MPMWORLDS: MATERIAL-POINT-METHOD SIMULATIONS FOR INFERRING AND EXTRAPOLATING PHYSICAL DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

To study the ability to infer physical dynamics from videos and extrapolate them forward in time, we assemble a dataset of 2D Material Point Method (MPM) physical simulations covering rich physical phenomena such as deformable objects, fluids, kinematic objects, and emitters. Every simulation pairs executable source code, a structured scene configuration, and a rendered video, which lets code generation and video generation be trained and conditioned on the same scene and evaluated on the same video. We study code generation and video diffusion approaches on this dataset, identifying their strengths and weaknesses by varying the amount of physically relevant side information. Our contributions are the dataset and this study. The study does not rank the two approaches; it characterizes how each one fails. The code generation models, beyond giving a working demonstration of automatic synthesis of MPM simulations, reveal that such an approach struggles with inferring physical parameters from visual input, but relative to video diffusion, produces physically and temporally stable extrapolations forward in time, while the video diffusion models more strongly identify geometric properties from visual input but produce physically implausible extrapolations. We test the main findings across three VLMs and three VDMs, including a sample-matched comparison and physically grounded metrics. An anonymized sample of the dataset is available on the **Kaggle website**.

1 INTRODUCTION

Physical simulation is used across robotics, science and engineering, videogames, cognitive science, and the arts. Up until recently, building a high-fidelity physical simulator handling rich physics, such as deformable objects, fluids, etc., meant painstaking programming using techniques such as Material Point Methods (MPM). But could generative models of source code dispense with the need to hand-program new simulations—and could generative models of videos further dispense with the need to have any simulation code at all? These questions are significant, because authoring precise simulation code is expensive and demands domain expertise, while generative video models might achieve much broader coverage of the everyday world than any symbolic simulation ever could.

The ability to infer the physical dynamics of a scene, and successfully extrapolate to future time points, is also a fundamental test of physical reasoning. While the literature has investigated the ability of generative video models to extrapolate physics forward in time (Kang et al., 2024), such works center on simple rigid body dynamics, not the richer dynamics such as fluids, deformable objects, and various different materials.

To answer these questions, we assemble a new dataset, MPMWorld, of physical simulations covering rich dynamics such as fluids, materials such as snow and sand, deformable bodies, and more (Figure 1). We then study the conditions under which generative models of source code (specifically VLMs) can produce accurate simulation code, given partial information such as a history of past frames, or privileged information such as underlying physical parameters. We also study the conditions under which generative video models (specifically video diffusion models, 'VDMs') can render the output of the desired simulation, without ever explicitly constructing the underlying code.

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

We find both methods achieve only partial success. Code generation excels at consistent long-horizon dynamics, but struggles with geometry. Video diffusion excels at geometry but hallucinates dynamics and deteriorates over long time horizons. Our main analysis uses a finetuned VLM (Qwen2.5-VL-7B) and a finetuned VDM (LongCat), and these patterns also hold when we add two more VLMs applied zero-shot (Kimi K3 and Qwen3.5-397B) and two more VDMs (Wan2.1-VACE, finetuned, and LTX-2.3, zero-shot). We probe the models to understand which elements of the simulations they struggle with, identifying specific classes of materials that are surprisingly challenging (or easy) for each approach.

In total we contribute the following:

1. A dataset of 2D physical simulations covering fluids, deformable objects, rigid bodies, emitters, 'motorized' objects such as pinwheels and conveyor belts, and more. Each simulation includes both *source code*, *scene configuration file*, and an associated *video of the simulation running*.
2. Neural network models trained on this dataset. We train a variety of models which either generate physics simulation source code, or which generate raw frames, and which condition on a variety of information sources, such as past frames or privileged information such as the numerical values of physical parameters or initial positions. We compare them against frontier models applied zero-shot.
3. The evaluative role of our dataset and models is to test learning-based methods for inferring physical dynamics, and extrapolating that inference forward in time. Our work analyzes which physical dynamics are challenging to reconstruct and extrapolate. But importantly the analysis offers constructive guidance for future work: By studying which forms of physically-relevant side information most improves the models, we hope to offer constructive guidance on what future modeling should focus on. We do not use the dataset to rank code generation against video diffusion; we use it to identify which physical properties each approach struggles to recover from video and which it struggles to carry forward in time. To support this, we add physically grounded metrics of mass and motion, and calibrate them against simulator state.

1.1 Problem Definition: Inferring and extrapolating physical dynamics.

Given partial information about the physical dynamics of a scene, such as a short video clip, we take *inferring physical dynamics* to mean the recovery (or reconstruction) of a forward simulation of that physical scene.

We consider scenes generated by physics simulation programs. We write ρ for one such program; different programs simulate different scenes with different objects, materials, etc. A program ρ produces a video v comprising a sequence of frames. Given a prefix of v up to time t, written $v_{\leq t}$, inferring the physical dynamics as a program means to recover a $\hat{\rho}$ equivalent to ρ. Extrapolating to future time points means predicting a video $\hat{v} = \hat{v}_{\leq t} || \hat{v}_{>t}$ (given $v_{\leq t}$).

Given the complexity of physical dynamics that we study, we do not expect inference and extrapolation to be easy. To understand why a model struggles with inference or extrapolation for a particular scene, we can provide physically-relevant *side information*, such as the positions of objects, their densities, etc. We extract different forms of side information from the underlying program, written $f(\rho)$. Models predict either a video or a program, i.e. estimate either $p(v \mid v_{\leq t}, f(\rho))$ or $p(\rho \mid v_{\leq t}, f(\rho))$. Figure 7 (Appendix C) illustrates the two inference-and-extrapolation pipelines studied in this work: executable simulation synthesis via VLMs and direct pixel-space continuation via VDMs.

2 MPMWorlds Dataset

To study physical dynamics inference and extrapolation, we introduce **MPMWorlds**, a simulation-based dataset of physical scenes. Each instance includes (i) executable simulation code, (ii) a structured configuration specifying the scene, and (iii) a rendered video of the resulting dynamics. The scene configuration specifies object types, materials, geometry, initial conditions, and appearance, while the simulation code implements the corresponding physical process. This paired representation enables controlled study of dynamics extrapolation, physical state inference, sensitivity to input modalities, and the role of explicit intermediate representations such as code.

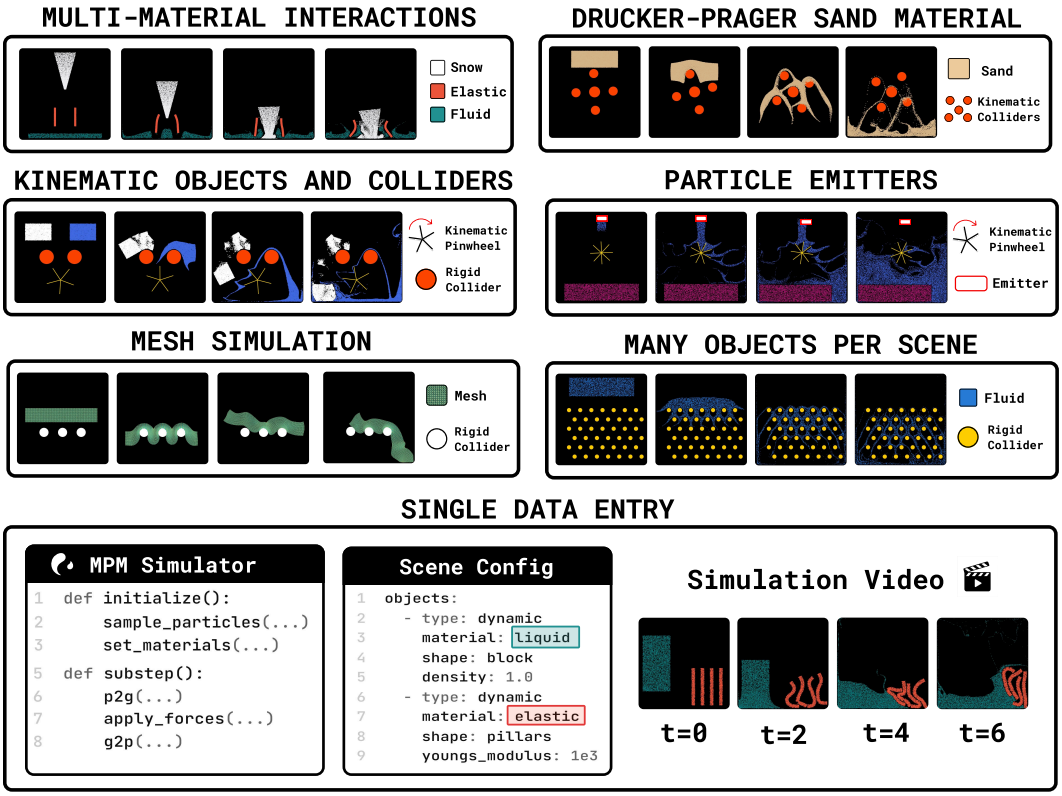

Figure 1: **MPMWorlds dataset overview.** The dataset includes diverse materials and physical interactions. Each entry contains the simulator code, the scene configuration, and the resulting video.

Each configuration is a structured YAML file containing object entries with fields for object type, geometry, material model, initial state, appearance, and optional kinematic or emitter behavior. For example, a dynamic body specifies its shape, color, constitutive model, material parameters, and initial position and velocity; emitters specify spawn region, material, and emission rate.

