

# Unsupervised Intuitive Physics from Visual Observations

Sebastien Ehrhardt<sup>1\*</sup>[0000–0002–6948–079X], Aron  
Monszpart<sup>2,3\*</sup>[0000–0003–0579–5300], Niloy Mitra<sup>2</sup>[0000–0002–2597–0914], and  
Andrea Vedaldi<sup>1</sup>[0000–0003–1374–2858]

<sup>1</sup> University of Oxford, Oxford, UK, {hyenal, vedaldi}@robots.ox.ac.uk  
<sup>2</sup> University College London, London, UK, {a.monzpart,n.mittra}@cs.ucl.ac.uk  
<sup>3</sup> Niantic, San Francisco, USA, aron@nianticlabs.com

**Abstract.** While learning models of *intuitive physics* is an active area of research, current approaches fall short of natural intelligences in one important regard: they require external supervision, such as explicit access to physical states, at training and sometimes even at test time. Some approaches sidestep these requirements by building models on top of handcrafted physical simulators. In both cases, however, methods cannot learn automatically new physical environments and their laws as humans do. In this work, we successfully demonstrate, for the first time, learning unsupervised predictors of physical states, such as the position of objects in an environment, *directly from raw visual observations and without relying on simulators*. We do so in two steps: (i) we learn to track dynamically-salient objects in videos using causality and equivariance, two non-generative unsupervised learning principles that do not require manual or external supervision. (ii) we demonstrate that the extracted positions are sufficient to successfully train visual motion predictors that can take the underlying environment into account. We validate our predictors on synthetic datasets; then, we introduce a new dataset, ROLL4REAL, consisting of real objects rolling on complex terrains (pool table, elliptical bowl, and random height-field). We show that it is possible to learn reliable object trajectory extrapolators from raw videos alone, without any external supervision and with no more prior knowledge than the choice of a convolutional neural network architecture.

**Keywords:** Unsupervised learning · Motion · Convolution Networks.

## 1 Introduction

A striking property of natural intelligences is their ability to perform accurate and rapid predictions of physical phenomena using only noisy sensory inputs. Even more remarkable is the fact that such predictors are learned without explicit supervision; rather, natural intelligences induce their internal representation of physics automatically from experience.

---

\* Authors contributed equally

Several authors have recently looked into the problem of learning physical predictors using deep neural networks in order to partially mimic this functionality. Early attempts predicted trajectories in hand-crafted spaces of physical parameters, such as positions and velocities, assuming that the ground-truth values of such parameters are fully observable during training. Others have considered performing predictions from visual observations, but used full supervision for training. Furthermore, while several papers [7,3] make use of simulators as a way to generate the required supervisory signals, limited work has been done in transferring such models to real data.

In this paper, we also investigate learning *physical predictors* using deep neural network. However, we do so in a **fully unsupervised manner**, learning from observations of unlabelled video sequences. In contrast to approaches such as the recent de-animation method of [39], we do not require synthetic data, nor do we rely on any handcrafted physical simulator for prediction. Our models are built directly from real data and learn intuitive physics models that empirically outperform more principled, but more brittle, models based on physical parameters [31].

Importantly, our goal is not to merely predict future frames in a video, a problem addressed before by several authors [19]. While we also predict future dynamics from a video stream, our goal is *not* to estimate appearance changes, but physical quantities such as object positions and velocities. So, where future frame prediction generates an image, our goal is to extract meaningful and actionable physical parameters from the data.

As a working example, we consider video footage of balls rolling on various surfaces, such as pool tables, bowls and random height-fields. Balls interact with the underlying environment (e.g., roll around obstacles) and among themselves (e.g., collide with each other). For rigorous assessment, in addition to considering several synthetic datasets, we also contribute a **new public dataset**, ROLL4REAL, containing a large number of such sequences captured in real-life. Methodologically, we make two contributions. First, inspired by [23], we show that an object **detector** can be learned in an **unsupervised manner** by tuning a convolutional detector to extract tracks that are maximally characteristic of the natural, causal ordering of the frames in a video. Second, we use these trajectories to learn **visual predictors** that automatically learn an internal representation of physics and can extrapolate the trajectory of the balls more reliably than even supervised approaches such as Interaction Networks (IN) [3] that use direct measurements of physical parameters.

Note that our goal, similar to other papers in this area, is not to come up with the best possible method for physical prediction. A handcrafted solution heavily engineered to use supervision, off-the-shelf trackers, and/or physical simulators may do better in raw predictive performance (although the task is in fact not simple, particularly as our terrains are complex and somewhat deformable). Rather, we focus on developing machines that can learn such physical predictors from raw input.

Empirically, we show that vision-based models more gracefully handle observation noise compared to approaches such as [7,3] that are learned using physical ground-truth parameters extracted from simulated scenarios. We also show that the Visual Interaction Network (VIN) of [22], which also propose a vision-based physical predictor, fails to account for the interaction of the objects and their environment, whereas more distributed tensor based approach succeeds.

The rest of the paper is organized as follows. We discuss related work in section 2. We then present the technical details of our approach in section 3. Next, we introduce the new ROLL4REAL data in section 4 and use the latter as well as several existing synthetic datasets to evaluate the approach in section 5. We summarise our findings in section 6.

## 2 Related Work

Existing work in learning physics can be organised according to several axes.

**Nature of the Representation of Physics:** A natural way to represent physics is to manually encode every object parameters and physical properties (mass, velocity, positions, etc.). From the earliest approaches [4] this has been widely used to represent physics and propagate it [3,7,26,32]. Some focusing on representing a small subset of physical parameters such as positions and velocities [37,38]. However, other approaches try to learn an implicit representation of physics, inspired by the success of implicit representation of dynamics [29,28,8,5]. Implicit physics are usually represented as activations in a deep neural network [10,36,20].

