CHAIN-OF-TIME: IN-CONTEXT PHYSICAL SIMULATION WITH IMAGE GENERATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel cognitively-inspired prompting method to stabilize and interpret the physical simulation produced by autoregressive image generation models, called *Chain-of-Time*. *Chain-of-Time* involves generating a series of intermediate images during a simulation, and it is motivated by mental simulation in humans, as well as in-context reasoning in LLMs. *Chain-of-Time* is used at inference time, and requires no additional fine-tuning. We apply this method to synthetic and real-world domains, including 2-D graphics simulations and natural 3-D videos. These domains cover various physical dynamics, ranging from rigid-body rolling to harmonic oscillation and pulley systems. We found that *Chain-of-Time* simulation stabilizes the physics simulated by the state-of-the-art image generation models. Beyond examining performance, we also analyzed the specific states of the world simulated by an image model at each time step, which sheds light on the dynamics underlying these simulations. This analysis reveals insights that are hidden from traditional evaluations of physical reasoning, highlighting cases where image generation models can simulate physical properties such as velocity, gravity, oscillations, and pulley systems. Like humans, image generation models are able to accurately predict future world states over short time periods. By providing a mechanism analogous to working memory, Chain-of-Time enables a model to stitch together a series of short-term predictions into an accurate long-term simulation.

1 INTRODUCTION

Recent developments enable auto-regressive image generation models (IGMs) to generate increasingly complex, realistic, and coherent images Chen et al. (2025); Cao et al. (2025b); Liu et al. (2023); Lu et al. (2024). Despite these improvements, however, these images often have distinct flaws, and may fail to capture real-world structures that are obvious to humans. Understanding the inner workings of Vision-Language Models (VLMs) such as IGMs has become a major topic in contemporary AI research (Dang et al., 2024; Chang et al., 2024; Goh et al., 2021; Bhalla et al., 2024; Zhang et al., 2024). In particular, there is a pressing question of how VLMs and image generation models represent physical properties which are required to predict how world states unfold over time. In this work, we present a zero-shot prompting method for stabilizing physical reasoning in image generation models, which is inspired by psychological theories of mental simulation in humans and allows us to analyze the step-by-step process that the models use to simulate physics over time.

To our knowledge, our work is unique in studying physical reasoning abilities of autoregressive Image Generation Models (IGMs), which are similar to autoregressive LLMs, except that they can take both images and text as input and generate images or text as output. IGMs are rapidly becoming the gold standard for text-to-image generation, with top AI companies like OpenAI and Google replacing their flagship text-to-image models (e.g. DALL-E) with IGMs (OpenAI, 2025a; DeepMind, 2025). Given that modern IGMs build on top of pre-trained vision-language models (OpenAI, 2025b), studying IGMs may also provide valuable insights into the inner workings of VLMs. Although our work is the first to study IGMs, prior work has evaluated the physical reasoning abilities of VLMs. Comprehensive benchmarks such as PhysBench (Chow et al., 2025) and WM-ABench (Gao et al., 2025) test VLMs on a wide array of physical simulation capabilities. Beyond VLMs, Meng et al. (2024) evaluates the extent to which text-to-image models – which generate images but do not take images as input – can generate images matching relational and physical constraints, using a separate VLM as an evaluator. Although these approaches provide valuable insights into how well VLMs can

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

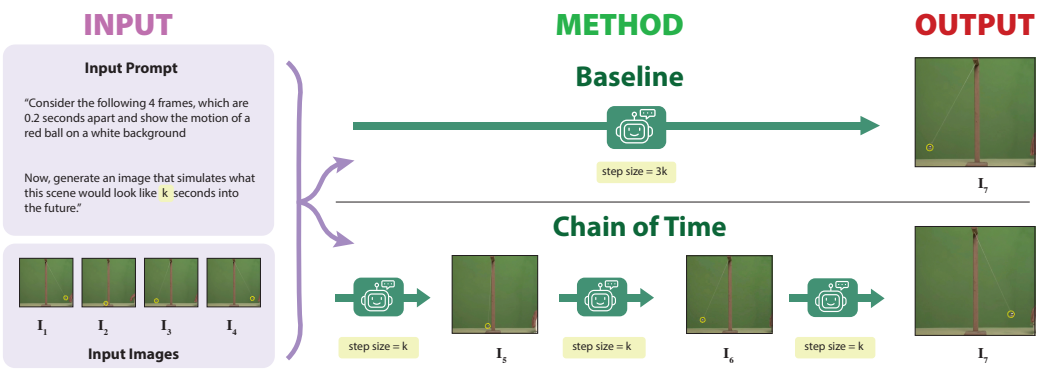

Figure 1: We study physical reasoning in multi-modal image generation models by providing the model with a sequence of input images of a scene at successive time steps, and having the model predict what the scene will look like some time in the future. Accurately predicting future world states requires reasoning about latent physical properties such as gravity. We propose a novel method, Chain-of-Time (Bottom), which generates a sequence of images corresponding to a step-by-step simulation of the scene as it unfolds over time. As a baseline, Direct Prediction (Top) directly generates the final state without intermediate steps. We use object segmentation algorithms to track object dynamics in both input and generated images (yellow dotted lines).

accomplish physical reasoning, they do not tell us *how* VLMs accomplish physical simulation. Our work aims to address this issue, providing a detailed analysis of the incremental processes underlying physical simulation.

We draw inspiration from approaches in cognitive science which empirically study analogous incremental processes when humans mentally simulate physical systems. Human mental simulation can be considered as a noisy Markov process where people simulate successive time steps in a latent world model, and there is also mounting evidence that working memory is essential for mental simulation in biological intelligence. We present a new method that endows IGMs with a module analogous to visual working memory, and enables them to perform physical simulation step-by-step, similar to what humans seem to do.

In this work, we adopt a theoretical framework of mental simulation from cognitive science to understand and stabilize physical reasoning and simulation abilities in Image Generation Models (Section 2). This framework helps us understand how IGMs reason about physical processes that unfold over time, by mapping input images to a latent state which is simulated with a Markov process to predict future time steps. In order to both stabilize the physical reasoning abilities of IGMs, and to expose an interpretable trace of intermediate reasoning steps, we propose a novel method for in-context physical simulation, which we call Chain-of-Time (Section 3, Fig. 1).

2 Mental Simulation in Humans

People can reason efficiently about the physical dynamics of everyday objects, in tasks that state-of-the-art generative AI models struggle with. In this section, we will consider psychological theories of how people reason about physics in order to develop methods for evaluating and improving physical reasoning in IGMs. For example, if you saw a pitcher full of juice begin to fall off of a table, you might quickly and intuitively predict what sequence of events would happen next. There are many competing theories that try to explain this 'intuitive physics'. One current proposal is that people rely on an 'internal physics engine' to carry out a mental simulation of a given scene (Battaglia et al., 2013; Ullman et al., 2017). While this proposal has been criticized (see for example Ludwin-Peery et al. (2021)), it finds support in cognitive science, computational modeling, cognitive development, and neuroscience (Fischer et al., 2016; Gerstenberg & Stephan, 2021; Allen et al., 2021; Fischer, 2021; Bass et al., 2021; Balaban & Ullman, 2025). More recent work suggests it is likely that humans combine various computations to carry out physical reasoning, mental simulation being just one component Hartshorne & Jing (2025); Sosa et al. (2025); Smith et al..

Research in cognitive science suggests that step-by-step mental simulation is an important component in human physical reasoning, and so we adopt this formalism for studying and improving upon the physical reasoning of current IGMs. For our purposes here, we consider a basic version of the mental physics engine framework: Suppose that an agent observes an image I that describes a scene at time t in a pixel-based format, and wants to predict the state of the scene at a later time. A mental physics engine is a probabilistic transition function that can achieve this by composing three sub-components: a de-renderer ϕ, a simulator τ, and an optional renderer ϕ^{-1}. The engine takes in the current image I_t, and de-renders it into the state of the world at that time, X_t. The engine then applies dynamic update rules to that state, corresponding to a transition τ that produces a distribution over future states of the world X_{t+1}. The engine may then render the state of the world back into a predicted image I_{t+1}.

A few notes on this overall formulation: First, while de-rendering has been studied in the context of intuitive physics in the past (e.g. Wu et al., 2017a;b; Smith et al., 2019), many other techniques exist for going from observations to physical states, and for our purposes here the specific technique is of less importance. Second, while the images I are pixel-based, the underlying physical state X is not, and corresponds to the 'game state' that describes in a lower-dimensional way the position, identity, and physical parameters of objects (Smith et al., 2019). Third, in computer graphics it is not strictly necessary to render the state of the world back into an image in order to answer various questions about the state, something that may hold for human mental physics as well (Balaban & Ullman, 2025).

More formally, the mental simulation formalism we consider here is:

$$p(X_t \mid I_t) = \phi(I_t) + N(0, \sigma_\phi) \qquad \text{De-rendering}$$
$$p(X_{t+1} \mid X_t) = \tau(X_t) + N(0, \sigma_\tau) \qquad \text{Simulation}$$
$$p(I_{t+1} \mid X_{t+1}) = \phi^{-1}(X_{t+1}) + N(0, \sigma_{\phi^{-1}}) \qquad \text{Rendering}$$

The noise parameters σ_ϕ, σ_τ, and $\sigma_{\phi^{-1}}$ account for perceptual noise in the de-rendering of I, the cognitive complexity of mental simulation of the underlying state X, and imperfect imagery in the rendering of the state back to an image.

In this formulation, the state and scene at timestep $t + 1$ depend only on the previous state and scene at timestep t. In other words, this is a linearly unfolding Markov Chain that moves from an initial observation I_0 step-by-step to a final state at time T, X_T, and optionally the predicted image at that time, I_T. While such step-by-step computations may underlie human mental simulation, it is unclear whether current IGMs tasked with predicting the future state of a scene I at time T go through a step-by-step simulation within their activations. However, even if current IGMs do not do so implicitly, this framework suggests that if models can engage in short prediction steps, then with in-context reasoning we may be able to bring IGMs more in line with human-like physical simulation.

3 Chain-of-Time Simulation

Chain-of-Time simulation is inspired by two bodies of prior literature: the cognitive science of mental simulation (described in Section 2), and in-context reasoning in LLMs. In-context reasoning methods with LLMs coerce a model to spell out intermediate reasoning steps in its output stream before giving a final answer. This may be through prompting, as in Chain-of-Thought reasoning (Kojima et al., 2022), or through specialized training regimes (Guo et al., 2025; Jaech et al., 2024). These methods can significantly improve LLM performance on a variety of tasks, extending the ability of LLMs to reason over complex problems with many individual steps. Reasoning chains are also distinct from traditional LLM tasks, since intermediate steps can be directly inspected by humans to interpret what the model is doing. Although in some cases a model's intermediate reasoning tokens may not align with its final answer (Turpin et al., 2023), Chain-of-Thought reasoning has proved a valuable tool for auditing the behavior of language models. Various theories have been developed to try to explain precisely why and how these methods work or occasionally fail (Wang et al., 2022; Merrill & Sabharwal, 2023; Prystawski et al., 2023).

Based on the prior literature in human mental simulation and in-context reasoning, we propose a novel method for improving physical reasoning with in-context simulations. We call this method *Chain-of-Time simulation* (Fig. 1). We treat VLMs as a derenderer ϕ and a simulator τ, and the IGMs as a renderer ϕ^{-1}. The basic physical simulation task we consider is as follows: given a sequence of

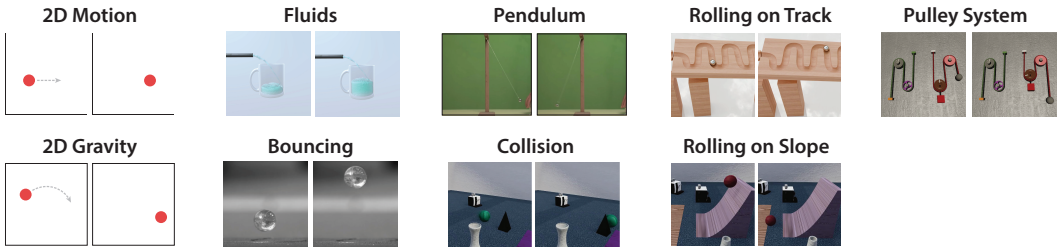

Figure 2: We use nine experimental domains to test models' ability to reason about different physical properties. Three of these domains (`2D Motion`, `2D Gravity`, `Fluids`) were synthetically generated and inspired by previous work on the cognitive science of intuitive physics, four of the domains (`Rolling on Slope`, `Collision`, `Rolling on Track`, `Pulley System`) were synthetically generated by prior works that evaluate physical reasoning in VLMs, and two of the domains (`Bouncing`, `Pendulum`) use real world videos sampled from various sources and demonstrate the generality of our approach.

input images $I_{0:t}$ up to a given time t, generate a new image $\widehat{I}_{t+k}$ that accurately depicts what the scene will look like k time steps into the future.

Chain-of-Time simulation involves two prompts (provided in App. C): first, a Simulation Instruction prompt that, along with a sequence of input images, instructs the model to simulate an image k seconds into the future. After the IGM generates a single image, we continue with our Simulation Follow-up prompt, which instructs the model to generate another image simulated an additional k seconds into the future until $t + k = T$, where $T = t + t_{length}$. In our experiments, we used $T = t + 2$ secs and the parameter k is determined by the number of intermediate steps $s \in \{2 \text{ steps}, 4 \text{ steps}\}$, where we instruct the model to generate another image every $k = t_{length}/s$ seconds after time t, until it reaches time T. To capture the Direct Prediction baseline, we use a method equivalent to Chain-of-Time but with a single timestep, prompting the IGM to predict $\widehat{I}_{t+k}$ given $I_{0:t}$, without intermediate prompts. In our case, to match the Chain-of-Time parameters, we consider $s = 1$ step as the Direct Prediction baseline.

