Learning to See Physics via Visual De-animation

Jiajun Wu
MIT CSAIL

Erika Lu
University of Oxford

Pushmeet Kohli
DeepMind

William T. Freeman
MIT CSAIL, Google Research

Joshua B. Tenenbaum
MIT CSAIL

Abstract

We introduce a paradigm for understanding physical scenes without human annotations. At the core of our system is a physical world representation that is first recovered by a perception module and then utilized by physics and graphics engines. During training, the perception module and the generative models learn by visual de-animation — *interpreting and reconstructing the visual information stream. During testing, the system first recovers the physical world state, and then uses the generative models for reasoning and future prediction.*

Even more so than forward simulation, inverting a physics or graphics engine is a computationally hard problem; we overcome this challenge by using a convolutional inversion network. Our system quickly recognizes the physical world state from appearance and motion cues, and has the flexibility to incorporate both differentiable and non-differentiable physics and graphics engines. We evaluate our system on both synthetic and real datasets involving multiple physical scenes, and demonstrate that our system performs well on both physical state estimation and reasoning problems. We further show that the knowledge learned on the synthetic dataset generalizes to constrained real images.

1. Introduction

Inspired by human abilities, we wish to develop machine systems that understand scenes. Scene understanding has multiple defining characteristics which break down broadly into two features. First, human scene understanding is *rich*. Scene understanding is physical, predictive, and causal: rather than simply knowing what is where, one can also predict what may happen next, or what actions one can take, based on the physics afforded by the objects, their properties, and relations. These predictions, hypotheticals, and counterfactuals are probabilistic, integrating uncertainty as to what is more or less likely to occur. Second, human scene understanding is *fast*. Most of the computation has to happen in a single, feedforward, bottom-up pass.

There have been many systems proposed recently to tackle these challenges, but existing systems have architectural features that allow them to address one of these features but not the other. Typical approaches based on inverting graphics engines and physics simulators [4] achieve richness at the expense of speed. Conversely, neural networks such as PhysNet [5] are fast, but their ability to generalize to rich physical predictions is limited.

We propose a new approach to combine the best of both. Our overall framework for representation is based on graphics and physics engines, where graphics is run in reverse to build the initial physical scene representation, and physics is then run forward to imagine what will happen next or what can be done. Graphics can also be run in the forward direction to visualize the outputs of the physics simulation as images of what we expect to see in the future, or under different viewing conditions. Rather than use traditional, often slow inverse graphics methods [4], we learn to invert the graphics engine efficiently using convolutional nets. Specifically, we use deep learning to train recognition models on the objects in our world for object detection, structure and viewpoint estimation, and physical property estimation. Bootstrapping from these predictions, we then infer the remaining scene properties through inference via forward simulation of the physics engine.

Without human supervision, our system learns by *visual de-animation*: interpreting and reconstructing visual input. We show the problem formulation in Figure 1. The simulation and rendering engines in the framework force the perception module to extract physical world states that best explain the data. As the physical world states are inputs to physics and graphics engines, we simultaneously obtain an interpretable, disentangled, and compact physical scene representation.

Our framework is flexible and adaptable to a number of graphics and physics engines. We present model variants that use neural, differentiable physics engines [1], and variants that use traditional physics engines, which are more mature but non-differentiable [2]. We also explore various

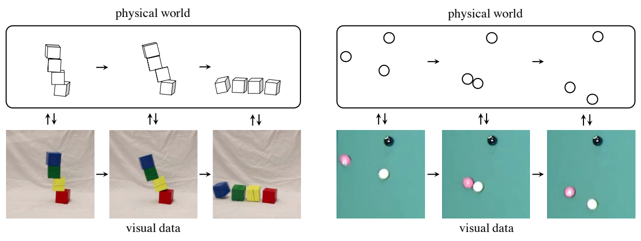


Figure 1: Visual de-animation — we would like to recover the physical world representation behind the visual input, and combine it with generative physics simulation and rendering engines.

graphics engines operating at different levels, ranging from mid-level cues such as object velocity, to pixel-level rendering of images.