**Hand-crafted vs Learned Dynamics:** Some approaches [37], including simulation-based ones [4,40], use physics by explicitly integrating parameters such as velocities. While this generally require extensive knowledge of the environment and object properties, other methods [3,7,26,32,10], integrate parameters of the scenarios through recurrent learnable predictors to make physical long term predictions.

**Physical vs Visual Observations:** Many approaches [3,7] assume direct access to physical quantities such as positions and velocities for prediction. If this first approach enable to make very accurate predictions it is however unlikely that such accuracy can be reached in the real-world. Others [4,20,21,40,36,13,33,17] take as input one or several frames of a scene to deduce physical properties (intuitive or explicit) or predict the next state of a system.

**Qualitative vs Quantitative Predictions:** While most of the papers discussed above consider *quantitative* predictions such as extrapolating trajectory, others have considered *qualitative* predictions focusing on *intuitive* physics, such as the stability of stacks of objects [4,20,21], the likelihood of a scenario [30] or the forces acting behind a scene [40]. Other papers are in between, and learn *plausible if not accurate physical predictions* [35,18,24], often for 3D computer graphics.

**Nature of the Supervision:** Most approaches are *passive and supervised*, as they are passive observer of physical scenarios and use ground truth information

about key physical parameters (positions, velocities, stability) during training. While this approaches require an expensive annotation of data, some work tried to learn from unsupervised data either through active manipulation [2,9] or using the laws of physics [33].

**Scenarios:** Two favorite scenarios in such experiments are bouncing balls, including billiard-like environment [14], and block towers [20]. As a variant, [36] consider balls subject to gravitational pulls, ignoring harder-to-model collisions. Most papers make use of simulated data, with limited validation on real data. A different approach [25] is to predict qualitative object forces and trajectories in fully-unconstrained natural images. The approach of [2] considers instead learning from active poking using a real-life robot. In most cases experiments are done on synthetic data. However, approaches such as [37,38,21] also used real data; [38] also contributed a dataset of videos of short physical experiments called *Phys-101*.

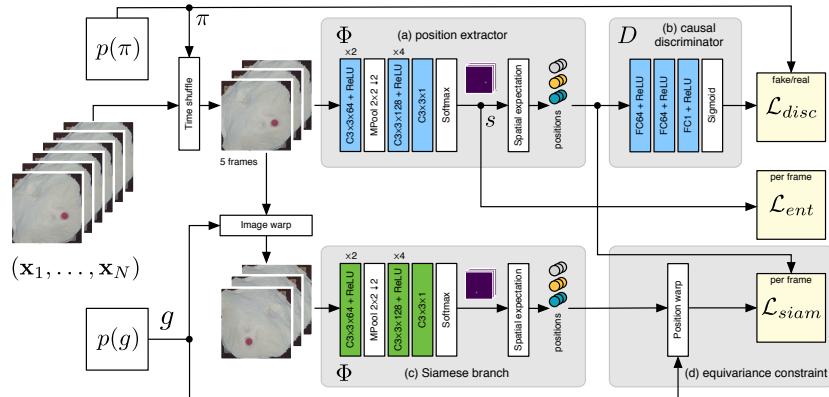
We relate to such previous work in that we also make physical predictions of the trajectory of ball-like objects. However, we differ in two significant ways. First, our approach, while using only passive observations, is *fully unsupervised*, and yet competitive if not more accurate than supervised counterparts. In particular, while [33,40] also do not use image labels, they use *a-priori* knowledge of physics for training (a fully-fledged simulator and renderer in the case of [40]). Second, we systematically test on *several real-life scenarios*, both in training and testing, using our new ROLL4REAL dataset. Compared to datasets such as *Phys-101*, ours allows testing long-term ball-rolling prediction in complex scenarios.

### 3 Method

Our goal is to construct a machine that can, given only raw videos and no supervision, learn physical parameters such as the position of the objects in the videos as well as proxies to physical laws that allow to predict the evolution of such parameters over time. For this, predicting appearance changes is not sufficient; instead, we decompose the problem in two steps. The first one is a method to discover and learn to extract object positions using as cue the fact that they should have a non-trivial causal dynamics (section 3.1). This tracker scales well to large datasets and is able to detect different type of objects without any further specification. Then, we use the resulting object trajectories to learn visual predictors that can extrapolate the object positions through time, embodying a proxy to the laws of mechanics (section 3.2).

#### 3.1 Unsupervised Detection and Tracking of Dynamic Objects

**Single-object Detection.** Let  $\mathbf{x}_t \in \mathbb{R}^{H \times W \times 3}$  be a RGB video frame and assume we are given video sequences  $\mathcal{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ , initially containing a single object moving in an environment, such as a rolling ball. Our goal is to learn a detector function  $\Phi(\mathbf{x}_t) = u_t \in \mathbb{R}^2$  that extracts the 2D position  $u_t$  of the



**Fig. 1. Overview of our unsupervised object tracker** Each training point consists of a sequence of five video frames. Top: the sequence is randomly permuted with probability 50%. The position extractor (a) computes a probability map  $s$  for the object location, whose entropy is penalised by  $\mathcal{L}_{ent}$ . The reconstructed trajectory is then fed to a causal/non-causal discriminator network (b) that determines whether the sequence is causal or not, encouraged by  $\mathcal{L}_{disc}$ . The bottom Siamese branch (c) of the architecture takes a randomly warped version of the video and is expected by  $\mathcal{L}_{siam}$  to extract correspondingly-warped positions in (d). Blue and green blocks contain learnable weights and green blocks are siamese shared ones. At test time only  $\Phi$  is retained.

moving object at any given time (Fig. 1). The challenge is to do so *without* access to any label for supervision or any a-priori information about object shape.