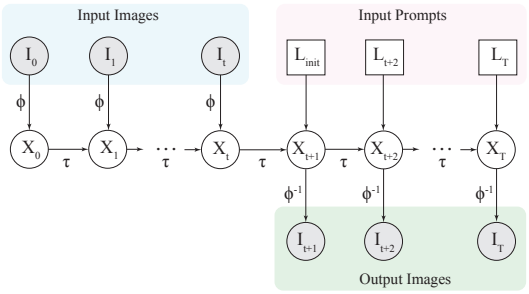

Figure 3: Chain-of-Time is a composition of three components: De-rendering ϕ, Simulation τ, and Rendering ϕ^{-1}. De-rendering operates by converting input images $I_0 \dots I_t$ into world states X_t, which represent a physical simulation over time. Chain-of-Time begins with an initial prompt L_{init} and iteratively generates a sequence of in-context output images $I_{t+1} \dots I_T$ with follow-up prompts $L_{t+1} \dots L_T$

Previous Work Prior work has developed in-context reasoning methods for VLMs that use images to represent individual reasoning steps. Our method differs from these approaches in a few important ways. Hu et al. (2024) proposed a method to solve simple reasoning problems with a VLM, such as geometry and spatial reasoning, and individual steps involve interleaved images and text outputs. Similar to our method, this approach requires no additional training. However, in contrast with *Chain-of-Time*, this method is a separate tool that the VLM invokes, which uses LLM code generation along with a Python interpreter to render intermediate sketches. Xu et al. (2025) proposed a method for planning in which a VLM is explicitly trained with reinforcement learning to generate sequences of images to solve tasks such as maze navigation. *Chain-of-Time* simulation, on the other hand, strives to stabilize physical simulation with IGMs where "steps" correspond to units of time, rather than planning, where "steps" correspond to actions an agent takes in an environment. Our method also differs in that it can be applied to out-of-the-box IGMs with no additional training. Further, our work presents an in-depth analysis of the sequence of images generated by *Chain-of-Time*, where we consider the accuracy of simulation at each individual step. Such an analysis was not used in Hu

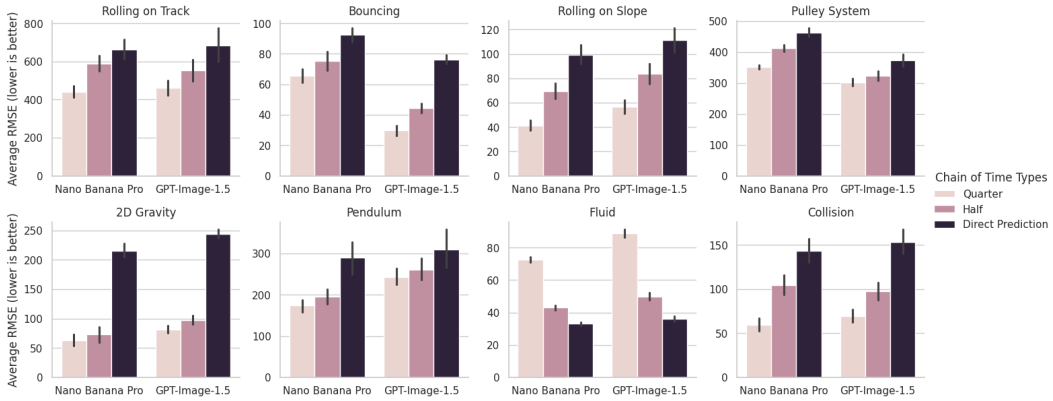

Figure 4: Prediction errors for the eight domains included in this analysis, averaged across all data for each domain. Prediction error is measured by taking the average RMSE between the ground-truth positions (location of focal object, or water level) and the final positions predicted by the IGM. Error bars are 95% Bootstrapped CI. We generally find a monotonic relationship between *Chain-of-Time* precision and performance when compared against the direct prediction baseline, indicating a more stable physical simulation under *Chain-of-Time*.

et al. (2024) or Xu et al. (2025), although Xu et al. (2025) develops learning metrics that evaluate step-wise accuracy, and both works show examples of generated image sequences.

Our work is also similar to approaches for evaluating physical reasoning ability in video generation models (Bansal et al., 2024; Wiedemer et al., 2025; Motamed et al., 2025; Yuan et al., 2025; Wei et al., 2025; Le et al., 2025). The approach of Zhang et al. (2025) is especially similar to our method, where they also use post-hoc image segmentation to track the state of objects in generated images, for the purpose of evaluating physical reasoning. Like prior work studying video models, we similarly evaluate pre-trained multimodal models with sequences of frames from videos, although importantly, we present a novel method which enables Image Generation Models to perform this task, which are not trained for video inputs or to generate video outputs. Additionally, Chain-of-Time uses a very small number of frames (4-5) for model inputs and outputs, compared to tens or hundreds of frames used with video models.

4 Experiments

We hypothesize that by using Chain-of-Time, IGMs will be able to achieve better accuracy than when using direct prediction. We use the frames created by IGMs using *Chain-of-Time* simulation to reveal the details about the simulation, including the initial states estimated by IGMs, physical interactions, and physical motion simulated by IGMs. To examine the overall validity and applicability of our method, we tested *Chain-of-Time* on nine experimental domains: **`2D Motion`**, **`2D Gravity`**, **`Fluids`**, **`Bouncing`**, **`Pendulum`**, **`Rolling on Slope`**, **`Collision`**, **`Rolling on Track`**, and **`Pulley System`** (Fig. 2). We analyzed our results in two ways: first, we measured the accuracy of predicted images relative to ground truth; and second, we visualized the temporal dynamics in simulated images (such as x and y location) to assess physical motions produced by the IGM's simulations to see if it approximately follows the motion indicated by the governing physics equations, if the motion is governed by physics equations with distinctive shapes like sin function or quadratic functions.

Stimuli Design We used 9 different physical domains as our stimuli. The stimuli used in `2D Motion`, `2D Gravity`were created in a simulation environment, and resemble stimuli in previous studies of intuitive physics Smith & Vul (2013), Bass et al. (2021), Gerstenberg & Stephan (2021). The stimuli used in `Fluids`were borrowed from Wang & Ullman (2025). In the `Bouncing`, `Pendulum`domains, we used real-world videos which have different physical parameters, e.g. different balls with different levels of "bounciness". In the `Rolling on Slope`, `Collision`, `Rolling on Track`, and `Pulley System`, we borrowed them from physical reasoning benchmarks such as Bear et al. (2021), Zheng et al. (2024) and video websites like BiliBili and YouTube. We simulated different speeds by re-

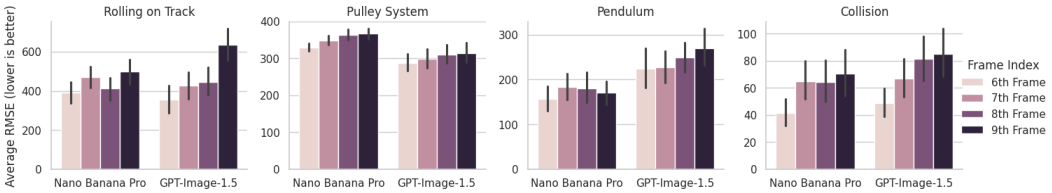

Figure 5: Prediction errors of each predicted frame for the four domains which have the most complicated object motion, averaged across all data for each domain under *Chain-of-Time* Quarter. Prediction error is measured by taking the average RMSE between the ground-truth positions (location of focal object) and the positions predicted by the IGM. Error bars are 95% Bootstrapped CI. We found that *Chain-of-Time* Quarter up to 2 seconds of simulation shows small but consistent error accumulation across these domains with complex motion.

sampling input videos at different frame rates. For further details of how we designed stimuli, please refer to App. B.

Image Generation Models We evaluated two state-of-the-art image generation models. The first image generation model we used is OpenAI's GPT Image 1.5 (gpt-image-1.5 [1]), as of December, 2025. The second image generation model is Google DeepMind's Nano Banana Pro (gemini-3-pro-image-preview) [2]. Both models support multi-turn chat interactions and are state-of-the-art models in image generation. For derendering and simulation, the two models respectively use the VLMs of GPT 5.2 and Gemini 3 Pro. Further, we also analyzed the newly released open-source IGM `HunyuanImage-3.0-Instruct` (Cao et al., 2025a). The experimental setup and results for this model are described in App. F. In addition to these models, we also empirically tested other closed and open source image generation models including Dall-E 3 and Qwen Image, but we found that Dall-E 3 was unable to simulate images of future world states with any reasonable accuracy, and Qwen Image failed at multi-turn image generation due to the lack of a VLM as a simulator. As an additional baseline, in App. G we also compare Chain-of-Time to physical simulation using a video generation model (`LTX-video`).

Evaluating Simulation Accuracy Next we describe our methods for evaluating the quality of intermediate and final simulated images (also see App. D). To track the specific positions of key objects, we used the Segment-Everything-Model 3 (SAM 3) [3] to identify object positions. In each case, we specified text descriptions of the object that the SAM 3 model should segment (e.g. *"tennis ball"*). For examples of successful segmentation generated by SAM 3 over the tasks we selected, see App. D.3. The analysis in `Fluids`was conducted using a traditional CV algorithm due to consistent detection errors from SAM 3 on `Fluids`(further details in App. D.4). We used the default setting for SAM3 and all the evaluation pipeline code is available at: `https://anonymous.4open.science/r/Chain-of-time-8E8F/`. We conducted a evaluation based on sub-sampling to ensure the accuracy and stability of this evaluation pipeline, as detailed in App. D.2.

Sampling Details At the start of each trial, the model was given 5 frames of a stimulus, showing the scene at 0, 0.5, 1.0, 1.5, and 2 seconds. Given these 5 frames, the model was asked to generate simulated frames at a time in the future, following the Initial Simulation Prompt. If *Chain-of-Time* Quarter-Step or *Chain-of-Time* Half-Step were used (see below), additional frames were generated following the Simulation Follow-Up Prompt. Details of the prompts we used can be found in App. C. In App. C.6, we also investigated the effect of varying the prompt complexity on the our method.

Chain-of-Time generates frames at different temporal resolutions, depending on a frame-rate parameter s. We considered two versions of *Chain-of-Time* with $s = 2$ steps and $s = 4$ steps. Since the final frame was 2 seconds into the future, *Chain-of-Time* with $s = 2$ steps generated 2 frames, corresponding to 1 and 2 seconds into the future, and*Chain-of-Time* with $s = 4$ steps generated 4 frames,

[1] `openai.com/index/image-generation-api/`

[2] `https://ai.google.dev/gemini-api/docs/image-generation`

[3] `https://ai.meta.com/sam3/`

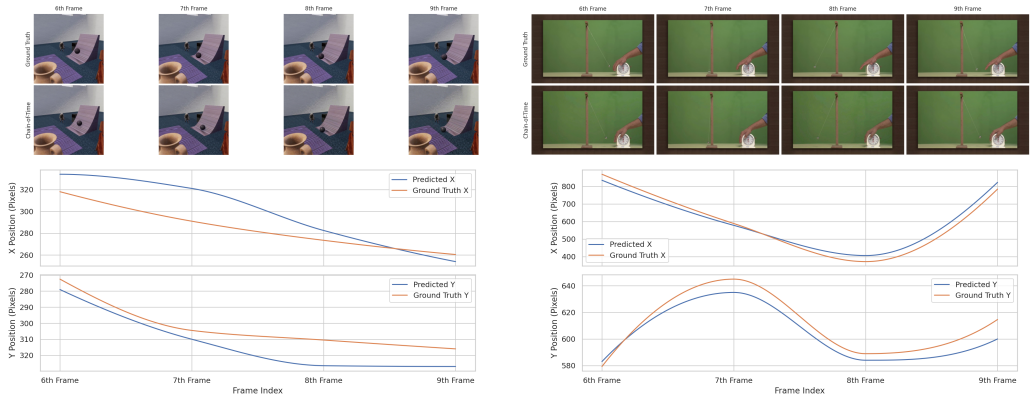

Figure 6: **Simulation analysis for `Rolling on Slope` and `Pendulum` domains** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom Left) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories, first accelerating sharply and then steadily decelerating. (Bottom Right) Extracted X and Y coordinates of the pendulum in each frame. We observe oscillatory motion in the pendulum for both the ground truth and simulated image sequences.

corresponding to 0.5, 1.0, 1.5, and 2.0 seconds into the future. For simplicity, we call *Chain-of-Time* with $s = 4$ steps *Chain-of-Time* Quarter, and *Chain-of-Time* with $s = 2$ steps *Chain-of-Time* Half. In addition, we had a baseline termed "Direct Prediction". In this method, we asked the model to directly generate the requested final frame, which is 2 seconds into the future. For all domains, we ran each stimulus 10 times (N=10) across all *Chain-of-Time* simulations and Direct Prediction.

5 Results

Accuracy Analysis We evaluated the IGM's accuracy in predicting the ground-truth value of the target variable (object location or water height) using three methods: *Chain-of-Time* Quarter, *Chain-of-Time* Half, and Direct Prediction (Baseline), for both Nano Banana Pro and GPT Image 1.5. As shown in Fig. 4, a finer *Chain-of-Time* step generally improves accuracy in seven of the eight domains for both models. We excluded the 2D Motion domain from the analysis in Fig. 4 due to its simplicity, but found results similar to those in the other domains. Overall, these results suggest that *Chain-of-Time* improves prediction accuracy across 2D and 3D tasks, spanning from simple to complex settings. However, performance differences between Nano Banana Pro and GPT Image 1.5 across domains indicate that gains over the Direct Prediction baseline also depend on the underlying capabilities of the base model. In App. H.5, we compared Chain-of-Time to a stronger baseline which extends Direct Prediction to use self-consistency sample selection with CoT Reasoning, and Chain-of-Time is superior in accuracy to this baseline as well. Chain-of-Time also consistently improves accuracy with the open-source IGM `HunyuanImage-3.0-Instruct` (except for the Fluids domain), and our results for this model broadly align with our results for the two closed-source IGMs (see App. F for further details). Finally, we find that Chain-of-Time with these IGMs performs similarly to a state-of-the-art video generation model (App. G).

Physical Motion Analysis In our prior analysis, we found that *Chain-of-Time* enables higher-fidelity physical simulation in most domains, measured by accuracy of the final generated image. This raises the question of precisely why Chain-of-Time improves the accuracy of physical simulation with IGMs. Similar to Chain-of-Thought reasoning with LLMs, Chain-of-Time simulation offers an interpretable trace of intermediate steps preceding the final output. Therefore, we evaluate the IGM's ability to simulate physical motion by examining its intermediate simulated images, focusing on whether *Chain-of-Time* stabilizes physical properties and produces object motions that are accurate and precise within the tested domains.

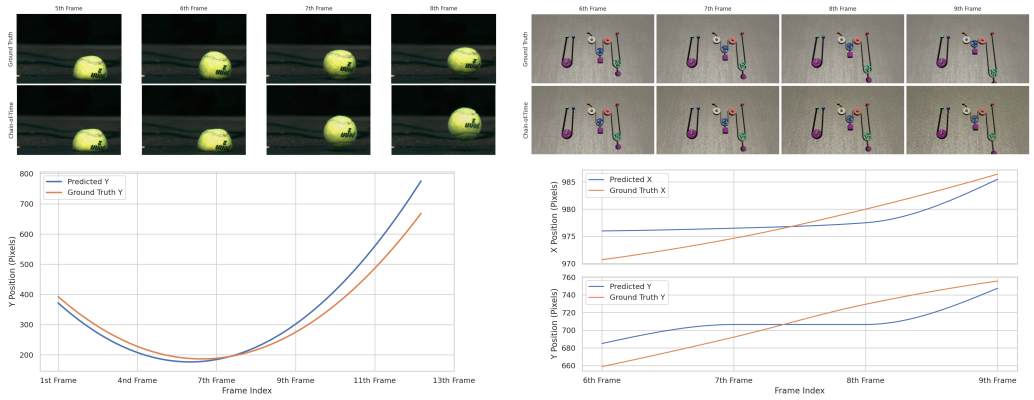

Figure 7: (Left) **Simulation analysis for `Bouncing` and `Pulley System` domains** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. For `Bouncing`, the images shown correspond to the "during" period of the collision, beginning at frame 5 of 12. (Bottom Left) Extracted Y coordinates of the tennis ball in each frame, for three independent simulation results of four frames each: one simulation before the bouncing collision, one during, and one after collision. We observe parabolic motion in the ball for both the ground truth and simulated image sequences. (Bottom Right) Extracted X and Y coordinates of the purple ball in each frame. Despite the complexity of this domain, image generation models are able to accurately simulate the ball's future motion.

We use `Rolling on Slope`, `Bouncing`, `Pendulum`, and `Pulley Systemas` illustrative examples, since the governing equations in these domains have common functional forms with geometrically distinctive trajectories. Because *Chain-of-Time* Quarter produces the greatest number of intermediate frames, we analyze the generated sequences from *Chain-of-Time* Quarter in these domains. We present one representative trial per domain here. Additional examples for these and other domains are provided in App. E.

As shown in Fig. 6, the IGM-generated sequence (second row) closely matches the ground-truth motion in the reference video (first row). Because the ramp is curved, the ball's speed changes nonlinearly as it rolls downward, producing nonlinear trajectories in both the x and y coordinates, as illustrated in the lower panel of Fig. 6. In the `Bouncing`domain, Fig. 7 shows that the IGM-generated motion likewise closely resembles the ground-truth sequence. We focus on the "during" interval, where the model must capture deformation during impact and the subsequent rebound, which requires modeling object deformation during impact and estimating physical parameters such as the coefficient of restitution. The IGM reproduces both the deformation dynamics and rebound behavior accurately, suggesting effective motion simulation and parameter inference from visual context. A characteristic signature of bouncing motion is that the Y position follows a quadratic function over time. In the lower panel of Fig. 7, we show that the IGM accurately recovers this functional form (upward U-shaped curve), further supporting its ability to simulate `Bouncing`.

