Vid2Param: Modelling of Dynamics Parameters from Video

Martin Asenov, Michael Burke, Daniel Angelov, Todor Davchev, Kartic Subr and Subramanian Ramamoorthy

Abstract— **Videos provide a rich source of information, but it is generally hard to extract dynamical parameters of interest. Inferring those parameters from a video stream would be beneficial for physical reasoning. Robots performing tasks in dynamic environments would benefit greatly from understanding the underlying environment motion, in order to make future predictions and to synthesize effective control policies that use this inductive bias. *Online* physical reasoning is therefore a fundamental requirement for robust autonomous agents. When the dynamics involves multiple modes (due to contacts or interactions between objects) and sensing must proceed directly from a rich sensory stream such as video, then traditional methods for system identification may not be well suited. We propose an approach wherein fast parameter estimation can be achieved directly from video. We integrate a physically based dynamics model with a recurrent variational autoencoder, by introducing an additional loss to enforce desired constraints. The model, which we call Vid2Param, can be trained entirely in simulation, in an end-to-end manner with domain randomization, to perform online system identification, and make probabilistic forward predictions of parameters of interest. This enables the resulting model to encode parameters such as position, velocity, restitution, air drag and other physical properties of the system. We illustrate the utility of this in physical experiments wherein a PR2 robot with a velocity constrained arm must intercept an unknown bouncing ball with partly occluded vision, by estimating the physical parameters of this ball directly from the video trace after the ball is released.**

I. Introduction

There is an ever growing need to perform robotic tasks in unknown environments. Reasoning about observed dynamics using ubiquitous sensors such as video is therefore highly desirable for practical robotics. Traditionally, the complexity of this reasoning has been avoided by investing in fast actuators [18] [37] and using very accurate sensing [27]. In emerging field applications of robotics, the reliance on such infrastructure may need to be decreased [20], while the complexity of tasks and environment uncertainty has increased [15]. As such, there is a need for better physical scene understanding from low-cost sensors and the ability to make forward predictions of the scene, so as to enable planning and control.

Recent advances have enabled video prediction conditioned on observations [31] and reasoning about complex physical phenomena [47]. Video streams provide a rich source of information, but it is often challenging to acquire the compressed structured representations of interests. Techniques for system identification, originally developed for process control domains, are aimed at this problem [33]. There are a number of different approaches to estimating parameters [25], and sometimes even model structure [32], from observed data.

Acquiring reduced representations of the environment and performing system identification have historically been disjointly solved, despite the rich contextual information images often contain. This may lead to slower inference or even failure of the optimization should the state space model reduction be inaccurate. Yet, solving the problem jointly brings a set of practical challenges. First, it is difficult to acquire the necessary training data to cover the possible variations of the task of interest and generalize to unseen data. Secondly, a model needs to learn to perform probabilistic inference of parameters of interest and generation from a sequence of images, and capture long-term dependencies. From a practical robotics perspective, the model needs to be sufficiently fast to be able to perform system identification on-the-fly, while performing inference directly from images.

In this paper, we focus on the case where a robot must perform robust system identification *online*, directly from a rich sensory stream such as video (including the implicit tasks of detecting and tracking objects). We pose the problem as learning an end-to-end model, where we regress from videos directly to parameters of interest. Furthermore, we structure the learned model so as to be able to make probabilistic future predictions in the latent space, to enable appropriate action selection. This allows for interacting in relatively unknown environments when system identification must be performed on-the-fly for the successful completion of a task.

We present a model that enforces physical conformity between videos and dynamical parameters of interests. We integrate an analytic simulator with a recurrent latent model [12] by introducing an additional loss term for encoder-decoder mapping from a given sensory input (vision) to physical parameters and states (position, velocity, restitution factor, gravity, etc. in a parametric description of a physics-based model). We show that such a model can be trained with suitable domain randomization [48] in simulation and deployed in a real physical system. This model, which we call Vid2Param, allows for forward predictions envisioning possible future trajectories based on uncertainty in the estimate of the physical parameters. We demonstrate that such a model can indeed perform accurate system identification directly from videos by demonstrating that we achieve similar levels of performance as traditional system identification methods, which have access to ground truth starting trajectories. We perform experiments on simulated and real recorded videos of a bouncing ball with different physical properties. To illustrate the utility of this capability, we demonstrate this model on the task of intercepting a bouncing ball with a relatively slow moving robot arm and standard visual sensing.

