			Summary scores		
		$S_{SSIM} \uparrow$	$S_{LPIPS} \uparrow$	$S_{obj} \uparrow$	$S_{evt} \uparrow$	$S_{mot} \uparrow$	$S_{val} \uparrow$	$S_{vis} \uparrow$	$S_{phys} \uparrow$	$S_{tot} \uparrow$
GPT-6-Astra	Codex	0.766	0.589	1.000	1.000	0.200	1.000	0.785	0.733	0.759
GPT-6-Luna	Codex	0.741	0.491	0.633	1.000	0.107	<u>0.500</u>	0.622	0.536	0.579
GPT-5.6-Luna	Codex	0.697	0.457	<u>0.800</u>	1.000	<u>0.209</u>	0.450	0.651	0.553	0.602
DeepSeek-V4.1-Flash	Claude Code	0.718	<u>0.519</u>	<u>0.800</u>	1.000	0.314	1.000	<u>0.679</u>	0.771	<u>0.725</u>
Claude Sonnet 5	Claude Code	0.774	0.511	0.533	1.000	0.056	<u>0.500</u>	0.606	0.519	0.562
Kimi K2.8	Kimi Code	0.685	0.442	<u>0.800</u>	1.000	0.097	1.000	0.642	0.699	0.671

Figure 13. **Appearance differences under equal event scores.** All six candidates receive an event score of 1, despite differences in stream color, glass visibility, and accumulated liquid. Claude Sonnet 5 has the highest SSIM (0.774) despite its pale rendering, but lower perceptual similarity than Astra (0.511 versus 0.589). Equal event scores thus coexist with appearance differences reflected in the frames and visual metrics.

D.1.4. Avocado Halves Entering Water



Component and summary scores. Best and second-best scores are **bold** and underlined, respectively.

Model	Harness	Visual metrics			Physical metrics			Summary scores		
		$S_{SSIM} \uparrow$	$S_{LPIPS} \uparrow$	$S_{obj} \uparrow$	$S_{evt} \uparrow$	$S_{mot} \uparrow$	$S_{val} \uparrow$	$S_{vis} \uparrow$	$S_{phys} \uparrow$	$S_{tot} \uparrow$
GPT-6-Astra	Codex	0.771	0.563	1.000	1.000	0.516	1.000	0.778	0.839	0.808
GPT-5.6-Sol	Codex	<u>0.702</u>	<u>0.519</u>	0.750	1.000	<u>0.339</u>	1.000	<u>0.657</u>	<u>0.780</u>	<u>0.718</u>
GPT-6-Luna	Codex	0.652	0.480	0.125	0.000	0.000	1.000	0.419	0.333	0.376
GPT-5.6-Luna	Codex	0.580	0.407	0.250	0.000	0.332	<u>0.500</u>	0.412	0.277	0.345
DeepSeek-V4.1-Flash	Claude Code	0.573	0.383	<u>0.875</u>	0.000	0.269	<u>0.500</u>	0.610	0.256	0.433
Kimi K2.8	Kimi Code	0.405	0.509	0.000	<u>0.471</u>	0.191	0.150	0.305	0.270	0.288

Figure 14. **Equal validity with different reconstruction fidelity.** GPT-6-Astra, GPT-5.6-Sol and GPT-6-Luna receive validity 1, but event scores of 1, 1, and 0, respectively. Luna’s validity report accepts a single composite rigid body in a homogeneous analytical medium: measured motion responds to effective gravity and linear drag. Passing these checks under the permitted simplification does not establish reproduction of the reference events.

D.1.5. Thai Flag



Component and summary scores. Best and second-best scores are **bold** and underlined, respectively.

Model	Harness	Visual metrics			Physical metrics			Summary scores		
		$S_{SSIM} \uparrow$	$S_{LPIPS} \uparrow$	$S_{obj} \uparrow$	$S_{evt} \uparrow$	$S_{mot} \uparrow$	$S_{val} \uparrow$	$S_{vis} \uparrow$	$S_{phys} \uparrow$	$S_{tot} \uparrow$
GPT-6-Astra	Codex	0.863	<u>0.711</u>	0.833	1.000	0.231	1.000	<u>0.802</u>	0.744	0.773
GPT-6-Luna	Codex	0.853	0.634	<u>0.667</u>	<u>0.000</u>	0.226	<u>0.500</u>	0.718	0.242	0.480
GPT-5.6-Luna	Codex	<u>0.861</u>	0.633	<u>0.667</u>	1.000	0.174	1.000	0.720	<u>0.725</u>	<u>0.722</u>
DeepSeek-V4.1-Flash	Claude Code	0.859	0.725	0.833	1.000	<u>0.272</u>	<u>0.500</u>	0.806	0.591	0.698
Qwen3.8-Flash	Claude Code	0.801	0.683	0.833	1.000	0.395	<u>0.500</u>	0.772	0.632	0.702
Kimi K2.8	Kimi Code	0.857	0.695	0.833	1.000	0.191	<u>0.500</u>	0.795	0.564	0.679

Figure 15. **Event completion with cloth self-intersections.** Qwen3.8-Flash receives an event score of 1, yet physical-validity checks detect self-intersections between nonadjacent cloth regions in 14 frames, capping validity at 0.5. This case shows that reproducing the visible event can coexist with invalid cloth geometry. Displayed frames also show differences in flag extension and cloth phase.

D.2. Physical Validity Failures

The six selected reconstructions receive $S_{\text{obj}} = S_{\text{evt}} = 1$, yet have explicit physical-validity failures. All displayed validity scores are resolved raw proposals from the original-condition checks; no default-pass assignments are used. This subsection illustrates rubric criteria evaluated on unchanged scenes; counterfactual criteria appear in Appendix D.3. Image pairs use the same reference/submission frame index. Plots identify one failure witness; scores summarize the corresponding original-condition rubric. Pixel fidelity and timing vary; these examples do not estimate failure prevalence.

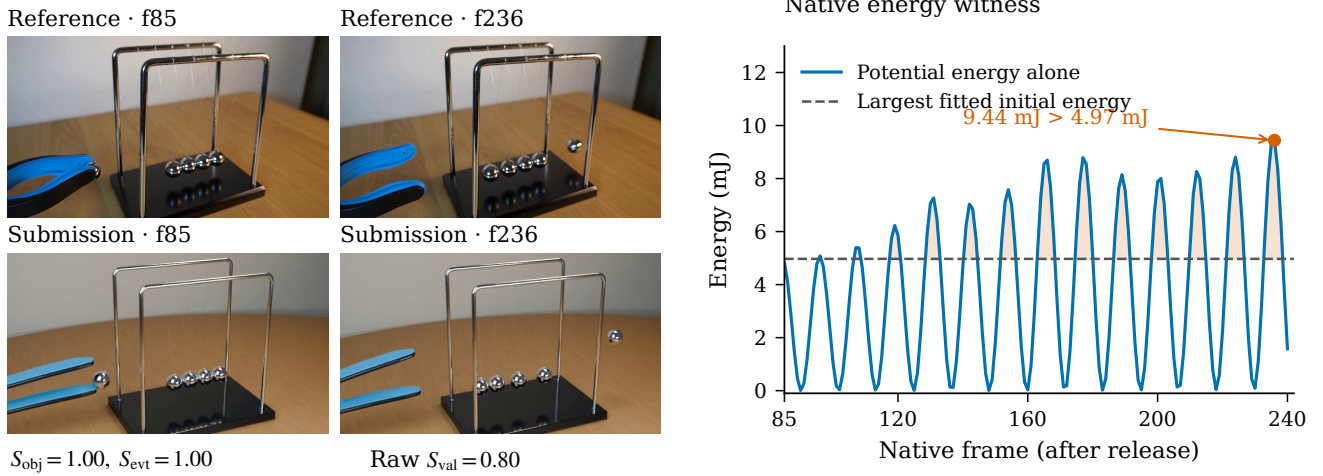


Figure 16. **GPT-6-Astra: energy addition in a Newton cradle.** After the gripper is clear at frame 85, four fits to the free pre-impact swing estimate initial total energy at approximately 4.971 mJ; the dashed line is the largest fitted estimate. At frame 236, gravitational potential alone reaches 9.441 mJ. With fixed supports and no admitted continuing work source, the native trajectory violates the passive energy budget (N3). This witness does not require differentiating impact-frame velocities.