For the `Pendulum`domain, Fig. 6 shows that the IGM generates realistic pendulum motion consistent with the physical dynamics of this domain. The coordinate traces in the lower panel of Fig. 6 exhibit the expected $\sin$ and $\cos$ dependence in x and y, respectively matching the standard Cartesian parameterization of a simple pendulum. In the `Pulley System`domain, Fig. 7 indicates that the IGM captures the dynamics of a more complex pulley system involving both fixed and movable pulleys. The generated sequence closely matches the ground-truth video (first row), accurately reproducing the coupled motion of the two masses connected with two movable pulleys and through two fixed pulleys. These results demonstrate the IGM's capability to correctly understand the characteristics of fixed and movable pulleys, and in general the pulley system dynamics.

Do Simulations Degrade Over Time? With the prior analysis on accuracy and physical motion, we found that *Chain-of-Time* substantially stabilizes the physical simulation and improves simulation accuracy in IGMs. Theoretically, we could run *Chain-of-Time* for indefinitely long time periods; however, we also might expect that with very long time horizons small errors could eventually

propagate into major discrepancies. Similarly, previous work in cognitive science studying mental simulation in humans suggested that accuracy of simulation actually degrades over time due to uncertainty Smith & Vul (2013). Therefore, we now aim to test whether *Chain-of-Time* simulation's accuracy degrades, i.e. diverges from ground truth, the longer a simulation runs for.

We use *Chain-of-Time* Quarter since it has the strongest overall performance (Fig. 4) and provides the most intermediate frames for analysis. We focus on four domains with complex dynamics and interactions: `Rolling on Slope`, `Pulley System`, `Pendulum`, and `Collision`. As shown in Fig. 5, for both Nano Banana Pro and GPT-Image-1.5, the average RMSE of the 6th–9th generated frames (corresponding to approximately 0.5, 1.0, 1.5, and 2.0 seconds into the future) steadily increases with longer simulation, but at a slow rate. This demonstrates that *Chain-of-Time* Quarter can maintain stable and robust simulations, although their accuracy may decrease with much longer time horizons.

In App. H we describe a number of additional experiments. First, in App. H.1 we vary the number of input frames and find diminishing returns where Chain-of-Time accuracy does not increase significantly with more than 5 input images. In App. H.2 we vary the number of intermediate simulation frames and similarly find diminishing effects with more frames, despite increasing costs. In App. H.3 we find that IGMs become less accurate when predicting further into the future, similar to what has been found with human mental simulation, although Chain-of-Time still improves accuracy. Finally, in App. H.4 we show that Chain-of-Time simulations can be stabilized by using a VLM "critic" that verifies intermediate simulation steps and has the IGM regenerate low-accuracy images.

6 Conclusion

Inspired by mental simulation in humans and in-context reasoning in large language models, we presented a prompting method for step-by-step physical simulation in Image Generation Models. With this method, *Chain-of-Time* simulation, a prediction horizon is divided into smaller time intervals, with mid-point frames being fed as input to the next step in an unfolding systematic process. We assessed *Chain-of-Time* for differing degrees of temporal precision, across a variety of physical reasoning tasks, and using quantitative metrics of step-wise accuracy compared to ground truth, and the qualitative metric of how well the IGM recovers physical parameters such as gravity and bounciness.

We also introduced a method for flexibly evaluating physical simulation accuracy in generated images by using a cutting-edge object segmentation algorithm (SAM 3). Evaluating physical simulation ability in IGMs presents an exciting new challenge for both AI evaluations and cognitive science: unlike for humans, where we must empirically probe mental simulations with secondary metrics like eye gaze, with IGMs we can directly analyze the images that the model generates.

Our results suggest that current Image Generation Models have some degree of physical simulation ability, but that ability may critically be limited in how far ahead into the future the IGM can simulate. This mirrors findings on the psychology of mental simulation in people, where people are much more accurate at predicting world states smaller amounts of time into the future, and mental simulation theoretically operates by chaining together sequences of short-term predictions. Analogously, *Chain-of-Time* stabilizes the physical simulation in IGMs by chaining together sequences of short-term predictions. This is evidenced by *Chain-of-Time* improving simulation accuracy for various domains involving physical reasoning. Another line of supporting evidence is our analysis of the intermediate images generated with *Chain-of-Time*, which show that improved accuracy in the final state is preceded by more accurate and stable predictions of intermediate states. This also demonstrates a secondary advantage of *Chain-of-Time*, which is that we can directly assess the simulated images produced by the IGM in order to evaluate the model's perception of physical parameters, motions, and interactions.

REFERENCES

Kelsey R Allen, Kevin A Smith, Laura-Ashleigh Bird, Joshua B Tenenbaum, Tamar R Makin, and Dorothy Cowie. Lifelong learning of cognitive strategies for physical problem-solving: the effect of embodied experience. *bioRxiv*, pp. 2021–07, 2021.

Halely Balaban and Tomer D Ullman. Physics versus graphics as an organizing dichotomy in cognition. *Trends in Cognitive Sciences*, 2025.

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.

Ilona Bass, Kevin A Smith, Elizabeth Bonawitz, and Tomer D Ullman. Partial mental simulation explains fallacies in physical reasoning. *Cognitive Neuropsychology*, 38(7-8):413–424, 2021.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

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.

Usha Bhalla, Alex Oesterling, Suraj Srinivas, Flavio Calmon, and Himabindu Lakkaraju. Interpreting clip with sparse linear concept embeddings (splice). *Advances in Neural Information Processing Systems*, 37:84298–84328, 2024.

Siyu Cao, Hangting Chen, Peng Chen, Yiji Cheng, Yutao Cui, Xinchi Deng, Ying Dong, Kipper Gong, Tianpeng Gu, Xiusen Gu, et al. Hunyuanimage 3.0 technical report. *arXiv preprint arXiv:2509.23951*, 2025a.

Yuefan Cao, Xuyang Guo, Jiayan Huo, Yingyu Liang, Zhenmei Shi, Zhao Song, Jiahao Zhang, and Zhen Zhuang. Text-to-image diffusion models cannot count, and prompt refinement cannot help. *arXiv preprint arXiv:2503.06884*, 2025b.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2025. URL https://arxiv.org/abs/2511.16719.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

Sixiang Chen, Jinbin Bai, Zhuoran Zhao, Tian Ye, Qingyu Shi, Donghao Zhou, Wenhao Chai, Xin Lin, Jianzong Wu, Chao Tang, Shilin Xu, Tao Zhang, Haobo Yuan, Yikang Zhou, Wei Chow, Linfeng Li, Xiangtai Li, Lei Zhu, and Lu Qi. An empirical study of gpt-4o image generation capabilities, apr 2025. URL https://arxiv.org/abs/2504.05979. Version 2, revised 10 Apr 2025.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Guizilini, and Yue Wang. Physbench: Benchmarking and enhancing vision-language models for physical world understanding. *arXiv preprint arXiv:2501.16411*, 2025.

Yunkai Dang, Kaichen Huang, Jiahao Huo, Yibo Yan, Sirui Huang, Dongrui Liu, Mengxi Gao, Jie Zhang, Chen Qian, Kun Wang, et al. Explainable and interpretable multimodal large language models: A comprehensive survey. *arXiv preprint arXiv:2412.02104*, 2024.

Google DeepMind. Addendum to gpt-4o system card: Native image generation, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

Jason Fischer. The building blocks of intuitive physics in the mind and brain, 2021.

Jason Fischer, John G Mikhael, Joshua B Tenenbaum, and Nancy Kanwisher. Functional neuroanatomy of intuitive physical inference. *Proceedings of the national academy of sciences*, 113 (34):E5072–E5081, 2016.

Qiyue Gao, Xinyu Pi, Kevin Liu, Junrong Chen, Ruolan Yang, Xinqi Huang, Xinyu Fang, Lu Sun, Gautham Kishore, Bo Ai, et al. Do vision-language models have internal world models? towards an atomic evaluation. In *ICLR 2025 Workshop on World Models: Understanding, Modelling and Scaling*, 2025.

Tobias Gerstenberg and Simon Stephan. A counterfactual simulation model of causation by omission. *Cognition*, 216:104842, 2021.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Joshua K Hartshorne and Mengguo Jing. Insights into cognitive mechanics from education, developmental psychology and cognitive science. *Nature Reviews Psychology*, pp. 1–15, 2025.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *arXiv preprint arXiv:2406.09403*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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

Minh-Quan Le, Yuanzhi Zhu, Vicky Kalogeiton, and Dimitris Samaras. What about gravity in video generation? post-training newton's laws with verifiable rewards. *arXiv preprint arXiv:2512.00425*, 2025.

Qihao Liu, Adam Kortylewski, Yutong Bai, Song Bai, and Alan Yuille. Discovering failure modes of text-guided diffusion models via adversarial search. *arXiv preprint arXiv:2306.00974*, 2023.

Wenquan Lu, Yufei Xu, Jing Zhang, Chaoyue Wang, and Dacheng Tao. Handrefiner: Refining malformed hands in generated images by diffusion-based conditional inpainting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 7085–7093, 2024.

Ethan Ludwin-Peery, Neil R Bramley, Ernest Davis, and Todd M Gureckis. Limits on simulation approaches in intuitive physics. *Cognitive Psychology*, 127:101396, 2021.

Fanqing Meng, Wenqi Shao, Lixin Luo, Yahong Wang, Yiran Chen, Quanfeng Lu, Yue Yang, Tianshuo Yang, Kaipeng Zhang, Yu Qiao, et al. Phybench: A physical commonsense benchmark for evaluating text-to-image models. *arXiv preprint arXiv:2406.11802*, 2024.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? *arXiv preprint arXiv:2501.09038*, 2025.

OpenAI. Addendum to gpt-4o system card: Native image generation, 2025a. URL https://openai.com/index/gpt-4o-image-generation-system-card-addendum/.

OpenAI. Introducing 4o image generation, 2025b. URL https://openai.com/index/introducing-4o-image-generation/.

Ben Prystawski, Michael Li, and Noah Goodman. Why think step by step? reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems*, 36:70926–70947, 2023.

Kevin Smith, Peter Battaglia, and Joshua Tenenbaum. Integrating heuristic and simulation-based reasoning in intuitive physics.

Kevin Smith, Lingjie Mei, Shunyu Yao, Jiajun Wu, Elizabeth Spelke, Josh Tenenbaum, and Tomer Ullman. Modeling expectation violation in intuitive physics with coarse probabilistic object representations. *Advances in neural information processing systems*, 32, 2019.

Kevin A Smith and Edward Vul. Sources of uncertainty in intuitive physics. *Topics in cognitive science*, 5(1):185–199, 2013.

Felix A Sosa, Samuel J Gershman, and Tomer D Ullman. Blending simulation and abstraction for physical reasoning. *Cognition*, 254:105995, 2025.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 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.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022.

YingQiao Wang and Tomer D Ullman. Resource bounds on mental simulations: Evidence from a liquid-reasoning task. *Journal of Experimental Psychology: General*, 2025.

Fanyue Wei, Kai Xu, Yizhuo Zhang, Pengzhan Sun, Junbin Xiao, and Angela Yao. Cophybench: Benchmarking physical reasoning from conditional video observation. *Openreview (Submitted to ICLR)*, 2025.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

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

Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 699–707, 2017b.

Yi Xu, Chengzu Li, Han Zhou, Xingchen Wan, Caiqi Zhang, Anna Korhonen, and Ivan Vulić. Visual planning: Let's think only with images. *arXiv preprint arXiv:2505.11409*, 2025.

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.

Chenyu Zhang, Daniil Cherniavskii, Antonios Tragoudaras, Antonios Vozikis, Thijmen Nijdam, Derck WE Prinzhorn, Mark Bodracska, Nicu Sebe, Andrii Zadaianchuk, and Efstratios Gavves. Morpheus: Benchmarking physical reasoning of video generative models with real physical experiments. *arXiv preprint arXiv:2504.02918*, 2025.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644, 2024.

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

A LIMITATIONS

The main limitation of our work is that we analyze only three Image Generation Models. This is because, as of submission time, there are only two AI model providers with fully featured autoregressive IGMs – Google and OpenAI – and we tested their most capable IGMs. These companies have stated that IGMs are now their gold standard for text-to-image generation (OpenAI, 2025a; DeepMind, 2025), although this technology is new and has not yet been adopted by other AI companies. In App. F, we also provide analyses with a recently released open source IGM (HunyuanImage-3.0-Instruct). Since this model is currently under development, it does not yet support sequences of chat messages with images, although this is planned for the near future.

We recognize that the small number of available IGMs limits how well we can establish the generalization of our method. However, we argue that this is similar to the state of research on LLMs when Chain-of-Thought reasoning was introduced. Kojima et al. (2022) showed Chain-of-Thought reasoning improved performance on only two LLMs. Although they tested open source LLMs, these models were significantly inferior to closed source LLMs and could not effectively perform zero-shot reasoning. Zero-shot Chain-of-Thought reasoning has since been proven effective on a wide variety of LLMs, including much more powerful LLMs. Similarly, given our consistent success across all three IGMs, we expect that Chain-of-Time simulation will similarly improve physical simulation with more powerful IGMs in the future.

B STIMULI DESIGN AND SPECIFICATION OF THE STIMULI

B.1 2D Motion

The stimulus shows a 2D red ball being launched at the left middle of the screen on a white background in a simulated environment. There is no gravity or friction enabled, and no visible boundaries on the white surface. The white surface is flat and featureless, allowing the red ball to travel without obstructions, as shown in Fig. 2. We varied the speed at which the ball rolled over the white surface with three different speeds: 100, 300, and 500 (in units of pixels/second). This gives us 3 stimuli in total. The stimuli and the generated image from the image generation model have the same resolution (1024*1024) natively.

B.2 2D Gravity

The stimulus shows a 2D red ball being launched at various positions (left-middle, left-bottom, right-middle, right-bottom) on a white background in a simulated environment. There is gravity enabled, and no visible boundaries on the white surface. The white surface is flat and featureless, allowing the red ball to travel without obstructions, as shown in Fig. 2. We varied the speed at which the ball rolled over the white surface with three different speeds: 230, 240, and 250 (in units of pixels/second) and with two different angles: $45°$ and $60°$. This gives us 24 stimuli in total. The stimuli and the generated image from the image generation model have the same resolution (1024*1024) natively.

B.3 Fluids

The stimulus shows a 3D glass mug being filled with water on a light sky-blue background in a simulated environment. The water is emitted from a dark-grey pipe into the glass mug, as shown in Fig. 2. There is gravity enabled, and the amount of water is limited so that the mug never overflows. We varied the flow rate: 25, 50, and 75 (in frames/second), the type of cups: small, medium, and large, and the ground truth water levels: 1/12 full, 3/12 full, 5/12 full, 7/12 full, 9/12 full. The stimuli are cropped to 1024*1024 resolution, keeping only the water pipe and the glass water mug. This will give us 45 stimuli in total. The generated images from the image generation model have the same resolution (1024*1024) natively.