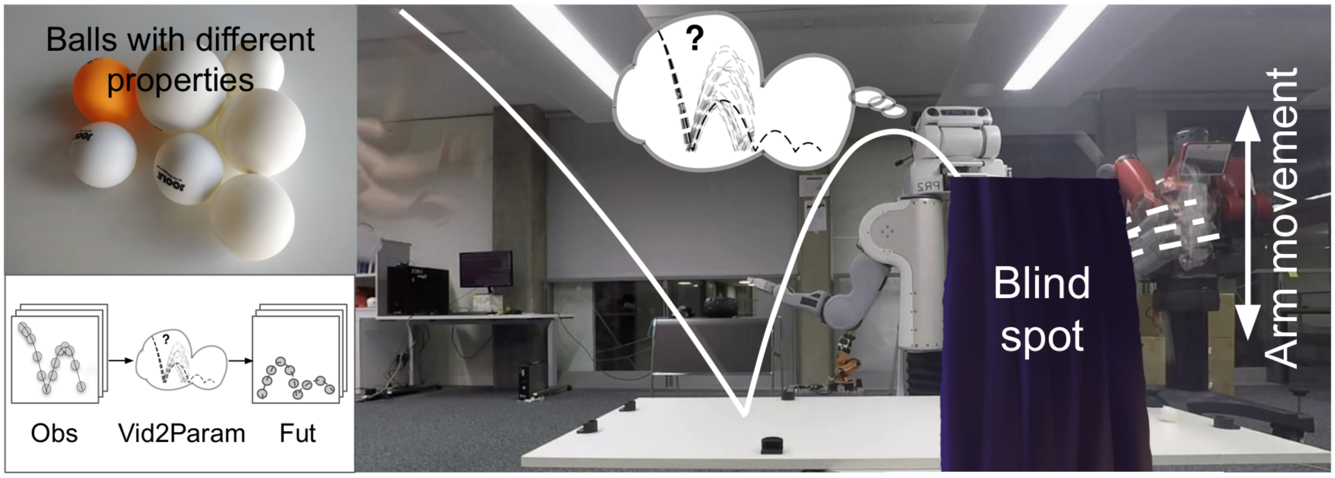


Fig. 1: **Overview and experimental setup.** We are interested in reasoning about the dynamics in an environment, by using a single video stream as a sensory input. To demonstrate the utility of our approach, we use a slowly actuated robotics arm to perform a 'stopping a bouncing ball' experiment, where the physical properties of the ball or height of the table are not known a priori and occlusions are present. Our model is able to perform online inference of the parameters of interests and generate plausible future trajectories. (Please refer to supplementary video for additional results.)

II. Related Work

A. System Identification

System identification (SysID) is concerned with the problem of determining the structure and parameters of a dynamical system, for subsequent use in controller design. The best developed versions of system identification methods focus on the case of linear time-invariant (LTI) systems, although almost all of these methods have also been extended to the case of nonlinear and hybrid dynamical systems. With these more complex model structures, the computational complexity of identification can be relatively high even for moderately sized data sets.

Examples of system identification procedures that could be applied to our problem domain, including the additional step of reducing model order, include the Eigen system realization algorithm [26] and Balanced POD (BPOD) [43] (which theoretically obtain the same reduced models [35]), and the use of feedforward neural networks [10]. BPOD can be viewed as an approximation to another popular method, Balanced truncation (BT) [44], which scales to larger systems.

Another way to approach the problem of identification is frequency domain decomposition [7]. Recent approaches in this vein include DMD [30] and Sindy [8], which allow for data driven, model-free system identification and can scale to high-dimensional data. When performing SysID directly from a rich sensory stream like video, it is not always clear what the optimal reduced representation should be [1]. We exploit the fact that a physics based model of objects can provide useful regularisation to an otherwise ill-posed identification problem.

B. Simulation alignment

When a parametric system model is available, simulation alignment can be performed to identify the parameters of the system. A standard approach is to perform least squares minimization or maximum likelihood estimation, for instance computing best fit parameters to align simulator traces to observed data [50]. When simulation calls are expensive, a prior over the parameter space can be enforced, e.g. Gaussian Processes, and Bayesian Optimization can be used [42] [41] [34] [5]. Our approach is closely related to [52] as we use supervision during the training phase of our model, and then use this learned approximation at test time. We also employ domain randomization while training our model [39] and our work follows a similar line of reasoning to that of [9], which aims to align a simulator to real world observations as the model is being trained. We focus on the problem of aligning a model to online observations at test time, for predictive purposes.

C. Learnable Physics Engines

There has been increasing interest in learnable physics engines - for example learning complex factorization (at the object or particle level) from data [36] [6], using particle based networks for fluid dynamics [45] and in robotics [13]. By representing the problem in terms of a set of learnable components (graph representing objects/particle and relations, Navier Stokes equations, linear complementary problem for the above mentioned tasks) a physics engine can be learned from raw data. Similar approaches have been shown to scale to video data [49]. We explore the complementary problem of system identification (with an analytical or learned simulator), and propose a direct optimization approach by learning an inverse probabilistic physics engine. This builds upon ideas presented in [50], where an analytical simulator is used with traditional system identification approaches. Closely related work is presented in [40], where surface properties are learned using Physics and Visual Inference Modules.

A related question to learning interactions between objects is that of learning a state space system to represent these. This has been explored for individual objects [28] [17], by using Kalman and Bayes filters for learning. State models and predictions have recently been explored in the context of videos involving multiple objects [22] through the use of Spatial Transformer Networks [23] and decompositional structures for the dynamics, as well as integrating the differential equations directly into a network [24].

D. Variational Autoencoder

Variational Autoencoders (VAEs) [29] have been extensively applied to image and video generation [16] [22]. Recently, VAEs have been used in reinforcement learning to improve generalization by learning a master policy from a set of similar MDPs [4]. Closely related work is that of [2] where Variational RNNs are used to learn 'residual physics' [51] [46]. The addition of loss terms to the reconstruction and KL error terms have also been proposed, allowing for enforcement of multiple desired constraints [21] [3]. We extend this line of work, by demonstrating that such constraints can be applied in a recurrent model to satisfy physics properties.

III. Vid2Param for online system identification from videos

Problem formulation Given a set of sensory observations $x_{1:t-1}$, we are interested in predicting future observations x_t using a low-dimensional dynamics representation $p\left(z_t|z_{1:t-1}\right)$.

$$\begin{aligned} p\left(x_t|x_{1:t-1}\right) &= \int p\left(x_t|z_t\right) p\left(z_t|z_{1:t-1}\right) p\left(z_{1:t-1}|x_{1:t-1}\right) dz \\ &= \int p\left(x_t|z'_t, \theta_t\right) p\left(z'_t, \theta_t|z'_{1:t-1}, \theta_{1:t-1}\right) \\ &\qquad\qquad p\left(z'_{1:t-1}, \theta_{1:t-1}|x_{1:t-1}\right) dz \end{aligned} \tag{1}$$

Here, we decompose the latent space $z = [z', \theta]$ into the physical dynamics parameters of interest, θ, and a remaining z' term, used for image reconstruction and to capture potentially un-modelled effects. We illustrate how this model can be learned in an end-to-end fashion from videos, and how we can use the predictions in the latent space for model predictive control.

