Interpreting Dynamic Scenes by a Physics Engine and Bottom-Up Visual Cues

Ilker Yildirim*, Jiajun Wu*, Yilun Du, Joshua B. Tenenbaum
{ilkery, jiajunwu, yilundu, jbt}@mit.edu

Brain & Cognitive Sciences, MIT
*Indicates equal contribution.

Abstract

Humans have the remarkable ability to infer many different physical properties of objects from observing dynamic scenes. Here, we propose a generative model for inferring physical object properties from real world videos of objects splashing in the water. At the core of our generative model is a 3D physics engine which models the object's interactions with the environment based on parameters such as density, surface area, shape, and volume. We infer these properties through the combination of bottom-up visual cues and a MCMC algorithm, which drives the physical simulation to match real world observations. Results show that our model makes accurate estimations of unobserved physical properties such as mass and density.

1 Introduction

Our visual system allows us to effectively determine the physical parameters of objects around us. Just by watching two objects collide, we can infer the approximate ratio of masses between objects and aspects of the material content of each object. How does our visual system recover so many physical parameters from a dynamical world?

Some recent theories of human perception posit that the brain understands physical or dynamical scenes via a mental physics engine that is noisy and probabilistic in nature [2, 8], different from typical discriminative methods [4, 7]. This approach has recently been extended to people's understanding of liquids [1, 5].

Previously, we have shown that a 3D physics engine combined with fast bottom-up recognition networks and MCMC can be used to interpret real world dynamical scenes such as an object sliding down on a ramp and colliding another [9]. Here, we extend this work in two ways: (1) we present examples from a database of a larger set of real world physical scenarios, and (2) we extend our model to interpreting real world videos of objects splashing in water.

2 Physics 101: A Physical Scene Dataset

Wu *et al*. [9] presented a database of real videos where an object sliding through a ramp and colliding to another object. Here we extend the database to include examples of three new scenarios (Figure 1) [10]: objects splashing in the water (rows 1 and 2), objects falling onto three different types of surfaces (rows 3 and 4), and objects hanging from two different springs with one more stiff than the other (rows 4 and 5). For each scenario, we filmed about 36 unique objects including 13 different types of materials: foam, cardboard, hollow wood, hollow plastic, metal coin, plastic block, metal pole, porcelain, wooden block, rubber, wooden pole, plastic rings, and plastic toys.

In the rest of this abstract, we concentrate on the scenario in which the objects splash in the water (rows 1 and 2, Figure 1). The components of this scenario are quite complex. Not only does the outcome depend on the object's density, but also on the shape of the object and the drag coefficients of the water and the air.

3 Model

We first describe a physics-based generative model of objects splashing in the water before we describe how we perform inference with that model given observations.

Physical object model Following [9], the core of our generative model is a physical object that is described by its shape, volume, density, surface area that interacts with the water, and the air and water drag coefficients applied on the object.

We model the prior distribution over object shape, k, as a set of shape primitives consisting of vertical cylinders, horizontal cylinders, and boxes.

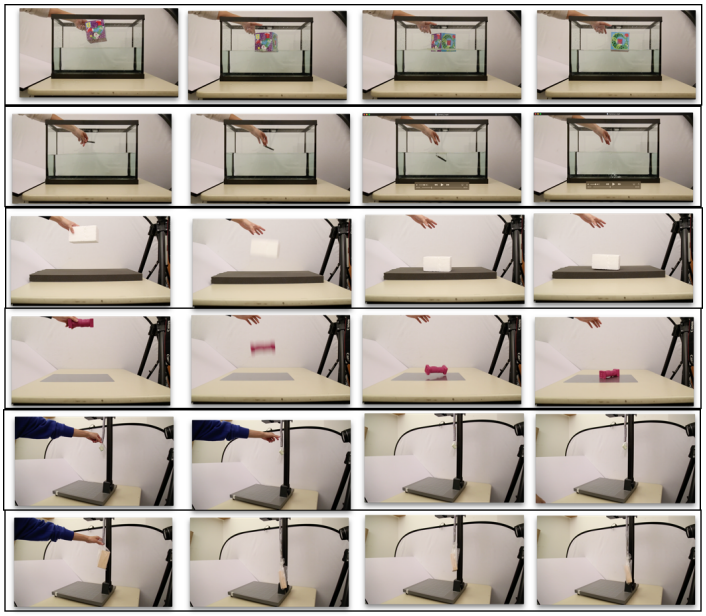

Figure 1: Examples of different scenes and objects in Physics 101

All three shape primitives have two degrees of freedom (height and radius for the cylinders; a square bottom, meaning equal width and depth, and height for the box). By scaling these two dimensions we obtain a sizable variety of different shapes. Instead of directly modeling these dimensions however, we represent them as surface area and volume, from which we can solve for these dimensions deterministically in all cases.