We start by implementing  $\Phi(\mathbf{x}_t)$  as a shallow Convolutional Neural Network (CNN) that extracts a scalar score  $f_v \in \mathbb{R}$  for each image pixel  $v \in \Omega = \{1, \dots, H\} \times \{1, \dots, W\}$ , resulting in a heat map. This is then normalised to a probability distribution using the softmax operator  $s_v = e^{f_v} / \sum_{z \in \Omega} e^{f_z}$  and the location  $u$  of the object is obtained as the expected value  $u = \sum_v v s_v$  [13].

We learn  $\Phi$  by combining two learning principles. The first one is **causality**. Applied to a video sequence, the detector produces a trajectory  $\Phi(\mathcal{X}) = (\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_N))$ . We expect that, when the detector locks properly on the rolling object, the trajectory is physically plausible (e.g., causal/smooth); at the same time, if the frames are shuffled by a random permutation  $\pi$ , the resulting trajectory should *not* be plausible anymore. We incorporate this constraint by learning a discriminator network  $D(\Phi(\mathbf{x}_{\pi_1}), \dots, \Phi(\mathbf{x}_{\pi_5}))$  that, for a subsequence, can distinguish between the natural ordering of the frames and a random shuffle (top row of Fig. 1). The permutation  $\pi$  is sampled with 50% probability as a consecutive sequence of 5 frames ( $\pi_{i+1} = \pi_i + 1$ ,  $i = 1, \dots, 4$ ) and with 50% uniformly at random. The discriminator is a 3 layers multi-layer perceptron followed by a sigmoid and the loss

$$\mathcal{L}_{disc}(D, \pi) = \begin{cases} -\log D, & \pi_{i+1} = \pi_i + 1, \quad i = 1, \dots, 4, \\ -\log(1 - D), & \text{otherwise.} \end{cases}$$

The second learning principle is **equivariance** (cf., [34,27]). This principle suggests that, if a transformation  $g$  is applied to a frame  $\mathbf{x}_t$  (e.g., a  $\pm\pi/2$  rotation), then the output of the detector should change accordingly:  $\Phi(g\mathbf{x}_t) = g\Phi(\mathbf{x}_t)$ . This is implemented as a Siamese branch (bottom row in Fig. 1) extracting 2D positions  $\Phi(g\mathcal{X}) = (\Phi(g\mathbf{x}_1), \dots, \Phi(g\mathbf{x}_N))$  from the rotated frames and comparing them to the rotated 2D positions extracted from the original frames using the  $L^2$  loss:  $\mathcal{L}_{siam} = \frac{1}{N} \sum_t \|g^{-1}\Phi(g\mathbf{x}_t) - \Phi(\mathbf{x}_t)\|^2$ .

Finally, in order to encourage the softmax operator to produce peaky distributions, we minimise the entropy of the resulting distribution  $\mathcal{L}_{ent} = -\sum_{v \in \Omega} s_v \log(s_v)$ . The final loss is therefore  $\mathcal{L} = \lambda_d \mathcal{L}_{disc} + \lambda_e \mathcal{L}_{ent} + \lambda_s \mathcal{L}_{siam}$ . In our experiment,  $\lambda_d = 1$ ,  $\lambda_e = 0.01$ , and  $\lambda_s = 0.001$ .

**Multi-object Tracking.** We now extend the method from detection of single objects to tracking of multiple objects. In order to do so, the network is finetuned to videos containing two or more moving objects of different appearance.

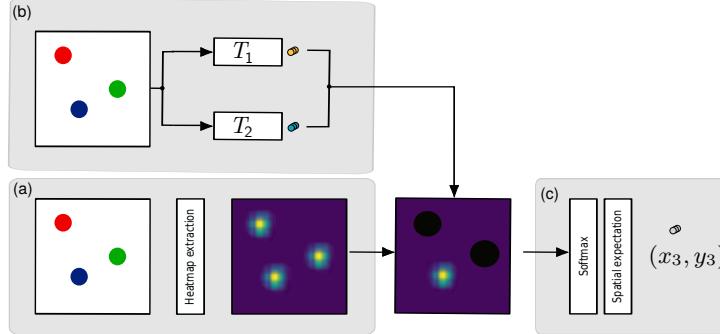
Since the network produces only a single pair of coordinates, it can formally estimate the location of a single object in the image. However, when multiple objects are present, the unsupervised learning process could still converge to an undesirable result, such as predicting the center of mass of several objects combined, or randomly jumping between objects over time. The first is discouraged by the entropy loss which prefers sharp heat map. The second is discouraged by the causality loss, as discontinuous trajectories would not look plausibly ordered and consistent.

In practice, our model learns to track consistently a single object selected at random among the visible ones. Once this is done, in the next iteration, a second object is detected by suppressing (setting to zero) a circular region of radius  $r$  around the first object location in the activations  $f_v$  immediately preceding the softmax operator, and the process is repeated for further object occurrences. Before the suppression we also add a positive bias to the activations  $f_v$  in order to consider the previously detected objects as zero probabilities in the new heatmap. Note that we consider the number of objects as given since it is in itself already a very challenging task that is under active research [12].

### 3.2 Trajectory Extrapolation Networks

We consider existing network modules for physical prediction. While these modules use external supervision in the original papers, here we apply them to the output of the unsupervised tracker of 3.1, hence training such physical extrapolators in a *fully unsupervised* manner for the first time.

We experiment in particular with *PosNet*, *DispNet*, and *ProbNet* from [11], configuring them to take as input the first four frames of a sequence and to produce as output the prediction of future object positions. These models learn an implicit representation of physics, which is extrapolated automatically by a recurrent propagation layer and used to extract estimates of the object positions. The difference between the models is that *PosNet* regresses positions from state, while *DispNet* and *ProbNet* regress displacements from state. Furthermore, *ProbNet* produces a probability estimate over trajectories.