Variational Recurrent Neural Networks We implement the inference process above using a modified recurrent VAE (VRNN) [12]. A VRNN consists of an RNN encoding the dynamics of the sequence, and a VAE conditioned on those dynamics, by including the hidden state of the RNN at each step. As shown in fig.2, the variational auto encoder φ_τ^{enc} (τ denotes the trainable parameters of the neural network) takes in a latent representation of the input $\varphi_\tau^x(x_t)$, in addition to the hidden state of an RNN, h_{t-1}. The encoder network then produces the mean and variance components, $\mu_{enc,t}$ and $\sigma_{enc,t}$ respectively, of a multivariate Gaussian distribution, q, that are also conditioned on h_{t-1}, thereby capturing information about the dynamics of the sequence until t:

$$\begin{aligned} q\left(z_t|x_{\leq t}, z_{<t}\right) = \mathcal{N}\left(\mu_{enc,t}, diag\left(\sigma^2_{enc,t}\right)\right), \\ \text{where } \mu_{enc,t}, \sigma_{enc,t} = \varphi_\tau^{enc}\left(\varphi_\tau^x\left(x_t\right), h_{t-1}\right) \end{aligned} \tag{2}$$

Similarly, the generative component of the VAE is also expanded by including the hidden state h_{t-1}:

$$\begin{aligned} p\left(x_t|z_{\leq t}, x_{<t}\right) = \mathcal{N}\left(\mu_{dec,t}, diag\left(\sigma^2_{dec,t}\right)\right), \\ \text{where } \mu_{dec,t}, \sigma_{dec,t} = \varphi_\tau^{dec}\left(\varphi_\tau^z\left(z_t\right), h_{t-1}\right) \end{aligned} \tag{3}$$

This means that the prior is no longer a standard normal distribution $\mathcal{N}(0,1)$ as in the case of a vanilla VAE, but is specified by a network conditioned on the hidden state h_{t-1}:

$$\begin{aligned} p\left(z_t|x_{<t}, z_{<t}\right) = \mathcal{N}\left(\mu_{prior,t}, diag\left(\sigma^2_{prior,t}\right)\right), \\ \text{where } \mu_{prior,t}, \sigma_{prior,t} = \varphi_\tau^{prior}\left(h_{t-1}\right) \end{aligned} \tag{4}$$

Finally, the RNN updates its hidden state h_t with a transition function f_ψ (with parameters ψ) by taking in a latent representation of the input $\varphi_\tau^x(x_t)$ a sample from $\varphi_\tau^z(z_t)$, and the previous hidden state h_{t-1}:

$$h_t = f_\psi\left(\varphi_\tau^x\left(x_t\right), \varphi_\tau^z\left(z_t\right), h_{t-1}\right) \tag{5}$$

VAEs are trained by minimising an evidence lower bound [29]. Given the VRNN modifications, the overall loss, including a Kullback-Leibler (KL) divergence and reconstruction loss term, becomes:

$$\begin{aligned} \mathbb{E}_{q(z_{\leq T}|x_{\leq T})}\Big[\sum_{t=1}^{T} \log p\left(x_t|z_{\leq t}, x_{<t}\right) - \\ \mathrm{KL}\left(q\left(z_t|x_{\leq t}, z_{<t}\right) \| p\left(z_t|x_{<t}, z_{<t}\right)\right)\Big]. \end{aligned} \tag{6}$$

In summary, the VRNN is a modified VAE that makes use of a learned low-dimensional dynamics model to make sequential predictions. However, the VRNN provides no guarantees that the latent representation is meaningful, and as a result is unsuited for use in control or for system identification.

Vid2Param We propose combining the encoder-decoder factorization with dynamics modelling in the latent space z, conditioned on parameters of interest, θ. We introduce an additional loss to the standard VRNN to encourage encoding of physically meaningful parameters, by including a Gaussian negative log likelihood [38] loss between part of the latent space z and the physical parameters θ we are interested in (e.g., gravity, restitution, position, etc., in the case of a bouncing ball). The loss terms are scaled with non-negative numbers α, β and γ and we let N_θ represent the size of the parameter vector.

$$\begin{aligned} &\mathbb{E}_{q(z_{\leq T}|x_{\leq T})}\Big[\sum_{t=1}^{T}\Big(\alpha \log p\left(x_t|z_{\leq t}, x_{<t}\right) - \\ &\qquad \beta \mathrm{KL}\left(q\left(z_t|x_{\leq t}, z_{<t}\right) \| p\left(z_t|x_{<t}, z_{<t}\right)\right) + \\ &\gamma \sum_{i=1}^{N_\theta}\big(\frac{1}{2}\ln(2\pi) + \frac{1}{2}\ln\left((\sigma^i_{enc,t})^2\right) + \frac{(\theta^i_t - \mu^i_{enc,t})^2}{2(\sigma^i_{enc,t})^2}\big)\Big)\Big] \end{aligned} \tag{7}$$

Given trained networks, we can now perform probabilistic inference of physical parameters from sensory data such as a sequence of images using the factored portion of the latent space directly.

$$P(\theta^i_t|x_{\leq t}) = \mathcal{N}(\mu^i_{enc,t}, \sigma^i_{enc,t}) \tag{8}$$

Additionally, we can sample plausible future extrinsic properties (eg. positions) by recursively updating the model predictions to generate future states.