Dataset Characteristics. We implement the simulations in 2D using the **Material Point Method** (MPM), rendering videos at 512×512 resolution and 30 FPS for 10 seconds. All simulations are written in Taichi, with each program explicitly defining the MPM state update, scene initialization, physical parameters, and time-stepping loop, without relying on high-level simulation APIs.

Appendix L gives a self-contained description of MPM and of the constitutive models that distinguish each material in our simulator.

2.1 Dataset Creation Pipeline

The dataset is generated using a partially synthetic pipeline combining human-written programs, LLM-based synthetic generation, and data augmentation through small perturbations. It is created entirely from scratch and does not rely on existing datasets or sources. The dataset generation pipeline has a hierarchical structure and we explain the steps below:

Step 1: Seed Simulations. We manually write 35 diverse "seed" simulations with varying object layouts, materials, geometries, initial velocities, kinematic elements (e.g., pistons, gates, rotating objects), and rigid colliders. Simulations are paired with a structured scene YAML config. Each seed is a complete, runnable program. Every generated scene records which seeds were used to produce it.

Step 2: Base Simulation Generation and Validation. We expand the dataset in the style of Self-Instruct (Wang et al., 2023) by prompting large language models (ChatGPT o3 and Gemini 2.5 Pro) with 2–3 seed examples to generate new simulation programs and corresponding scene configurations, by combining and extending the provided human written code. Generated simulations are executed and filtered using automated checks for runtime failures, trivial outputs, and severe numerical

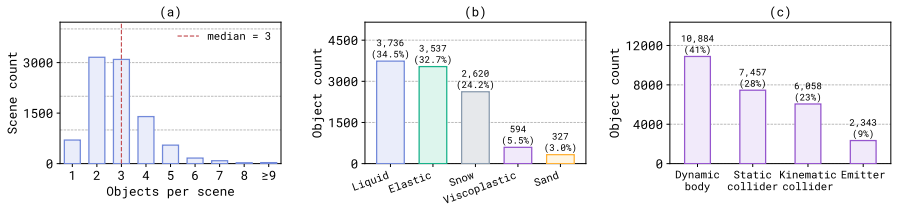

Figure 2: Dataset statistics for **MPMWorlds** across base scenes. (a) Distribution of number of object per scene. (b) Distribution of dynamic-body material types across the dataset. (c) Distribution of object types illustrating the diversity of interaction mechanisms present in the dataset.

instability. Harness-level Taichi and CUDA faults get a single repair attempt. We additionally validate consistency between generated programs and configuration files using an LLM-based verification pass checking object counts, materials, and initial conditions, with flagged failures manually inspected. Physical validity is established by the deterministic execution checks, not by this pass; we additionally re-audit the full dataset with no model in the loop (Section N.1). This process yields 1,093 valid **Parent** simulations, 55.4% of which passed validation on the first attempt, without the repair.

We further expand the dataset by prompting the LLM to modify previously generated scenes through changes such as adding or removing objects, altering materials or geometries, and modifying interactions, resulting in 8,111 **Child** simulations; 95.4% of modified programs validate on the first attempt. Together, the Parent and Child simulations constitute the core 9,204 **Scene Templates** that define the unique physical setups in our dataset.

Step 3: Automated Perturbations. We augment the dataset by perturbing numerical parameters, e.g. object positions, subject to spatial constraints that avoid invalid layouts, such as overlapping objects or objects outside the domain. This reduces bias from LLM-chosen initial configurations, introduces variability in initial conditions, and aids generalization.

2.2 Dataset Statistics

MPMWorlds contains 95,805 rendered simulations, including 9,204 Scene Templates (comprising the full set of Parent and Child simulations) and associated perturbations. Base scenes define the physical setup, while perturbations vary numerical parameters.

Figure 2 summarizes dataset statistics. Scenes have a median of 3 objects and mean of 2.9 objects, with a tail of more complex interactions involving dynamic bodies, colliders, and emitters. The dataset spans multiple classes of physical materials, including liquids, elastic materials, snow, viscoplastic materials, and sand; liquids and elastic materials are most common, followed by snow, while sand and viscoplastic materials provide additional deformation regimes. Overall, 22.5% of scenes contain at least two distinct material types, and 5.3% contain three or more.

Object types include dynamic deformable bodies, static colliders, kinematic colliders, and emitters, enabling interactions such as moving boundaries, externally driven motion, and continuous material injection, substantially increasing the diversity and difficulty of long-horizon dynamics prediction.

Quality and diversity. A re-audit with no model in the loop finds that 97.6% of parent scenes implement the material model their configuration declares. In a user study, five external annotators inspected the rendered videos of 860 scenes and found 97.9% of declared objects present and 96.3% of scenes with each material behaving as its configuration declares. The dataset contains 3,709 distinct compositions of objects and mechanisms, and only 1.4% of scenes have a near-duplicate (Appendix N.1).

3 Experimental Setup and Evaluation

3.1 Input Conditions

We evaluate each model under different input regimes designed to identify which physical properties prove difficult to infer from visual observations alone and if one type of information is much easier to infer from visual information for one approach (VLM or VDM) than the other.

All regimes include the prefix $v_{\leq t}$ (about 2.5 seconds). *Frames Only* gives nothing else; *Full Configuration* adds the complete YAML as $f(\rho)$; *No Materials* removes material identities and parameters from it; and *No Positions* removes initial positions. Definitions are in Appendix M.

For instance, for a scene containing an elastic block, No Materials removes that object's `constitutive_model` field and its material parameters, but keeps its position, shape, and color.

3.2 Models

We compare two types of models, code generation models (VLMs) that are asked to produce executable simulation code and video diffusion models (VDMs) that directly predict future frames. Each family contains a model finetuned on MPMWorlds and models applied zero-shot.

Vision-Language Models. VLMs predict future dynamics indirectly by generating executable MPM simulation code. Given prefix frames and optional configuration information, the VLM outputs a full Python (Taichi) simulation program $\hat{\rho}$, which when executed predicts video $\hat{v}$.

We use Qwen2.5-VL-7B-Instruct (Bai et al., 2025) as the base VLM, which we finetune. We also evaluate two frontier open-weights models zero-shot under their official configurations: Kimi K3 (Team et al., 2026), a 2.8T-parameter model about 400× larger than Qwen2.5-VL-7B, and Qwen3.5-397B-A17B (Qwen Team, 2026). We do not evaluate Gemini 2.5 Pro or o3, since they were used to generate the dataset, so no evaluated model took part in its construction. Our main analysis (Section 4) uses the finetuned Qwen2.5-VL-7B and LongCat; the other models test the generality of its conclusions (Section 4.2). Scaffold details are in Appendix B.

Video Diffusion Models. VDMs predict future dynamics directly in pixel space without constructing an explicit physical representation. We use the 14B LongCat video continuation model (Team et al., 2025) as our diffusion baseline. The model is conditioned on the observed video prefix and, when available, the same textual scene information used for the VLM input regimes. We also finetune Wan2.1-VACE-14B (Team Wan et al., 2025; Jiang et al., 2025), converting it from masked prediction to video continuation, and apply LTX-2.3 (22B) (HaCohen et al., 2025) zero-shot.

3.3 Training and Inference

For all experiments, we construct a 10k-example subset of the dataset with increased sampling weight on base scenes. We evaluate two held-out regimes enabled by the hierarchical dataset structure. The primary **test split** is partitioned at the Parent simulation level: all modifications (Children) and perturbations derived from a held-out base simulation are assigned to test, so models never observe the underlying scene template during training. This evaluates generalization to entirely novel physical scenes.

We additionally construct a **modification validation split** by withholding some modified descendants of training base scenes. This split intentionally preserves partial template overlap while changing object layouts, materials, etc., providing an intermediate regime between exact memorization and fully novel scene generalization.

To improve training and inference efficiency, videos are resized to 256×256 resolution at 15 FPS. For VLM training, we use full supervised finetuning. For VDMs, we observed little difference between full finetuning and LoRA adaptation and therefore use LoRA for all diffusion experiments.

At inference time, we generate multiple candidate programs from the VLM and select among them using evaluation metrics computed on the predicted $\hat{v}_{\leq t}$. Specifically, we sample 15 candidate programs per test example. In contrast, we observe relatively low diversity across diffusion samples under different random seeds and therefore draw a single sample from the VDM. We use 50 denoising

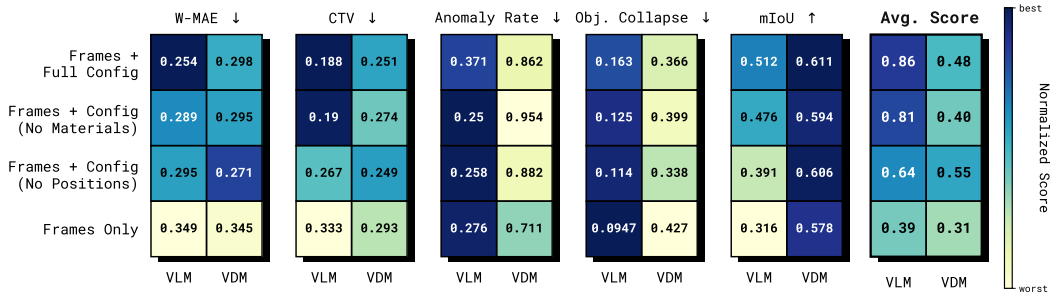

Figure 3: Comparison of VLM- and VDM-based extrapolation across input conditions and evaluation metrics. Values are averaged over the held-out test split. Lower is better for W-MAE, CTV, anomaly rate, and object collapse, while higher is better for mIoU. The final column reports the mean normalized score across metrics for each model and input condition. The VLM in this figure is the finetuned Qwen2.5-VL-7B with top-1 prefix selection per metric over 15 attempts; the VDM is the finetuned LongCat.

steps for diffusion inference, which empirically worked well and which calibrates the inference time compute budget of the two models: 50 denoising steps takes about as long as sampling 15 programs. Because matching compute still gives the VLM a selection step the VDM does not get, Section 4.2 also matches the two families on samples: models are compared on shared scenes, each drawing five samples per scene, and we report the expected score of a single sample, best-of-5 selection, and an oracle upper bound. This comparison is sample-matched, not compute-matched.