B.4 Bouncing

The stimulus shows a real world 3D object being launched from the top and bouncing back after hitting the ground as shown in Fig. 2. There is gravity enabled, and the objects each have a specific coefficient of restitution. We varied the coefficient of restitution of the object by including various balls made with different materials, and the falling rate. The velocity we varied for all trials includes 10, 15, 20, and 50 (frames / second). We will also separate each stimulus into three partitions. The first partition is *before*, which is the sequence of frames that shows the motion of the ball right before it hits the ground and starts to deform. The second partition is *during*, which is the sequence of frames that shows the motion of the ball hitting the ground, deforming, and beginning to bounce back. The third partition is *after*, which is the sequence of frames that shows the ball bouncing back upwards. This gives us 54 stimuli in total.

The stimulus images were resized to 1080*720, and the generated images from the image generation model are in 1536*1024, and were resized into 1080*720.

B.5 Pendulum

The stimulus shows a real world 3D simple pendulum oscillating between left and right as shown in Fig. 2. There is gravity enabled, and we vary the oscillation frequency of the simple pendulum in 10 and 15 (frames / second). We have 9 videos with different simple pendulums. This gives us 18 stimuli in total.

The stimulus images were resized to 1K resolution, with an aspect ratio of 16:9, 16:9, 4:5, 16:9, 2:3, 2:3, 16:9, 2:3, and 4:5.

B.6 Rolling on Slope

The stimulus from Bear et al. (2021) shows a 3D object beginning to roll down from the top of a curved ramp to the bottom, as shown in Fig. 2. The scene includes synthetic environments with other objects, intended to mimic the real physical world. There is gravity enabled, and we vary the speed of the object rolling in 10 and 15 (frames / second). We have 9 videos with different 3D objects varying by color and shape. This gives us 18 stimuli in total.

The stimulus images had a resolution of 512×512 pixels, with an aspect ratio of 1:1 for all the videos.

B.7 Collision

The stimulus from Bear et al. (2021) shows a 3D object colliding with another object, as shown in Fig. 2. The scene includes synthetic environments with other objects, intended to mimic the real physical world. There is gravity enabled, and we vary the speed of the object rolling in 10 and 15 (frames / second). We have 9 videos with different 3D objects varying by color and shape. This gives us 18 stimuli in total.

The resolution of the stimuli were 512*512 resolution, with an aspect ratio of 1:1 for all the videos.

B.8 Rolling on Track

The stimulus from YouTube shows various 3D objects rolling in complex tracks set up in simulated environments. The scene is extremely complex and realistic, with many realistic physics assets in order to closely resemble real-world scenes. We vary the speed of the object rolling in 10 and 15 (frames / second). We have 7 videos with different 3D objects varying by color and shape. This gives us 14 stimuli in total.

The stimulus images were resized to 1920*1080 resolution, with an aspect ratio of 9:16, 9:16, 9:16, 9:16, 9:16, 16:9, and 16:9.

B.9 Pulley System

The stimulus from Zheng et al. (2024) shows a complex pulley system consisting of multiple pulley systems in a simulated environment. Each pulley system consists of multiple static or movable pulleys, and a complex background to mimic the common real world complex pulley system. The gravity is enabled, and we vary the speed of the object moving on the pulley in 10 and 15 (frames / second). We have 13 videos with different 3D objects varying by color and shape. This gives us 26 stimuli in total.

The stimulus images were resized to 1920*1080 resolution, with an aspect ratio of 16:9 for all videos.

C PROMPTS

We defined two parameters for the prompts we used in all nine domains.

For parameter **number of seconds forward**, which is k, we chose from $k \in [0.5, 1.0, 2.0]$.

For parameter **scene content**, we choose it based on the domain. In `2D Motion`, we define the parameter as the following: *"a ball rolling on a surface"*. In `2D Gravity`, we define the parameter as the following: *"a ball being launched from a specific position and angle"*. In `Fluids`, we define the parameter as the following: *"a glass mug being filled with water, at a constant rate"*. For `Bouncing`, when simulating *before* and *during* partitions, we define the parameters as the following: *"a bouncy ball falling towards the ground"*. When simulating *after* partition, we define the parameters as the following: *"a bouncy ball bouncing upward after hitting the ground"*. In `Pendulum`, we define the parameter as the following: *"a bob swinging back-and-forth in simple pendulum motion"*. In `Rolling on Slope`, we define the parameter as the following: *"an object rolling down an inclined ramp"*. In `Collision`, we define the parameter as the following: *"a object moving towards and then colliding with another object that is lighter (in absolute mass)."*. In `Rolling on Track`, we define the parameter as the following: *"a ball rolling inside a track"*. In `Pulley System`, we define the parameter as the following: *"a running complex pulley system"*.

C.1 PROMPTS FOR ALL DOMAINS

In all domains, for our *Chain-of-Time* simulation method, as well as our Direct Prediction baseline, models are provided the following prompt, with different methods (*Chain-of-Time* Quarter, Half, and Direct Prediction) varying the `{{scene content}}` and `{{number of seconds forward}}` parameter:

Simulation Instruction Prompt

```
Consider the following sequence of 5 images, which show {scene_content}. Each
    image frame video is precisely 0.5 seconds after the last frame.

Please generate an image that continues this sequence, simulating what the scene
    will look like {number of seconds forward} seconds further into the future
    after the last frame.

{{image sequence}}
```

For *Chain-of-Time*, we use the following prompt to elicit subsequent simulation steps from the IGM:

Simulation Follow-Up Prompt

```
Continue this simulation {number of seconds forward} seconds into the future
    after the last frame that you generated.
```

C.2 ADDITIONAL EXPERIMENT: VARYING PROMPTS FOR ALL DOMAINS

In addition to the prompt we provided above, we also probed the effect of prompt complexity on our method. In this part, we conducted an additional experiment with the aim of exploring the prompt sensitivity of our method.

To investigate the method's sensitivity to the prompt, we designed another version of the prompt that provide more information than the prompt we listed in section C. The prompts below were designed to intentionally include more details about the physical simulation task itself, more detailed simulation instructions, and more details on the generated images themselves. These prompts therefore contrast with our simple general prompt in App.C.

For prompts, we designed three parameters for the prompts, and we used them in all nine domains, if necessary.

For parameter **number of seconds forward**, which is k, we chose from $k \in [0.5, 1.0, 1.5, 2.0]$.

For parameter **direction**, we choose from leftbottom, leftmiddle, rightmiddle, and rightbottom.

For parameter **scene content**, we choose it based on the domain.

In `Fluids`, we define the parameter as the following: *"a glass mug being filled with water, at a constant rate"*.

For `Bouncing`, when simulating *before* and *during* partitions, we define the parameters as the following: *"a bouncy ball falling towards the ground"*. When simulating *after* partition, we define the parameters as the following: *"a bouncy ball bouncing upward after hitting the ground"*.

In `Pendulum`, we define the parameter as the following: *"a bob swinging back-and-forth in simple pendulum motion"*.

In `Rolling on Slope`, we define the parameter as the following: *"an object rolling down a inclined ramp"*.

In `Collision`, we define the parameter as the following: *"a object moving towards and then colliding with another object that is lighter (in absolute mass)"*.

In `Rolling on Track`, we define the parameter as the following: *"a ball rolling inside a track"*.

In `Pulley System`, we define the parameter as the following: *"a running complex pulley system"*.

C.3 Prompt for `2D Motion`

In the `2D Motion` domain, for our *Chain-of-Time* simulation method, as well as our Direct Prediction baseline, models are provided the following prompt, with different methods (*Chain-of-Time* Quarter, Half, and Direct Prediction) varying the `number of seconds forward` parameter:

Simulation Instruction Prompt

```
Consider the following 5 frames, which show the motion of a red ball on a white
background. Note that each frame is precisely .5 seconds apart.

Now, please generate an image that simulates what this scene would look like
{number of seconds forward} Seconds into the future.

Make sure that your image is 2d and consists of a single red circle on a solid
white background. Ensure that the circle is exactly the same size as the input
images. Assume that there is no friction, the ground is flat, and the ball can
pass through objects.

{{image sequence}}
```

For *Chain-of-Time*, we used the following prompt to elicit subsequent simulation steps from the IGM:

Simulation Follow-Up Prompt

```
Now, simulate additional {number of seconds forward} seconds into the future.
```

C.4 Prompts for `2D Gravity`

In the `2D Gravity` domain, for our *Chain-of-Time* simulation method, as well as our Direct Prediction baseline, models are provided the following prompt, with different methods (*Chain-of-Time*

Quarter, Half, and Direct Prediction) varying the {direction} and {number of seconds forward} parameters:

Simulation Instruction Prompt

```
Consider the following 5 frames, which show the projectile motion of a red ball
being launched from the {direction}. Each frame is precisely 0.5 seconds apart.

You will generate an image that simulates the position of the red ball
{number of seconds forward} seconds forward into the future after the last
frame.

Assume that there is gravity and the red ball continues to follow the projectile
motion shown in the frames provided. The scene is viewed from the side, so
gravity pulls downward.

Make sure that the generated image is 2-D, that it contains a circle exactly the
same size and color as the circle in the frames provided, and that the
background color is white.

{{image sequence}}
```

For *Chain-of-Time*, we use the following prompt to elicit subsequent simulation steps from the IGM:

Simulation Follow-Up Prompt

```
Generate an additional image that simulates the position of the red ball
{number of seconds forward} seconds forward into the future after the last
frame that you generated.

Assume that there is gravity and the red ball continues to follow the projectile
motion shown in the frames provided. The scene is viewed from the side, so
gravity pulls downward.

Make sure that the generated image is 2-D, that it contains a circle exactly the
same size and color as the circle in the frames provided, and that the
background color is white.
```

C.5 Prompts for Fluids and Bouncing

In domains Fluids and Bouncing, for our *Chain-of-Time* simulation method, as well as our Direct Prediction baseline, models are provided the following prompt, with different methods (*Chain-of-Time* Quarter, Half, and Direct Prediction) varying the {scene content} and {number of seconds forward} parameters:

Simulation Instruction Prompt

```
Consider the following sequence of 5 images, which show {scene_content}.
Each image frame video is precisely 0.5 seconds after the last frame.

Please generate an image that continues this sequence, simulating what the scene
will look like {number of seconds forward} seconds further into the future
after the last frame. Make sure that the generated image is from exactly the
same perspective as the input images and that the background color remains the
same color.

{{image sequence}}
```

For *Chain-of-Time*, we use the following prompt to elicit subsequent simulation steps from the IGM:

Simulation Follow-Up Prompt

```
Continue simulating this scene {number of seconds forward} seconds into the
future after the last frame that you generated. Make sure that the generated
image is from exactly the same perspective as the input images and that the
background color remains the same color.
```

C.6 Prompts for all other domains

In all other domains, for our *Chain-of-Time* simulation method, as well as our Direct Prediction baseline, models are provided the following prompt, with different methods (*Chain-of-Time* Quarter, Half, and Direct Prediction) varying the `{scene content}` and `{number of seconds forward}` parameters:

Simulation Instruction Prompt

```
Consider the following sequence of 5 images, which show {scene_content}.
Each image frame video is precisely 0.5 seconds after the last frame.

Please generate an image that continues this sequence, simulating what the scene
will look like {number of seconds forward} seconds further into the future
after the last frame. Make sure that the generated image is from exactly the
same perspective as the input images and that the background color remains the
same color. In addition, make sure the object moving in the images you generated
follows the same motion shown in the images provided.

{{image sequence}}
```

For *Chain-of-Time*, we use the following prompt to elicit subsequent simulation steps from the IGM:

Simulation Follow-Up Prompt

```
Continue this simulation {number of seconds forward} seconds into the future
after the last frame that you generated. Make sure that the generated image is
from exactly the same perspective as the input images, the background color
remains the same color, and the object moving in the image you generated follows
the same motion shown in the images provided.
```

Here are the experimental results obtained using the prompts listed in App.C.6:

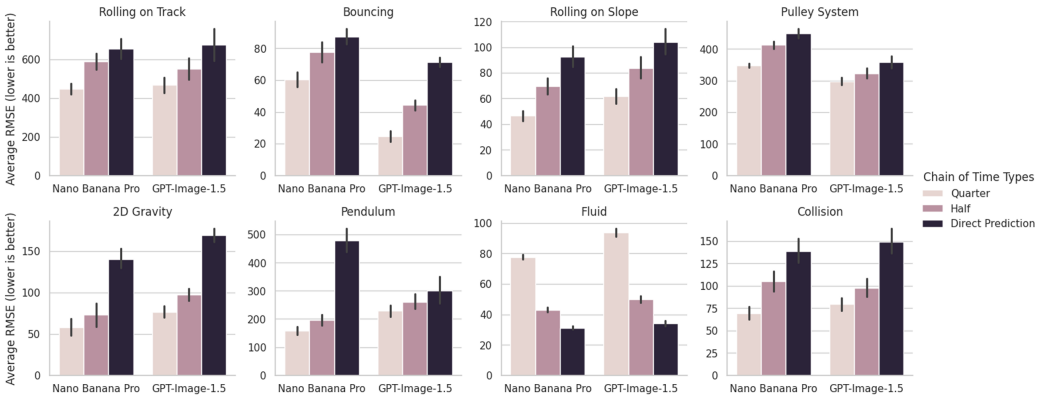

Figure 8: Prediction errors for the eight domains included in this analysis, averaged across all data for each domain using prompts listed in App. C.6. Prediction error is measured by taking the average RMSE between the ground-truth positions (location of focal object, or water level) and the final positions predicted by the IGM. Error bars are 95% Bootstrapped CI. We observe similar prediction between the results generated by the prompts in App. C.6 and the results generated by the prompts in App. C, which is shown in Fig. 4.

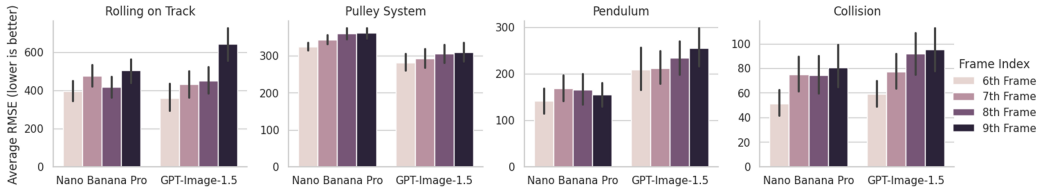

Figure 9: Prediction errors of each predicted frame for the four domains which have the most complicated object motion, averaged across all data for each domain under *Chain-of-Time* Quarter, using the data generated by App. C.6. Prediction error is measured by taking the average RMSE between the ground-truth positions (location of focal object) and the positions predicted by the IGM. Error bars are 95% Bootstrapped CI. We found that the result generated by using prompt listed in App. C.6 shows similar quantitative values and patterns when compared with the result generated by using prompt listed in App. C, which is shown in Fig. 5.

D Details of Object Detection Algorithms

D.1 SAM3 Details

In our analysis, we utilized Segment Everything Model 3Carion et al. (2025) to track the objects of interest in all domains except `Fluids`. The prompts that we used to extract the objects of interest are listed below by domain.