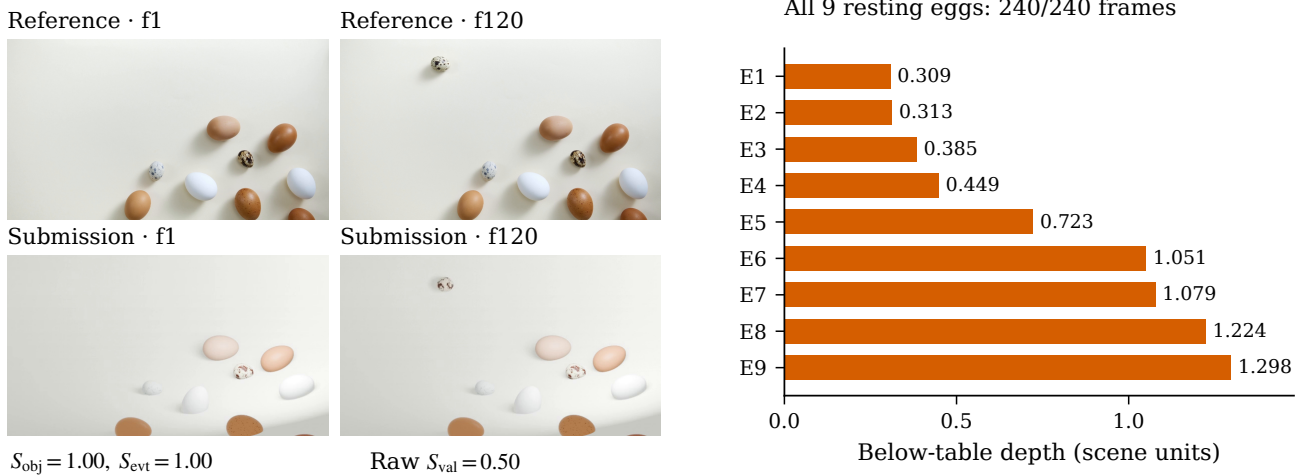


Figure 17. **GPT-5.6-Sol: resting eggs embedded in the tabletop.** All nine resting eggs intersect the actual evaluated table surface throughout 240/240 frames (E1). Their deepest below-surface vertices are 0.309–1.298 scene units beneath the local surface, with persistent material-boundary intersections. The bars describe the resting eggs only, excluding the mover’s isolated initial overlap.

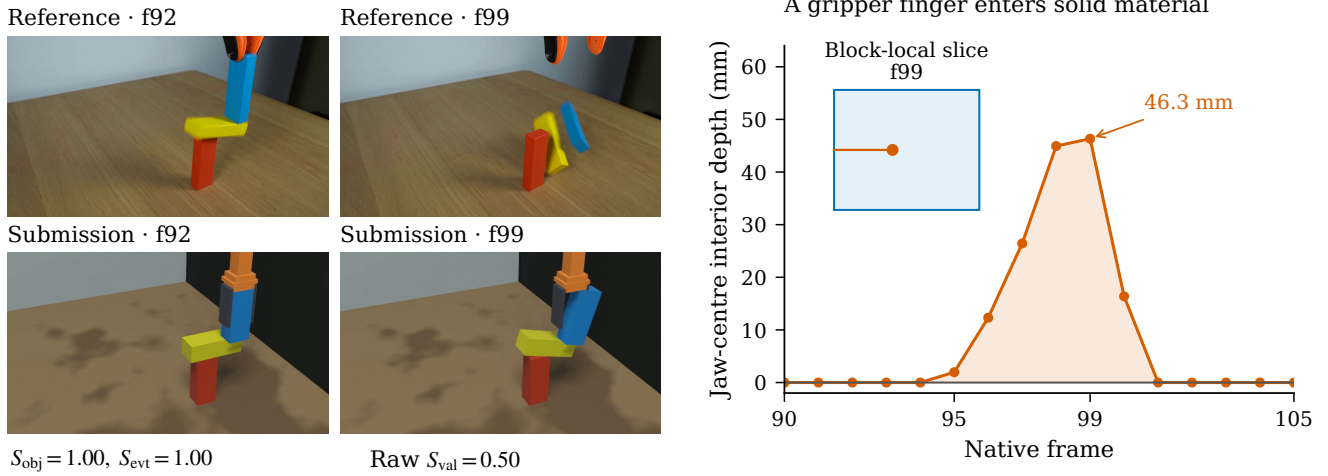


Figure 18. **DeepSeek-V4.1-Flash: a gripper passes through a released block.** The right jaw’s centre lies inside the blue block on frames 95–100, reaching 46.3 mm from the nearest block face at frame 99 (K1). The inset is a block-local cross-section through that centre, derived from measured coordinates. The plotted quantity is centre interior depth, not a universal surface-penetration depth; the original-condition score also includes an independent guide-attachment failure.

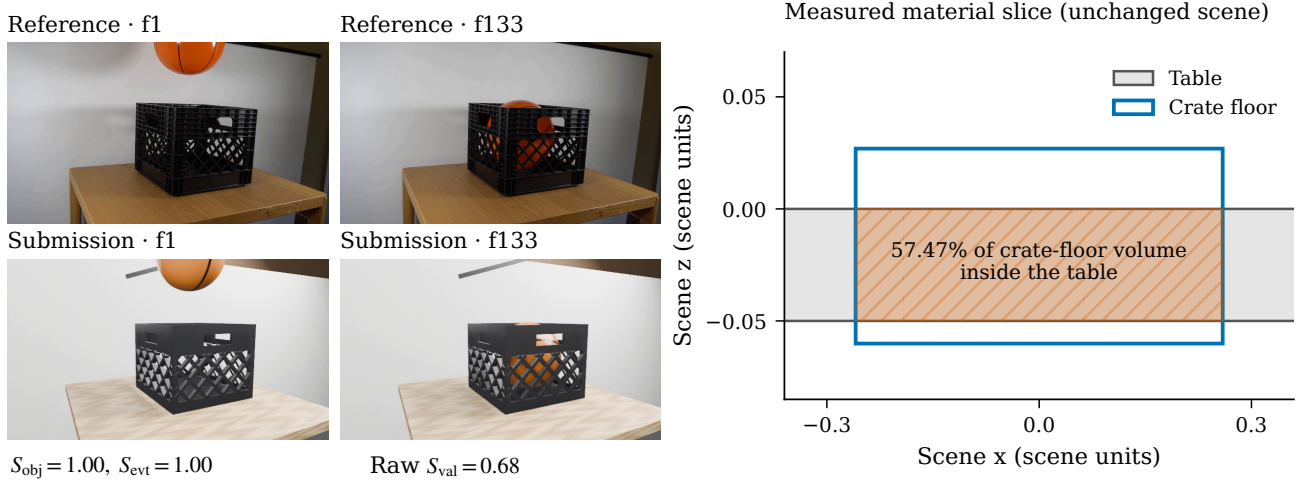


Figure 19. **GLM-5.3-Flash: the crate floor occupies the wooden tabletop.** The measured solid intervals overlap by $0.51864 \times 0.51864 \times 0.05000$ scene units, embedding 57.47% of the crate-floor volume in the table (B1). The right panel is a section through the actual axis-aligned box materials, with their common material shaded. Both bodies remain unchanged throughout all 240 frames; this is a persistent original-state overlap, not a failure inferred by modifying the scene.