3.4 Evaluation Metrics

Metrics. We report five visual metrics: mask IoU (mIoU) for spatial overlap, windowed motion-activity error (W-MAE) for the amount of motion, color total variation (CTV) for appearance, object collapse (OCS) for whether predicted regions persist, and temporal anomaly rate for abrupt appearance changes absent from the ground truth. For the evaluation in Section 4.2, we add five physically grounded metrics: envelope violation, mass-trajectory shape error, and per-material mass loss for mass, and speed ratio and flow agreement for motion. For VLMs, these agree closely with the same quantities computed from simulator particle state (Pearson r 0.90–0.999, Table 8). For VLMs we also read the constitutive model from the emitted program to measure material recovery. Evaluation metrics are defined in Appendices E and K.

Metrics enable Best-of-N sampling for program generation. Because we sample multiple programs at test time, the VLM can benefit from Best-of-N sampling: Each program can be executed and its video compared to the ground truth prefix $v_{\leq t}$ under the above metrics, with the best-scoring program being returned. Note that selection only reads the prefix $v_{\leq t}$; the suffix $v_{>t}$ remains hidden from the learner at test time and is used solely to evaluate the selected program. The same selection cannot help a VDM: the context frames it returns are identical to the ones it was given, so all of its samples have the same prefix score.

4 Results and Analysis

Unless otherwise stated, results are on the *fully held-out test split.* Section 4.1 analyzes the finetuned Qwen2.5-VL-7B and LongCat, with the VLM selected by prefix score over 15 programs; Section 4.2 tests whether the conclusions hold for more models and at a matched sampling budget.

4.1 Main Analysis

How do executable simulation models compare to direct video prediction? Code generation and direct video prediction exhibit substantially different strengths and failure modes under long-horizon physical extrapolation task (Figure 3, Table 1). Code generation (w/ VLMs) generally outperforms direct video prediction (w/ VDMs) on most metrics and input conditions, particularly

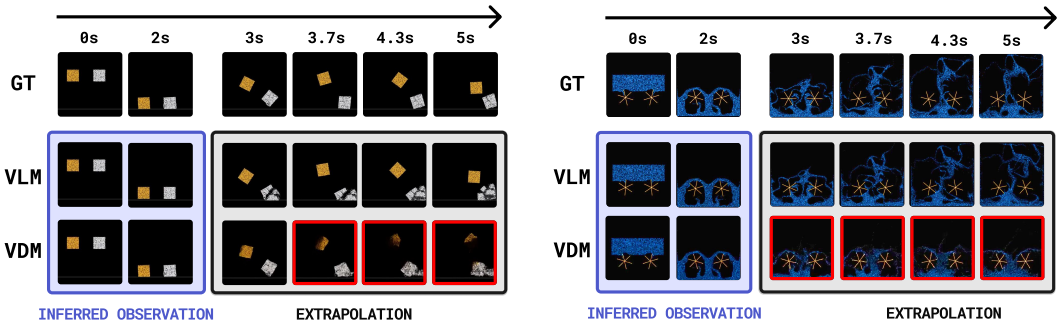

Figure 4: **VDM failure modes in long-horizon physical extrapolation. Left:** In an elastoplastic scene, the VLM correctly maintains rigid object permanence and trajectory. The VDM suffers from object collapse, causing the bouncing block to fade and vanish during extrapolation. **Right:** Subjected to high-energy kinematic colliders, the VLM preserves complex fluid volume and splashing dynamics, whereas the VDM prediction fails to capture high frequency motion.

on temporal stability and long-horizon consistency. VDMs remain better on spatial overlap, and Section 4.2 shows which of these VLM advantages depend on prefix selection. The largest differences between VLMs and VDMs appear in anomaly rate and object collapse, where executable simulations remain substantially more stable over time. Because generated programs are executed inside a physically grounded simulator, successful VLM generations typically preserve object persistence and temporal smoothness unless the predicted code itself becomes numerically unstable. In contrast, VDM predictions frequently exhibit hallucinated motion, disappearing regions, abrupt appearance changes, or temporally inconsistent dynamics, leading to significantly worse anomaly and collapse scores. Representative long-horizon failure cases are shown in Figure 4.

The main exception is mIoU, where VDMs consistently outperform VLMs across all input conditions. This suggests that diffusion models are comparatively better at preserving coarse spatial occupancy and approximate object placement directly from visual observations. However, these spatially plausible extrapolation often deteriorate temporally, producing physically inconsistent motion or unstable long-horizon behavior despite maintaining visually reasonable foreground overlap.

How sensitive are models to structured scene information? Figure 3 shows that VLM performance is highly sensitive to the availability of structured scene information. Removing material or positional information substantially degrades performance, particularly for appearance fidelity (CTV) and motion consistency (W-MAE). In contrast, VDM performance changes comparatively little across input regimes, suggesting that current video diffusion models make limited use of additional structured physical information even when it is provided explicitly. Instead, they appear to rely primarily on short-term visual extrapolation.

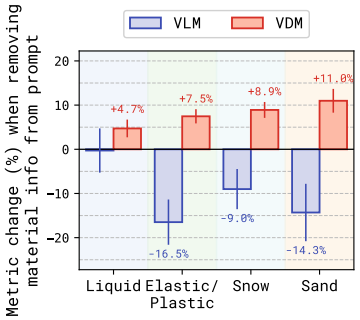

Figure 5: Performance change after removing material information, grouped by material family.

Which materials are hardest to infer from visual information? Figure 5 measures how performance changes when material information is removed from the structured prompt, relative to the full-configuration setting. For VLMs, removing material information has little effect on liquid scenes but substantially degrades elastic/plastic, snow, and sand scenes, suggesting that fluid-like behavior is either more readily inferred from visual context or less sensitive to exact material parameters, whereas coherent or history-dependent materials require explicit material information. Inspecting the emitted programs, we find that the finetuned Qwen2.5-VL implements the correct constitutive family in 84.4% of runs when the prompt states the material, and in 78.7% when it must be inferred from the frames alone. In contrast, VDMs slightly improve when material information is removed, suggesting that they rely primarily on visual data and do not benefit from textual material specifications. In absolute terms (Figure 11, Appendix H), VLMs do worst on sand, likely because its Drucker–Prager

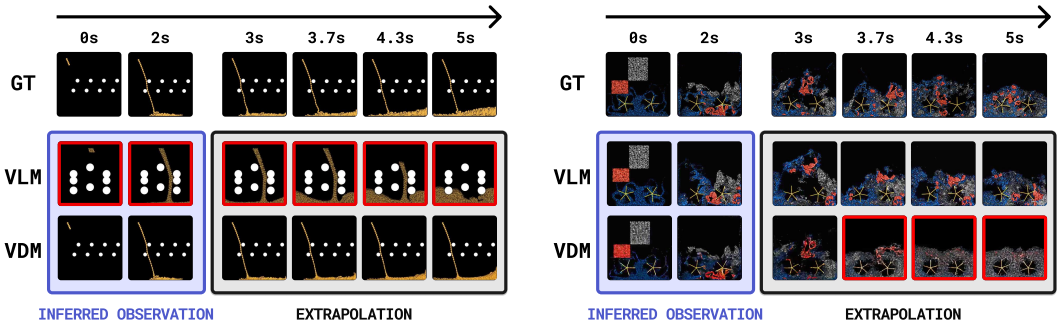

Figure 6: **VLM sensitivity to spatial inputs and compositional scaling. Left:** When explicit positional coordinates are withheld, the VLM struggles with visual state estimation, hallucinating an incorrect obstacle layout (red boxes) that alters the physical trajectory. The VDM, relying directly on pixels, preserves the geometry. **Right:** When provided with full configurations, the VLM scales seamlessly to compositional extremes (simultaneous liquid, elastic, and granular materials interacting with kinematic colliders). In contrast, the VDM suffers a catastrophic compositional breakdown, merging distinct materials into a single static average.

model (Klár et al., 2016) is the hardest constitutive model in our dataset to implement, and VDMs do worst on elastic and plastic scenes, where they lose track of bouncing bodies over time.

How do failure modes evolve over time? VDM motion error grows faster with the prediction horizon than VLM error, and VDM collapses occur early and keep accumulating, while VLM rollouts mostly remain stable after the first frames (Figures 12 and 13, Appendix I).

Figure 6 contrasts the two failure modes.

How do models perform on variants of training scenes? On the modification split, where test scenes are new variants of training scene families, both model classes improve, and the VLMs improve more (Appendix F).