D.1.1 `2D Motion`,`2D Gravity`

Prompt for Object Detection in `2D Motion`and `2D Gravity`

A Red Ball

D.1.2 `Bouncing`

Prompt for Object Detection in `Bouncing`

1. A transparent ball
2. A ball with smiley face
3. A ball with Thomas the Tank Engine
4. A black ball
5. A tennis ball
6. A soccer ball
7. A tennis ball
8. A purple ball
9. A tennis ball

D.1.3 `Pendulum`

Prompt for Object Detection in `Pendulum`

1. A Silver Ball
2. A Silver Ball
3. A Silver Object
4. A Silver Ball
5. A Blue Ball
6. A Brass Ball
7. A Silver Ball
8. A Brass Ball
9. A Silver Object

D.1.4 `Rolling on Slope`

Prompt for Object Detection in `Rolling on Slope`

1. A Brown Ball
2. A Brown Object
3. A Brown Ball
4. A Dark Purple Ball
5. An Orange Object
6. A Dark Purple Ball
7. A Green Cube
8. A Dark Green Cube
9. A Dark Green Object

D.1.5 `Collision`

Prompt for Object Detection in `Collision`

1. A Black Object
2. A Green Object
3. A Green Ball
4. A Green Ball
5. A Green Ball
6. A Green Cube
7. A Green Ball
8. A Green Cylinder
9. A Green Object
10. A Green Ball

D.1.6 `Rolling on Track`

Prompt for Object Detection in `Rolling on Track`

1. An Orange Ball
2. A Silver Ball
3. A Silver Ball
4. A Red Ball
5. A Silver Ball
6. A Silver Ball
7. A Silver Ball

D.1.7 `Pulley System`

Prompt for Object Detection in `Pulley System`

1. A White Cube
2. A Yellow Cube
3. A Yellow Ball
4. An Orange Cube
5. A Purple Ball
6. A Blue Ball
7. A Purple Ball
8. A Cyan Ball
9. A White Cube
10. A White Ball
11. A Grey Ball
12. A Flesh-Color Ball
13. A Red Ball

D.2 SAM3 Accuracy Check

To ensure the stability and the accuracy of our evaluation pipeline, we investigated the robustness of the segmentation masks generated by SAM3 by conducting a mask evaluation based on subsampling. We sub-sampled 3 generated frames for each stimuli in each subdomain, and had a human evaluator to evaluate the correctness of SAM3 segmentation masks. We found that SAM3 received the maximum possible score from the human evaluator in each domain, except for fluids; for the fluids domain, our human evaluator assigned the maximum score to the masks generated by our hand-engineered algorithm. To demonstrate segmentation mask validity for the reader, we included a sample of these segmentation masks generated by SAM3 and fluid detector in App. D.3. While we believe that a more ideal validation would compare SAM3 masks to ground truth masks generated by humans, this process would be very resource intensive and beyond the scope of our present work.

D.3 Examples of SAM3 Segmentation

Here we present the example segmentation masks 10, 11, 12, 13 generated by SAM3 for the domains we used in the experiment.

D.4 COMPUTER VISION ALGORITHM USED IN THE FluidsDOMAIN

We found SAM 3 to be unreliable in appropriately tracking the water level in the `Fluids` domain. When using SAM, it consistently identified the stream of water pouring in addition to the water in the mug. This segmentation mask presented challenges for our metric for this domain, which is the top edge of the water level. In order to better identify the water level of the mug, we instead appealed to traditional hand-engineered (so to speak) computer vision algorithms.

The computer vision algorithms we used to detect the water blob and water level in the images generated by the image generation models were coded by ChatGPT-5. The code description provided by the model is as follows:

`Detect the water "blob" inside a glass mug and report its top/bottom y.`

Inputs

- `IGM_path`: path to RGB/RGBA image.
- `y_cut`: only analyze rows $y \geq y_cut$ (ignore the upper part of the image).
- `alpha_thresh`: pixels with alpha $\leq$ this are treated as transparent (ignored).
- `hsv_lower` / `hsv_upper`: HSV thresholds for sky-blue liquid (OpenCV $H \in [0, 179]$).
- `erode_iters`, `close_kernel`: morphology parameters to suppress thin rims/stream and fill gaps.
- `coverage_main` / `coverage_fallback`: minimum horizontal coverage (fraction of bbox width) required for a row to be considered liquid; a fallback is used if the main threshold yields none.
- `ignore_top_rows`: discard the first N rows inside the ROI to avoid picking the crop boundary.

Steps

1. Read the image with `cv2.IMREAD_UNCHANGED` so the alpha channel (if present) is preserved.
2. Build an alpha mask: alpha_mask $= (\alpha >$ alpha_thresh$)$. If no alpha channel exists, use an all-ones mask (all pixels opaque).
3. Crop to the region of interest (ROI): rows $y \geq y_cut$. Crop the alpha mask the same way.
4. Convert the ROI to HSV and threshold with $[$`hsv_lower`, `hsv_upper`$]$ to obtain a binary color mask for the sky-blue liquid.
5. AND the color mask with the alpha mask to suppress fully/mostly transparent background.
6. Morphological cleanup:
   - Erode (`erode_iters` times) to remove thin bright rims and the pouring stream.
   - Close (kernel size $=$ `close_kernel`) to reconnect the eroded blob and fill small holes.
7. Find connected components (external contours) and select the largest-area component as the liquid blob.
8. Compute the blob's bounding box (x, y, w, h) in ROI coordinates. Inside this box:
   - For each row, count foreground pixels (row_counts).
   - Mark rows as liquid if row_counts $\geq$ coverage_main $\times\, w$.
   - If none qualify, relax to coverage_fallback $\times\, w$.
   - Discard candidate rows whose index $<$ `ignore_top_rows`.
9. Determine water-level bounds:
   - Top y (global) $= y_cut + y + \min($valid_row_indices$)$ if valid rows exist; otherwise $y_cut + \min($contour_y$)$.
   - Bottom y (global) $= y_cut + y + \max($valid_row_indices$)$ if valid rows exist; otherwise $y_cut + \max($contour_y$)$.
10. Optional visualization:

- Red rectangle: blob bounding box.
- Green line: top y.
- Blue line: bottom y.

Notes

- HSV bounds are intentionally broad; tune to lighting and hue variations.
- Erosion suppresses 1–2 px rims and stream artifacts that otherwise bias the "top" level.
- The row-coverage rule prefers rows where a substantial horizontal span is filled, making the estimate robust to foam/splash highlights.
- Complexity is approximately $O(HW)$ per frame (thresholding + morphology + a single bbox scan).

And Fig. 14 is an example of the algorithm detecting the water level in images generated in `Fluids`.

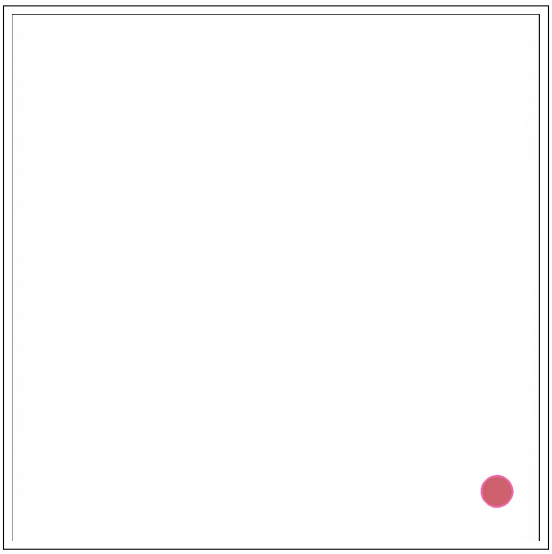

Figure 10: Examples of the SAM3 detecting the red ball for stimuli used in `2D Gravity`.

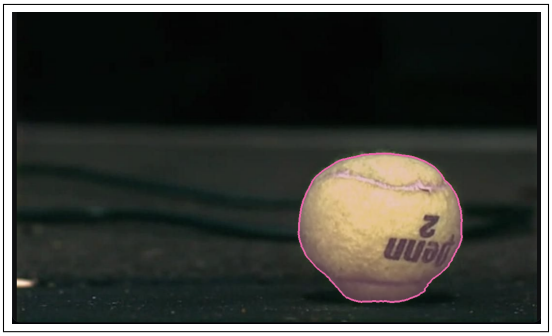

Figure 11: Examples of the SAM3 detecting the tennis ball for stimuli used in `Bouncing`.

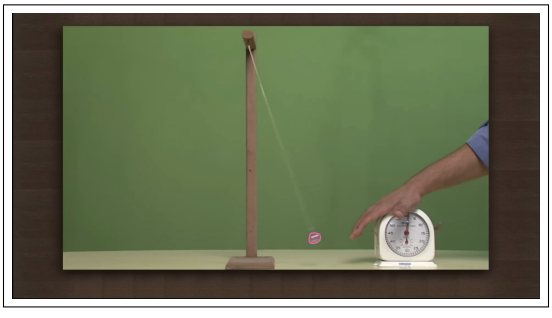

Figure 12: Examples of the SAM3 detecting the silver ball for stimuli used in `Pendulum`.

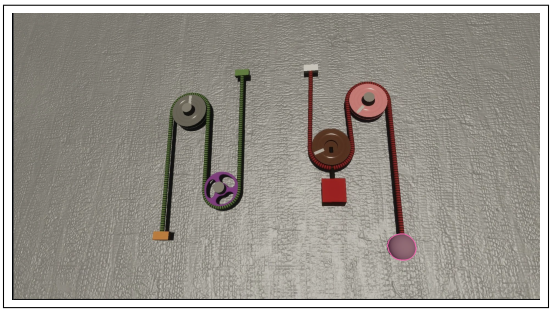

Figure 13: Examples of the SAM3 detecting the grey ball for stimuli used in `Pulley System`.

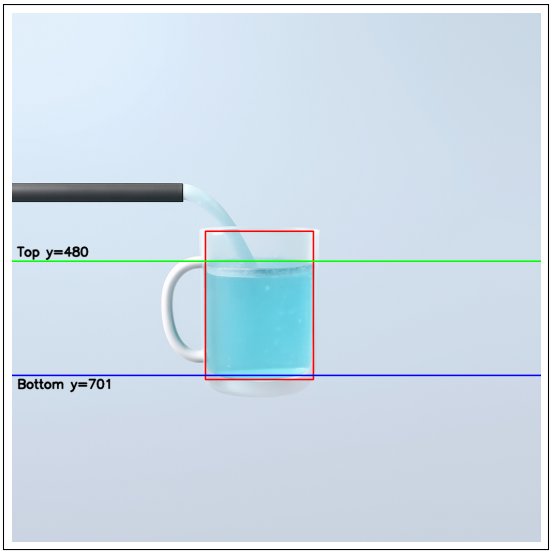

Figure 14: Examples of the algorithm detecting the water level for stimuli used in `Fluids`.

E ADDITIONAL ANALYSIS

E.1 THE Fluids DOMAIN

In Section 5, we observed that *Chain-of-Time* Quarter and Half exhibited higher average error than Direct Prediction in the fluid-dynamics domain. This motivated us to examine how the model perceives the initial state sequence $X_{0:t}$, which in the case of `Fluids`includes physical parameters such as flow rate and initial water level.

We analyzed the simulated image sequences $\widehat{I}_{t:T}$ generated by the IGM in `Fluids`. Specifically, we examined the predicted water-level time series for stimuli with slow (25 frames/s) and fast (75 frames/s) flow rates, and with low ($1/12$ and $3/12$ full) versus high ($7/12$ and $9/12$ full) initial water levels. As shown in Fig. 15, the IGM is sensitive to the initial water level—lowering the initial level shifts the intercept (dotted to solid line; left panel)—but largely insensitive to flow rate—the slope remains similar for 25 and 75 frames/s (right panel).

The sensitivity to the initial water level indicates that the IGM leverages salient visual cues in the inputs, whereas its insensitivity to flow rate suggests substantial error in estimating that parameter. Flow rate is a critical physical variable in this task, ensuring that the glass mugs are filled with the correct amount of water over time. These findings imply that errors in estimating the initial state $X_{0:t}$ can occur, especially for parameters that are not directly observable. Such estimation error accumulates over the rollout: as the *Chain-of-Time* step size k decreases (i.e., more, finer steps), the accumulated error grows. This helps explain why *Chain-of-Time* underperforms Direct Prediction in the fluid-dynamics domain, and why the Quarter configuration performs worse than Half and Direct Prediction.

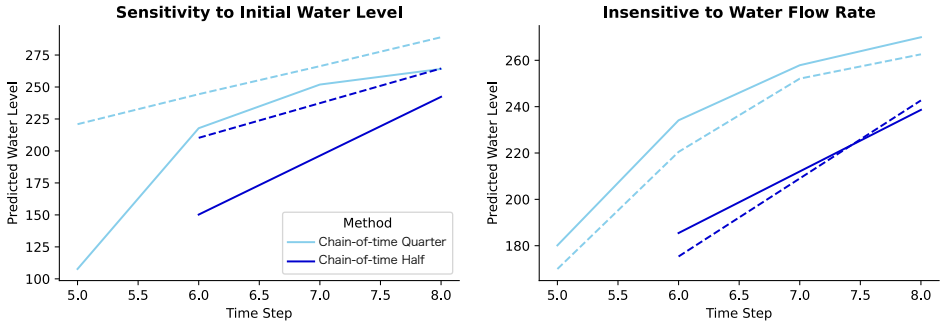

Figure 15: In the `Fluids` domain, we find that the IGMs are able to simulate water levels increasing over time. Here we show water levels in generated images steadily increasing as a function of time. (Left) The model is sensitive to initial water levels, with solid lines representing a low water level at the initial time of simulation t and dotted lines representing a high initial water level. (Right) However, the model is insensitive to the flow rate of water, with water level consistently increasing at the same rate during simulations. Solid lines represent a slow flow rate and dotted lines represent a 3x faster flow rate.

E.2 THE BouncingDOMAIN

Here are the additional trials in Bouncing:

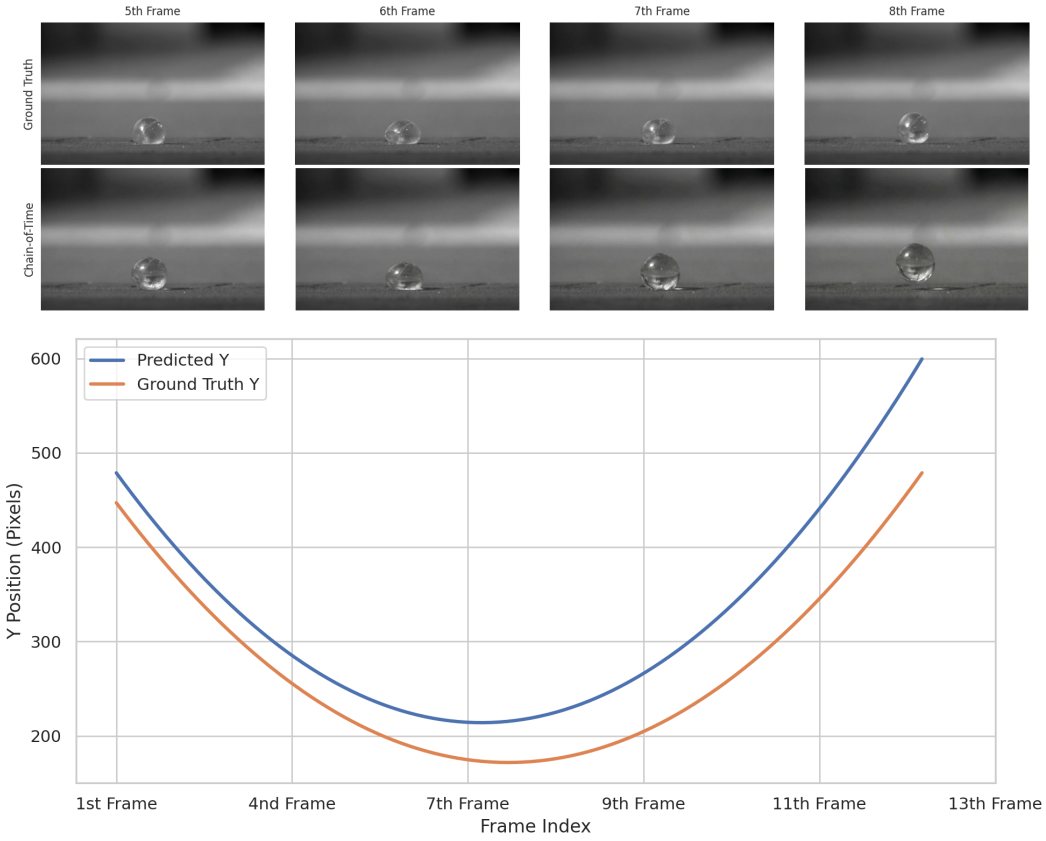

Figure 16: **Simulation analysis for Bouncing domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. The images shown correspond to the "during" period of the collision, beginning at frame 5 of 12. (Bottom) Extracted Y coordinates of the transparent ball in each frame, for three independent simulation results of four frames each: one simulation before the bouncing collision, one during, and one after collision. We observe parabolic motion in the ball for both the ground truth and simulated image sequences.