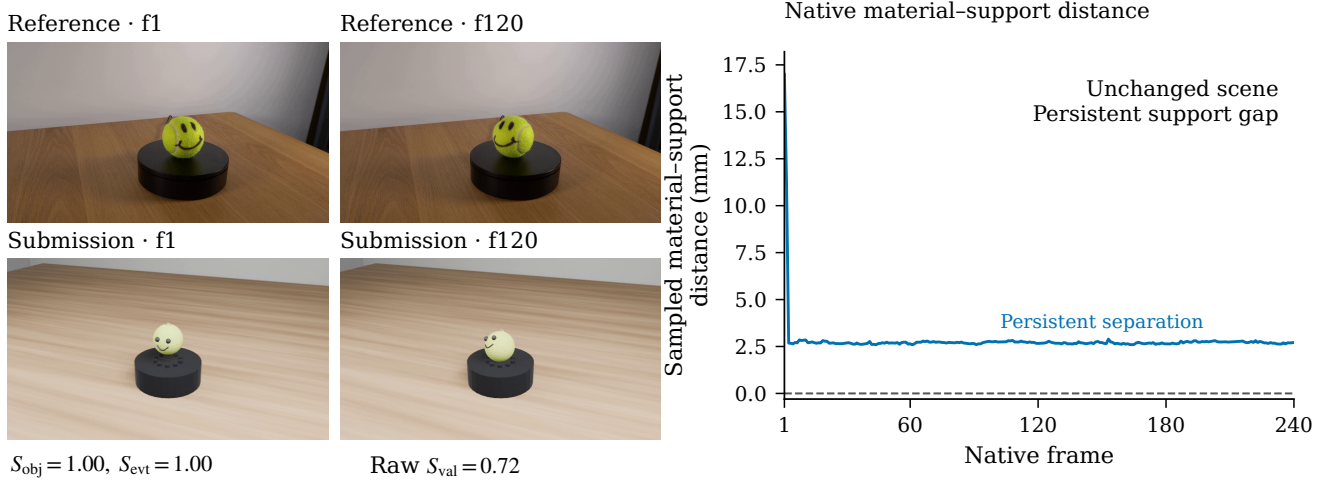


Figure 20. **Claude Sonnet 5: rotation supported by an oversized collision proxy.** Joined top decoration enlarges the spherical collider beyond the ball material, producing persistent remote support/traction (S2). The main curve is the minimum over sampled vertices, edge midpoints and face centroids, not an exact global surface-distance minimum. The unchanged scene continues to support and rotate the ball while sampled material distances remain positive; the dashed line denotes zero material separation. Only original-state measurements are displayed.

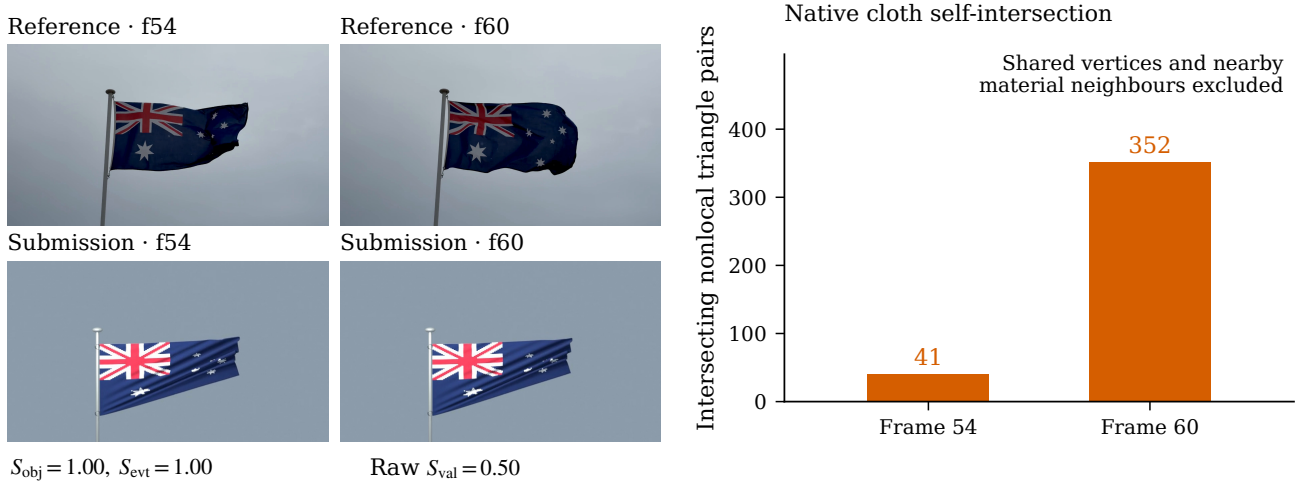


Figure 21. **GPT-6-Luna: nonadjacent flag material crosses itself.** Exhaustive checks of unchanged Blender-scene frames 54 and 60 find 41 and 352 intersecting nonlocal triangle pairs (F2). Pairs sharing a material vertex or lying within a two-quad material neighbourhood are excluded; a frame-54 witness joins patches separated by 18 material rows. These are geometric pair counts at the two inspected frames, not independent events or a claim about the first onset of crossing. The raw validity score is the resolved value 0.50.

D.3. Counterfactual Mechanism Checks

A reconstruction can pass checks along its original trajectory while omitting a physical interaction that the trajectory never exercises. The five selected reconstructions below illustrate this gap: a rolling body does not encounter a nearby dish, the cue ball remains on the table, or a bearing keeps a rotor away from the surface below. Changing the release or connection activates the missing interaction. A failure is established by measured material intrusion or unsupported motion together with a responsive control, rather than by trajectory change alone.

Each selected reconstruction scores 1.0 on its original-condition checks but fails a counterfactual criterion of its physical-validity rubric. The paired reports use identical source and saved-world hashes. The native scene views show the tested intervention states, paired with measurements of geometry and motion. The examples illustrate failures of counterfactual rubric criteria; they do not estimate a failure rate.