Is Full Configuration just YAML-to-code translation? No. A deterministic baseline that compiles each YAML into a program beats nearest-YAML retrieval, but the finetuned VLM exceeds it on mIoU, W-MAE, and CTV, because the YAML does not define a unique program (Appendix J).

Can prefix quality predict extrapolation quality? A VLM program re-renders the observed prefix, so its reconstruction quality is known at test time. Using it to route each example to the VLM or the VDM continuation improves on either model alone (Table 3); we treat this gate as a diagnostic, not a deployable model (Appendix G).

4.2 Generality of the Findings

The analysis above rests on one VLM and one VDM, and on letting the VLM choose among 15 programs. We test whether the conclusions of the analysis survive changing either choice. We add the other four models of Section 3.2. Qwen2.5-VL, Kimi K3, LongCat, and Wan are compared on a 105-scene subset of the test set on which all four produce a scorable output, each drawing five samples per scene (Table 4, Figure 14). This comparison is sample-matched but not compute-matched, since sampling 15 programs costs about as much as one diffusion sample. Qwen3.5 (one sample per scene) and LTX-2.3 (two input regimes) are evaluated on their own scenes and reported separately (Table 6, Appendix J).

Do the findings generalize across models and scale? Yes. In the shared-scene comparison, LongCat and Wan beat both VLMs on mIoU in all four regimes, and scale does not close the gap: Kimi K3, about 400× larger than our finetuned Qwen2.5-VL, peaks at 0.385, below LongCat's worst regime (0.592). Conversely, Qwen2.5-VL and Kimi K3 both have 1.6–3.2× fewer temporal anomalies than both VDMs in every regime. VLMs still depend heavily on side information: going from Full Config to Frames Only lowers mIoU by 1.70× for Qwen2.5-VL and 1.58× for Kimi K3, against 1.04× for LongCat and no drop for Wan.

Do the findings depend on candidate selection? Mostly not. With one sample per model and no selection, VLMs still produce fewer temporal anomalies, and Qwen2.5-VL still collapses less than both VDMs (Table 4). Best-of-5 selection mainly widens margins the VLMs already have, for example from 2.21× to 2.98× on anomaly rate (average VDM over average VLM, Table 5). The exception is W-MAE and CTV: at one sample and at best-of-5, the VDMs are better on both, so the VLM's advantage on these metrics in Figure 3 likely reflects that it could search over 15 programs and choose among them based on the observed prefix. Kimi K3 does not beat both VDMs on object collapse at one sample, but it does under oracle selection (0.127 against 0.222 and 0.232, Table 5), so its collapses likely come from a minority of unstable programs.

Do the findings hold under physical metrics? Both VLMs have lower per-material mass loss and a speed ratio closer to 1 than both VDMs in every regime (Table 4). LongCat has the highest flow agreement in every regime, matching its lowest W-MAE.

5 Related Works

Further related work is in Appendix A.

Programmatic Physical Reasoning and Simulation Code Generation. Recent work explores executable or symbolic representations for physical reasoning. Scene2Prog (Liu et al., 2019) infers programmatic structure from visual scenes, VisPhyWorld (Liang et al., 2026) studies code-driven physical reconstruction, and PhysCodeBench (Xie et al., 2026) evaluates LLM-based physics simulation synthesis. Our setting instead couples executable MPM programs with rendered rollouts and structured scene configurations, enabling controlled comparison between code-space and pixel-space prediction for deformable multi-material dynamics.

6 Discussion

We introduce a new dataset of physical simulations which help understand the strengths and weaknesses of code generation and video diffusion for physical inference and extrapolation. Despite being limited to 2D physics, the simulations prove challenging for current approaches, owing to the rich variety of physical materials and their interactions instantiated in the data. By feeding privileged information to the models we obtain insight on how to move forward: VLMs benefit strongly from explicit scene information, suggesting they struggle to recover materials, positions, and parameters from vision alone.

Limitations. Our dataset is limited to 2D MPM simulations and does not capture the full material or visual richness of the real world, including hair, cloth, and highly stiff bodies. Because target videos are rendered from executable programs, the benchmark may favor code generation on temporal stability in ways that do not transfer to natural video. VDMs nevertheless achieve higher spatial overlap on the shared scenes, while VLMs have fewer temporal anomalies even in the one-sample comparison without candidate selection (Table 4). We interpret these results as failure modes on this benchmark, not a general ranking of model families. Our physical metrics are extracted from video and can be calibrated against simulator state only for VLMs. Future work should extend the benchmark to 3D scenes and real-world deformable dynamics.

AI Use Statement

Large language models are part of our dataset pipeline: as described in Section 2.1, ChatGPT o3 and Gemini 2.5 Pro generate and modify simulation programs and their scene configurations from human-written seed simulations, and an LLM checks that each program is consistent with its configuration. All generated simulations are filtered by deterministic execution checks, and we audit the dataset without a model in the loop and with external annotators (Appendix N.1).

We also used generative AI tools to assist with writing and software development: editing and restructuring the manuscript text, formatting tables, implementing plotting code in an author-designed figure style, and finding references. The authors reviewed and verified all AI-assisted text, code, and references, and checked all reported numbers against the experimental results. The authors made all

decisions about the dataset, methodology, experiments, and analysis, and take full responsibility for the content of this work.

Ethics Statement

More broadly, realistic physical video generation may be misused to produce misleading simulations, making controlled evaluation and failure analysis important for responsible deployment.

References

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. *arXiv:1908.05656*, 2019.

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. *arXiv preprint arXiv:2406.03520*, 2024.

Hritik Bansal, Clark Peng, Yonatan Bitton, Roman Goldenberg, Aditya Grover, and Kai-Wei Chang. Videophy-2: A challenging action-centric physical commonsense evaluation in video generation. *arXiv preprint arXiv:2503.06800*, 2025.

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

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

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Jing Gu, Xian Liu, Yu Zeng, Ashwin Nagarajan, Fangrui Zhu, Daniel Hong, Yue Fan, Qianqi Yan, Kaiwen Zhou, Ming-Yu Liu, et al. " phyworldbench": A comprehensive evaluation of physical realism in text-to-video models. *arXiv preprint arXiv:2507.13428*, 2025.

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

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. LTX-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2025.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics*, 37(4):150:1–150:14, 2018.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, 2016.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. VACE: All-in-one video creation and editing. *arXiv preprint arXiv:2503.07598*, 2025.

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

Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. Drucker-prager elastoplasticity for sand animation. *ACM Trans. Graph.*, 35(4), July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925906. URL https://doi.org/10.1145/2897824.2925906.

Minchen Li, Chenfanfu Jiang, Zhaofeng Luo, Wenxin Du, Chang Yu, Žiga Kovačič, and Tianyi Xie. *Physics-Based Simulation*. March 2026. doi: 10.5281/zenodo.20597655. URL https://doi.org/10.5281/zenodo.20597655. Open-source online book. Live version available at https://phys-sim-book.github.io/.

Shiqian Li, Kewen Wu, Chi Zhang, and Yixin Zhu. I-PHYRE: Interactive physical reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1bbPQShCT2.

Jiarong Liang, Max Ku, Ka-Hei Hui, Ping Nie, and Wenhu Chen. Visphyworld: Probing physical reasoning via code-driven video reconstruction, 2026. URL https://arxiv.org/abs/2602.13294.

Yunchao Liu, Zheng Wu, Daniel Ritchie, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Learning to describe scenes with programs. In *International conference on learning representations*, 2019.

Fanqing Meng, Jiaqi Liao, Xinyu Tan, Wenqi Shao, Quanfeng Lu, Kaipeng Zhang, Yu Cheng, Dianqi Li, Yu Qiao, and Ping Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation. *arXiv preprint arXiv:2410.05363*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, et al. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *arXiv preprint arXiv:2504.16074*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, et al. Longcat-video technical report. *arXiv preprint arXiv:2510.22200*, 2025.

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

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, 2020.

Hsiao-Yu Tung, Mingyu Ding, Zhenfang Chen, Daniel Bear, Chuang Gan, Josh Tenenbaum, Dan Yamins, Judith Fan, and Kevin Smith. Physion++: Evaluating physical scene understanding that requires online inference of different physical properties. *Advances in Neural Information Processing Systems*, 36:67048–67068, 2023.

Tomer D Ullman, Elizabeth Spelke, Peter Battaglia, and Joshua B Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in cognitive sciences*, 21(9):649–665, 2017.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

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

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

Tianyidan Xie, Peiyu Wang, Yuyi Qian, Yuxuan Wang, Rui Ma, Ying Tai, Song Wu, Qian Wang, Lanjun Wang, and Zili Yi. Physcodebench: Benchmarking physics-aware symbolic simulation of 3d scenes via self-corrective multi-agent refinement. *arXiv preprint arXiv:2604.23580*, 2026.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020.

Jianhao Yuan, Fabio Pizzati, Francesco Pinto, Lars Kunze, Ivan Laptev, Paul Newman, Philip Torr, and Daniele De Martini. Likephys: Evaluating intuitive physics understanding in video diffusion models via likelihood preference. *arXiv preprint arXiv:2510.11512*, 2025.