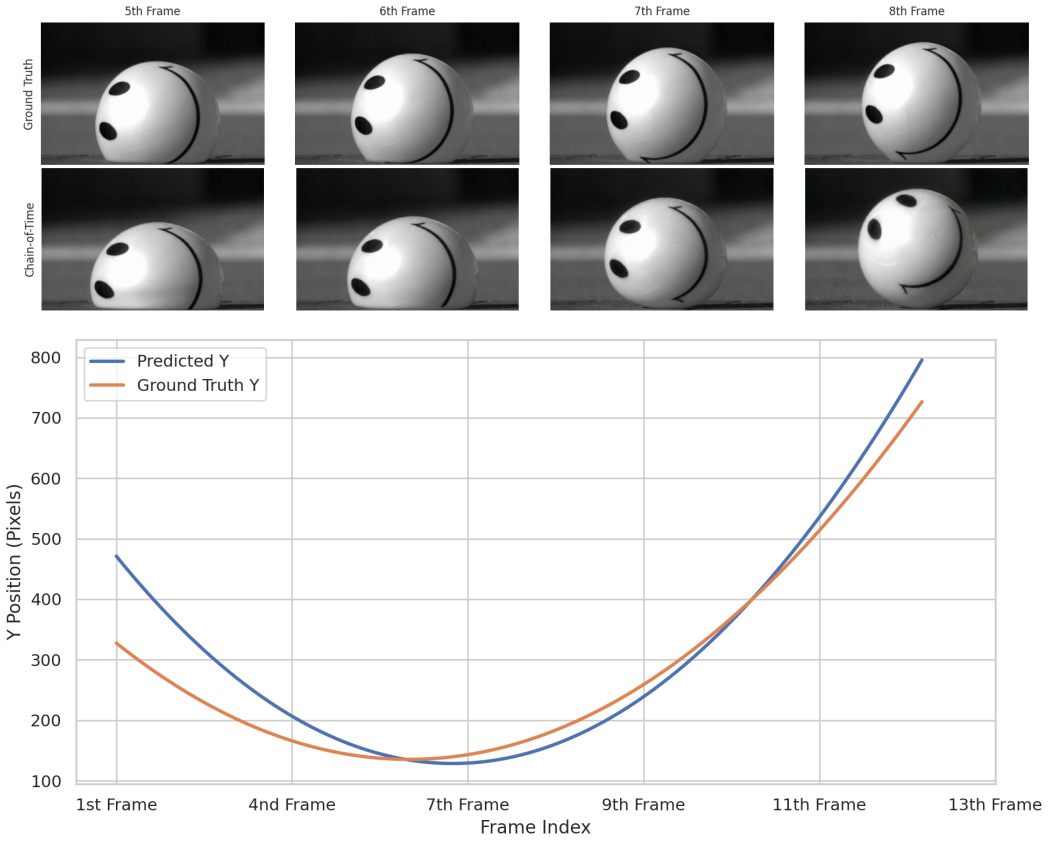

Figure 17: **Simulation analysis for `Bouncing` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. The images shown correspond to the "during" period of the collision, beginning at frame 5 of 12. (Bottom) Extracted Y coordinates of the smiley face ball in each frame, for three independent simulation results of four frames each: one simulation before the bouncing collision, one during, and one after collision. We observe parabolic motion in the ball for both the ground truth and simulated image sequences.

E.3 THE PendulumDOMAIN

Here are the additional trials in Pendulum:

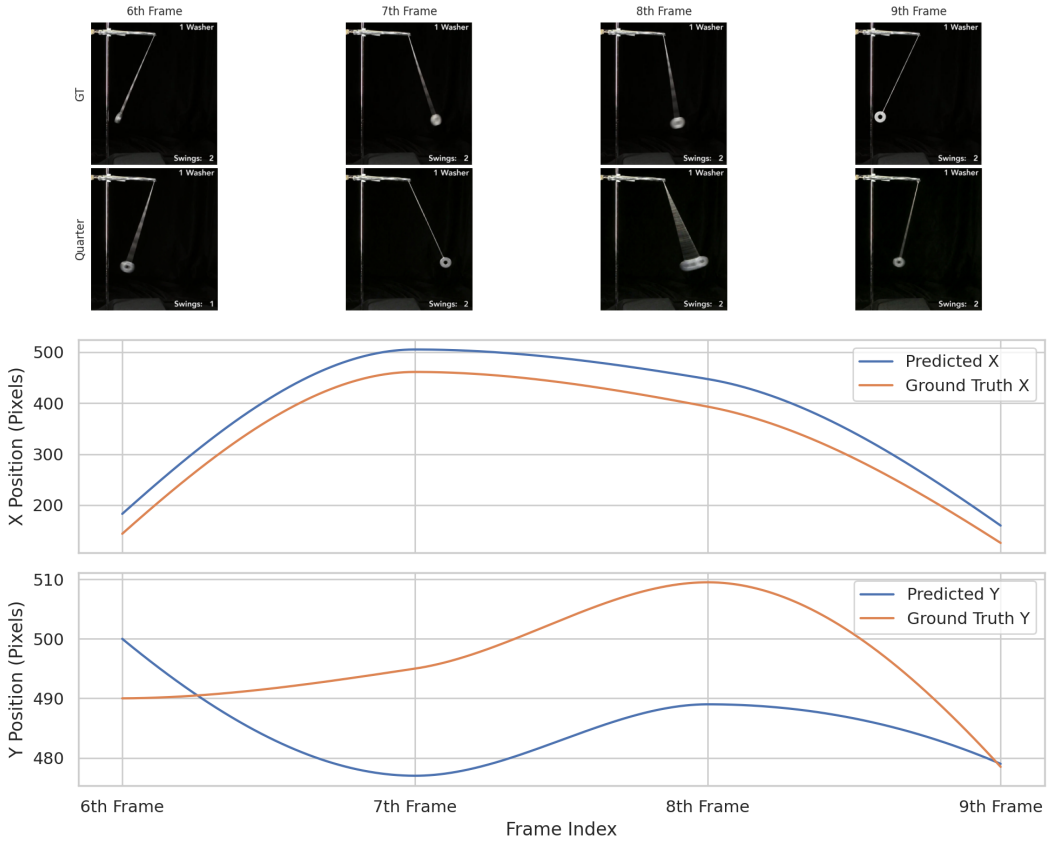

Figure 18: **Simulation analysis for Pendulum domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. (Bottom) Extracted X and Y coordinates of the pendulum in each frame. We observe oscillatory motion in the pendulum for both the ground truth and simulated image sequences.

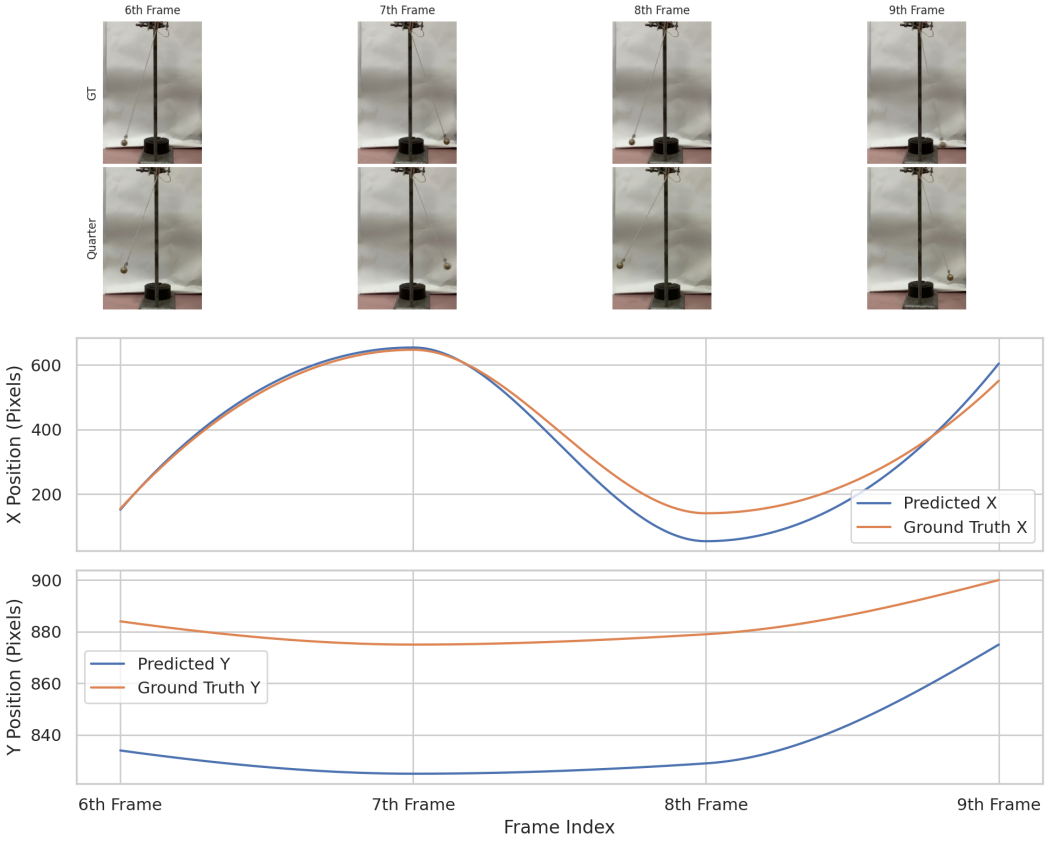

Figure 19: **Simulation analysis for `Pendulum` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. (Bottom) Extracted X and Y coordinates of the pendulum in each frame. We observe oscillatory motion in the pendulum for both the ground truth and simulated image sequences.

E.4 THE `Rolling on Slope` DOMAIN

Here are the additional trials in `Rolling on Slope`:

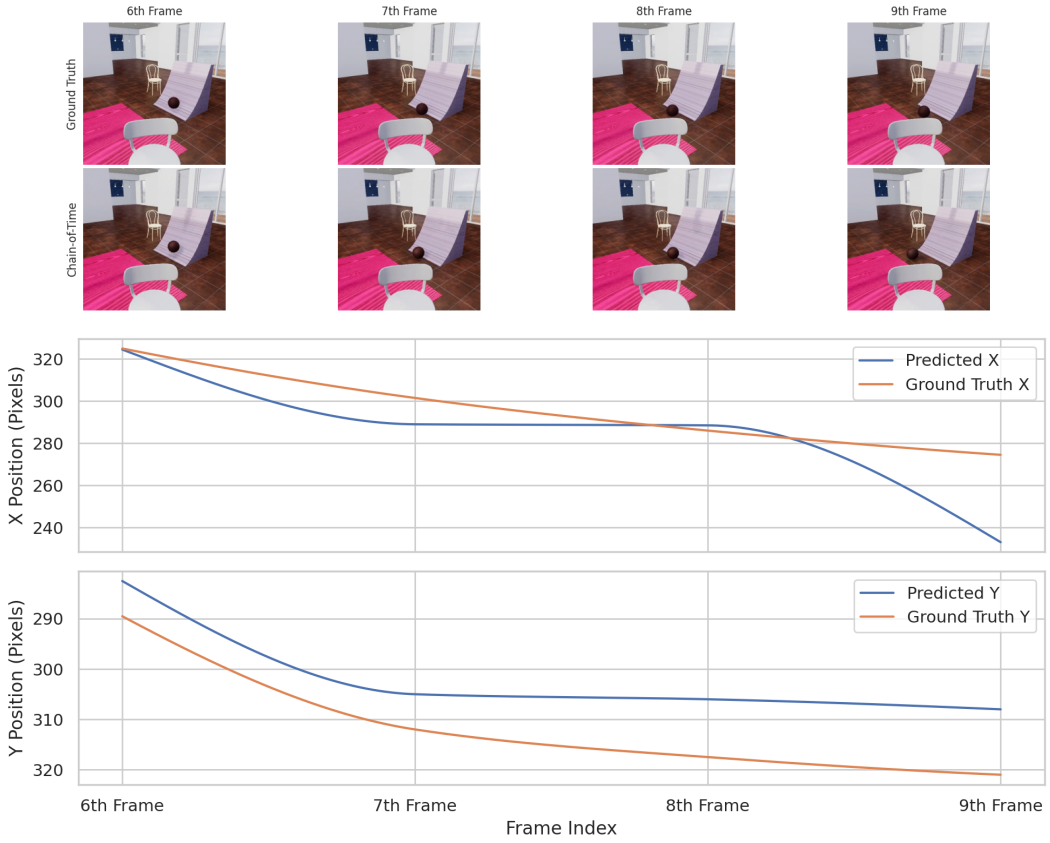

Figure 20: **Simulation analysis for `Rolling on Slope` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories, first accelerating sharply and then steadily decelerating.

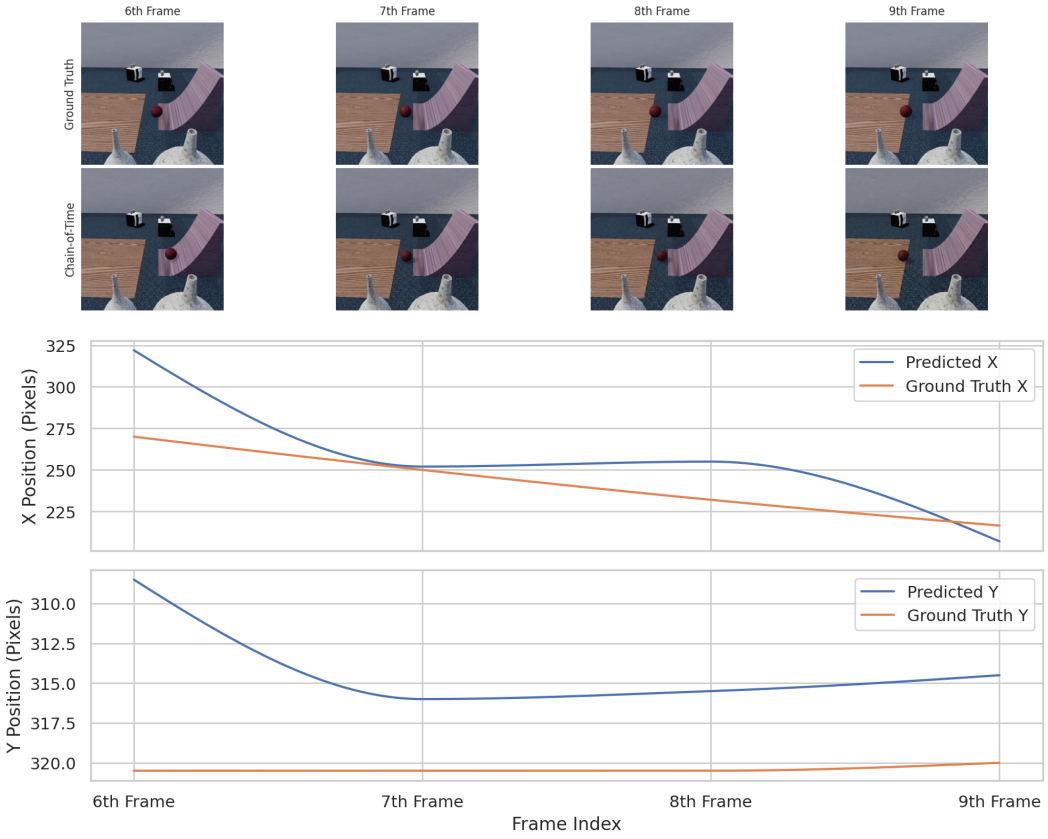

Figure 21: **Simulation analysis for `Rolling on Slope` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories, first accelerating sharply and then steadily decelerating.