$$P(\theta^i_t|x_{<t}, \theta_{<t}) = \mathcal{N}(\mu^i_{prior,t}, \sigma^i_{prior,t}) \tag{9}$$

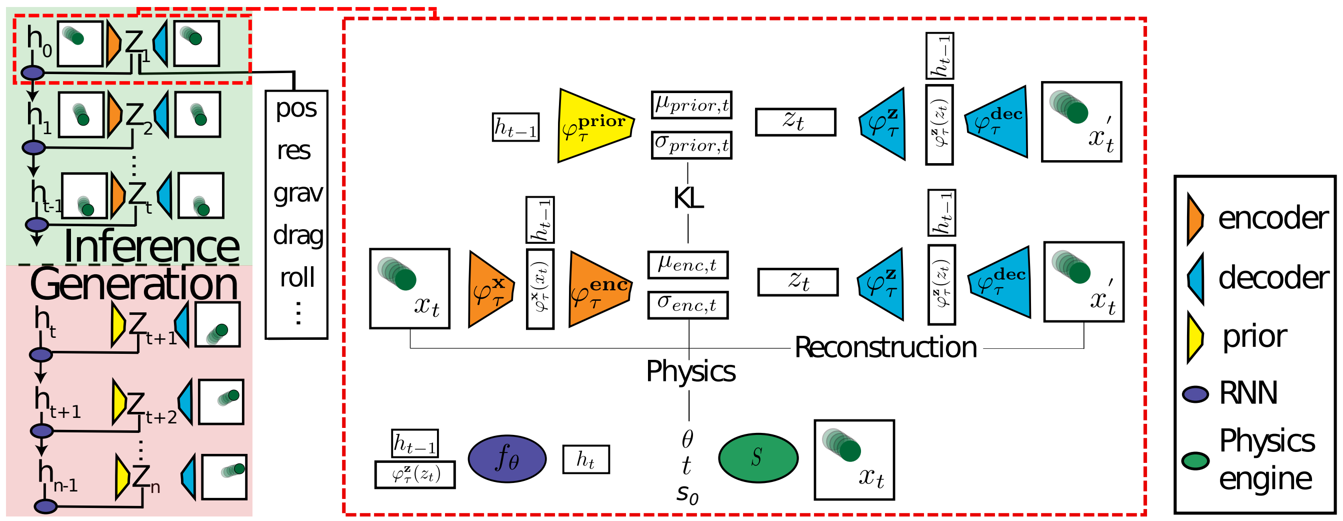


Fig. 2: **Technical details and notation.** We propose an end-to-end model for performing system identification directly from unstructured input, such as video. We based our model on a Variational Recurrent Neural Network (VRNN) [12]. To encode the physical properties of interests we propose an additional Gaussian negative log likelihood loss between the parameters of interest and part of the latent space. The inference and generation overview of the model (left) and the training procedure at each frame (right) can be seen in the figure.

As a final modification, we exclude x_t from the recurrent step, since all necessary information is already present in z_t. This speeds up the prediction in the latent space, as x does not need to be reconstructed and fed back into the network at every step. As such we can make recursive future predictions entirely in the latent space,

$$h_t = f_\psi \left(\varphi^z_\tau\left(z_t\right), h_{t-1}\right). \qquad (10)$$

To summarise, the contributions of this paper include:

1) Extension of the VRNN model with a loss term to encode dynamical properties.
2) Enabling faster future predictions in the latent space, along with uncertainty quantification through the identified parameters.
3) Evaluation of speed and accuracy of identification, against alternate approaches to system identification
4) Demonstration on a physical robotic system, in a task requiring interception of a bouncing ball whose specific physical parameters are unknown a priori, requiring online identification from the video stream.

IV. Experiments

First, we perform a series of experiments on simulated videos, when ground truth is explicitly available. We compare our method against existing system identification methods, evaluating speed of estimation and accuracy of the identified parameters. Next, we evaluate our method on a set of real videos. Finally, we perform a physical experiment involving online system identification from a camera feed.

A. Setup

We use a bouncing ball as an example hybrid dynamical system. This is a particularly useful example, as the dynamics of the ball vary depending on the ball state, making system identification particularly challenging from high dimensional sensor data using classical techniques. The governing dynamics of the bouncing ball can be described using the following set of ordinary differential equations:

$$S = \begin{cases} s_t = s_0 + \dot{s}_0 t + \frac{1}{2}\ddot{s}t^2, \ddot{s} = g - d & \text{Free fall} \\ \dot{s}^y_t = -e\dot{s}^y_{t-1}, \text{when } \dot{s}^y_t < 0; s^y_t = 0 & \text{Bounce} \\ \dot{s}^x_t = r\dot{s}^x_{t-1}, \text{when } \ddot{s}^y_t = 0 & \text{Rolling} \end{cases} \qquad (11)$$

We use $s, \dot{s}, \ddot{s}$ for the current position, velocity and acceleration respectively, e is the coefficient of restitution and r the rolling resistance. Additional dynamic effects are often observed such as air drag $d = -c\dot{s}\sqrt{(\dot{s}^x)^2 + (\dot{s}^y)^2}/m$, where c is the drag constant and m is mass. Thus the acceleration becomes $\ddot{s} = g + d$, where g is the gravitational force ($g^x = 0$). As such, the system is completely determined by the initial state of the system $s, \dot{s}, \ddot{s}$ and its physical properties $e, r, m, c, g \in \theta$. It should be noted that the real world behaviour of any specific ball could deviate from this model depending on its shape, initial spin (Magnus effect), the presence of wind, and so on.

We make the following assumptions: 1) there is a single moving object, the bouncing ball 2) the ball bounces off a flat surface. We do not assume to know the physical properties of the ball, the height of the surface or the exact velocities of the ball, and we use a single low quality camera for sensing.