A Extended Related Work

Video Generation Models as World Models. Recent work increasingly views generative video models as implicit world models (Ha & Schmidhuber, 2018; Bruce et al., 2024). Several benchmarks evaluate physical plausibility in generated videos, including VideoPhy (Bansal et al., 2024; 2025), LikePhys (Yuan et al., 2025), PhyWorldBench (Gu et al., 2025), and analyses of physical-law consistency in video generation (Kang et al., 2024). Unlike prior work focused primarily on perceptual plausibility or short-horizon events (Qiu et al., 2025; Meng et al., 2024), our setting studies long-horizon continuation of deformable multi-material dynamics with paired executable simulations and structured scene representations.

Physical Reasoning Benchmarks. Prior benchmarks evaluate physical reasoning through mechanics puzzles, causal reasoning, and interactive simulation, including PHYRE (Bakhtin et al., 2019), CLEVRER (Yi et al., 2020), Physion (Bear et al., 2021), Physion++ (Tung et al., 2023), and I-PHYRE (Li et al., 2024). PHYBench (Qiu et al., 2025) studies formal physics reasoning in language models. In contrast, our dataset evaluates generative physical continuation, where models must extrapolate future dynamics either through executable simulation synthesis or direct video generation.

Intuitive Physics and Physical Reasoning. Prior work studies physical reasoning as the ability to infer and predict physical dynamics from visual observations. Cognitive theories propose that human intuitive physics operates through approximate internal simulation (Ullman et al., 2017), while benchmarks such as Physion (Bear et al., 2021) and Physion++ (Tung et al., 2023) evaluate physical understanding in controlled environments. Learned relational and object-centric dynamics models include Interaction Networks (Battaglia et al., 2016), latent physical-property inference from videos (Wu et al., 2015; 2016), and program-like scene representations (Liu et al., 2019). Our work instead studies whether modern generative models can recover executable or implicit forward models of complex physical scenes.

B Model Details

At inference time, the model receives sampled frames from the prefix, optional side information, and a lightweight program scaffold containing only boilerplate imports and function signatures. The scaffold fixes non-semantic code structure, such as function names and ordering, while the model generates the simulation logic, object initialization, material parameters, and update rules. This focuses learning on the physically meaningful parts of the program rather than arbitrary formatting choices.

C Task Pipelines

These problems are practically relevant because extrapolations forward in time can serve as a world model for e.g. planning or intuitive physics purposes, and inferring the underlying program is valuable for engineers or artists wishing to work with the physics code. We consider a variety of partial information sources in part because end-users may wish to specify a physical simulation with more than just images or video, but also provide specific physical parameters, while keeping the rest unspecified.

The recovered physical simulation could be represented explicitly as source code, or it could be implicitly represented as neural activations, but in either case, it should support *extrapolation to future time points*, meaning that we can predict what the scene will look like at any point in the future.

D Full test set evaluation results

Table 1 gives the values of Figure 3 with standard errors.

D.1 Example Results

Finally, both model families degrade substantially in the frames-only regime, indicating that long-horizon physical extrapolation remains difficult even when models observe several seconds of

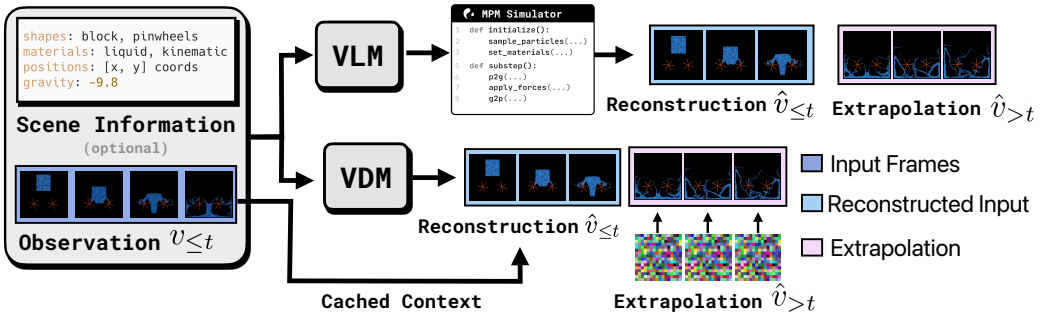

Figure 7: **Reconstruction and extrapolation pipelines.** Models must predict a full video sequence ($\hat{v} = \hat{v}_{\leq t} \parallel \hat{v}_{>t}$) from an initial observation ($v_{\leq t}$). **Top:** The VLM synthesizes a simulation program that executes from $t = 0$, explicitly reconstructing the input (blue) alongside the future extrapolation (pink). **Bottom:** The VDM operates in pixel space, using the input strictly as conditioning to generate future frames.

Table 1: Visual metrics for shared input regimes across VLM and VDM on **test holdout set** (top-1 per metric, mean ± 1 SEM). **Blue bold** = best, red = worst per metric.

	Input Regime	WMAE ↓	CTV ↓	Anomaly ↓	Collapse ↓	mIoU ↑	SEMD ↓	MSE ↓
VLM	Frames + Full Config	**0.254 ± 0.026**	**0.188 ± 0.017**	0.371 ± 0.042	0.163 ± 0.026	0.512 ± 0.021	0.064 ± 0.008	0.027 ± 0.002
	Frames + No Material Info	0.290 ± 0.025	0.190 ± 0.014	**0.250 ± 0.038**	0.125 ± 0.025	0.476 ± 0.020	0.066 ± 0.004	0.028 ± 0.002
	Frames + No Position Info	0.295 ± 0.023	0.267 ± 0.014	0.258 ± 0.040	0.114 ± 0.024	0.391 ± 0.016	0.075 ± 0.004	0.036 ± 0.002
	Frames Only	0.349 ± 0.025	0.333 ± 0.017	0.276 ± 0.040	**0.095 ± 0.023**	0.316 ± 0.016	0.117 ± 0.015	0.037 ± 0.002
VDM	Frames + Full Config	0.298 ± 0.018	0.251 ± 0.007	0.862 ± 0.028	0.366 ± 0.034	**0.611 ± 0.015**	**0.044 ± 0.002**	**0.019 ± 0.001**
	Frames + No Material Info	0.295 ± 0.018	0.274 ± 0.009	0.954 ± 0.017	0.399 ± 0.036	0.594 ± 0.015	0.045 ± 0.002	**0.019 ± 0.001**
	Frames + No Position Info	0.271 ± 0.017	0.249 ± 0.007	0.882 ± 0.026	0.338 ± 0.032	0.606 ± 0.015	**0.044 ± 0.002**	**0.019 ± 0.001**
	Frames Only	0.345 ± 0.018	0.293 ± 0.011	0.711 ± 0.037	0.427 ± 0.035	0.578 ± 0.017	0.049 ± 0.004	0.020 ± 0.001

rollout history. However, the degradation is qualitatively different: VLM failures typically arise from incorrect physical implementation or unstable generated programs, whereas VDM failures are dominated by accumulated temporal drift, object instability, and incoherent motion over longer horizons. Additional qualitative results are shown in Figures 8 and 9.

Additional qualitative examples of model failure modes are provided in Figures 8 and 9, illustrating object-collapse failures, material misidentification, and spatial reasoning errors across both model classes.

E Evaluation Metrics

Metrics that compare the two model families must operate on rendered video, because VDMs do not expose simulator state; where simulator state exists, for the ground truth and for executed VLM programs, we calibrate the physical metrics against it (Table 8). We evaluate spatial overlap, motion, appearance, object persistence, and temporal coherence. Tables 1 and 2 also report MSE and sliced Earth Mover's Distance (SEMD), which we do not use as primary metrics because they correlate poorly with long-horizon physical consistency.

E.1 Mask Intersection-Over-Union

For each frame, RGB images are thresholded into binary foreground masks and the Intersection-over-Union is computed frame-wise.

$$\mathrm{IoU}_t = \frac{|M_t^{\mathrm{GT}} \cap M_t^{\mathrm{pred}}|}{|M_t^{\mathrm{GT}} \cup M_t^{\mathrm{pred}}|},$$

with IoU$t = 1$ when both masks are empty. The final score is the mean over all video frames.

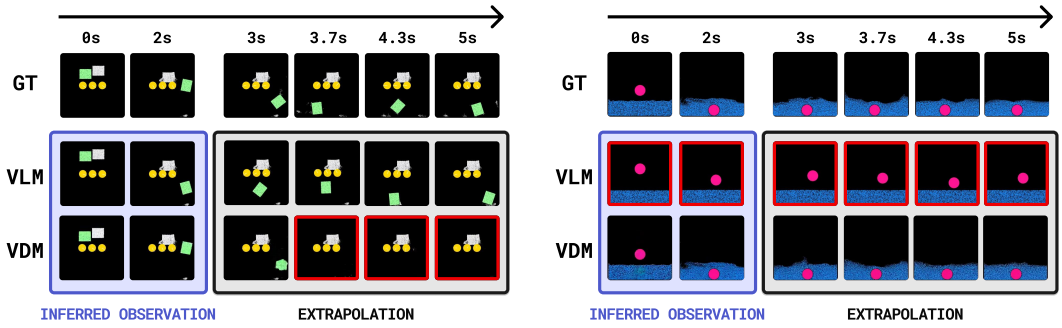

Figure 8: **Failure modes in object permanence and material inference. Left:** In an elastoplastic scene, the VLM correctly maintains rigid object trajectories, while the VDM suffers from severe object collapse, causing the bouncing green block to disappear during extrapolation. **Right:** When explicit material properties are withheld from the input prompt, the VLM sometimes struggles with material inference from frames, hallucinating incorrect interaction dynamics (e.g., treating the liquid surface as a rigid boundary). The VDM, relying purely on pixel patterns, more accurately captures object materials.