E.5 THE Collision DOMAIN

Here are the additional trials in `Collision`:

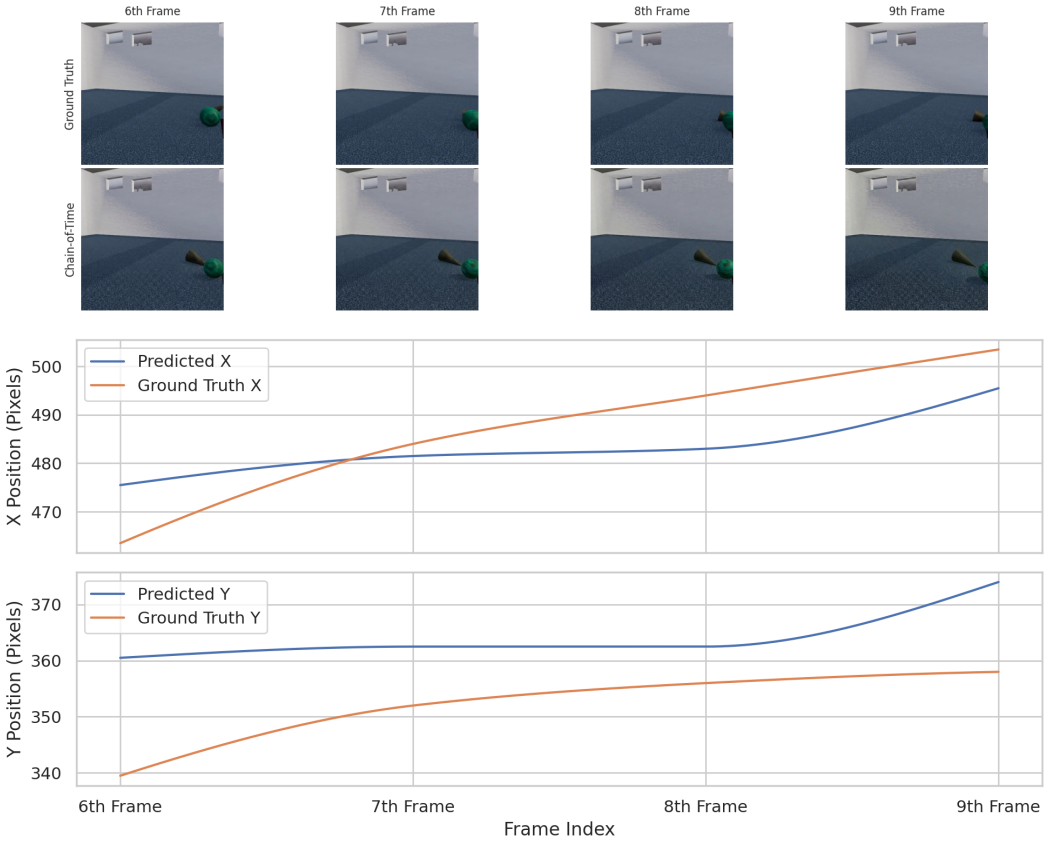

Figure 22: **Simulation analysis for `Collision` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories.

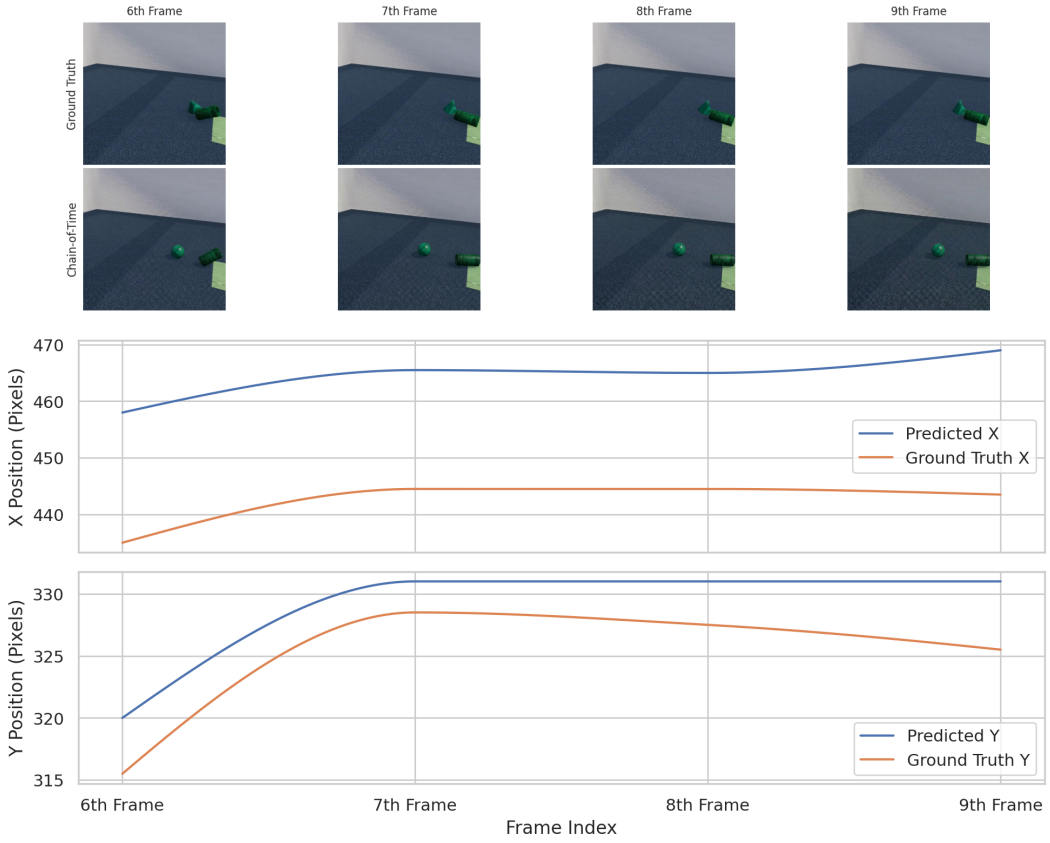

Figure 23: **Simulation analysis for `Collision` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories.

E.6 THE Rolling on Track DOMAIN

Here are the additional trials in `Rolling on Track`:

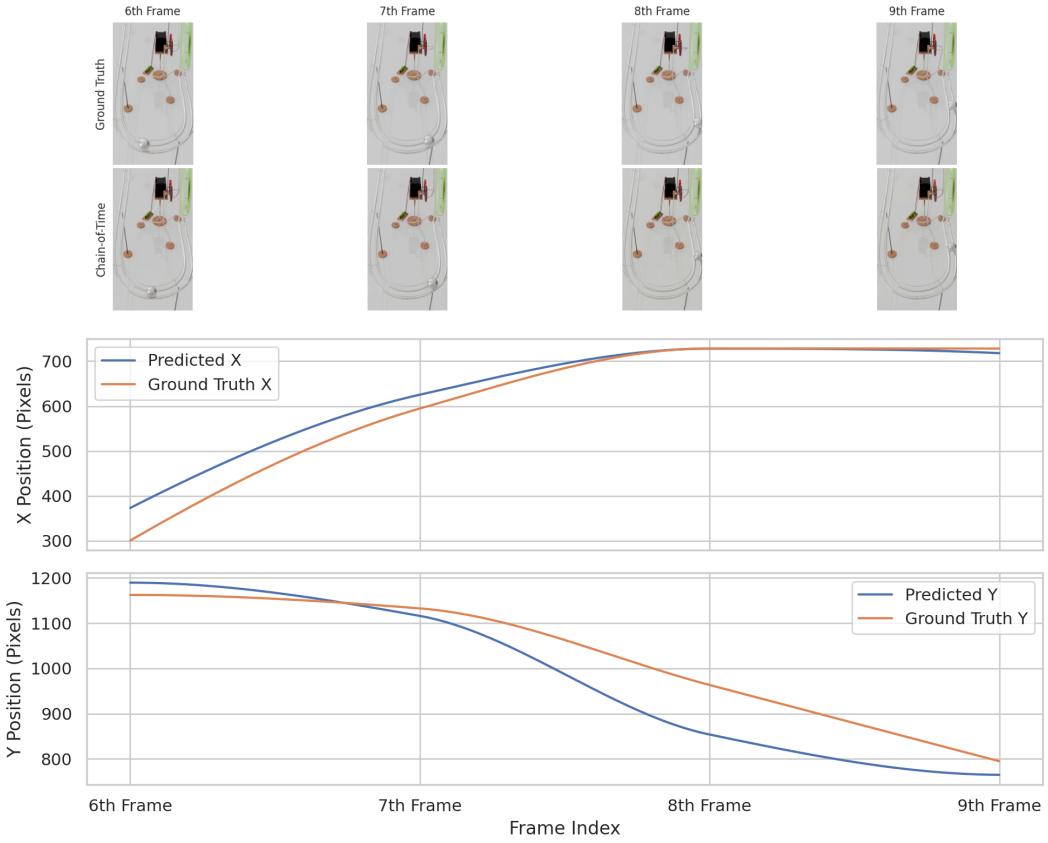

Figure 24: **Simulation analysis for `Rolling on Track` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories.

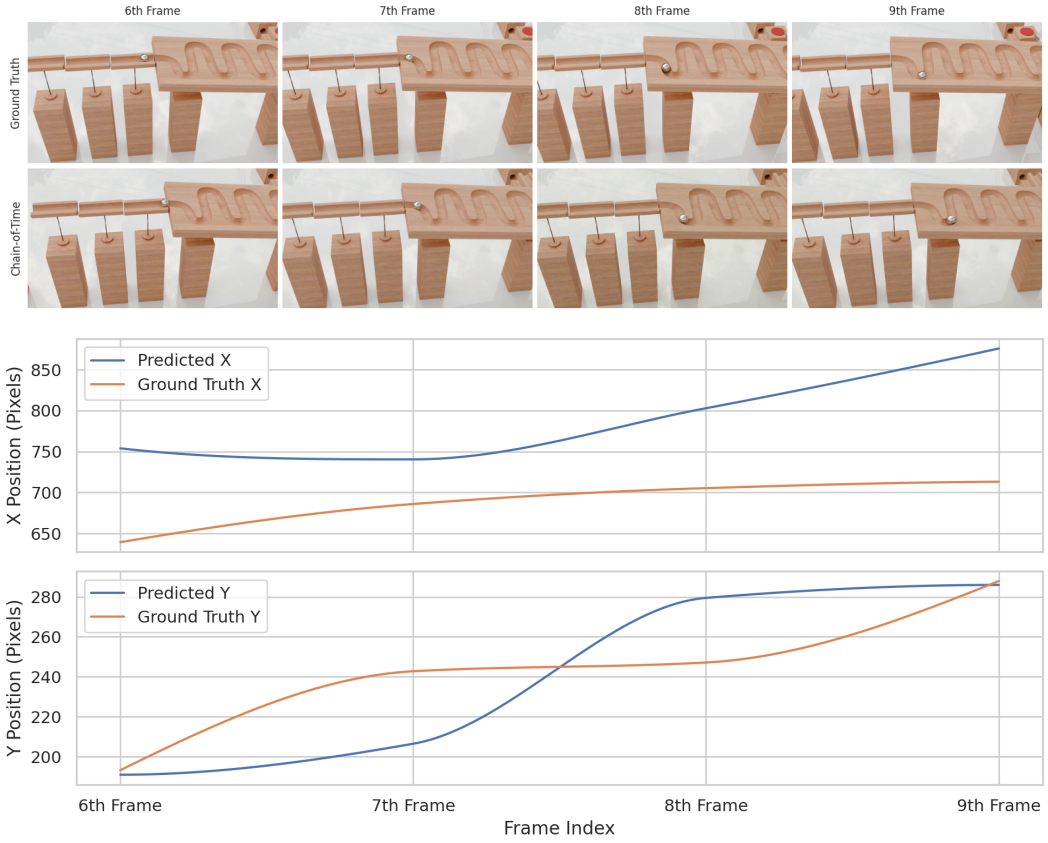

Figure 25: **Simulation analysis for `Rolling on Track` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for a single trial. Frame indexing begins at 6 because the first 5 frames are provided as input to the model. (Bottom) Extracted X and Y coordinates of the ball in each frame. The ground truth and simulated objects follow very similar trajectories.

E.7 THE `Pulley System` DOMAIN

Here are the additional trials in `Pulley System`:

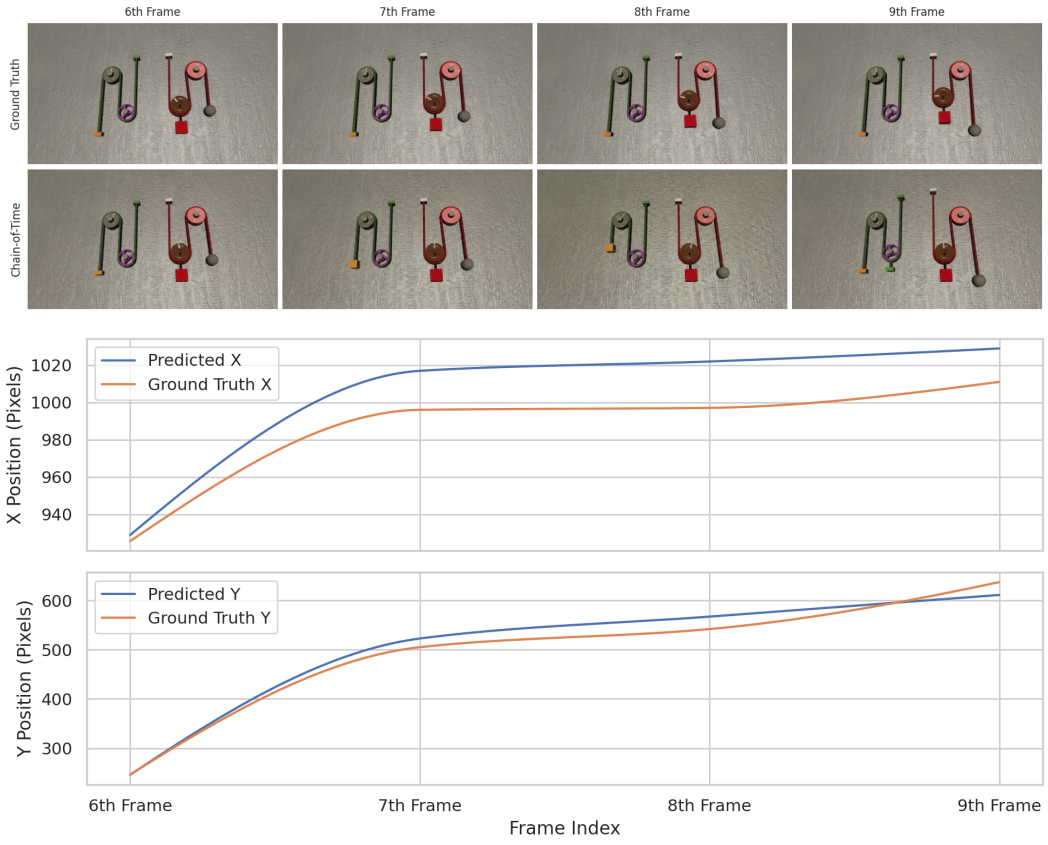

Figure 26: **Simulation analysis for `Pulley System` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. (Bottom) Extracted X and Y coordinates of the grey ball in each frame. Despite the complexity of this domain, image generation models are able to accurately simulate the ball's future motion.