We model the volume of an object, h, so that it can vary in the range of 0.01 to $100cm^3$, and the surface $s \sim \text{Uniform}(0.1, 20)$ (in cm^2). We further model the density $\rho \sim \text{Uniform}(0.001, 2)$, water drag coefficient $D_w \sim \text{Uniform}(0.001, 2)$, and air drag coefficient, D_a, which is deterministically related to D_w through $D_a = D_w \times 10^{-3}$ due to the approximate ratio of their densities.

Finally, we model the observation start frame as a random variable $t \sim \text{Uniform}(5, 30)$. The observation frame represents the time step at which we start matching of physics engine simulation to observed values.

Physics engine The next major component of our generative model is a fully fledged physics engine, namely Bullet physics engine [3], which we denote as χ. The physics simulation takes as input the object's volume, surface area, and shape to generate an approximate scaling factor for each dimension of the object. Using this input and information about the objects drag coefficient and density, the physics engine then models the objects fall into water by applying impulses to the object proportional to the volume of the object submerged to surface. The physics engine then takes our specifications and steps the simulation forward in time, to generate simulated velocity vectors v_s and x_s.

Bottom-up recognition model We link our simulations to real world videos by using the observed velocity vectors and positions in the video. Using bottom-up visual cues, we lift the raw observations from the videos to velocity and position vectors as the following. We first perform background subtraction to isolate the foreground of the moving object using a Gaussian mixture model (with five components), and then track the resulting video of the isolated object using the KLT tracker [6]. We also use the tracker's output to make a bottom-up estimate of the volume of the moving object (see Inference for more).

Likelihood function Given observation vectors v_o and x_o, the recovery of the physical object representation $T = \{k, s, h, \rho, D_a, D_w\}$ can be formalized as

$$P(T|v_o, x_o, \chi(\cdot)) \propto P(v_o, x_o|v_s, x_s) \cdot P(v_s, x_s|T, \chi(\cdot), t) \cdot P(t) \cdot P(T), \quad (1)$$

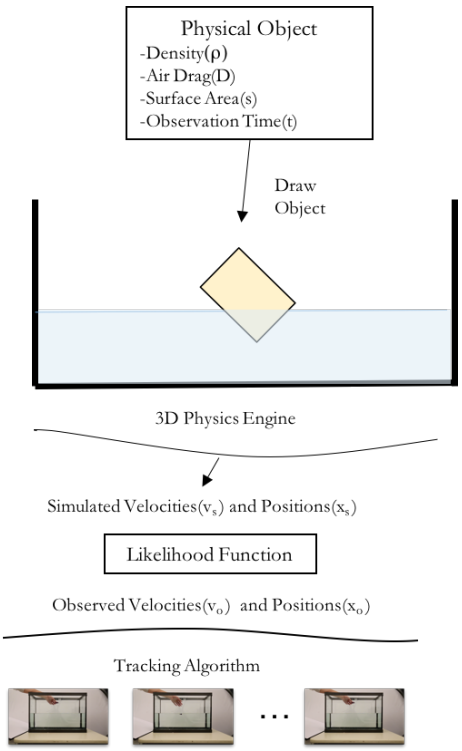

Figure 2: Schematic of inferring physical properties

where the likelihood function is defined to be

$$P(v_o, x_o | v_s, x_s) = N(v_o, x_o | v_s, x_s, \Sigma). \quad (2)$$

We select the initial t frames from the simulation and use as many steps as necessary to set the lengths of v_s and x_s to be the same length as v_o and x_o. The length of v_o and x_o vary though, depending on the length of the video that the model is watching. In all simulations, we fix Σ to be equal to 1, though changes in values of Σ do not seem to effect the model. In our experiments, we found that both velocity profiles and positional information are important for accurately inferring the physical information.

3.1 Inference

Note that the current posterior represented by Equation 1 is degenerate. To alleviate this problem, and to speed up inference in the model, we heuristically estimate the parameters of object shape based on the statistics of the tracked points provided by the KLT tracker [6]. We then compute the approximate volume of the object h from them.