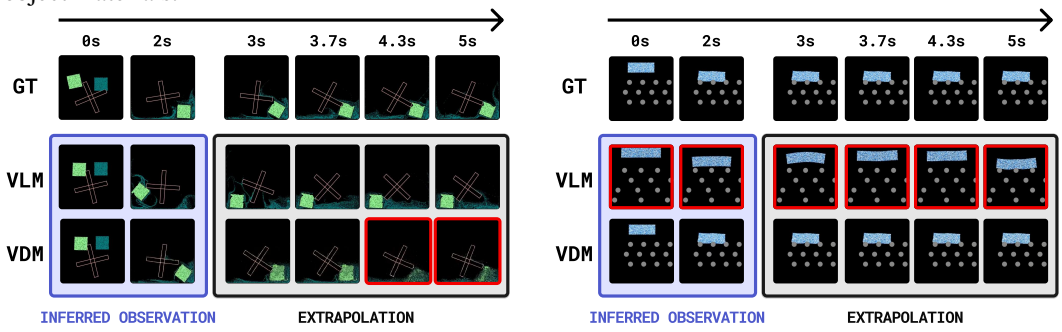

Figure 9: **Contrasting failure modes in complex dynamics and spatial reasoning. Left:** In a scene featuring high-energy multi-material interactions (a kinematic pinwheel, fluid, and a rigid block), the VLM accurately preserves object permanence and fluid volume. The VDM suffers from temporal object collapse, causing the rigid block to unphysically dissipate into the surrounding fluid during extrapolation (red boxes). **Right:** When explicit positional coordinates are limited, the VLM struggles with visual state estimation, hallucinating an incorrect geometric layout for the obstacle pegs (red boxes). The VDM, relying directly on pixel-space conditioning, successfully preserves the underlying spatial geometry.

E.2 Object Collapse Score (OCS)

We measure prediction-internal temporal object coherence by checking whether objects generated early in the predicted rollout remain visible over time. We segment foreground components in an early frame using brightness thresholding and connected components, merge components with near-identical colors, and use the surviving components as predicted object/material regions. Each region defines an initial area $a_n^{(0)}$ and reference color $\mathbf{c}_n$.

At each later frame t, we estimate the area $a_n^{(t)}$ of region n by counting foreground pixels whose chromaticity and brightness remain close to $\mathbf{c}_n$. The object-level collapse score is

$$s_n^{(t)} = \max\left(0, 1 - \frac{a_n^{(t)}}{a_n^{(0)}}\right),$$

and the per-video score is the worst collapse over all initialized regions and frames:

$$\mathrm{OCS} = \max_{n,t>0} s_n^{(t)}.$$

Lower values indicate that predicted objects or material regions remain stable; values near one indicate that at least one generated region nearly disappears during the predicted video. For example

a value of 0.8 means on averaged over all test samples in that group, the single worst object in each generated video lost 80% of its initial pixel count at some point during the predicted video.

E.3 Temporal Anomaly Rate (RTSJ)

For each frame, we divide the image into coarse spatial regions and compute HSV color histograms within each region. Temporal jumps are measured by comparing histogram statistics across neighboring temporal windows, producing a localized spatiotemporal change signal. We then spatially aggregate this signal to obtain a jump magnitude J_t for each timestep t.

For both the prediction and ground truth, we compute jump magnitudes J_t^{pred} and J_t^{GT}. The excess jump is

$$e_t = \max\left(J_t^{\text{pred}} - J_t^{\text{GT}}, 0\right).$$

A prediction is flagged as anomalous if

$$\text{anomaly} = \mathbf{1}\left[\max_t e_t > \tau\right].$$

The reported anomaly rate is the proportion of test samples flagged as anomalous.

E.4 Windowed Motion Activity Error (W-MAE)

We measure whether the prediction exhibits a similar amount of temporal activity as the ground-truth video, independent of exact spatial alignment. Motion activity is computed from frame-to-frame intensity changes aggregated over coarse spatial regions and temporally smoothed using a sliding window. W-MAE is the normalized difference between predicted and ground-truth activity magnitudes over time.

E.5 Color Total Variation (CTV)

To evaluate appearance and material-composition accuracy, we compare foreground color distributions between predicted and ground-truth rollouts. At each frame, foreground pixels are discretized into RGB histograms and compared using total variation distance.

F Modification test set values

How well do models perform on modifications of previously observed scenes? Figure 10 compares the fully held-out test split with the modification validation split, where models are evaluated on modified descendants of training scene families. Both model classes improve on the modification split, confirming that fully novel scene templates are harder than new variants of familiar scene structures.

The improvement is larger for VLMs, which gain across nearly all metrics and input regimes, especially in motion fidelity and temporal stability. VDMs also improve, but collapse and anomaly rates remain high, while mIoU remains their strongest metric. Thus, the VLM–VDM gap is not explained solely by memorization or scene novelty, suggesting that executable simulation generation benefits from familiar interaction structure, whereas pixel-space prediction remains better at geometric overlap but weaker at long-horizon dynamics. Figure 6 highlights these contrasting failure modes: VLMs struggle with visual state estimation when positional information is removed, while VDMs fail under compositional multi-material interactions despite preserving coarse geometry.

G Prefix-Quality Threshold Gating

As a diagnostic, we fit per-metric thresholds on prefix reconstruction quality to choose between the VLM and the VDM continuation. The resulting gate improves over the better single model on every metric and input regime (Table 3), indicating that prefix reconstruction quality is predictive of continuation quality; because thresholds are fitted separately for each metric and condition, we do not treat this as a deployable hybrid model.

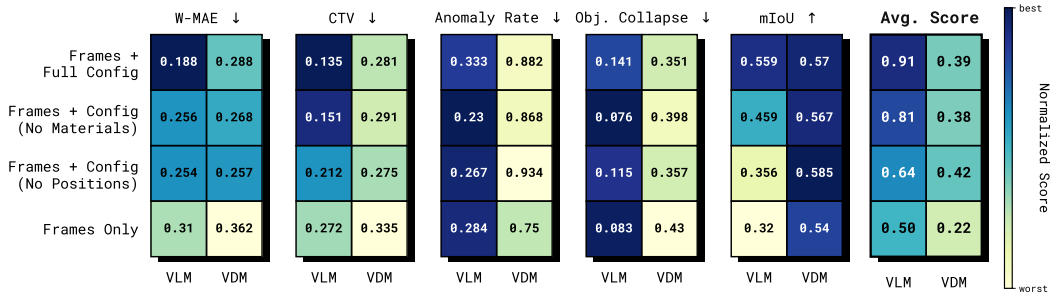

Figure 10: Comparison of VLM- and VDM-based extrapolation across input conditions and evaluation metrics. Values are averaged over the **modification test split**. Lower is better for W-MAE, CTV, anomaly rate, and object collapse, while higher is better for mIoU. The final column reports the mean normalized score across metrics for each model and input condition.

Table 2: Visual metrics for shared input regimes across VLM and VDM on **modification validation set** (top-1 per metric, mean ± 1 SEM). **Blue bold** = best, red = worst per metric.

	Input Regime	WMAE ↓	CTV ↓	Anomaly ↓	Collapse ↓	mIoU ↑	SEMD ↓	MSE ↓
VLM	Frames + Full Config	**0.188 ± 0.025**	**0.135 ± 0.014**	0.333 ± 0.051	0.141 ± 0.028	0.559 ± 0.023	0.053 ± 0.005	0.018 ± 0.002
	Frames + No Material Info	0.256 ± 0.031	0.151 ± 0.017	**0.230 ± 0.045**	**0.076 ± 0.022**	0.459 ± 0.028	0.081 ± 0.013	0.022 ± 0.002
	Frames + No Position Info	0.254 ± 0.027	0.212 ± 0.017	0.267 ± 0.048	0.115 ± 0.029	0.356 ± 0.023	0.082 ± 0.012	0.026 ± 0.002
	Frames Only	0.310 ± 0.030	0.272 ± 0.020	0.284 ± 0.048	0.083 ± 0.023	0.320 ± 0.022	0.090 ± 0.006	0.028 ± 0.002
VDM	Frames + Full Config	0.288 ± 0.025	0.281 ± 0.018	0.882 ± 0.037	0.351 ± 0.043	0.570 ± 0.024	0.075 ± 0.017	0.015 ± 0.001
	Frames + No Material Info	0.268 ± 0.025	0.291 ± 0.015	0.868 ± 0.039	0.398 ± 0.045	0.567 ± 0.023	0.067 ± 0.014	0.015 ± 0.001
	Frames + No Position Info	0.257 ± 0.023	0.275 ± 0.016	0.934 ± 0.029	0.357 ± 0.043	**0.585 ± 0.023**	**0.050 ± 0.005**	**0.014 ± 0.001**
	Frames Only	0.362 ± 0.028	0.335 ± 0.021	0.750 ± 0.050	0.430 ± 0.044	0.540 ± 0.025	0.086 ± 0.019	0.015 ± 0.001

For each metric m and task condition, we compute a prefix score $p_i^m = m(\hat{x}_i^{\mathrm{VLM}}, x_i)$ between the predicted and ground-truth prefix, along with continuation scores $c_{i,\mathrm{VLM}}^m$ and $c_{i,\mathrm{VDM}}^m$ for the corresponding continuations.