We demonstrate our system on real and synthetic datasets across multiple domains: synthetic billiard videos [3], in which balls have varied physical properties, real billiard videos from the web, and real images of block towers from Facebook AI Research [5].

2. Visual De-animation

Our visual de-animation (VDA) model consists of an efficient inverse graphics component to build the initial physical world representation from visual input, a physics engine for physical reasoning of the scene, and a graphics engine for rendering videos. We show the framework in Figure 2. In this section, we first present an overview of the system, and then describe each component in detail.

The first component of our system is an approximate inverse graphics module for physical object and scene understanding, as shown in Figure 2-I. Specifically, the system sequentially computes object proposals, recognizes objects and estimates their physical state, and recovers the scene layout.

The second component of our system is a physics engine, which uses the physical scene representation recovered by the inverse graphics module to simulate future dynamics of the environment (Figure 2-II). Our system adapts to both neural, differentiable simulators, which can be jointly trained with the perception module, and rigid-body, non-differentiable simulators, which can be incorporated using methods such as REINFORCE [7].

The third component of our framework is a graphics engine (Figure 2-III), which takes the scene representations from the physics engine and re-renders the video at various levels (*e.g.* optical flow, raw pixel). The graphics engine may need additional appearance cues such as object shape or color (Figure 2c). Here, we approximate them using simple heuristics, as they are not a focus of our paper. There is a tradeoff between various rendering levels: while pixel-level reconstruction captures details of the scene, rendering at a more abstract level (*e.g.* silhouettes) may better generalize. We then use a likelihood function (Figure 2d) to evaluate the difference between synthesized and observed signals, and compute gradients or rewards for differentiable and non-differentiable systems, respectively.

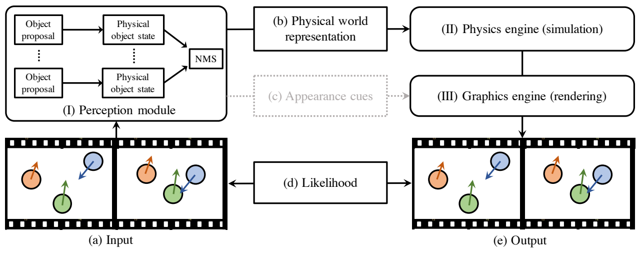


Figure 2: Our visual de-animation (VDA) model contains three major components: a convolutional perception module (I), a physics engine (II), and a graphics engine (III). The perception module efficiently inverts the graphics engine by inferring the physical object state for each segment proposal in input (a), and combines them to obtain a physical world representation (b). The generative physics and graphics engines then run forward to reconstruct the visual data (e). See Section 2 for details.

Our model combines efficient and powerful deep networks for recognition with rich simulation engines for forward prediction. This provides us two major advantages over existing methods: first, simulation engines take an interpretable representation of the physical world, and can thus easily generalize and supply rich physical predictions; second, the model learns by explaining the observations — it can be trained in a self-supervised manner without requiring human annotations.

3. Evaluation

We evaluate variants of our frameworks in two scenarios: synthetic billiard videos and real block towers.

3.1. Billiard Tables: A Motivating Example

We begin with synthetic billiard videos to explore end-to-end learning of the perceptual module along with differentiable simulation engines. We explore how our framework learns the physical object state (position, velocity, mass, and friction) from its appearance and/or motion.

Data For the billiard table scenario, we generate data using the released code from [3]. We updated the code to allow balls of different mass and friction. We used the billiard table scenario as an initial exploration of whether our models can learn to associate visual object appearance and motion with physical properties. As shown in Figure 3, we generated three subsets, in which balls may have shared or differing

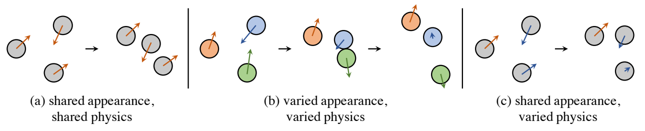

(a) shared appearance, shared physics
(b) varied appearance, varied physics
(c) shared appearance, varied physics

Figure 3: The three settings of our synthetic billiard videos: (a) balls have the same appearance and physical properties, where the system learns to discover them and simulate the dynamics; (b) balls have the same appearance but different physics, and the system learns their physics from motion; (c) balls have varied appearance and physics, and the system learns to associate appearance cues with underlying object states, even from a single image.