D.3.1. A Nearby Obstacle Has No Contact Response

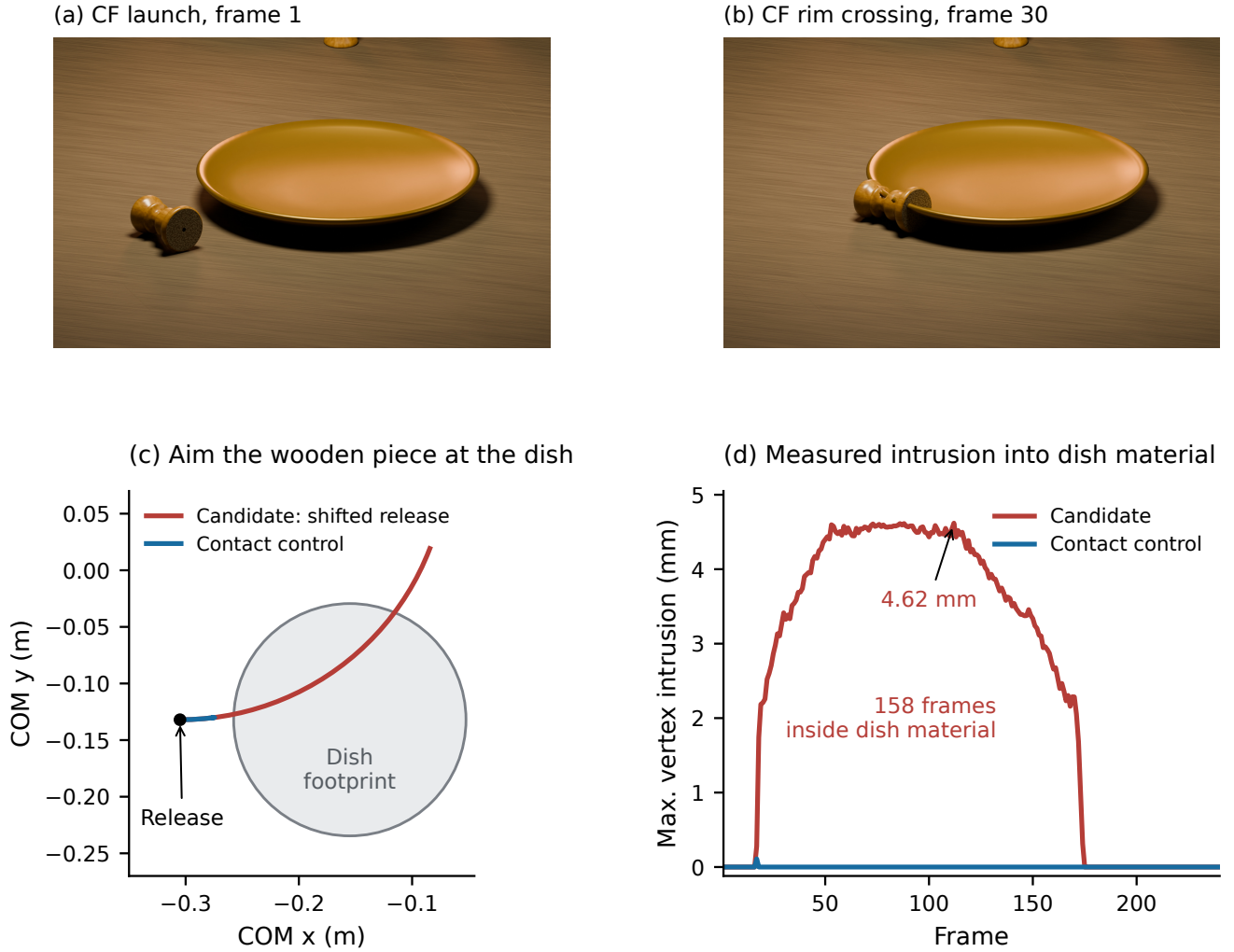


Figure 22. **GPT-6-Astra: the wooden piece passes through the dish.** The views show the shifted release before contact and during rim crossing. The original release avoids the dish. A shifted release starts 32.25 mm clear of its surface but puts piece vertices inside the actual closed dish material for 158 frames, reaching 4.619 mm depth. The grey disc in (c) is only the dish footprint; the failure witness in (d) measures intrusion into its three-dimensional material. Removing the dish leaves the candidate’s centre-of-mass trajectory unchanged. An independent contact control using the source mesh, mass, inertia and integration scale arrests the approach, with only 0.110 mm maximum local contact intrusion. Thus the issue is absent obstacle response, not merely overlap in the image or the top-down projection.