We then sweep thresholds τ on a held-out validation set and select the threshold minimizing the average continuation error:

$$\tau^{m*} = \arg\min_{\tau} \frac{1}{N} \sum_{i=1}^{N} \begin{cases} c_{i,\mathrm{VLM}}^m, & p_i^m < \tau, \\ c_{i,\mathrm{VDM}}^m, & \text{otherwise}, \end{cases}$$

for lower-is-better metrics. For higher-is-better metrics such as mIoU, the inequality is reversed.

At inference time, we sample multiple VLM candidates, select the candidate with the best prefix score, and apply the learned threshold to decide whether to use the VLM or VDM continuation. Thresholds are learned independently for each metric and task condition.

Table 3: Prefix-quality gate against the better of the two single models on the test split, for each metric and input regime (VLM: finetuned Qwen2.5-VL-7B with top-1 prefix selection over 15 attempts; VDM: finetuned LongCat). Thresholds are fitted on a validation set, separately for each metric and regime.

	W-MAE ↓		CTV ↓		Anomaly ↓		Collapse ↓		mIoU ↑	
Input regime	gate	best	gate	best	gate	best	gate	best	gate	best
Full Config	0.144	0.254	0.132	0.188	0.001	0.371	0.126	0.163	0.633	0.611
No Materials	0.176	0.289	0.139	0.190	0.002	0.250	0.073	0.125	0.629	0.594
No Positions	0.197	0.271	0.193	0.249	0.001	0.258	0.063	0.114	0.617	0.606
Frames Only	0.221	0.345	0.213	0.293	0.001	0.276	0.044	0.095	0.617	0.578

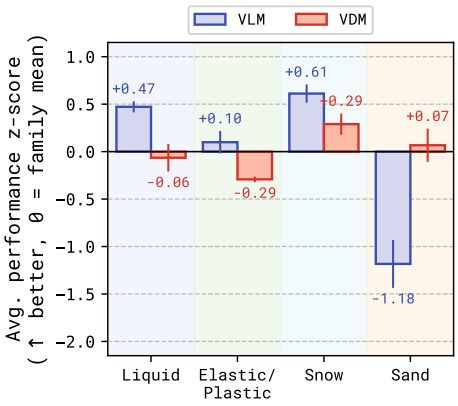

Figure 11: Average normalized performance by material family. Zero denotes each model-family mean across materials.

H Material Sensitivity

Figure 11 aggregates normalized metric performance across tasks for each material family, for the finetuned Qwen2.5-VL and LongCat. VLMs perform best on liquid and snow scenes and worst on sand, whose Drucker–Prager model (Klár et al., 2016) is among the most difficult constitutive models in our dataset to implement. Although Drucker–Prager sand is also an elastoplastic material (Li et al., 2026), we separate it from elastic/plastic scenes because it exhibits granular flow, frictional yielding, and loss of coherent object shape. VDMs perform worst on elastic and plastic scenes, where bouncing bodies must keep coherent trajectories over time.

I Temporal cumulative collapse rate

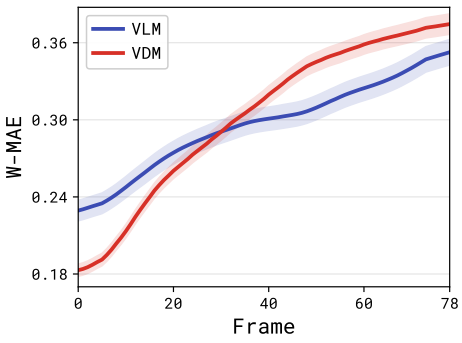

Figure 12: Moving-average W-MAE over continuation frames. VDM motion error grows faster with prediction horizon than VLM error.

Figure 12 shows the moving-average W-MAE across extrapolation frames, averaged over input conditions. The VDM's motion-activity error grows quickly after the prefix and remains consistently above the VLM curve, indicating that direct video prediction increasingly deviates from the ground-truth motion magnitude over time. The VLM also degrades with horizon, but more gradually, suggesting that executable simulation provides a more stable long-horizon dynamics prior.

We observe a similar pattern for collapse failures. Figure 13 plots the cumulative fraction of samples that have collapsed by each extrapolation frame. VDM collapses occur early and continue accumulating throughout the rollout, reaching a substantially higher final collapse rate. In contrast, VLM rollouts exhibit far fewer collapses and mostly remain stable after the initial frames.

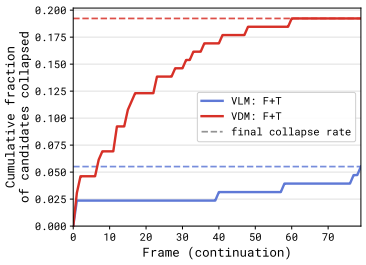

Figure 13: Cumulative collapse rate over extrapolated frames. VDM rollouts collapse earlier and more often than VLM rollouts under the full-configuration setting.

J Full Results for Additional Models and Sampling Budgets

Table 4 and Figure 14 compare four models on the 105 test scenes where all four produce a scorable attempt. Table 6 reports Qwen3.5 and LTX-2.3 separately, since requiring them to succeed as well would halve the shared scene set. Table 5 reports the sampling budgets. For VDMs, selecting samples on their own likelihood gives no improvement.

Deterministic full-configuration baseline. The YAML does not define a unique program; it does not, for example, specify contact handling. A deterministic YAML-to-program baseline performs better than nearest-YAML program retrieval, but it does not define a ceiling on task performance (Table 7).

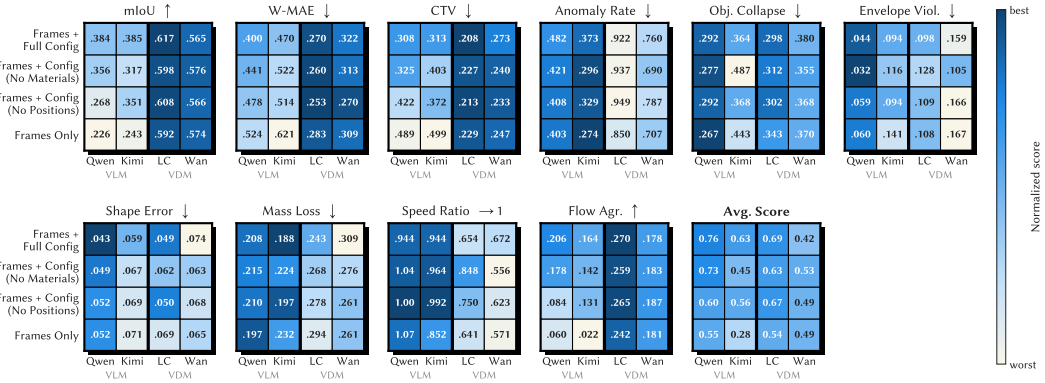

Figure 14: Model comparison at the one-sample budget; values as in Table 4. Color is the normalized score per metric (darker is better).

Table 4: Model comparison at the one-sample budget. Code models at the mean over every attempt, diffusion models at their 5-seed mean: both the expected value of a single draw. Bold marks the best value in each regime, not a significant difference. Envelope violation for GT is mass lost / mass gained, since material compacts in a rendered MPM scene and 0 is not attainable. Anom = temporal-anomaly rate, Coll = object collapse, Env = envelope violation, Shape = mass-trajectory shape error, Mass = per-material mass loss, Speed = speed ratio, Flow = flow agreement.

Regime	Model	Family	mIoU ↑	W-MAE ↓	CTV ↓	Anom ↓	Coll ↓	Env ↓	Shape ↓	Mass ↓	Speed → 1	Flow ↑
Full Config	Qwen2.5-VL-7B (FT)	VLM	0.384	0.400	0.308	0.482	**0.292**	**0.044**	**0.043**	0.208	**0.944**	0.206
	Kimi K3	VLM	0.385	0.470	0.313	**0.373**	0.364	0.094	0.059	**0.188**	**0.944**	0.164
	LongCat (FT)	VDM	**0.617**	**0.270**	**0.208**	0.922	0.298	0.098	0.049	0.243	0.654	**0.270**
	Wan2.1-VACE (FT)	VDM	0.565	0.322	0.273	0.760	0.380	0.159	0.074	0.309	0.672	0.178
No Materials	Qwen2.5-VL-7B (FT)	VLM	0.356	0.441	0.325	0.421	**0.277**	**0.032**	**0.049**	**0.215**	1.040	0.178
	Kimi K3	VLM	0.317	0.522	0.403	**0.296**	0.487	0.116	0.067	0.224	**0.964**	0.142
	LongCat (FT)	VDM	**0.598**	**0.260**	**0.227**	0.937	0.312	0.128	0.062	0.268	0.848	**0.259**
	Wan2.1-VACE (FT)	VDM	0.576	0.313	0.240	0.690	0.355	0.105	0.063	0.276	0.556	0.183
No Positions	Qwen2.5-VL-7B (FT)	VLM	0.268	0.478	0.422	0.408	**0.292**	**0.059**	0.052	0.210	**1.000**	0.084
	Kimi K3	VLM	0.351	0.514	0.372	**0.329**	0.368	0.094	0.069	**0.197**	0.992	0.131
	LongCat (FT)	VDM	**0.608**	**0.253**	**0.213**	0.949	0.302	0.109	**0.050**	0.278	0.750	**0.265**
	Wan2.1-VACE (FT)	VDM	0.566	0.270	0.233	0.787	0.368	0.166	0.068	0.261	0.623	0.187
Frames Only	Qwen2.5-VL-7B (FT)	VLM	0.226	0.524	0.489	0.403	**0.267**	**0.060**	**0.052**	**0.197**	**1.072**	0.060
	Kimi K3	VLM	0.243	0.621	0.499	**0.274**	0.443	0.141	0.071	0.232	0.852	0.022
	LongCat (FT)	VDM	**0.592**	**0.283**	**0.229**	0.850	0.343	0.108	0.069	0.294	0.641	**0.242**
	Wan2.1-VACE (FT)	VDM	0.574	0.309	0.247	0.707	0.370	0.167	0.065	0.261	0.571	0.181
Ground truth			1.000	0.000	0.000	0.000	0.000	0.024 / 0.031	0.000	0.000	1.000	1.000

Table 5: Sampling budget, pooled over the four input regimes. b@5 ranks the best of five on the observed prefix over all five-sized subsets; orc takes the best of five per scene using the held-out ground truth and acts as an upper bound. VDMs have no b@5 column because prefix ranking is equivalent to a random draw for them. Figure 3 reports the VLM at top-1 over every attempt, a stronger selector, so its values are higher.