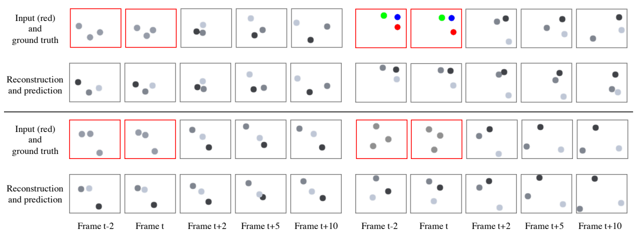

Input (red) and ground truth
Reconstruction and prediction
Input (red) and ground truth
Reconstruction and prediction
Frame t-2
Frame t
Frame t+2
Frame t+5
Frame t+10
Frame t-2
Frame t
Frame t+2
Frame t+5
Frame t+10

Figure 4: Results on the billiard videos, comparing ground truth videos with our predictions. We show two of three input frames (in red) due to space constraints. Left: balls share appearance and physics (I), where our framework learns to discover objects and simulate scene dynamics. Top right: balls have different appearance and physics (II), where our model learns to associate appearance with physics and simulate collisions. It learns that the green ball should move further than the heavier blue ball after the collision. Bottom right: balls share appearance but have different frictions (III), where our model learns to associate motion with friction. It realizes from three input frames that the right-most ball in the first frame has a large friction coefficient and will stop before the other balls.

appearance (color), and physical properties. For each case, we generated 9,000 videos for training and 200 for testing.

(I) Shared appearance and physics (Figure 3a): balls all have the same appearance and the same physical properties. This basic setup evaluates whether we can jointly learn an object (ball) discoverer and a physics engine for scene dynamics.

(II) Varied appearance and physics (Figure 3b): balls can be of three different masses (light, medium, heavy), and two different friction coefficients. Each of the six possible combinations is associated with a unique color (appearance). In this setup, the scene de-rendering component should be able to associate object appearance with its physical properties, even from a single image.

(III) Shared appearance, varied physics (Figure 3c): balls have the same appearance, but have one of two different friction coefficients. Here, the perceptual component should be able to associate object motion with its corresponding friction coefficients, from just a few input images.

Results Our formulation recovers a rich representation of the scene. With the generative models, we show results in scene reconstruction and future prediction. We compare two variants of our algorithm: the initial system has its perception module and neural physics engine separately trained, while the full system has an additional end-to-end fine-tuning step, as discussed above. We also compare with a baseline, which has the sample perception model, but in prediction, simply repeats object dynamics in the past without considering interactions among them.

Scene reconstruction: given input frames, we are able to reconstruct the images based on inferred physical states. For evaluation, we compute pixel-level MSE between reconstructions and observed images. We show qualitative results in Figure 4.

Future prediction: with the learned neural simulation engine, our system is able to predict future events based on physical world states. We show qualitative results in Figure 4. Our model achieves good performance in reconstructing the scene, understanding object physics, and predicting scene dynamics. See caption for details.

