Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning

Jiajun Wu*
EECS, MIT
jiajunwu@mit.edu

Ilker Yildirim*
BCS MIT, The Rockefeller University
ilkery@mit.edu

Joseph J. Lim
EECS, MIT
lim@csail.mit.edu

William T. Freeman
EECS, MIT
billf@mit.edu

Joshua B. Tenenbaum
BCS, MIT
jbt@mit.edu

1 Introduction

Our visual system is designed to perceive a physical world that is full of dynamic content. Consider yourself watching a Rube Goldberg machine unfold: as the kinetic energy moves through the machine, you may see objects sliding down ramps, colliding with each other, rolling, entering other objects, falling — many kinds of physical interactions between objects of different masses, materials and other physical properties. How does our visual system recover so much content from the dynamic physical world? What is the role of experience in interpreting a novel dynamical scene?

Recent behavioral and computational studies of human physical scene understanding push forward an account that people's judgments are best explained as probabilistic simulations of a realistic, but mental, physics engine [1, 5]. Specifically, these studies suggest that the brain carries detailed but noisy knowledge of the physical attributes of objects and the laws of physical interactions between objects (*i.e.*, Newtonian mechanics). To understand a physical scene, and more crucially, to predict the future dynamical evolution of a scene, the brain relies on simulations from this mental physics engine.

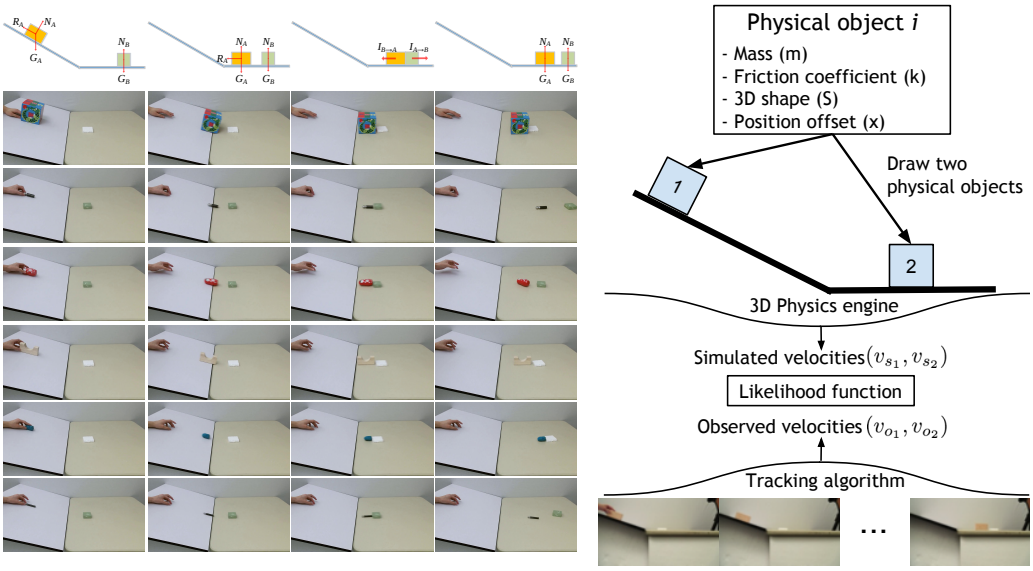
Here, we build on the idea that humans utilize a realistic physics engine as part of a generative model to interpret real-world physical scenes. We name our model Galileo. The first component of our generative model is the physical object representations, where each object is a rigid body and represented not only by its 3D geometric shape (or volume) and its position in space, but also by its mass and its friction. All of these object attributes are treated as latent variables in the model, and are approximated or estimated on the basis of the visual input.

The second part is a fully-fledged realistic physics engine — in this paper, specifically the Bullet physics engine [2]. The physics engine takes a scene setup as input (*e.g.*, specification of each of the physical objects in the scene, which constitutes a hypothesis in our generative model), and physically simulates it forward in time, generating simulated velocity profiles and positions for each object.