**Fig. 2. Multiple object unsupervised tracker** (a) We first extract an object heatmap with the method described in 3.1. (b) Then we mask the objects detected by previously trained tracker ( $T_1$  and  $T_2$ ) on the heatmap by zeroing out the values around a circular area around their center. (c) Finally we extract position from this last heatmap with masked values.

We also consider the *Visual Interaction Network* (VIN) module and its variant *Interaction Network from State* (IFS) [36]. While VIN uses only visual inputs for prediction just like the other networks, IFS works with an explicit state vector of physical parameters, which we set as the stacking of the 2D positions for four past frames which starts with positions extracted from our tracker. Additionally, in the synthetic experiments (first part of Table 2), IFS uses velocity and in BOWLS experiments the ground-truth ellipsoid axes parameters are appended to the state to inform the model of the shape of the ground. IFS and VIN are trained following [11]; in particular, this means that VIN uses the same setting as the original paper ( $32 \times 32$  pixels images).

We also note that while VIN and models from [11] have essentially the same core concepts (they consist of a first feature extractor module to extract implicit physical state, a recurrent propagation module to propagate the state, and an extractor module to get desired physical parameters from the state) their main difference resides in the structure of the propagated state. While VIN used a vector state representation, each of *PosNet*, *DispNet*, and *ProbNet* use a tensor representation.

All such models are trained by showing the network four initial frames of a sequence and the output of the unsupervised tracker up to time  $T_{\text{train}} \in \{15, 20\}$  frames. At test time, the networks, which are recurrent, are used to extrapolate the trajectory up to an arbitrary time  $T$ , also starting from four video frames. We test in particular  $T = T_{\text{train}}$  and  $T \gg T_{\text{train}}$  to assess the generalization capabilities of the models learned by the network.

In addition, for some experiments on single object we also consider *linear* and *quadratic* extrapolators as baselines. In both cases we fit a first (respectively second) order polynomial to the 10 first positions given as input (hence with a significant advantage compared to the networks which only observe four frames).



**Fig. 3. Physical setup** In each of the three real-world scenarios (POOLR, BOWLR, HEIGHTR), we show the experimental setup (left) and a sample data frame (right).

#### 4 ROLL4REAL: A New Benchmark Dataset

In the absence of a suitable real-world dataset to evaluate intuitive physics on objects rolling on complex terrains, we created a new benchmark, ROLL4REAL (R4R).

**Dataset Content.** R4R consists of 1118 short  $256 \times 256$  videos containing one or two balls rolling on three types of terrains (Fig. 3): a flat pool table (POOLR), a large ellipsoidal ‘bowl’ (BOWLR), and an irregular height-field (HEIGHTR). More specifically, there are 151 videos (avg. 99 frames/video) for the POOLR dataset with one ball; 216 videos (522 frames/video) for the BOWLR dataset with one ball; 543 videos (356 frames/video) for the HEIGHTR dataset with one ball; and 208 videos (206 frames/video) for the HEIGHTR dataset with two balls. We rolled a total of 7 differently colored balls for the HEIGHTR and BOWLR datasets, varying from 3.5 cm to 7 cm in diameter. The height-field surface fits into a  $70 \times 70 \times 28$  cm<sup>3</sup> bounding box, with 76 cm diameter. The bowl was created using a 70 cm diameter ball, and is 60 cm high. Videos were randomly split into *train*, *validation*, and *test* sets. Ground-truth annotations are provided for the test split.

**Dataset Collection.** Both the bowl and height-field terrains were modeled using paper mâché on scaffolds, using a large inflatable ball and a custom-made wire-mesh frame, respectively. For the the POOLR dataset, balls were rolled on the table, while for the other settings, balls were manually dropped from a small height and allowed to roll on. The setup was imaged using a fixed camera (Samsung Galaxy S8) from the top. The POOLR dataset was captured at 30fps (due to low light), while all the others at 240fps in order to reduce motion blur and later downsampled to 80fps. Videos were cropped to only focus on the scenario of interest, i.e., ball(s) and terrain, and trimmed to retain the portion of the video containing motion. We rolled a total of 7 different balls: a pink foam ball (7 cm diameter), a fluorescent yellow tennis ball (6.8 cm), a blue and an orange ping-pong ball (4 cm), a black squash ball wit two yellow dots (4 cm), and a green and a brown cork ball (3.5 cm).

In order to create ground-truth tracks for the ball centers, we used a template-based tracker using zero-normalized cross-correlation in the LAB color space, and tracked each frame along with a smoothness term over time. The setup was manually initiated by providing suitable template. The raw results were then manually inspected, corrected, and saved as ground-truth. We found that due to environment jitter (the ball rolling on the different terrains often created vibration or deformation in the BOWLR and HEIGHTR datasets), differences in

lighting across some experiments and different ball colors, the template-based tracker was not perfect and manual inspection was required.

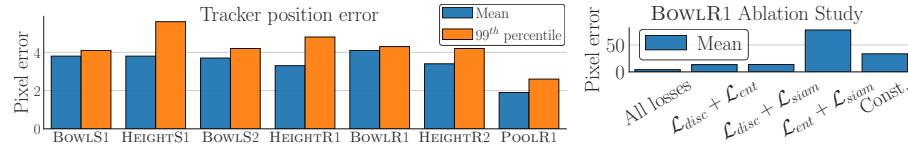
It is worth noting that, while this process was enough to produce ground truth annotations for the test set, the method does not scale due to the need for manual verification and correction. While our aim here is to show the feasibility of learning physics in an unsupervised manner, such problems show that our deep tracker also has an applicative advantage compared to these traditional handcrafted approaches.