Once we set the volume to its bottom-up approximated value, we use MCMC to determine the values of the remaining four latent variables. Given volume, we initialize the surface area, s, such that all dimensions of the object are approximately equal. We initialized density, ρ, as 0.5–half the density of water. We implement a Metropolis-Hastings algorithm to perform posterior inference, where at each MCMC sweep we set the proposal distribution for density, ρ, and surface area, s, to be Uniform$(-0.05, 0.05)$ centered at the current value, and the proposal distribution for water drag coefficient, D_w, to be Uniform$(-0.0005, 0.0005)$ again centered at the current value. Remember that $D_a = 0.001 \times D_w$ as we assume that the water is $1{,}000$ times denser than the air. The proposal for observation start frame, t, is uniform over the set $\{-1, 1\}$, meaning that the starting frame could be proposed to be up or down by one frame. To help better mixing in our chains, at every 20 steps, we set the proposal distribution to be much larger – for density and surface area to be Uniform$(-0.5, 0.5)$ and for water drag coefficient to be Uniform$(-0.002, 0.002)$.

4 Experimental Results

To estimate the mass of each measured objects, 50 MCMC chains were run on each input video, and final density results for the MCMC chain were multiplied by the volume of the object to get the overall mass prediction for every single object. Figure 4 shows some final MCMC based posterior distributions the surface area and mass. The median mass results from the

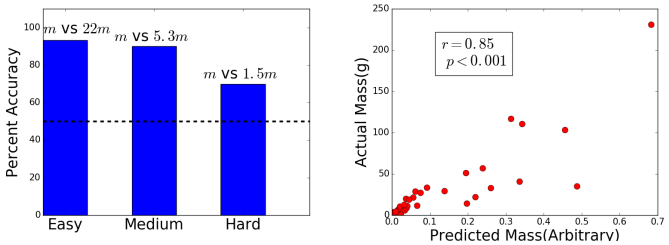

Figure 3: Left: Results of prediction on the heaviest object in pairs of objects in easy tasks (median mass of 22 times mass of other item), moderate tasks (median mass 5.3 times mass of other item) and difficult tasks (median mass 1.5 times mass of other item). Right: Plot of predicted mass of objects vs the actual mass of objects.

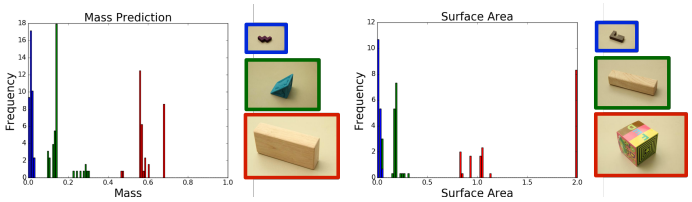

Figure 4: Left: Prediction of MCMC algorithm on the mass distribution on 3 objects. Right: Prediction of MCMC on the surface area distribution of 3 objects.

MCMC chains were then used to make mass related judgments about an objects.

To evaluate our system, we performed an experiment in which we showed two videos to our system each featuring a different object. The model had to determine which of the two objects was heavier. A series of 30 easy tasks were generated in which the median difference between the masses of the two alternative objects was 22.01. A series of 30 medium tasks were generated in which the median difference between the masses of objects was 5.3 times the other. Finally, a series of 30 difficult tasks were generated in which the median difference between of the masses of objects was 1.5 that of the other . As the left panel in Figure 3 shows, our model performed highly accurately even with the hardest set of tasks. This indicates that our model's estimates of the density and volume of objects just by watching them splash in the water are quite accurate.

As a further analysis of the mass estimates of our model, we also plotted our models prediction about the masses (as density times the volume) of the 33 unique objects. (The MCMC algorithm did not converge for the other 3 objects.) Results are presented in a scatter plot in Figure 3. We were surprised by the remarkably high correlations between our model's estimates and the true mass values ($r = 0.85$).

[1] Christopher J Bates, Ilker Yildirim, Joshua B Tenenbaum, and Peter W Battaglia. Humans predict liquid dynamics using probabilistic simulation. In *CogSci*, 2015. 1

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

[3] Erwin Coumans. Bullet physics engine. *Open Source Software: http://bulletphysics. org*, 2010. 1

[4] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. In *ICLR*, 2016. 1

[5] James Kubricht, Chenfanfu Jiang, Yixin Zhu, Song-Chun Zhu, Demetri Terzopoulos, and Hongjing Lu. Probabilistic simulation predicts human performance on viscous fluid-pouring problem. In *CogSci*, 2016. 1

[6] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. 1, 2

[7] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016. 1

[8] 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. 1

[9] Jiajun Wu, Ilker Yildirim, Joseph J Lim, William T Freeman, and Joshua B Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*, 2015. 1

[10] Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016. 1