	Qwen2.5-VL-7B (FT)			Kimi K3			LongCat (FT)		Wan2.1-VACE (FT)	
Metric	mean	b@5	orc	mean	b@5	orc	mean	orc	mean	orc
mIoU ↑	0.309	0.402	0.463	0.324	0.447	0.495	0.604	0.645	0.570	0.620
W-MAE ↓	0.461	0.317	0.191	0.532	0.680	0.246	0.266	0.208	0.303	0.223
CTV ↓	0.386	0.259	0.213	0.397	0.542	0.196	0.220	0.189	0.248	0.199
Anomaly rate ↓	0.429	0.332	0.081	0.318	0.221	0.076	0.914	0.771	0.736	0.488
Object collapse ↓	0.282	0.144	0.077	0.416	0.405	0.127	0.314	0.222	0.368	0.232
Per-material mass loss ↓	0.208	0.211	0.052	0.210	0.178	0.046	0.271	0.196	0.277	0.178

Table 6: Additional model baselines. n is the number of test scenes on which the model produced a scorable output; since each model has its own scenes, rows are not directly comparable. Qwen3.5 is prompted at the one-sample budget; LTX-2.3 is zero-shot, at its 5-seed mean, and was run in two input regimes.

Model	Regime	n	mIoU ↑	W-MAE ↓	CTV ↓	Anom ↓	Coll ↓	Env ↓	Shape ↓	Mass ↓	Speed → 1	Flow ↑
Qwen3.5-397B	Full Config	83	0.200	0.695	0.523	0.273	0.545	0.222	0.112	0.326	0.605	0.077
Qwen3.5-397B	No Materials	80	0.207	0.660	0.525	0.268	0.544	0.202	0.133	0.337	0.688	0.073
Qwen3.5-397B	No Positions	72	0.149	0.682	0.558	0.234	0.548	0.212	0.124	0.322	0.654	0.065
Qwen3.5-397B	Frames Only	85	0.102	0.741	0.580	0.183	0.554	0.169	0.092	0.353	0.465	−0.005
LTX-2.3	Full Config	103	0.509	0.383	0.340	0.920	0.542	0.765	0.140	0.664	1.275	0.124
LTX-2.3	Frames Only	103	0.513	0.362	0.303	0.860	0.493	0.508	0.123	0.574	1.168	0.151

Table 7: Deterministic full-configuration baseline and nearest-YAML retrieval, against the VLMs at best-of-5 in the Full Configuration regime.

Metric	Deterministic	Retrieval (nearest YAML)	Qwen2.5-VL (FT)	Kimi K3
mIoU ↑	0.369	0.292	0.488	0.501
W-MAE ↓	0.371	0.430	0.271	0.658
CTV ↓	0.339	0.526	0.202	0.492
Anomaly rate ↓	0.364	0.414	0.412	0.243
Object collapse ↓	0.169	0.196	0.184	0.395

K PHYSICALLY GROUNDED METRICS

All five metrics are computed from rendered video, the one representation available for both model families. *Envelope violation* measures whether material disappears or appears beyond what the ground-truth simulator itself does; *mass-trajectory shape error*, whether the amount of material rises and falls the way it should; *per-material mass loss*, the same per material rather than in aggregate; *speed ratio*, whether material moves the right amount (1 is correct); and *flow agreement*, whether it moves in the right direction, from optical flow. Table 8 calibrates each metric against simulator state.

Table 8: Calibration of each metric over test scenes where the VLM produces an executable program, between metrics computed from the ground-truth video and the render of the VLM's program, and from the ground-truth particle state and the VLM's simulated particle state.

Metric	Pearson r	Spearman ρ	Mean bias
Mass-trajectory shape error	0.999	0.938	$+0.002$
Speed ratio	0.961	0.948	-0.019
Envelope violation	0.962	0.822	$+0.009$
Flow agreement	0.930	0.956	-0.027
Per-material mass loss	0.904	0.900	$+0.011$

L THE MATERIAL POINT METHOD

MPM represents material as particles that carry mass, velocity, and deformation state, and uses a background grid to compute forces and resolve contact. Each timestep performs (i) a particle-to-grid transfer of mass and momentum, (ii) a grid update that applies forces, gravity, and boundary conditions, and (iii) a grid-to-particle transfer that updates particle velocities and positions. We use MLS-MPM (Hu et al., 2018); see Jiang et al. (2016) and Li et al. (2026) for the derivation. Grid transfers and grid updates are shared by every material. Materials differ only in the particle-level constitutive step: the stress computed from the deformation gradient, the return mapping for plastic materials, and which deformation state each material tracks. Fluids, for example, set the shear modulus to zero, so they carry no resistance to shape change and flow freely; snow and sand apply a plastic return mapping that limits the elastic deformation a particle can hold. We impose no prior on material parameters during generation.

M INPUT REGIMES

Table 9: Configuration fields provided in each input regime. All regimes also receive the video prefix $v_{\leq t}$.

Field	Full Config	No Materials	No Positions	Frames Only
Object type, shape, geometry, color	✓	✓	✓	–
Initial position	✓	✓	–	–
Initial velocity, kinematic and emitter behavior	✓	✓	✓	–
Constitutive model and material parameters	✓	–	✓	–

N DATASET CONSTRUCTION DETAILS

N.1 DATASET QUALITY AND DIVERSITY

Validity. We audit the dataset without an LLM in the loop by parsing each Taichi program (AST and regex analysis) and matching it against its YAML configuration: 97.6% of parent scenes and 94.2% of all scene templates implement the declared material model, and 97.8% of objects in parent scenes are present as declared. Perturbation scenes inherit their parent's materials and are not tested separately. Five external annotators inspected the rendered videos of 860 scenes spanning 599 of the 1,093

parent scenes (54.8%). They found 97.9% of declared objects present and 96.3% of scenes showing each material behaving as declared, with no detectable annotator effect (Cochran's Q, $p = 0.41$); on object presence the study gives 98.3% [97.7, 98.8], consistent with the automatic audit. Every released scene completes its 300-frame rollout without diverging, and the median CFL number across deformable objects is 0.48.

Diversity. We measure the diversity of the dataset along three axes, the structure of its scenes, the text of its programs, and its rendered videos, and we check for near-duplicates.

Structural diversity. Describing each scene by the multiset of its object types, materials, and mechanisms (e.g., emitters, moving colliders), ignoring coordinates, sizes, colors, and all numeric parameters, gives 3,709 distinct compositions; 80% of parent scenes match no other scene, and of a seed's median of 103 descendants, 92 have a composition found nowhere else in its lineage.

Program diversity. Under TF-IDF vectors of program text and histograms of AST node types, generated programs are $0.91\times$ as varied among themselves as the 35 seeds, and sit $1.26\times$ further from their nearest seed than a seed does from its nearest other seed.

Video diversity. Relative to the distance between two seeds, a generated scene's distance to its nearest seed is 1.09–1.12 for motion (RAFT optical flow (Teed & Deng, 2020) pooled over material pixels), 1.05–1.10 under V-JEPA 2 (Assran et al., 2025), and 0.85–0.88 for appearance (DINOv2 (Oquab et al., 2023)). Two positional perturbations of one scene differ by 0.208 of the median distance between seeds, and 98.9% of generated scenes are further than that from every seed.

Near-duplicates. With the threshold set from a scene's distance to its own deliberate edit, 1.4% of scenes have a near-duplicate under DINOv2, and 0.2% if the pair must also match in motion under V-JEPA 2.

Across structure, program text, and motion, the generated scenes are as varied as the hand-written seeds are among themselves.

Generation prompt. Apart from the in-context seeds, the prompt contains only a task statement and hard constraints: a grid resolution of at most 128, exactly 300 frames, a $[0, 1]^2$ domain with specified boundary conditions, and a mandatory stability guard, among others. No code scaffold, interaction template, or parameter prior is used during generation, and the YAML is written alongside each program rather than given as input. The generating model never sees the rendered output of a program.

O Compute Resources.

All VLM and VDM experiments were trained on a server with 4 NVIDIA RTX A6000 GPUs. Training each model required approximately 12 hours. Additional compute was used for dataset generation, rendering, and evaluation.