## 5 Results and Discussion

**Implementation Details** For all networks trained on every dataset, weights were initialised using Xavier initialization [15]. The learning rate was initially set to  $10^{-4}$  and was progressively decreased by a factor of 10 when no improvements were found over  $K$  epochs ( $K = 100$  for the synthetic datasets). Training was stopped when the loss did not decrease for  $2K$  consecutive epochs. Before processing images, we resized all dataset images to  $128 \times 128$  pixels to fit in the GPU memory. We used TensorFlow [1] on a single NVIDIA Titan X GPU for all the experiments.

### 5.1 Unsupervised Tracker

We first evaluate our unsupervised object detector and tracker and compare against currently state-of-the-art trackers. We report results in Table 1 against the following trackers: 1. Optical Flow Lucas-Kanade (OFLK) from OpenCV[6] library; 2a. Flownet2-simple, which computes pairwise flowfields using FlowNet2 [16] and follows the velocity vectors; 2b. Flownet2-blob, where we after computing the flowfields from FlowNet2 [16], update the positions as the center of the blobs found in the flowfield. If no blob was detected, we updated the position according to 2a; 3. LAB: a template tracker similar to section 4 without any manual corrections. Note that these methods need manual initialization at the objects positions (expect for LAB) or templating which needs more work with growing object count and/or variety. In addition to POOLR, BOWLR, and HEIGHTR from ROLL4REAL, we also consider two synthetic datasets from [11] in Fig. 4: BOWLS for the ellipsoidal bowl with one or two balls and HEIGHTS for the random height-fields. Fig. 4-left reports the mean and 99<sup>th</sup> percentile pixel error of the extracted object positions against ground-truth averaged over multiple runs of our experiments. Even though the trackers perform well in practice, they suffer from *large variance*. For example, OFLK went off-track 15% of the time on the BOWLR dataset, 10% for the HEIGHTR, and 30% for POOLR. In contrast, ours never loses track of the object. The 99th percentile reported in Fig. 4 shows that the offset is almost constant generally due to the detection occurring on the edge of the objects. Overall, our method learns to track objects robustly in a diverse range of complex scenarios.



**Fig. 4. Tracker errors and Ablation study** Left: Tracker errors on different dataset. The errors are consistently small across dataset and show that our tracker can perform well on a different range of real situations. Right: Ablation study. We try different combination of tracker losses on the BowLR dataset. ‘Const.’ indicates that we are predicting a constant point at the center of the image for reference. For left and right, position errors are reported in pixels. The number of balls in the datasets is appended to the name of the dataset.

**Table 1. Tracker results across real datasets** The reported numbers are the average (left) and the variance (right) of the pixel error. All numbers refer to  $128 \times 128$  images.

	POOLR	BowLR	HEIGHTR	HEIGHTR 2B.	
1. Optical Flow Lucas-Kanade	23.3	965	5.6	275	<b>2.7</b>
2a. FlowNet2-simple	41.4	767	30.4	715	16.6
2b. FlowNet2-blob	3.9	12.1	<b>2.2</b>	4.8	4.6
3. LAB w/o manual correction	<b>0.3</b>	<b>0.1</b>	16.4	247	8.3
4. Ours	1.9	0.2	4.1	<b>0.5</b>	3.3
					102
					<b>1.2</b>

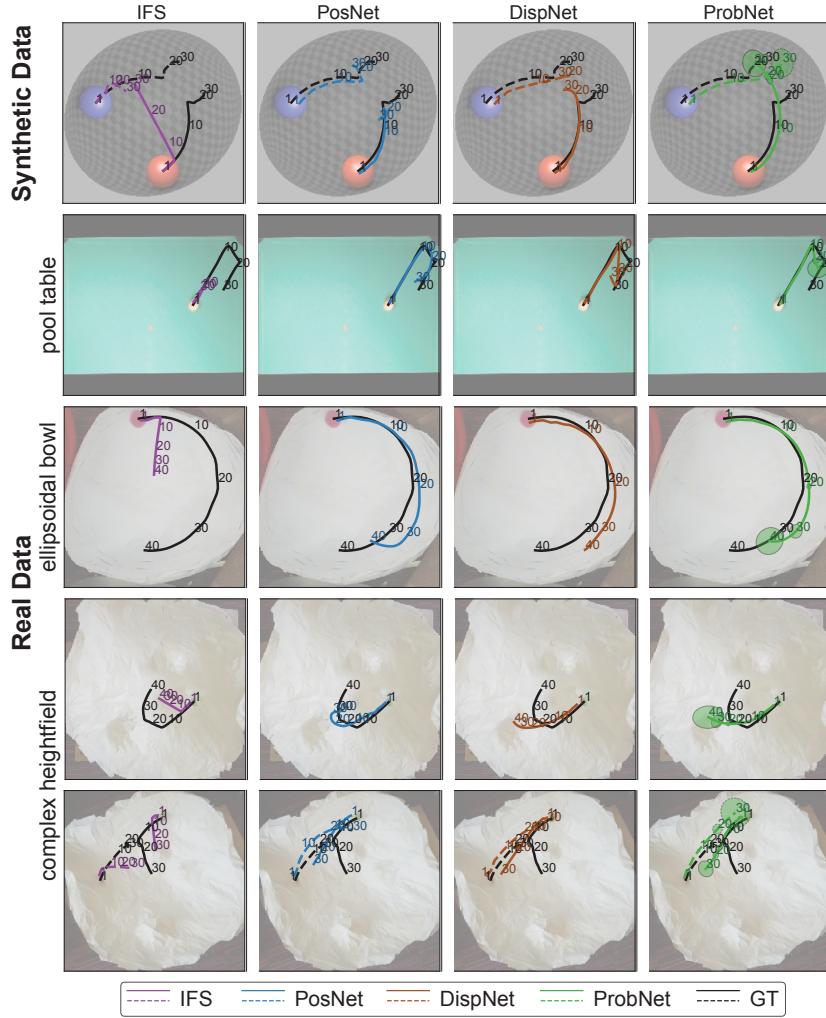
Importantly, since our tracker does not use any manual annotations it scales easily to larger synthetic datasets, multiple objects, and different object appearances within the same dataset by just providing more example data.