We use a parallel adaptive ODE solver to simulate data described by eq.11. We use these simulated trajectories to generate a sequence of images. We generate 10000 training and 100 test videos, with 200 timesteps/10 seconds, 28×28, with $e \in [0.6, 1.0]$, $g \in [-6.81, -12.81], d \in [0.05, 0.0005], r \in [0.0, 0.7]$. We split the parameters into 10 sub-ranges and alternately use them for training and testing, so no parameters used in the training data are available in the test data.

For the real videos and robot experiment, we trained a separate model with 5000 videos, 100×50 with 75 timesteps/10 seconds and the same physical parameters. Additionally, we add motion blur based on the velocity and black-out part of the frames to account for some of the missing/noisy data typically exhibited when using low-cost cameras. Additionally, we randomize the height of the plane on which the ball bounces. Our encoder-decoder network follows a similar architecture to [11] and for the RNN we use a standard LSTM network. We set $\alpha = 1$, $\beta = 1$ and $\gamma = 10$ throughout our experiments (eq.7). We use an NVidia 1080 Ti for training and laptop NVidia Quadro M2000M GPUs for testing the model. We use MSE as an accuracy metric throughout the paper and normalize positions and parameters.

B. System identification

In this experiment, we evaluate the speed and accuracy of the proposed method against different simulation alignment approaches. We evaluate the likelihood of the observed trajectory with respect to simulated trajectories by performing MLE estimation (similarly to [50]). We sample 2000 trajectories by placing a uniform prior over the physical parameters and compare the simulated trajectories against those observed. We select the parameters which generated the least error between the trajectories, and compare these against parameters that generated the observed trajectory.

Secondly, we extend this by imposing a Gaussian Process prior over the parameters of interest and performing Bayesian Optimization [19] [41]. We use Expected Improvement as an acquisition function and use 20 optimisation steps. The baselines have access to the initial velocity, n number of positions (as such we don't need a tracker as in [50]) and an optimized ODE solver for sampling.

In contrast, our trained model receives *only* the video as an input, and no other parameters. Speed and accuracy benchmarks are shown in fig.3. It can be seen that our method has similar or better performance, despite not having access to the initial ground truth trajectory of the ball.

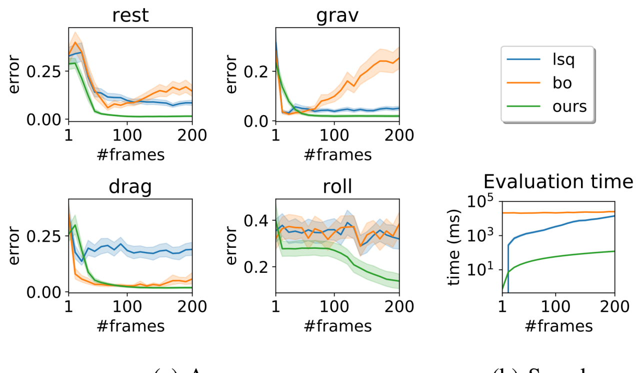


Fig. 3: **Performance of different system identification methods with variable number of observed frames.** (a) Overall error of the predicted normalized parameters (b) Speed of computation. We denote [50] as 'lsq' and [19] as 'bo'.

C. Forward predictions

Here, we evaluate the future prediction accuracy as frames are observed. In addition to previous baselines in Sec.IV-B, we add an additional non-parametric model for system identification that approximates responses using a set of modes with different frequencies [30]. Three sets of predictions are evaluated - after 20, 50 and 100 frames respectively - until the end of the video at 200 frames, as shown in fig.4. We visualize example model predictions and their associated uncertainty in fig.5. Importantly, the proposed approach becomes more certain as additional frames are observed, highlighting the probabilistic nature of Vid2Param.

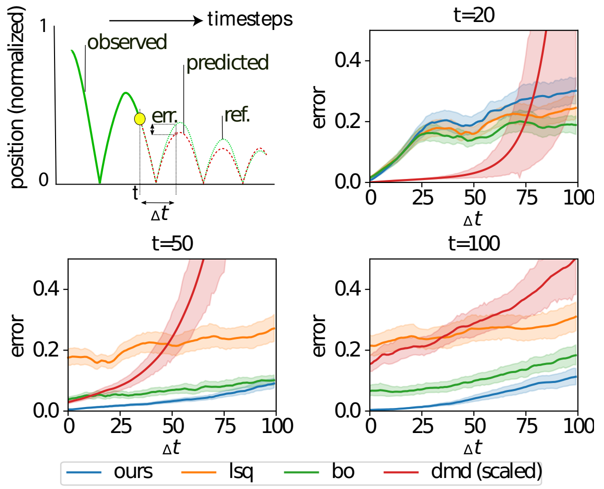


Fig. 4: **Accuracy of forward prediction.** The accuracy is evaluated after 20, 50 and 100 frames are observed, predicting for the next 100 frames. The DMD error is scaled down 1k, 50 and 5 times respectively for predictions after 20, 50 and 100 observations. We note [50] as 'lsq', [19] as 'bo' and [14] as 'dmd'.

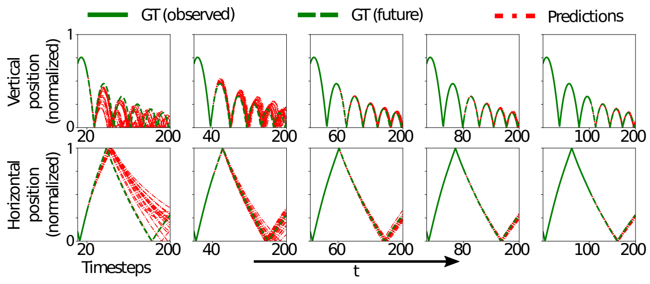


Fig. 5: **Forward prediction uncertainty** after different numbers of observed frames (20, 40, 60, 80 and 100). Observed trajectory (green), future ground truth (dashed-green), predictions (red).