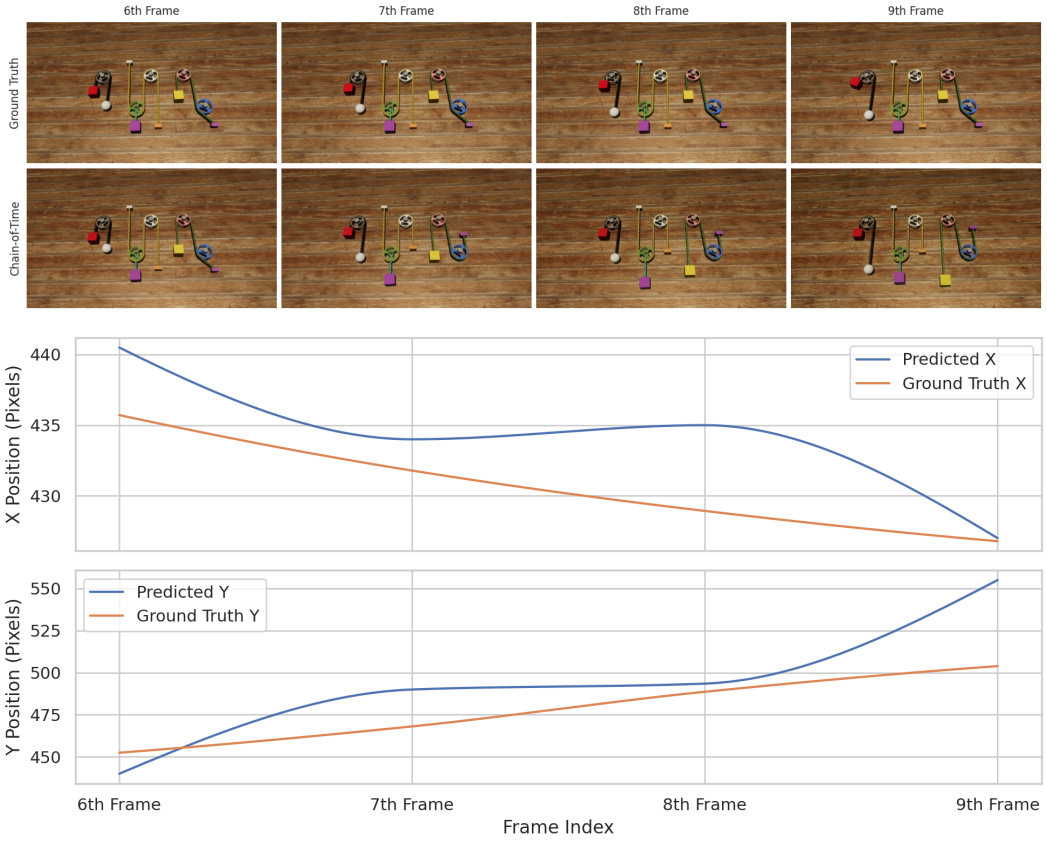

Figure 27: **Simulation analysis for `Pulley System` domain** (Top) Comparison between ground truth images and images generated by Chain-of-Time (quarter) for an example trial. (Bottom) Extracted X and Y coordinates of the white ball in each frame. Despite the complexity of this domain, image generation models are able to accurately simulate the ball's future motion.

F Open Source Model Results

In addition to the closed-source models we evaluated in our main paper, we also evaluated an open source autoregressive IGM that was recently released, `HunyuanImage-3.0-Instruct` (Cao et al., 2025a). However, since this model is currently under development, a necessary feature for Chain-of-Time simulation – multi-turn chat – is not yet available as it is for closed-source models (Nano Banana Pro and GPT-Image-Gen 1.5), although it is on the roadmap in the future.

Since this is the only commercial-grade open-source autoregressive image generation model available in the market right now, we adapt our Chain-of-Time method to operate without multi-turn chat. Instead of having the IGM generate each image iteratively as a new chat message (Fig. 1), we instead append simulated frames to the list of input images, and constrain this list to a limit of 3 input images, which for now is the maximum number of input images for this model. When multi-turn chat is added as a feature and the limit of 3 input images is expanded for this model in the near future, we will include results for this IGM along with our other models in our manuscript.

Here are the results from HunyuanImage-3.0-Instruct when we evaluated the domains with this model:

Method	2D Gravity	Bouncing	Rolling on Slope	Fluid	Pendulum
Chain-of-time Quarter	112 ± 5.2	72 ± 3.1	103 ± 3.2	92 ± 6.7	274 ± 4.8
Chain-of-time Half	133 ± 4.1	87 ± 5.9	117 ± 2.8	78 ± 4.2	312 ± 3.3
Direct Prediction	152 ± 6.7	103 ± 4.2	128 ± 3.6	64 ± 5.4	357 ± 3.6

Table 1: RMSE results across different methods and domains.

Our results with HunyuanImage-3.0-instruct are broadly similar to our findings with Nano Banana Pro and GPT-Image-Gen. We find that Chain-of-time consistently improves the performance of HunyuanImage-3.0-instruct over the Direct Prediction baseline, with the best performance at quarter-time. As with the other IGMs, this is true for all domains except fluids.

G Comparison with Video Model

Although our work studies IGMs, Chain-of-Time simulation is similar in some ways to video generation models, which also can be used to predict future frames given a sequence of input images. It is worth noting, though, that video models predict future frames in a dramatically different manner from Chain-of-Time, since these models typically rely on hundreds of input frames while Chain-of-Time only uses a few sparse input frames, and since video models are explicitly trained on sequential image prediction whereas IGMs are not.

We ran and analyzed a state-of-the-art open source video model (`LTX-video`) on two of our domains: the Pendulum and Rolling on Slope domains. Before describing our results, we would like to reiterate an important difference between our method - Chain-of-Time - and video generation models, which is that our method requires far less data as input, only a few image frames. When we tested video generation models with the same sparse input data that we use for Chain-of-Time, their output was incoherent, for example showing no motion at all. Therefore, in the following analysis we provided as input to LTX-video all frames of the first 2 seconds of the video and prompted the model to generate 2 seconds into the future at 30 frames per second using a prompt similar to what we have in App. C. We found that the simulations generated by the video model are less accurate (higher RMSE) or comparably accurate when compared to quarter-time Chain-of-Time, despite the former having 8 times more input frames and 15 times more output frames:

This indicates that video generation models may not possess any fundamental advantage in terms of simulation accuracy under a fixed future horizon when compared with an IGM paired with Chain-of-Time simulation, despite using substantially more computational resources.

Model Type	RMSE in Pendulum Domain	RMSE in Rolling on Slope Domain
LTX-Video	143 ± 19.1	89.8 ± 11.2
Nano Banana Pro	169 ± 14.6	46 ± 5.1
GPT-Image-Gen 1.5	243 ± 20.2	52 ± 6.6

Table 2: RMSE results across pendulum and rolling on slope domains. Three models were tested here: LTX-Video, Nano Banana Pro, and GPT-Image-Gen 1.5.

H Additional Experiments

H.1 Varying The Number of Input Frames

We set the number of input frames to 5, but our method could theoretically work with different numbers of input frames, for example, 2,3,4,6. Therefore, we performed an additional analysis of the impact of input frames in selected domains, aiming to investigate and address the impact of the number of input frames on the final accuracy between Chain-of-time Quarter and Direct Prediction. Varying the number of input frames also changes the temporal gaps in the input. We varied the number of input frames to [2,3,4,5,6], equally spaced out across a 2-second interval, so that e.g. with 2 input frames, the frames are 2 seconds apart and thus in-distribution for our Direct Prediction baseline. Vice versa, if input frames are not equally spaced out across a 2-second interval, e.g. 3,4,5,6 input frames, such interval is out-of-distribution for our Direct Prediction baseline. Our results for the Rolling on Slope domain are shown below:

Domain	2 Frames	3 Frames	4 Frames	5 Frames	6 Frames
Rolling on Track	12.3%	21.7%	27.3%	31.8%	33.2%
Bouncing	18.7%	32.7%	42.0%	49.2%	53.5%
Rolling on Slope	14.3%	35.9%	42.0%	50.5%	56.1%

Table 3: Accuracy improvements and gains over DP with Chain-of-Time across different input frame counts in three different domains.

We see the relative improvement in accuracy of Chain-of-Time (quarter time) compared to Direct Prediction. And according to the result above, accuracy gain diminishes as the number of in-context frames increases, and 3-5 in-context frames are sufficient to generate simulations with reasonable accuracy.

H.2 Varying The Number of Output Frames

In our main experiment, we set the number of output frames in [4,2,1], corresponding to Chain-of-time Quarter, Chain-of-time Half, and Direct Prediction. But theoretically, the number of output frames could vary, meaning that it could be more than 4, or it could be 3. As you increase the number of output frames generated, you are increasing your computational cost too. Therefore, the relationship between the number of output frames and accuracy is important to understand.

To investigate this relationship, we performed a simple trade-off analysis between accuracy gains and token consumption. Since in our setup, the token consumption is varied only when an additional intermediary image is generated, we consider token consumption in terms of how many additional intermediate images are generated, with which we will analyze the accuracy gain (reduction in RMSE) vs one additional intermediary image generated, considering all subdomains. Here is the result:

Additional Image Generation	1	2	3
Accuracy Gain	11.07%	13.1%	14.47%

Table 4: Accuracy gain across additional image generations.

In addition, we investigated the relationship between the tokens needed for 1% accuracy gain when additional images are generated. Here is the result:

Additional Image Generation	1	2	3
Tokens needed for 1% Accuracy Gain	309.03	540.98	1043.86

Table 5: Tokens needed for each 1% accuracy gain across additional image generations.

According to this analysis, we could see that the relationship between accuracy gain and token consumption is not linear, indicating that generating more images yields diminishing returns in terms of accuracy gain, and it costs more tokens for 1% accuracy gain as accuracy gain increases.

H.3 Analysis of Longer Time Horizons

We recognize that the longer time horizon is an important topic to explore in our method. To address this point, we conducted an additional experiment which tests our method's ability when simulating longer horizons of up to 10 seconds on Pendulum and Rolling on Track domains. The results of these analyses are shown below:

Model Type	CoT Quarter RMSE	CoT Half RMSE	Direct Prediction RMSE
Nano Banana Pro	232 ± 25.3	337 ± 24.8	420 ± 30.1
GPT-Image-Gen 1.5	293 ± 31.2	373 ± 29.2	436 ± 35.9

Table 6: RMSE results in the Pendulum domain.

Model Type	CoT Quarter RMSE	CoT Half RMSE	Direct Prediction RMSE
Nano Banana Pro	580 ± 37.1	646 ± 25.3	730 ± 30.8
GPT-Image-Gen 1.5	562 ± 34.2	651 ± 28.6	710 ± 32.7

Table 7: RMSE results in the Rolling on Track domain.

Here, we find that Chain-of-Time still vastly outperforms Direct Prediction, although the errors are higher than we find with the 2-second data we tested in our main experiments. This suggests that our method will generalize to videos with much longer time horizons than 2 seconds.

H.4 VLM-as-Critic For Error Correction

In Section 5, we found that error accumulates as the time horizon increases. Here, we investigate one method that could potentially serve as the method to reduce error accumulation.

We implemented a VLM-as-critic strategy [4], aiming to correct incremental errors during the Chain-of-Time. A VLM is implemented as a *"critic"* to verify if an intermediary image is physically plausible, considering the previous images (input images + previously generated intermediary images). Running a preliminary experiment on the Pendulum domain, we found that on average the accuracy (RMSE) improved by 18%, and the VLM reduces the accumulation of visual artifacts and physical inaccuracies, thus stabilizing the simulation process. In future work building on Chain-of-time, we plan to explore the potential of a VLM as a source of reward signals for both inference-time simulation and model fine-tuning.

H.5 New Baseline with Self-Consistency Selection with CoT Reasoning over Multiple Sampled Images

In the original Direction Prediction baseline, we averaged across all samples in each domain and calculated the average RMSE as an interval. But here, we aim to create an different baseline using self-consistent selection with CoT reasoning. For each stimuli within each domain, direct prediction will generate N=10 FINAL FRAMES. We use these N=10 generated FINAL FRAMES from Direct Prediction as the input along with the original 5 input frames back to the VLM. We use a VLM-as-judge that uses CoT reasoning to determine which of these images is the most accurate FINAL FRAME. The final RMSE for each stimuli within each domain is calculated using this final frame.

[4] We thank an anonymous reviewer for this suggestion.

Averaging across all stimuli for each domain, this forms a new baseline. Here is the comparison between the experimental data shown in Fig. 4 and the new baseline.

Domains	Rolling on Track	Bouncing	Rolling on Slope	Pulley System	2D Gravity	Pendulum	Fluid	Collision
Nano-Banana Pro (Original Baseline)	667 ± 53	93 ± 5.7	98 ± 6.2	462 ± 10.4	226 ± 12.8	288 ± 47.2	34 ± 2.3	142 ± 12.4
Nano-Banana Pro (New Baseline)	703	108	94	502	197	253	23	152
Nano-Banana Pro (Chain-of-Time Half)	582 ± 51	72 ± 5.2	69 ± 6.2	411 ± 15.1	74 ± 12.5	187 ± 16.9	43 ± 1.9	104 ± 10.6
Nano-Banana Pro (Chain-of-Time Quarter)	428 ± 27	65 ± 4.3	46 ± 5.1	339 ± 11.9	65 ± 8.4	169 ± 14.6	72 ± 1.8	58 ± 9.9
GPT-Image-1.5 (Original Baseline)	693 ± 97	74 ± 2.8	112 ± 9.7	378 ± 8.1	246 ± 11.7	306 ± 57.1	37 ± 1.8	152 ± 10.9
GPT-Image-1.5 (New Baseline)	682	57	103	355	214	317	34	131
GPT-Image-1.5 (Chain-of-Time Half)	562 ± 68	48 ± 3.7	88 ± 7.2	320 ± 18.5	93 ± 11.2	260 ± 24.5	47 ± 2.5	90 ± 12.1
GPT-Image-1.5 (Chain-of-Time Quarter)	446 ± 31	29 ± 4.2	52 ± 6.6	297 ± 13.2	80 ± 9.4	243 ± 20.2	90 ± 3.5	68 ± 9.9

Table 8: Prediction errors for the eight domains included in this analysis, averaged across all data for each domain, compared to the new baseline. Prediction error is measured by taking the average RMSE between the ground-truth positions (location of focal object, or water level) and the final positions predicted by the IGM. Error bars are 95% Bootstrapped CI. We generally find a monotonic relationship between *Chain-of-Time* precision and performance when compared against the new baseline as well, consistent with the results in Fig. 4.

I STATEMENTS

I.1 LANGUAGE MODEL USE

LLMs were used in this work for literature review, for coding assistance with constructing the computer vision algorithm described in App. D.4, and for proofreading the manuscript.

I.2 REPRODUCIBILITY

Code and data will be shared with the camera-ready version of this work. While the reproducibility of our results on closed source IGMs is limited by whether these models are still publicly available in the future, in App. F, we show results with an open-source IGM. The closed-source IGMs were tested using their respective official APIs, and the LTX-Video was run on a single A100 for 3 hours.

I.3 IMPACT AND RISKS

We provide methods for improving physical simulation in IGMs, and for systematically evaluating accuracy in simulated frames. While our results are unlikely to have immediate societal impacts, in the future, this may shape how we evaluate physical reasoning and related abilities in IGMs. Further, this work paves the way towards more powerful IGMs that may optimize directly for Chain-of-Time simulation ability, similar to how reasoning LLMs optimize directly for Chain-of-Thought Reasoning ability (Jaech et al., 2024).