We also conducted an ablation study on the BowLR dataset to measure the impact of each loss term. Fig. 4-right shows that, while each loss contributes to the final results, the best performance is obtained when all the terms are combined.

## 5.2 Unsupervised Physics Extrapolation

**Supervised vs Unsupervised (Single Ball Synthetic Datasets).** We now compare training predictors using either ground-truth object positions or the output of the unsupervised tracker. All predictors observe only  $T_0 = 4$  frames as input (either positions or video frames) except VIN which uses  $T_0 = 6$  and the least squares baselines which use  $T_0 = 10$ . All the networks were trained to predict  $T_{train}$  positions. Table 3 reports the average errors at time  $T_{train}$  and  $2T_{train}$  to measure the ability of predictors to generalise beyond the training regime.

We see that the *Net* models (*ProbNet*, *DispNet*, *PosNet*) perform well using ground-truth positions or the unsupervised tracker outputs (e.g. *PosNet* error for BOWLS/HEIGHTS is 2.9/6.4 supervised vs 4.9/6.9 unsupervised), whereas IFS does not handle the transition well (3.3/10.4 to 13.3/23.1) and Linear, Quadratic



**Fig. 5. Qualitative performance comparison for the various methods against ground-truth trajectories** Top-to-bottom: two balls colliding on an ellipsoidal bowl; single ball colliding against the walls of a pool table; single ball rolling on an ellipsoidal bowl; single ball rolling on complex height-field; and two balls rolling on complex height-field. The top row is on synthetic data, while the other rows are on real-data. The green ellipsoids in the last column show the variance of the predictions estimated by *ProbNet* at selected locations.

and VIN are not competitive. The latest result shows a clear advantage of tensor-based state representations compared to vector based one. This suggests that modelling objects positions is done better by a representation which is spatially distributed. IFS also seems very sensitive to defects in the supplied annotations,

**Table 2. Long term predictions compared on synthetic datasets with model trained with ground-truth from simulator** All the models (except VIN, Linear, and Quadratic) are given  $T_0 = 4$  frames as input and train to predict first  $T_{train}$  positions. We report the average pixel error and perplexity for *PosNet* model at two different times. Perplexity, shown in bracket, is defined as  $2^{-\mathbb{E}[\log_2(p(x))]}$  where  $p$  is the estimated posterior distribution. *State* shows either the carried forward state is a physical quantity (Exp.), or an implicit vector or tensor (Imp.)

			BOWLS- $T_{train} = 20$		HEIGHTS- $T_{train} = 20$	
			$T = T_{train}$	$2 \times T_{train}$	$T_{train}$	$2 \times T_{train}$
Method	Input	State	With positions from simulator			
Linear	2D pos.	Exp.	61.9	20.1	21.3	61.9
Quadratic	2D pos.	Exp.	11.7	93.1	26.7	126.0
IFS	2D pos.	Exp.	3.3	<b>8.9</b>	10.4	27.6
VIN	Visual	Imp.	24.0	30.2	42.6	42.7
<i>PosNet</i>	Visual	Imp.	<b>1.6</b>	24.4	7.2	24.6
<i>DispNet</i>	Visual	Imp.	2.5	20.6	7.7	25.8
<i>ProbNet</i>	Visual	Imp.	2.9 (32.1)	21.8 (54.0)	<b>6.4 (9.5 )</b>	<b>22.5 (12.7)</b>
Method	Input	State	With positions from <i>unsupervised</i> tracker			
IFS	2D pos.	Exp.	13.3	<b>23.6</b>	23.1	38.3
VIN	Visual	Imp.	24.7	30.3	45.8	48.0
<i>PosNet</i>	Visual	Imp.	4.3	29.9	<b>6.6</b>	25.6
<i>DispNet</i>	Visual	Imp.	<b>3.9</b>	25.6	6.8	<b>22.7</b>
<i>ProbNet</i>	Visual	Imp.	4.9 (6.3)	27.0 (20.6)	6.9 (8.3 )	23.3 (13.4)

**Table 3. Long term predictions using one ball and real data** The table has the same format as Table 2. All models are trained using the unsupervised tracker, input and state are the same as Table 2, and we report pixel error (perplexity) at  $T$ .

Method	POOLR- $T_{train} = 15$		HEIGHTR- $T_{train} = 20$		BowlR- $T_{train} = 20$	
	$T = T_{train}$	$2 \times T_{train}$	$T_{train}$	$2 \times T_{train}$	$T_{train}$	$2 \times T_{train}$
IFS	26.0	37.5	48.0	58.1	26.2	39.1
VIN	50.9	40.8	40.2	47.3	33.9	33.0
<i>PosNet</i>	4.6	21.4	<b>5.6</b>	29.0	<b>5.6</b>	23.0
<i>DispNet</i>	<b>3.8</b>	23.6	<b>5.6</b>	<b>28.5</b>	6.5	<b>22.6</b>
<i>ProbNet</i>	4.7(6.3)	<b>16.3</b> (11.3)	5.7(5.8)	30.0(22.5)	6.8(6.8)	23.5(13.8)

since its knowledge of the environment is very limited, error correction is very challenging for it.