The third part of Galileo is the likelihood function. We evaluate the observed real-world videos with respect to the model's hypotheses using the velocity vectors of objects in the scene. We use a standard tracking algorithm to map the videos to the velocity space.

Now, given a video as observation to the model, physical scene understanding in the model corresponds to inverting the generative model by probabilistic inference to recover the underlying physical object properties in the scene. Here, we build a video dataset to evaluate our model and humans on real-world data, which contains 150 videos of different objects with a range of materials and masses over a simple yet physically rich scenario: an object sliding down an inclined surface, and potentially collide with another object on the ground. Note that in the fields of computer vision and robotics, there have been studies on predicting physical interactions or inferring 3D properties of objects for various purposes including 3D reasoning [4, 8] and tracking [6]. However, none of them

* indicates equal contributions. The authors are listed in alphabetical order.



(a) (b)

Figure 1: (a) Snapshots of the dataset. (b) Overview of the model. Our model formalizes a hypothesis space of physical object representations, where each object is defined by its mass, friction coefficient, 3D shape, and a positional offset w.r.t. an origin. To model videos, we draw exactly two objects from that hypothesis space into the physics engine. The simulations from the physics engine are compared to observations in the velocity space, a much "nicer" space than pixels.

focused on learning physical properties directly, and nor they have incorporated a physics engine with representation learning.

Here, our approach suggests one potential computational path to the development of the ability to perceive physical content in static scenes. In the spirit of the Helmholtz machine [3], we train a recognition model (*i.e.*, sleep cycle) that is in the form of a deep convolutional network, where the training data is generated in a self-supervised manner by the generative model itself (*i.e.*, wake cycle: real-world videos observed by our model and the resulting physical inferences). Interestingly, this computational solution asserts that the infant starts with a relatively reliable mental physics engine, or acquires it soon after birth.

2 Scenario

We seek to learn physical properties of objects by observing videos. Among many scenarios, we consider an introductory setup: an object is put on an inclined surface; it may either slide down or keep static due to gravity and friction, and may hit another object if it slides down.

This seemingly simple scenario is physically highly involved. The observed outcome of these scenario are physical values which help to describe the scenario, such as the velocity and moving distance of objects. Causally underlying these observations are the latent physical properties of objects such as the material, density, mass and friction coefficient. As shown in Section 3, our Galileo model intends to model the causal generative relationship between these observed and unobserved variables.

3 Galileo: A Physical Object Model

The gist of our model can be summarized as probabilistically inverting a physics engine in order to recover unobserved physical properties of objects. We collectively refer to the unobserved latent variables of an object as its *physical representation* T. For each object i, T_i consists of its mass m_i, friction coefficient k_i, 3D shape V_i, and position offset p_i w.r.t. an origin in 3D space.

The next component of our generative model is a fully-fledged realistic physics engine that we denote as ρ. Specifically we use the Bullet physics engine [2] following the earlier related work. The physics engine takes a specification of each of the physical objects in the scene within the

basic ramp setting as input, and simulates it forward in time, generating simulated velocity vectors for each object in the scene, v_{s_1} and v_{s_2} respectively – among other physical properties such as position, rendered image of each simulation step, *etc*.

In light of initial qualitative analysis, we use velocity vectors as our feature representation in evaluating the hypothesis generated by the model against data. We employ a standard tracking algorithm (KLT point tracker [7]) to "lift" the visual observations to the velocity space. That is, for each video, we first run the tracking algorithm, and we obtain velocities by simply using the center locations of each of the tracked moving objects between frames. This gives us the velocity vectors for the object on the ramp and the object on the ground, v_{o_1} and v_{o_2}, respectively.