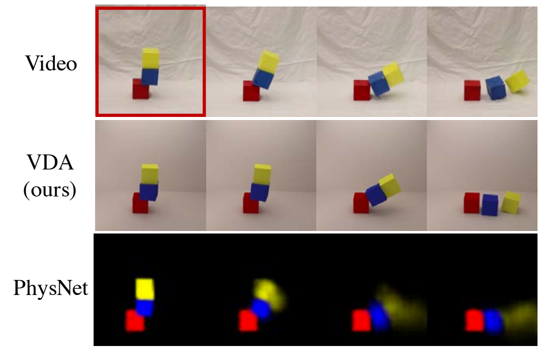
3.2. The Blocks World

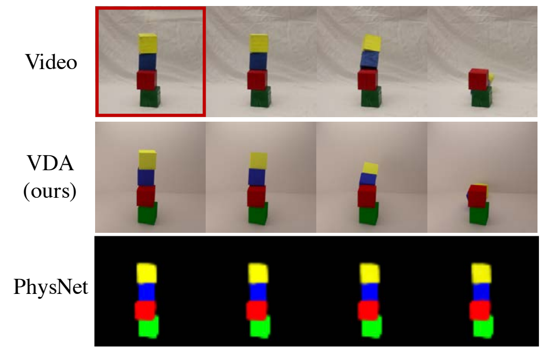
We now look into a different scenario — block towers.

Data [5] built a dataset of 492 images of real block towers, with ground truth stability values. Each image may contain 2, 3, or 4 blocks of red, blue, yellow, or green color. Though the blocks are the same size, their sizes in each 2D image differ due to 3D-to-2D perspective transformation. Objects are made of the same material and thus have identical mass and friction.

Results We show results on two tasks: scene reconstruction and stability prediction. For each task, we compare three variants of our algorithm: the initial system has its perception module trained without fine-tuning; an intermediate system has joint end-to-end fine-tuning, but without considering the physics constraint; and the full system considers both reconstruction and physical stability during fine-tuning.

We show qualitative results on scene reconstruction in Figures 5a and 5d, where we also demonstrate future prediction results by exporting our inferred physical states into Blender. We show quantitative results on stability prediction in Table 5b, where we compare our models with PhysNet [5] and GoogleNet [6]. All given a static image as test input, our algorithms achieve higher prediction accuracy (75% vs. 70%) efficiently (¡10 milliseconds per image).





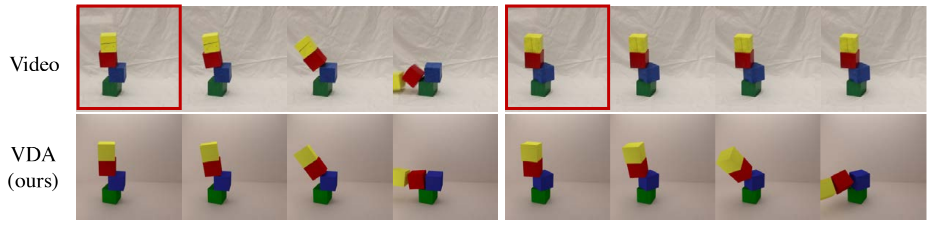
(a) Our reconstruction and prediction results given a single frame (marked in red). From top to bottom: ground truth, our results, results from [5].

Methods	# Blocks 2	3	4	Mean
Chance	50	50	50	50
Humans	67	62	62	64
PhysNet	66	66	73	68
GoogLeNet	70	70	70	70
VDA (init)	73	74	72	73
VDA (joint)	75	76	73	75
VDA (full)	76	76	74	75

(b) Accuracy (%) of stability prediction on the blocks dataset

Methods	2	3	4	Mean
PhysNet	56	68	70	65
GoogLeNet	70	67	71	69
VDA (init)	74	74	67	72
VDA (joint)	75	77	70	74
VDA (full)	76	76	72	75

(c) Accuracy (%) of stability prediction when trained on synthetic towers of 2 and 4 blocks, and tested on all block tower sizes.



(d) Our reconstruction and prediction results given a single frame (marked in red)

Figure 5: Results on the blocks dataset [5]. For quantitative results (b), we compare three variants of our visual de-animation (VDA) model: perceptual module trained without fine-tuning (init), joint fine-tuning with REINFORCE (joint), and full model considering stability constraint (full). We also compare with PhysNet [5] and GoogLeNet [6].

Our framework also generalizes well. We test out-of-sample generalization ability, where we train our model on 2- and 4-block towers, but test it on all tower sizes. We show results in Table 5c.

References

[1] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *ICLR*, 2017.

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

[3] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. In *ICLR*, 2016.

[4] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[5] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *ICML*, 2016.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[7] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *MLJ*, 8(3-4):229–256, 1992.