The main weakness of the *Net* models is that their performance degrades as prediction extends beyond the training horizon  $2T_{train}$ , whereas IFS generalizes more. At least *ProbNet* explicitly indicates that the model is uncertain when this occurs.

**Synthetic vs Real (One Ball Datasets).** On real datasets (Table 3), the *Net* models uniformly outperform others at both  $T_{train}$  and  $2T_{train}$ , with errors comparable to the synthetic case. Note that the real datasets in ROLL4REAL are particularly challenging due to the non-idealities of the surface (e.g. the BowlR surface is slightly elastic and wobbles as the ball rolls).

**Table 4. Long term predictions using two balls on real and synthetic data**  
Table layout and measures are the same as Table 2. Models are trained with positions from tracker, input and state are the same as Table 2, and we report pixel error (perplexity) at  $T$ .

Method	BOWLS 2b.- $T_{train} = 15$		HEIGHTR 2b.- $T_{train} = 15$	
	$T = T_{train}$	$2 \times T_{train}$	$T_{train}$	$2 \times T_{train}$
IFS	18.4	30.0	15.6	26.6
VIN	41.3	45.8	45.9	39.8
<i>PosNet</i>	<b>5.0</b>	<b>13.4</b>	<b>5.4</b>	<b>12.5</b>
<i>DispNet</i>	5.5	24.7	6.2	15.4
<i>ProbNet</i>	5.6 (7.3)	20.6 (13.7)	6.8 (7.9)	16.9 (12.4)

**One vs Multiple Balls (Real and Synthetic Datasets).** Finally, we move to cases where the balls are interacting with the environment and with each others due to collisions. This is particularly challenging when no ground-truth is used as multiple object tracking is much harder to achieve in an unsupervised setting than tracking a single object.

As shown in Table 4, the *Net* models still perform well. Due to memory limitations, models were trained for a slightly shorter time span  $T_{train}$ ; since the corresponding predictions are shorter term, their errors are a little lower than before. Overall, the results show that neither perfect ground-truth annotations nor a very large dataset is required to train a reliable physical extrapolator. Still, we noticed that collisions were difficult to predict in the HEIGHTR dataset (see the bottom row of Fig. 5), probably because such events are rare during training. In contrast, this seems to be much better handled by the models in the synthetic dataset (First row of Fig. 5).

### 5.3 Unsupervised Physics Interpolation

As in [11], we also study the interpolation problem considering their *InterpNet* configuration. We compare the latter to the extrapolation network *DispNet* trained over a longer horizon  $T_{train} = \{30, 40\}$ . *InterpNet* has the same architecture has *DispNet* with the difference that, in addition to the first  $T_0$  frames of the sequence, *InterpNet* additionally takes as input the last video frame as well. The first extracted state is used to regress the first  $T_0$  positions as well as the positions at time  $T_{train}$ , so that this state is explicitly encouraged to encode information about the last position of the object as well. In Table 5 and Table 6 we see that *InterpNet* managed to reduce the error in most cases. However in this case, compared to results in [11] *InterpNet* performs poorer on synthetic dataset and estimation of the intermediate states seems to be more challenging. Our interpretation is that the imperfect nature of the training data creates several possible path that this model is unable to solve. Finally we also noticed that the heightfield datasets seem to be very challenging as training for longer horizons didn't reduce the error as much as it does on the 'bowl.'

**Table 5. Extrapolation vs interpolation: one ball datasets** One ball datasets synthetic and real. Models are trained with positions from tracker. Pixel error at different time  $T$ .

$T$	POOLR $T_{\text{train}} = 30$			BOWLS $T_{\text{train}} = 40$				HEIGHTS $T_{\text{train}} = 40$				BOWLR $T_{\text{train}} = 40$				HEIGHTR $T_{\text{train}} = 40$			
	10	20	30	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40
<i>DispNet</i>	3.1	5.6	10.1	3.8	4.0	4.2	4.2	5.2	8.2	13.2	19.1	4.1	5.0	5.5	6.9	4.3	6.6	9.4	12.7
<i>InterpNet</i>	4.5	5.6	3.1	3.8	4.2	4.0	3.8	4.5	6.4	6.5	4.2	6.3	6.5	4.8	3.7	4.0	5.0	4.8	4.3

**Table 6. Extrapolation vs interpolation: two balls datasets** Two balls datasets synthetic and real. Models are trained with positions from tracker. Pixel error at different time  $T$ .

$T$	BOWLS 2b $T_{\text{train}} = 30$			HEIGHTR 2b $T_{\text{train}} = 30$		
	10	20	30	10	20	30
<i>DispNet</i>	4.3	6.9	9.7	5.2	8.9	13.4
<i>InterpNet</i>	4.2	5.0	4.1	6.5	6.9	7.6

## 6 Conclusions

We presented a method that can learn to track physical objects such as balls rolling on complex terrains using only raw video sequences and no supervision. Combined with recent neural networks that can learn an implicit representation of physics, such a system is able to extrapolate object trajectories over time while accounting for object-environment and object-object interactions. To the best of our knowledge, this is the first time that learning long-term physics extrapolation without access to supervision or handcrafted simulators has been demonstrated. Through an extensive benchmark we also demonstrated the superiority of tensor-based state representation that were able to produce satisfactory results on real data without the need of large datasets.

We also contributed a new dataset, ROLL4REAL, of real-life video sequences for complex scenarios such as ball rollings on pool tables, bowls, and height-field, showing that all such methods are applicable to the real world. This data will be made publicly available.

In this work we used different colored objects to make them distinguishable, which in practice is one of the main limitation of our work. We plan to address this issue by using same colored objects and build a tracker that would be trained to detect all objects at once removing the need for iterative training.