D. Varying physical properties and sensitivity analysis

In this experiment, we evaluate how well Vid2Param can estimate physical parameters when they are changing as the video is unrolled (using a model where parameters are assumed to stay constant throughout the video). Therefore, this is a test of robustness or sensitivity of the model. We generate a new dataset, wherein the parameters change every 50 frames (2.5 seconds) in a given video. The results are shown in fig.6, and show that the proposed model can infer changing parameters, provided there is enough system excitation to facilitate this. For example, gravitation coefficients can only be inferred if the ball is bouncing, while the rolling coefficient can be inferred if the ball is rolling.

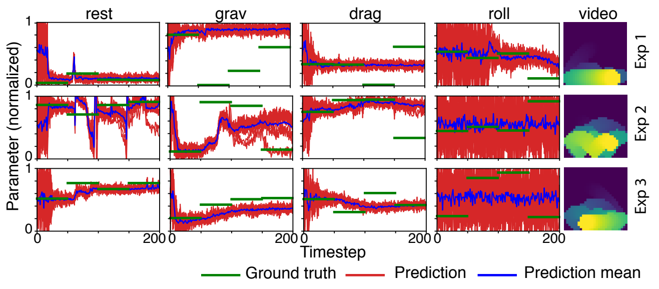


Fig. 6: **System Identification from video with varying physical parameters.** The physical parameters of the bouncing ball change every 50 frames (4 times per video). We plot ground truth (green), predicted samples (red), and the predicted - mean (blue). Given enough excitation, our model can detect the change in the parameters.

We also perform sensitivity analysis over multiple conditions. First, we test for extrapolation over unseen parameters by training on the first half of the range for different parameters - eg. $e_{train} \in [0.6, 0.8]$ (for restitution). We evaluate performance for increasing parameter deviations, $\Delta params$, in the test data - $e_{test} \in [0.8, 1.0]$. Secondly, we evaluate the performance when training with increasing additive parameter noise. Finally, we train with simpler physics model (fixed drag and rolling coefficient) and test how well we can estimate gravity and restitution when testing with the full model as the video is unrolled. Results can be seen in fig.7, with extrapolation performance slowly decaying outside of training regions and consistent performance even with large percentage of noise added to the training parameters.

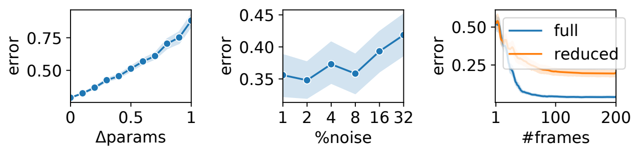


Fig. 7: **Sensitivity analysis.** (Left) Extrapolation of parameters (Middle) Training with noisy data (Right) Training with less accurate model - fixed drag and rolling coefficient.

E. Real videos

Here, we evaluate how well our method can scale to real videos. We record a set of videos, lasting between 1-5 seconds, of different types of bouncing balls - rubber, tennis and ping-pong balls. Since the exact physical properties of the balls are not available, we instead use the accuracy of the forward predictions as an evaluation metric. We compare the last ten positions of the ground truth position of the ball, with the forward predictions of a model as the video is unrolled. Here, we compare our method against [50], where we generate 5000 uniform samples of all physical parameters, horizontal and vertical velocities, horizontal and vertical position in a small region around the starting location of the ball - in order to account for some of the noise in the real videos. In fig.8 we show the convergence of our method for different types of balls, as well as the accuracy of the forward predictions as the video is observed.

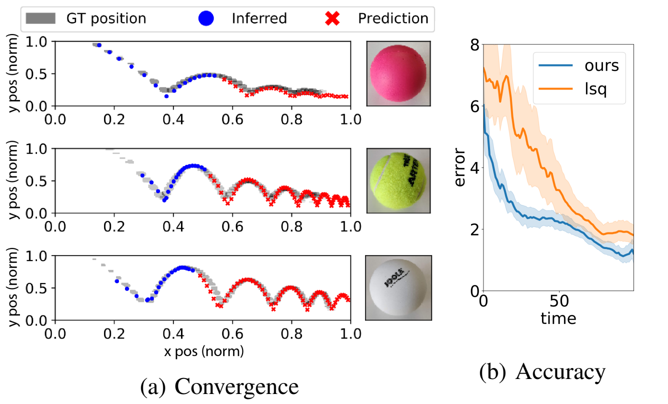
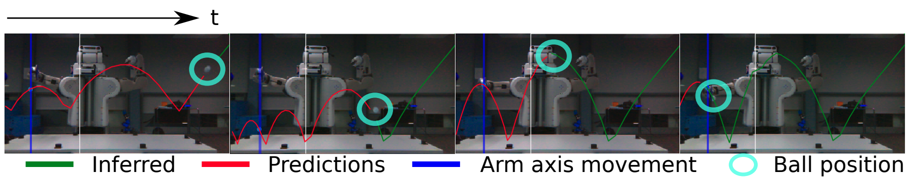


Fig. 8: **Experiments with real videos.** We evaluate the performance of our method on real videos with bouncing balls with different physical properties (rubber, tennis, ping-pong). As the video is unrolled we compare the future predictions for the very last 10 frames of each video (the long term prediction accuracy) as explicit ground truth over the physical parameters is not available (a) Example model predictions for the three different types of balls, overlaid on the extracted positions from the images (b) Accuracy of the last 10 forward predictions as the video is observed. We denote [50] as 'lsq'.