Given a pair of observed velocity vectors, v_{o_1} and v_{o_2}, the recovery of the physical object representations T_1 and T_2 for the two objects via physics-based simulation can be formalized as:

$$P(T_1, T_2 | v_{o_1}, v_{o_2}, \rho(\cdot)) \propto P(v_{o_1}, v_{o_2} | v_{s_1}, v_{s_2}) \cdot P(v_{s_1}, v_{s_2} | T_1, T_2, \rho(\cdot)) \cdot P(T_1, T_2). \quad (1)$$

where we define the likelihood function as $P(v_{o_1}, v_{o_2} | v_{s_1}, v_{s_2}) = N(v_o | v_s, \Sigma)$, where v_o is the concatenated vector of v_{o_1}, v_{o_2}, and v_s is the concatenated vector of v_{s_1}, v_{s_2}. The dimensionality of v_o and v_s are kept the same for a video by adjusting the number of simulation steps we use to obtain v_o according to the length of the video. But from video to video, the length of these vectors may vary. In all of our simulations, we fix Σ to 0.05, which is the only free parameter in our model.

The posterior distribution in Equation 1 is intractable. In order to alleviate the burden of posterior inference, we use the output of our recognition model to predict and fix some of the latent variables in the model. Specifically, we determine the V_i, or $\{t_i, x_i, y_i, z_i\}$, using the output of the tracking algorithm, and fix these variables without further sampling them. Furthermore, we fix values of p_is also on the basis of the output of the tracking algorithm.

Once we initialize and fix the latent variables using the tracking algorithm as our recognition model, we then perform single-site Metropolis Hasting updates on the remaining four latent variables, m_1, m_2, k_1 and k_2. At each MCMC sweep, we propose a new value for one of these random variables, where the proposal distribution is Uniform$(-0.05, 0.05)$. In order to help with mixing, we also use a broader proposal distribution, Uniform$(-0.5, 0.5)$ at every 20 MCMC sweeps.

4 Experiments

In this section, we conduct experiments from multiple perspectives to evaluate our model. Specifically, we use the model to predict how far objects will move after the collision; whether the object will remain stable in a different scene; and which of the two objects is heavier based on observations of collisions. For every experiment, we also conduct behavioral experiments on Amazon Mechanical Turk so that we may compare the performance of human and machine on these tasks.

In the outcome prediction experiment, our goal is to measure and compare how well human and machines can predict the moving distance of an object if only part of the video can be observed. Specifically, for behavioral experiments on Amazon Mechanical Turk, we first provide users four full videos of objects made of a certain material, which contain complete collisions. In this way, users may infer the physical properties associated with that material in their mind. We select a different object, but made of the same material, show users a video of the object, but only to the moment of collision. We finally ask users to label where they believe the target object (either cardboard or foam) will be after the collision, *i.e*., how far the target will move. We tested 30 users per case.

We compare three kinds of predictions: human feedback, Galileo output, and, as a baseline, a uniform estimate calculated by averaging ground truth ending points over all test cases. Figure 2 shows the Euclidean distance in pixels between each of them and the ground truth. We can see that human predictions are much better than the uniform estimate, but still far from perfect. Galileo performs similar to human in the average on this task. Figure 3 shows, for some test cases, heat maps of user predictions, Galileo outputs (orange crosses), and ground truths (white crosses).

The second experiment is to predict which of two objects is heavier, after observing a video of a collision of them. For this task, we also randomly choose 50 objects, we test each of them on 50 users. For Galileo, we can directly obtain its guess based on the estimates of the masses of the objects.

Figure 4 demonstrates that human and our model achieve about the same accuracy on this task. We also calculate correlations between different outputs. Here, as the relation is highly nonlinear, we