D.3.2. Support Extends Beyond the Physical Table

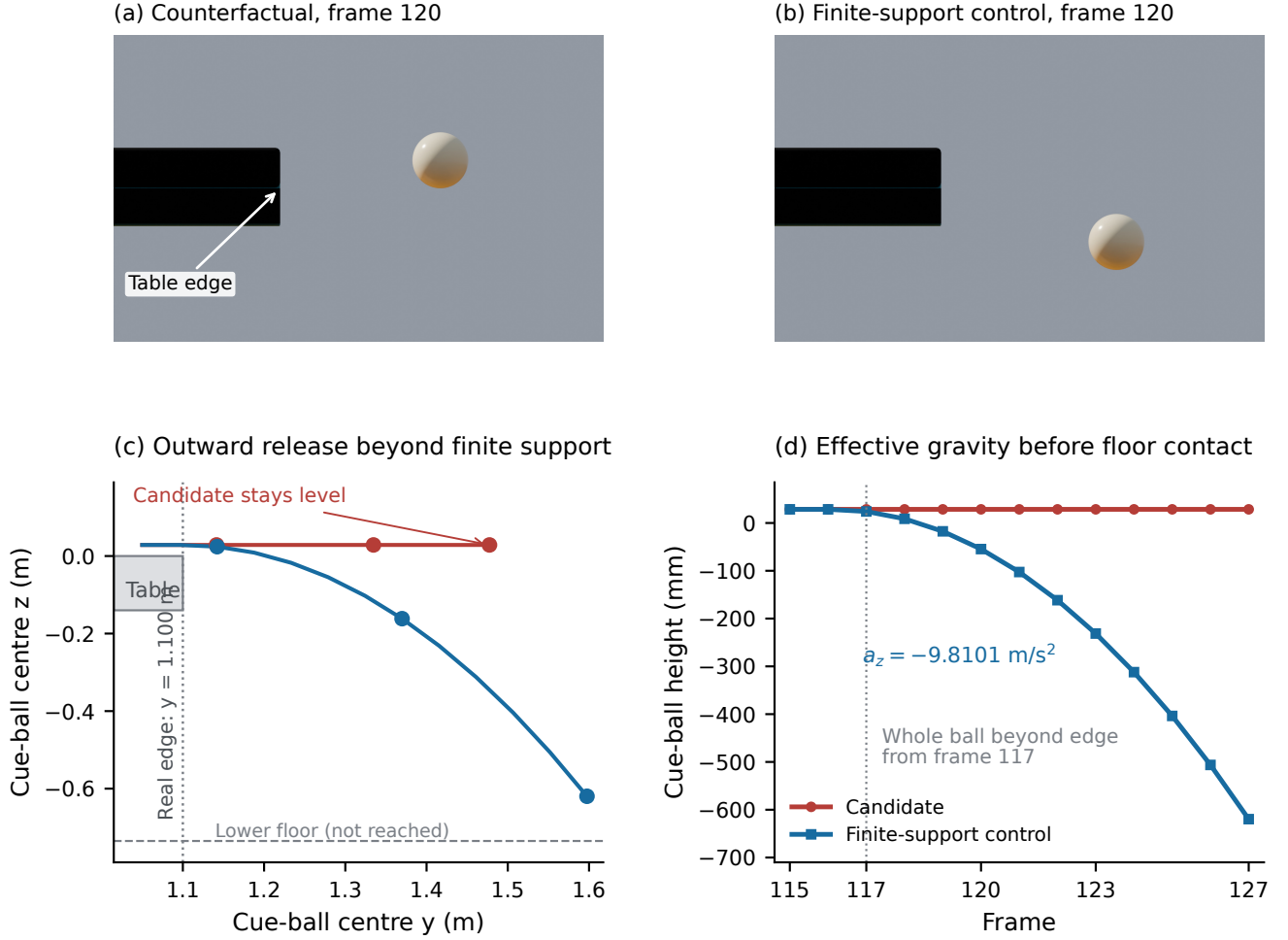


Figure 23. **GPT-6-Astra: the cue ball remains suspended beyond the table edge.** The views compare the outward-release candidate and finite-table control at frame 120. An outward release moves the entire cue-ball envelope beyond the actual support boundary at $y = 1.100$ m from frame 117. Its centre remains at approximately 28.575 mm height through frame 240 (124 samples), despite the absence of table material below it. An explicit infinite-support control reproduces the candidate trajectory exactly. Restricting support to the finite tabletop reproduces the prefix through frame 116 and then gives $a_z = -9.8101 \text{ m/s}^2$. Only frames 115–127 are shown for that control, before it reaches the lower floor; its later floor-contact behavior is outside this witness. The plotted table patch marks the top support boundary, not a scaled cross-section of the entire table.

What the intervention distinguishes. The original-condition checks see rolling supported motion and do not exercise this upper boundary. The intervention tests the spatial extent of the support law against the actual scene material. The finite-support and infinite-support controls separate a missing boundary condition from an ineffective gravity change. The defect is in the reconstructed support mechanism; it is not inferred from the ball leaving the camera view.

D.3.3. A Bearing Hides a Missing Table Collision

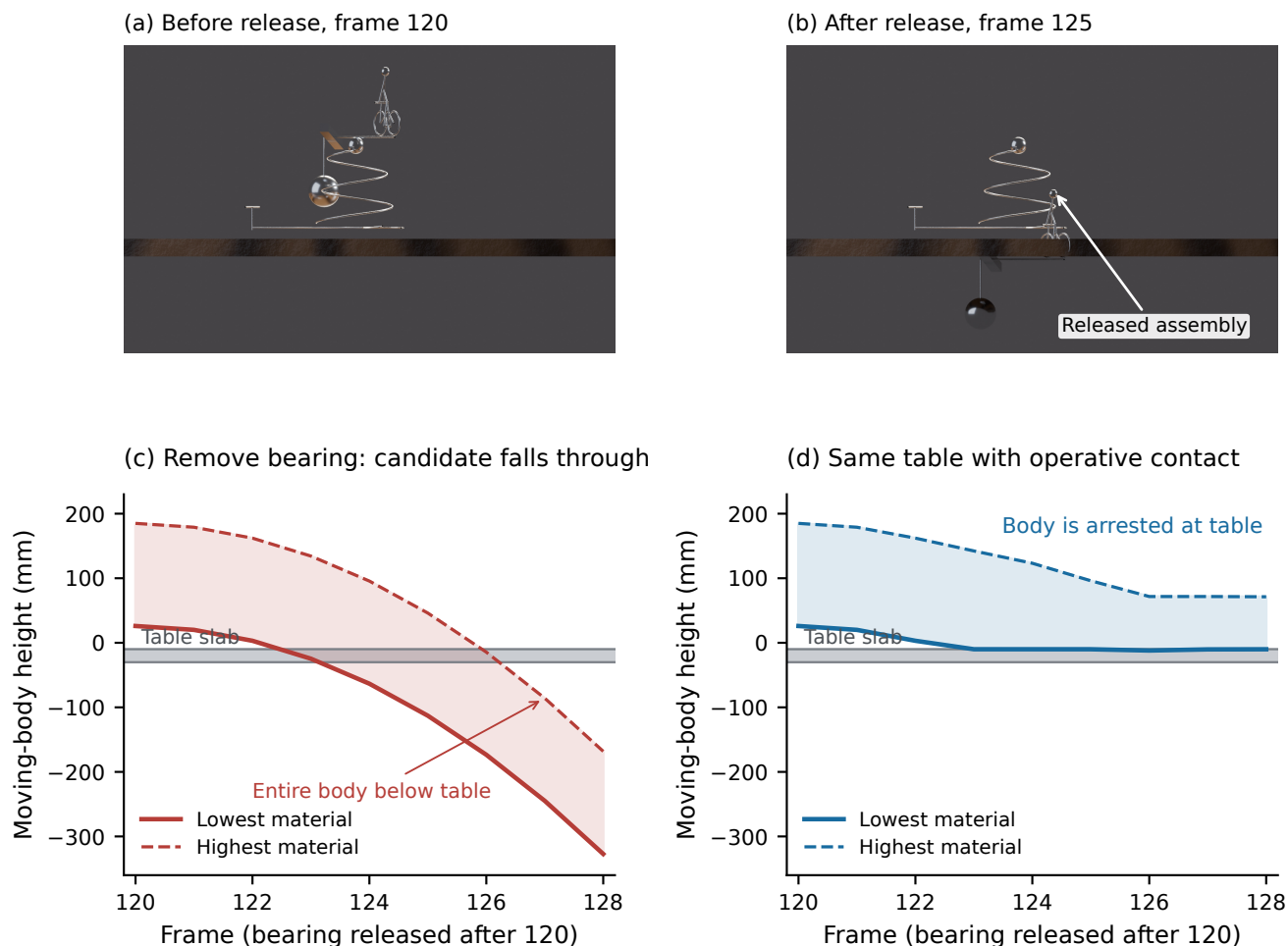


Figure 24. **DeepSeek-V4.1-Flash (max effort): releasing the rotor exposes missing table contact.** The views show the assembly just before release and intersecting the table at frame 125. The original bearing supports the moving assembly. The intervention retains the exact trajectory through frame 120, then removes the sole base-to-beam hinge. The body intersects the unchanged table on frames 123–126 and lies entirely below the table bottom at frame 127. The shaded red and blue bands show each branch’s measured lowest-to-highest material extent, not a filled-body cross-section. Giving the same table geometry an operative collision path arrests the body at its surface; table-removal and disabled-contact controls reproduce the candidate’s free fall. Falling after a bearing is removed is expected. Passing through the table is the counterfactual failure.

Why the control matters. Removing a legitimate support does not itself establish that a reconstruction is invalid. Here, the bearing-retained branch and the unchanged prefix isolate the release, while the table-contact control demonstrates that the available surface can stop the body. The intervention reveals a contact mechanism that the supported original motion did not require. Full source identities, intervention records and trajectory data are retained with the figures.