F. PR2 Robot experiments

Finally, we evaluate the accuracy and speed of our method in an experiment where the PR2 robot uses its arm to intercept a bouncing ball from a visual feed, using a standard low-cost camera as sensory input (please refer to supplementary video). Firstly, the camera is calibrated with respect to the arm movements, so that predictions of the ball in the image, correspond to the same position of the gripper. No calibration with respect to the bouncing surface, position/velocity mappings, size of the ball, etc. are needed since these should be robustly dealt with by the model trained on randomized physics in simulation. The difference between two consecutive frames are fed directly into our model and the latent predictions are unrolled until the future predicted horizontal position is approximately the same as the horizontal position of the gripper of the arm. Then the generated vertical position of the ball is sent as a positional set point to the arm. An experimental run usually lasts for 2-3 seconds, during which the PR2 robot must infer the physics of the ball, predict its future trajectory and execute an action to intercept it. Our model runs at 20Hz on a standard laptop GPU, using IKFast for inverse kinematics of the arm. We use different types of ping-pong balls, in order to test how well our model can reach to balls with different physical properties. After each experiment (each throw of a ball), the model state is reset - so in each experiment we evaluate how well we can perform online system identification. Results can be seen in fig.9.



Random Policy	8/35 (23%)
Random Policy (2x)	10/35 (29%)
Vid2Param	**27/35 (77%)**

Fig. 9: **Experiments with PR2.** We perform a set of experiments, where a slowly actuated arm of the PR2 robot must intercept a bouncing ball, using only video as sensory input. We evaluate our method against a random policy feeding actions at 1Hz/2Hz. We demonstrated that our method can perform fast inference over the physical parameters of interest (e.g. restitution factor, height of the table) and correct future trajectories. Each experiment lasts about 2 seconds. At each throw of the ball, the model state is reset - so system identification is performed at every throw of the ball. (Please see the supplementary video for additional results.)

V. Results and Analysis

Unified model for physical reasoning. In this work we present a model for inference of physical parameters and generation of plausible future states from videos. We observe that such a model can be trained in an end-to-end fashion and perform accurate system identification in simulated and real settings, and subsequently used for control. We constrain our experiments to a single object and show that using just a video stream we can perform *online* system identification. Importantly, the proposed approach is able to generalise well to images captured from a real camera, despite only being trained on simulated data. This highlights the value of sim2real techniques for interpreting physical parameters in various applications, and its potential to enable reasoning about physical properties, from relatively low fidelity sensors bootstrapped by learning from simulation.

System identification. We observe that the proposed method can accurately infer different physical parameters, outperforming baselines from the literature. The magnitude of gravity and air drag can usually be inferred from observing just a few frames. Air drag can usually be inferred after observing a few more frames, as it is a function of both horizontal and vertical velocity, rather than just the vertical velocity as in the case of gravity. Restitution factors can be inferred a few frames after the ball has bounced for the first time. The rolling coefficient has a higher error, which starts to decrease towards the end of the videos. While traditional system identification methods can infer physical properties, which have a clear effect on the trajectory (such as restitution), it is challenging to infer properties that jointly contribute to a certain effect. By using domain randomization and a sim2real approach, our method can learn the difference in parameters of trajectories with similar appearance even when trained with noisy data. As such we can accurately estimate parameters with similar effect on the dynamics (such as gravity and air drag), as well as parameters whose effects are not observed until the end of the trajectory (such as rolling coefficients). Moreover, we also demonstrate that to an extent detect change in the parameters, as a video is unrolled (although this requires system excitation), and extrapolate to unseen parameters.

Forward predictions. We have shown that our model can perform forward predictions in the latent space, over parameters of interest such as physical state variables. The forward predictions bring out key aspects of the evolution of uncertainty, such as high variability before a bounce and lower variability soon after, high variability over the stopping point before rolling is observed, etc. The proposed approach outperforms both parametric and non-parametric baselines in its ability to accurately perform forward predictions.

Limitations and future work. We observe in robotics experiments that our model performs well in real settings. Nevertheless, we experienced some limitations arising from making the predictions based on a single image, e.g. the ball passing behind or in front of the gripper. Thus in the future it will be beneficial to extend this line of work by inferring future predictions from multiple sources of video stream from different locations, or incorporating depth sensing. We note that our proposed model is independent of the choice of encoder, decoder and training data. It would be of interest to explore how such a model would perform with richer training data (e.g. multiple objects) by using advanced domain randomization [53] or different encoder-decoder structure [6].

VI. Conclusions

This paper presents a method for online system identification from video. We benchmark the proposed approach against existing baselines from the literature, showing it outperforms these both in terms of speed and accuracy of identification. We then demonstrate the utility of this approach with the task of stopping a bouncing ball with a robot arm, performing online identification from a camera feed and using the proposed model for inference of the physics parameters. Further, we show its ability to generate future predictions of the ball position, laying the groundwork for much more sophisticated predictive motion planning schemes.

References

[1] A. Achille and S. Soatto. A separation principle for control in the age of deep learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:287–307, 2018.
[2] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *IROS*, pages 3066–3073. IEEE, 2018.
[3] D. Angelov, Y. Hristov, and S. Ramamoorthy. Using causal analysis to learn specifications from task demonstrations. AAMAS '19, pages 1341–1349, 2019.
[4] I. Arnekvist, D. Kragic, and J. A. Stork. Vpe: Variational policy embedding for transfer reinforcement learning. *ICRA*, 2019.