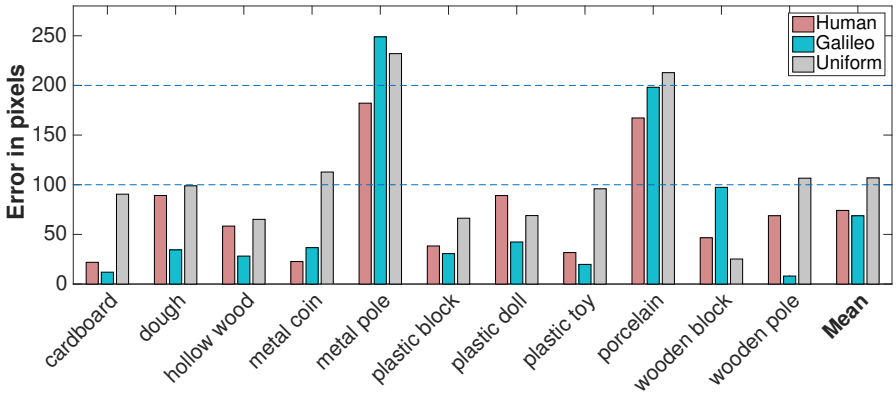


Figure 2: Mean errors in numbers of pixels of human predictions, Galileo outputs, and a uniform estimate calculated by averaging ground truth ending points over all test cases

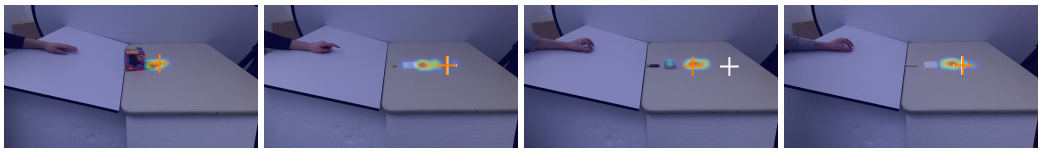


Figure 3: Heat maps of user predictions, Galileo outputs (orange crosses), and ground truths (white crosses).

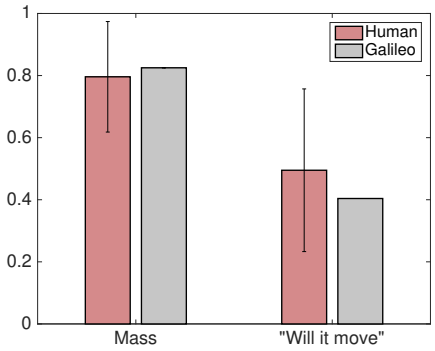


Figure 4: Average accuracy of human predictions and Galileo outputs on the tasks of mass prediction and "will it move" prediction. Error bars indicate standard deviations of human accuracies.

Mass	Spearman's Coeff
Human vs Galileo	0.51
Human vs Truth	0.68
Galileo vs Truth	0.52

"Will it move"	Pearson's Coeff
Human vs Galileo	0.56
Human vs Truth	0.42
Galileo vs Truth	0.20

Table 1: Correlations between pairs of outputs in the mass prediction experiment (in Spearman's coefficient) and in the "will it move" prediction experiment (in Pearson's coefficient).

calculate Spearman's coefficients. From Table 1, we notice that human responses, machine outputs, and ground truths are all positively correlated.

Our third experiment is to predict whether a certain object will move in a different scene, after observing one of its collisions. On Amazon Mechanical Turk, we show users a video containing a collision of two objects. In this video, the angle between the inclined surface and the ground is 20 degrees. We then show users the first frame of a 10-degree video of the same object, and ask them to predict whether the object will slide down the surface in this case. For Galileo, it is straightforward to predict the stability of an object in the 10-degree case using estimates from the 20-degree video.

Interestingly, both humans and the model are at chance on this task (Figure 4), and their responses are reasonably correlated (Table 1). Moreover, both subjects and the model show a bias towards saying "it will move." Future controlled experimentation and simulations will investigate what underlies this correspondence.

References

[1] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *PNAS*, 110(45):18327–18332, 2013.

[2] Erwin Coumans. Bullet physics engine. *Open Source Software: http://bulletphysics. org*, 2010.

[3] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[4] Zhaoyin Jia, Andy Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d reasoning from blocks to stability. *IEEE TPAMI*, 2014.

[5] Adam N Sanborn, Vikash K Mansinghka, and Thomas L Griffiths. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review*, 120(2):411, 2013.

[6] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1130–1137. IEEE, 2013.

[7] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. *International Journal of Computer Vision*, 1991.

[8] Bo Zheng, Yibiao Zhao, Joey C Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Detecting potential falling objects by inferring human action and natural disturbance. In *ICRA*, 2014.