D.3.4. A Different Reconstruction Omits Contact with a Chess Piece

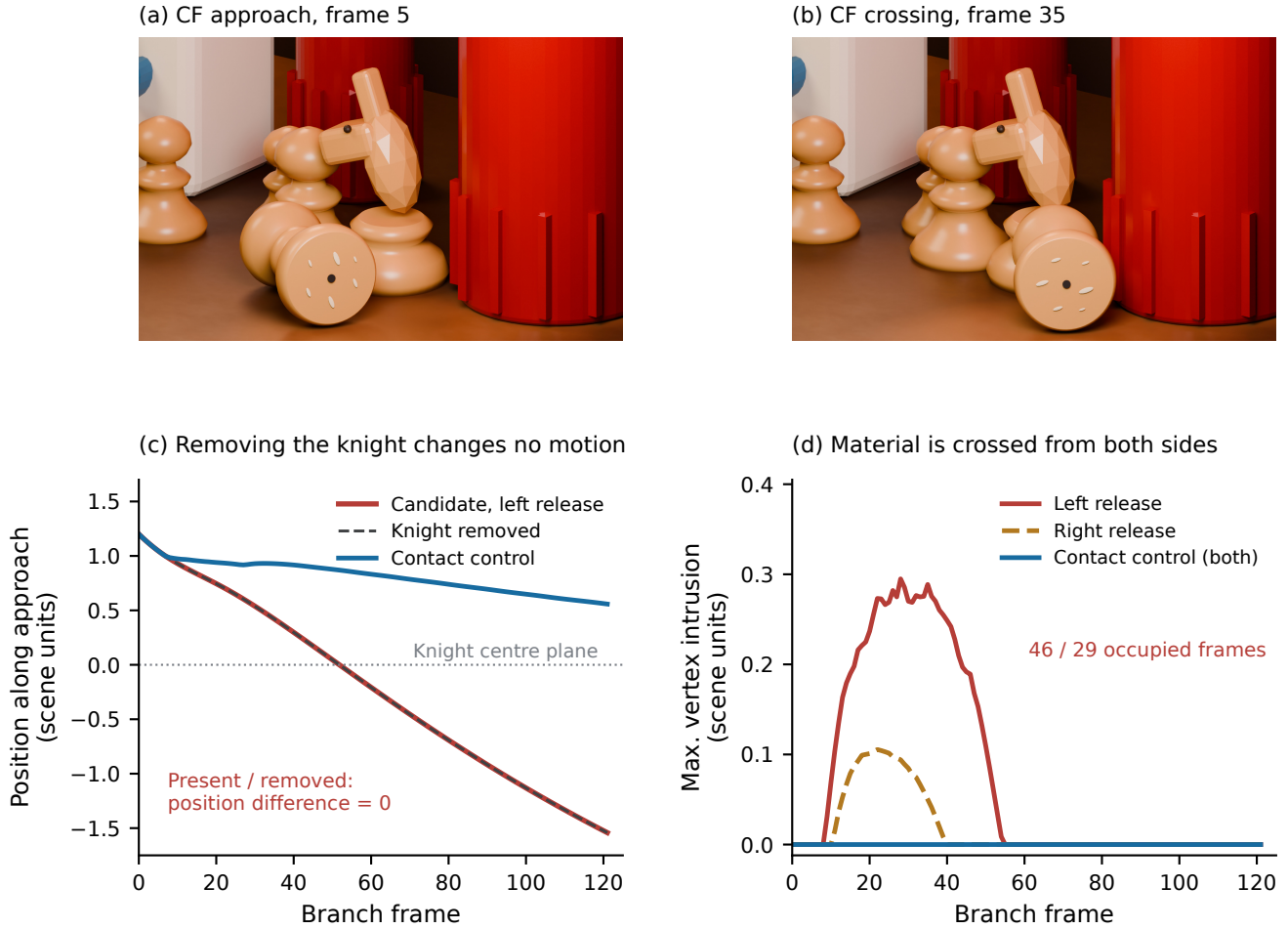


Figure 25. **GPT-5.6-Sol: the rolling piece passes through the knight.** The views follow the left-release branch before and during the crossing. This reconstruction of the same rolling task contains a nearby closed knight assembly whose five material meshes have no collision owner. Two initially separated, table-supported releases put the moving piece inside knight material for 46 and 29 frames, with maximum interior vertex depths of 0.295 and 0.105 scene units. Depth is the largest measured vertex intrusion in either direction between the two meshes. In (c), the left-release positions and velocities are exactly unchanged when the knight is removed, across all 122 recorded samples; crossing the centre plane is only a path illustration, while (d) measures actual material intrusion. Adding passive contact to the unchanged target geometry produces a response and avoids material intrusion for both releases. Repeated runs and doubled solver substeps retain the failure. Length values use native scene units.

What the original trajectory leaves untested. The original-condition checks certify the rolling body’s rigidity, continuity and table support. They do not require a response from the nearby knight because the original motion never reaches it. The changed releases activate this missing contact path. This extends the dish example in Fig. 22 to a different submitted program and a different obstacle selected from its own scene.