[5] M. Asenov, M. Rutkauskas, D. Reid, K. Subr, and S. Ramamoorthy. Active localization of gas leaks using fluid simulation. *IEEE Robotics and Automation Letters*, 4(2):1776–1783, 2019.

[6] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, pages 4502–4510, 2016.

[7] R. Brincker, L. Zhang, and P. Andersen. Modal identification of output-only systems using frequency domain decomposition. *Smart materials and structures*, 10(3):441, 2001.

[8] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[9] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *ICRA*, 2019.

[10] S. Chen, S. Billings, and P. Grant. Non-linear system identification using neural networks. *International journal of control*, 51(6):1191–1214, 1990.

[11] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, pages 2172–2180, 2016.

[12] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *NeurIPS*, pages 2980–2988, 2015.

[13] J. Degrave, M. Hermans, J. Dambre, et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, 13, 2019.

[14] N. Demo, M. Tezzele, and G. Rozza. PyDMD: Python Dynamic Mode Decomposition. *The Journal of Open Source Software*, 3(22):530, 2018.

[15] Z. Erickson, H. M. Clever, G. Turk, C. K. Liu, and C. C. Kemp. Deep haptic model predictive control for robot-assisted dressing. In *ICRA*, pages 1–8. IEEE, 2018.

[16] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *NeurIPS*, pages 3225–3233, 2016.

[17] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *NeurIPS*, pages 3601–3610, 2017.

[18] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa. Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system. In *ICRA*, pages 181–187. IEEE, 2006.

[19] J. González. Gpyopt: A bayesian optimization framework in python, 2016.

[20] H. Hastie, K. Lohan, M. Chantler, D. A. Robb, S. Ramamoorthy, R. Petrick, S. Vijayakumar, and D. Lane. The orca hub: Explainable offshore robotics through intelligent interfaces. *arXiv preprint arXiv:1803.02100*, 2018.

[21] Y. Hristov, A. Lascarides, and S. Ramamoorthy. Interpretable latent spaces for learning from demonstration. In *CoRL*, volume 87, pages 957–968. PMLR, 29–31 Oct 2018.

[22] J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles. Learning to decompose and disentangle representations for video prediction. In *NeurIPS*, pages 515–524, 2018.

[23] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NeurIPS*, pages 2017–2025, 2015.

[24] M. Jaques, M. Burke, and T. Hospedales. Physics-as-inverse-graphics: Joint unsupervised learning of objects and physics from video. *arXiv preprint arXiv:1905.11169*, 2019.

[25] T. A. Johansen and B. A. Foss. Identification of non-linear system structure and parameters using regime decomposition. *Automatica*, 31(2):321–326, 1995.

[26] J.-N. Juang and R. S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *J GUID CONTROL DYNAM*, 8(5):620–627, 1985.

[27] A. Kapur, A. Kapur, N. Virji-Babul, G. Tzanetakis, and P. F. Driessen. Gesture-based affective computing on motion capture data. In *International conference on affective computing and intelligent interaction*, pages 1–7. Springer, 2005.

[28] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

[29] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[30] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[31] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[32] Y. Li, H.-L. Wei, S. A. Billings, and P. G. Sarrigiannis. Identification of nonlinear time-varying systems using an online sliding-window and common model structure selection (cmss) approach with applications to eeg. *International Journal of Systems Science*, 47(11):2671–2681, 2016.

[33] L. Ljung. System identification. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2001.

[34] T. Lopez-Guevara, N. K. Taylor, M. U. Gutmann, S. Ramamoorthy, and K. Subr. Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation. In *CoRL*, pages 77–86, 2017.

[35] Z. Ma, S. Ahuja, and C. W. Rowley. Reduced-order models for control of fluids using the eigensystem realization algorithm. *THEOR COMP FLUID DYN*, 25(1-4):233–247, 2011.

[36] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. F. Fei-Fei, J. Tenenbaum, and D. L. Yamins. Flexible neural representation for physics prediction. In *NeurIPS*, pages 8813–8824, 2018.

[37] A. Namiki, Y. Imai, M. Ishikawa, and M. Kaneko. Development of a high-speed multifingered hand system and its application to catching. In *IROS*, volume 3, pages 2666–2671. IEEE, 2003.

[38] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *ICNN'94*, volume 1, pages 55–60. IEEE, 1994.

[39] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, pages 1–8. IEEE, 2018.

[40] S. Purushwalkam, A. Gupta, D. M. Kaufman, and B. Russell. Bounce and learn: Modeling scene dynamics with real-world bounces. *ICLR*, 2019.

[41] F. Ramos, R. C. Possas, and D. Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[42] D. Romeres, G. Prando, G. Pillonetto, and A. Chiuso. On-line bayesian system identification. In *2016 European Control Conference (ECC)*, pages 1359–1364. IEEE, 2016.

[43] C. W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013, 2005.

[44] M. G. Safonov and R. Chiang. A schur method for balanced-truncation model reduction. *IEEE Transactions on Automatic Control*, 34(7):729–733, 1989.

[45] C. Schenck and D. Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *CoRL*, pages 317–335, 2018.

[46] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung. Neural lander: Stable drone landing control using learned dynamics. *ICRA*, 2019.

[47] T. Takahashi and M. C. Lin. Video-guided real-to-virtual parameter transfer for viscous fluids. In *Transactions on Graphics (SIGGRAPH ASIA 2019)*, 2019.

[48] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30. IEEE, 2017.

[49] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *NeurIPS*, pages 4539–4547, 2017.

[50] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NeurIPS*, pages 127–135, 2015.

[51] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *arXiv preprint arXiv:1903.11239*, 2019.

[52] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *ICRA*, pages 528–535. IEEE, 2016.

[53] C. Zheng, T.-J. Cham, and J. Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *ECCV*, pages 767–783, 2018.