Finally, we also plan to train the tracker and the extrapolator end-to-end, further improving tracking of multiple objects. We also aim at improving the generalisation of the predictors beyond the training regime; we believe that the key is to factor knowledge about the environment and the object dynamics to allow the models to remember the first better over longer time spans.

**Acknowledgements.** The authors would like to gratefully acknowledge the support of ERC 677195-IDIU and ERC SmartGeometry StG-2013-335373 grants.

## References

1. Abadi, et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org
2. Agrawal, P., et al.: Learning to Poke by Poking: Experiential Learning of Intuitive Physics. In: Proc. NIPS. pp. 5074–5082 (2016)
3. Battaglia, P., et al.: Interaction networks for learning about objects, relations and physics. In: Proc. NIPS. pp. 4502–4510 (2016)
4. Battaglia, P., Hamrick, J., Tenenbaum, J.: Simulation as an engine of physical scene understanding. PNAS **110**(45), 18327–18332 (2013)
5. Bhattacharyya, A., et al.: Long-term image boundary prediction. In: Thirty-Second AAAI Conference on Artificial Intelligence. AAAI (2018)
6. Bradski, G.: The OpenCV Library. Dr. Dobb’s Journal of Software Tools (2000)
7. Chang, M.B., et al.: A compositional object-based approach to learning physical dynamics. In: Proc. ICLR (2017)
8. Chiappa, S., et al.: Recurrent environment simulators (2017)
9. Denil, M., et al.: Learning to perform physics experiments via deep reinforcement learning. Deep Reinforcement Learning Workshop, NIPS (2016)
10. Ehrhardt, S., others.: Learning A Physical Long-term Predictor. arXiv e-prints arXiv:1703.00247 (Mar 2017)
11. Ehrhardt, S., et al.: Learning to Represent Mechanics via Long-term Extrapolation and Interpolation. arXiv preprint arXiv:1706.02179 (Jun 2017)
12. Eslami, S.A., et al.: Attend, infer, repeat: Fast scene understanding with generative models. In: Advances in Neural Information Processing Systems. pp. 3225–3233 (2016)
13. Finn, C., et al.: Deep spatial autoencoders for visuomotor learning. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on. pp. 512–519. IEEE (2016)
14. Frakiadaki, K., et al.: Learning visual predictive models of physics for playing billiards. In: Proc. NIPS (2016)
15. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)
16. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks
17. Kansky, K., et al.: Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In: International Conference on Machine Learning. pp. 1809–1818 (2017)
18. Ladický, et al.: Data-driven fluid simulations using regression forests. ACM Trans. on Graphics (TOG) **34**(6), 199 (2015)
19. Lee, A.X., et al.: Stochastic adversarial video prediction. arXiv preprint arXiv:1804.01523 (2018)
20. Lerer, A., Gross, S., Fergus, R.: Learning physical intuition of block towers by example. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. pp. 430–438 (2016)
21. Li, W., Leonardis, A., Fritz, M.: Visual stability prediction and its application to manipulation. AAAI (2017)
22. Luc, P., Neverova, N., Couprie, C., Verbeek, J., LeCun, Y.: Predicting deeper into the future of semantic segmentation. ICCV (2017)

23. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: European Conference on Computer Vision. pp. 527–544. Springer (2016)
24. Monszpart, A., Thuerey, N., Mitra, N.: SMASH: Physics-guided Reconstruction of Collisions from Videos. ACM Trans. on Graphics (TOG) (2016)
25. Mottaghi, R., et al.: Newtonian scene understanding: Unfolding the dynamics of objects in static images. In: IEEE CVPR (2016)
26. Mrowca, D., et al.: Flexible Neural Representation for Physics Prediction. ArXiv e-prints (2018)
27. Novotny, D., et al.: Self-supervised learning of geometrically stable features through probabilistic introspection (2018)
28. Oh, J., et al.: Action-conditional video prediction using deep networks in atari games. In: Advances in Neural Information Processing Systems. pp. 2863–2871 (2015)
29. Ondruska, P., Posner, I.: Deep tracking: Seeing beyond seeing using recurrent neural networks. In: Proc. AAAI (2016)
30. Riochet, R., et al.: IntPhys: A Framework and Benchmark for Visual Intuitive Physics Reasoning. ArXiv e-prints (2018)
31. Sanborn, A.N., Mansinghka, V.K., Griffiths, T.L.: Reconciling intuitive physics and newtonian mechanics for colliding objects. Psychological review **120**(2), 411 (2013)
32. Sanchez-Gonzalez, A., et al.: Graph networks as learnable physics engines for inference and control (2018)
33. Stewart, R., Ermon, S.: Label-free supervision of neural networks with physics and domain knowledge. In: AAAI. pp. 2576–2582 (2017)
34. Thewlis, J., Bilen, H., Vedaldi, A.: Unsupervised learning of object frames by dense equivariant image labelling. In: Advances in Neural Information Processing Systems (NIPS). pp. 844–855 (2017)
35. Tompson, J., et al.: Accelerating Eulerian Fluid Simulation With Convolutional Networks. ArXiv e-print arXiv:1607.03597 (2016)
36. Watters, N., et al.: Visual interaction networks: Learning a physics simulator from video. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 4542–4550. Curran Associates, Inc. (2017)
37. Wu, J., et al.: Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In: Proc. NIPS. pp. 127–135 (2015)
38. Wu, J., et al.: Physics 101: Learning physical object properties from unlabeled videos. In: Proc. BMVC (2016)
39. Wu, J., et al.: Learning to see physics via visual de-animation. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems (NIPS) 30, pp. 153–164. Curran Associates, Inc. (2017)
40. Wu, J., et al.: Learning to see physics via visual de-animation. In: Proc. NIPS (2017)