D.3.5. A Solid Token Behaves as if It Were Absent

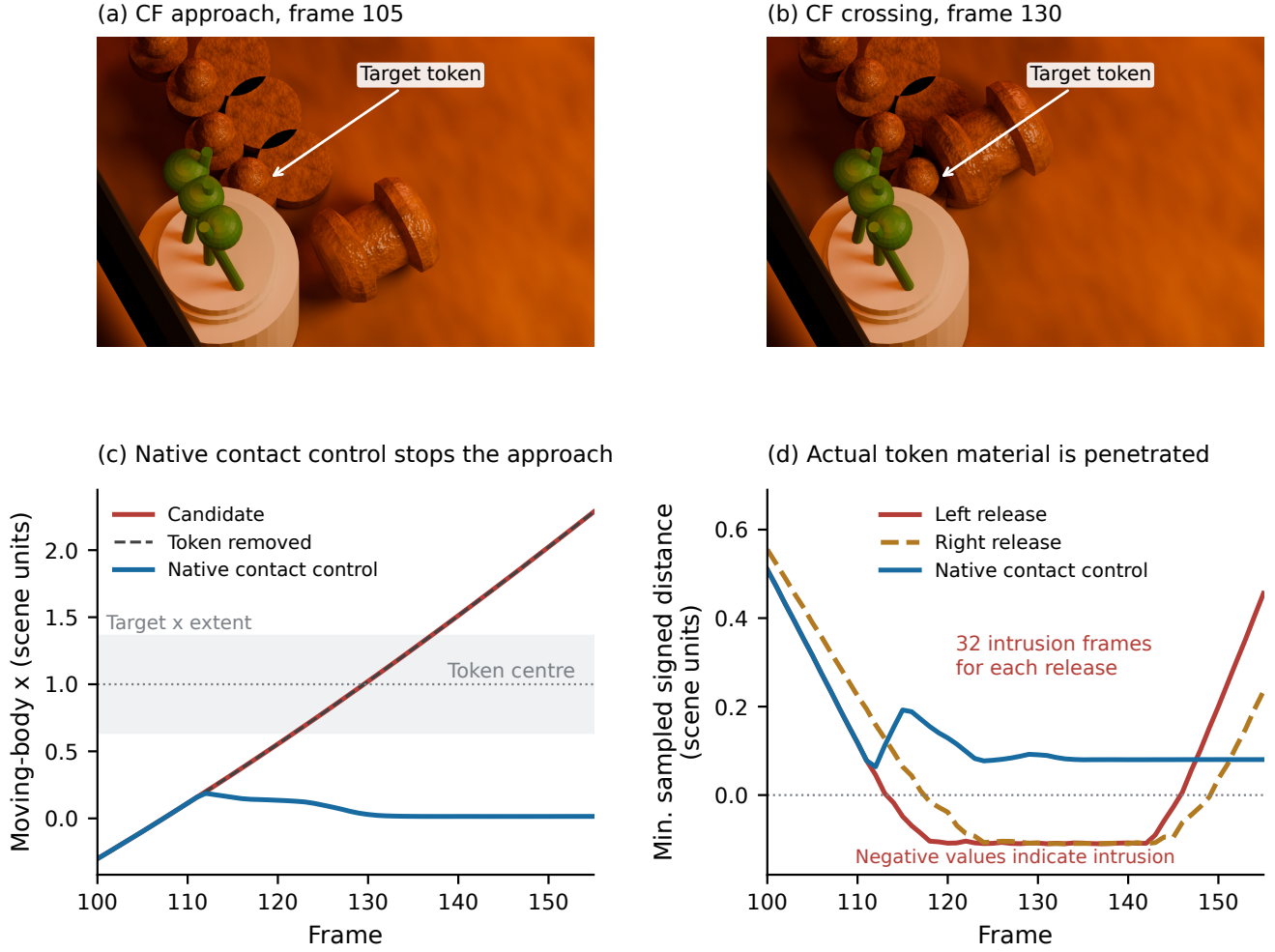


Figure 26. **GPT-5.6-Luna: the rolling piece traverses a solid wooden token.** The views track the left-release branch as the body enters the selected token. The selected target is a filled cylinder with no hole and no physical contact owner. Two target-reaching releases produce mesh intersections on frames 114–145 and 118–149, respectively: 32 consecutive frames each. The minimum cylinder signed distance over sampled moving-body surface points is approximately -0.110 scene units in both branches. For the left release, removing the token changes none of the 240 sampled positions. A native contact control matches the prefix through frame 111 and then stops the approach without token penetration. Its collider responds before the actual material surface, with a dense measured separation of 0.0622 scene units; it is a sensitivity control, not an exact surface-contact trajectory. The grey band in (c) is only the target’s projected x extent; (d) gives the material-distance witness.

Scope of the comparison. The original motion avoids this obstacle, so its table-support checks can pass while the obstacle remains physically inactive. The intervention exposes the mismatch between rendered solid material and the program’s operative collision scene. The initial opposite-side placement blocked by another real object remains in the archived test matrix; the plotted opposite-side branch uses a separately measured, target-reaching release. All three reconstructions of this rolling task retain their own geometry and source scale; no common artificial obstacle is inserted to create the failure.

D.4. Cross-Harness Verification

Harness instructions may influence when an agent checks its reconstruction and how it reserves execution budget for corrections. We explore this possible connection through two DeepSeek-V4.1-Flash tasks where Codex completes execution checks after its final revisions and obtains better event scores than Claude Code. Both agents perform earlier tests; the difference is whether the final repairs are followed by another successful run and output inspection. The galleries show videos generated by the benchmark verifier from the submitted programs, while Appendix D.4.3 examines the agents’ own verification steps and the harness guidance behind them.

D.4.1. Checking Rolling After Contact Changes

In the wooden-piece task, Claude Code performs prototype measurements but exhausts its Blender allowance before executing the final rewritten program. Its last checks find no saved world or completed frame set, and the finalizer refuses the submission (steps 199–205). Codex also initially produces a stalled object. It detects the stall, tests collision shapes and table resolution, and revises the contact geometry and incline (132–160). Its twentieth launch renders the final program, after which it inspects output frames and diagnostics and finalizes (163–169).

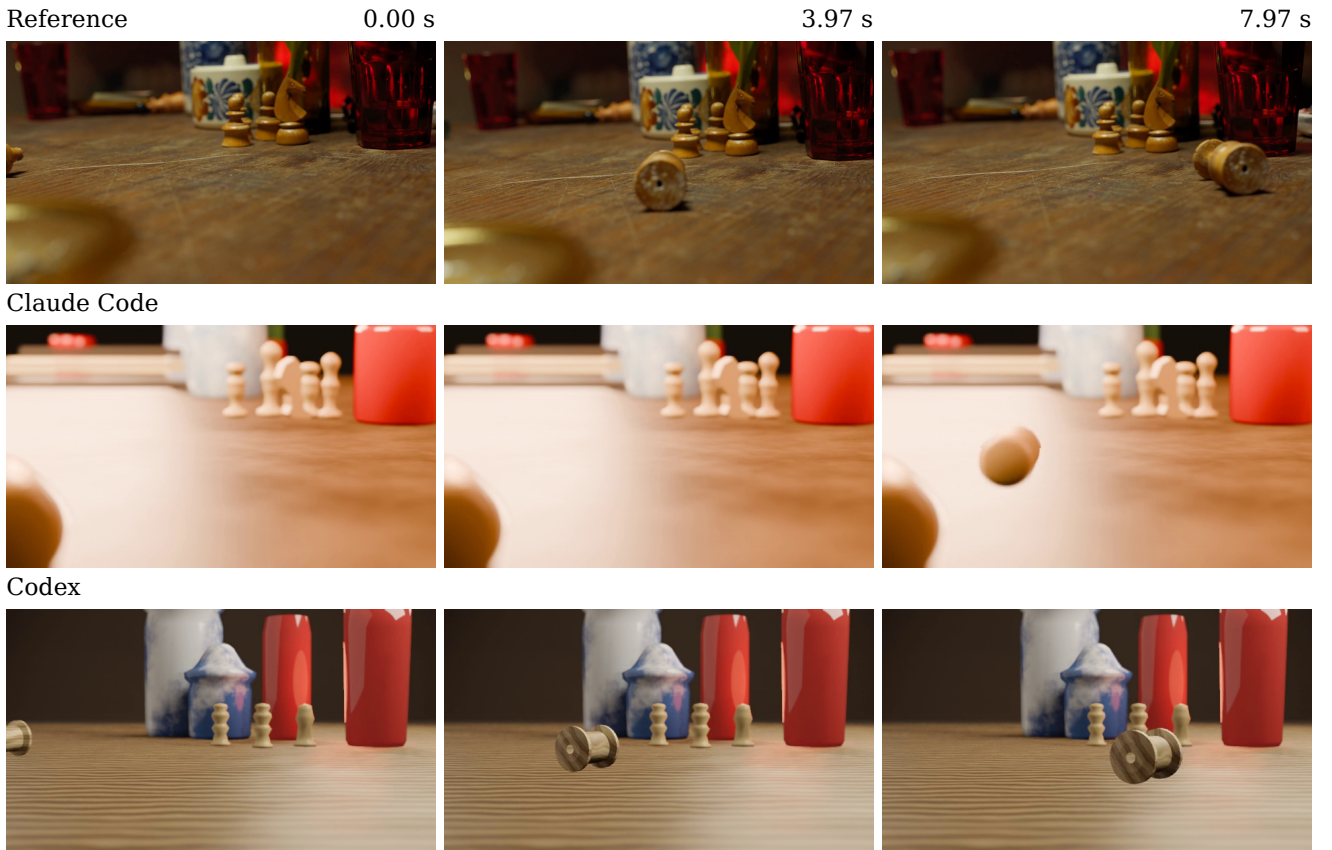


Figure 27. **DeepSeek-V4.1-Flash: wooden-piece rolling.** Codex tests and revises the contact setup before completing the final render and checking its motion.

Outcome. The benchmark event score rises from 0 for Claude Code to 1 for Codex. Appearance remains imperfect: both receive $S_{\text{obj}} = 0.286$. Codex’s tests are followed by contact changes, a new full run, and recovery of the rolling event. Claude Code’s final revision has static checks and earlier-prototype evidence, but no successful execution after the final changes. This pattern suggests that completing behavioral checks within the launch budget may matter for event fidelity.

D.4.2. Checking Accumulation Across a Pour

In the wine-pouring task, Claude Code’s final production bake aborts. It then changes the solver substeps, initial-liquid clearance, and resolution, but has no remaining launch budget to execute those fixes; the agent’s output directory remains empty and finalization fails (224–248). Codex inspects sampled frames from a complete rendered sequence, identifies an excessive filling rate, and twice adjusts the inlet and reruns the program (80–90). It then inspects separated frames and checks a saved-scene preview at frame 240 against the delivered frame (91–94).

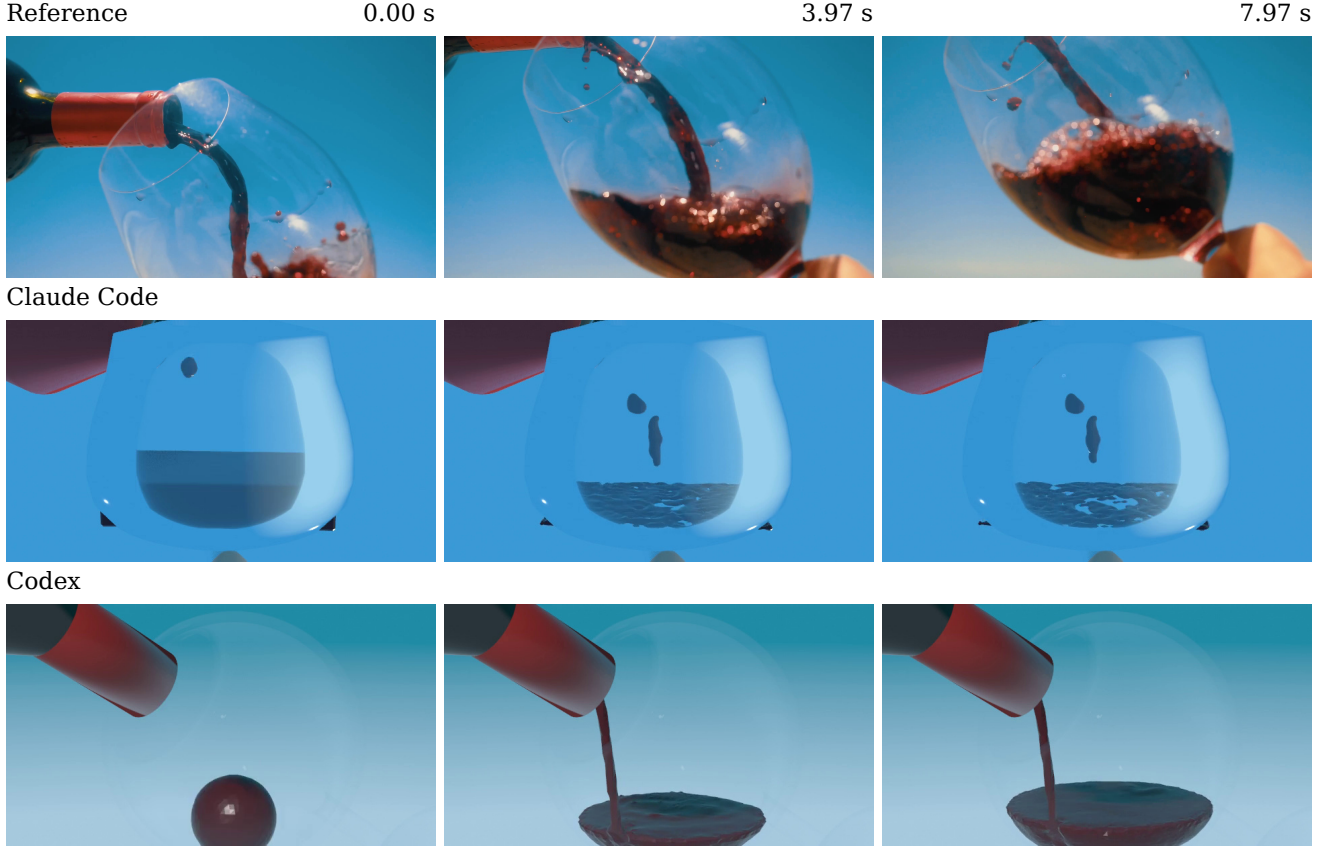


Figure 28. **DeepSeek-V4.1-Flash: wine pouring.** Codex’s checks lead to repeated inlet corrections and a new full render after each correction.

Outcome. The event score increases from 0 to 0.857, and S_{vis} from 0.521 to 0.601. Codex still fails the event item requiring the tilted, visibly supported glass; not every metric improves. Its frame-240 preview has zero pixel difference from the corresponding delivered frame, which establishes that single-frame agreement only. The sequence illustrates how checks across time can reveal a filling-rate error and guide further corrections when execution budget remains.

D.4.3. Harness Instructions and Completed Verification

What the prompts require. The archived Codex 0.150.1 sessions explicitly instruct the agent, in the recorded `never` approval mode, to “proactively run tests, lint and do whatever you need to ensure you’ve completed the task.” Its task prompt also starts with a reminder to reserve the last Blender launch for final validation. After removing this prefix, the benchmark task text is byte-identical across harnesses for both selected tasks: it asks the agent to build a minimal runnable scene, verify it, refine it, and run `finalize_submission`. Thus Claude Code also receives an explicit task-level verification requirement.

Version-matched local captures of Claude Code 2.1.274 and 2.1.278 emphasize faithful reporting of failed tests and skipped steps; we did not observe the same broad proactive-testing instruction in those captures. This part of the comparison uses local reproductions because the historical full Claude Code system prompts were not retained.

Table 9. Prompt evidence for the selected DeepSeek-V4.1-Flash configurations.

Evidence	Claude Code	Codex
Benchmark task	Explicitly requires verification and finalization.	Same task body; an additional leading reminder repeats the final-validation budget requirement.
Native prompt	Historical full prompt unavailable. Version-matched local captures emphasize faithful reporting of tests and skipped steps.	Historical session metadata records an explicit proactive-testing instruction under <code>never</code> .
Interpretation	Earlier checks are present, but the final revisions in these two cases remain unexecuted.	The final revisions in these two cases are run and inspected before the attempt ends.

The observed verification sequences. Table 10 connects the prompt comparison to concrete checks, corrections, and subsequent runs. The two Codex traces contain no observed bypass of the launch cap, backdating of the submitted source, or substitution of older frames for the final revision. The screenshots and scores describe verifier-stage videos; the table describes what the agents themselves managed to execute and inspect.

Table 10. Final-revision verification in the two selected tasks. Step numbers refer to each case’s own agent trace.

Task	Claude Code	Codex
Wooden piece	Revises the final program after the cap is reached; static checks pass, but execution is blocked and output checks fail (166–205).	Detects stalling, revises contact settings, executes the final program at launch 20, then checks frames and finalizes (132–169).
Wine pouring	After a bake abort, changes numerical settings without a successful subsequent execution; finalizer reports missing frames (224–248).	Detects excessive accumulation, adjusts the inlet twice, reruns after each edit, inspects final frames, and checks a saved-scene preview (80–94).

Possible connection to harness guidance. The event-score differences in these examples may be related to the harness’s emphasis on proactive verification, the timing of behavioral checks, and the allocation of the limited launch budget. Codex’s instructions explicitly encourage verification, and its successful revisions here are followed by simulation and output inspection. The two cases illustrate how such guidance may help an agent turn detected defects into tested corrections and better event reconstruction.