Learning Physical Dynamical Systems for Prediction and Control: A Survey

Julienne LaChance
Princeton University
jl40@cs.princeton.edu

Abstract

This paper reviews recent papers in learning physical dynamics as a function of image or video data. The ability to describe the evolution of dynamical systems is essential to constructing accurate algorithms for a variety of tasks in computer vision. Capturing the physics of a system is critical to video prediction, due to the necessity of simulating the interaction between various agents and their environment, such as motion prediction of a bouncing ball. Accurate physical models are also essential to controlling an agent which can interact with the visual environment, such as a robotic arm which learns to manipulate physical objects by collecting visual information through a camera. We survey the various methods for representing a dynamical system in isolation, methods for prediction and control, and multi-modal extensions. We summarize the major roadblocks and prospective trends in each domain, and highlight several notable works for their contributions towards building robust, generalizable models.

1. Introduction

In this survey, we consider the problem of utilizing raw visual input, possibly augmented with extra-sensory data, to determine a representation of a physical, dynamical system. The physics determined by the modeling process can then be used to predict future states of the system or apply controls to modify the state. Here, we explore a variety of strategies for recovering the internal physics of a system, including parametric models, in which physical parameters must be determined within fixed dynamical equations, and more approximate methods, such as causality models and graph-based models. Oftentimes, very simple physical systems are utilized to analyze the effectiveness of such methods, including gravitational systems, rigid body dynamics, spring-mass systems, or scenarios in which the algorithm must detect when a stack of objects will fall. Motion tracking is another widely studied problem. We will subsequently review the current methods for utilizing such dynamical models for video prediction and control.

Much of the inspiration of these models is derived from how humans experience and interact with the world. If a human is presented with a novel object, he or she could anticipate the trajectory of the object if it were to be thrown or dropped, all as a function of raw sensory inputs. In 2011, Hamrick et al. performed experiments to suggest that humans use internal models of physics to predict how dynamical systems evolve [22]. In this study, human judgments regarding properties of physical systems were found to closely correlate with those made by a model observer using simulations based on realistic physical dynamics and sampling-based approximate probabilistic inference. A number of older works from the field of cognitive science explore the internal physical model hypothesis, including [25], [32], [23], and [47].

When it comes to predicting subsequent frames within a video, one might consider works such as [34] by Michaelski et al., which predict the motion of a bouncing ball. However, overly simplified methods for generating new video frames directly from past frames are wrought with drawbacks. Such methods do not generalize well to novel environments. Moreover, they do not provide any real interpretable notion of the agents within the system nor their influence on their surrounding environment, and they may not scale well with large training data sets. An analogously naïve prediction method developed to determine the future visual snapshots of a robotic arm is presented by Boots et al. in [7].

Within the controls systems engineering community, the internal physical model formulation has been well explored. The methods presented in the control literature frequently involve running multiple internal simulations of a system and choosing the control strategy which optimized the output state. D. Q. Mayne summarizes the use of internal models for action planning in [33]. A more recent work by Oh et al., [39], provides a method for learning internal physical models with task supervision (with Q control) for the purpose of playing Atari games. In order to extend to more general environments, however, it would be beneficial to design more generalizable methods which do not require extensive task supervision, as do those in the controls literature. In

the following sections, we emphasize the contributions of various researchers towards the goal of producing accurate physical predictions with minimal human supervision.

2. Parameter estimation for fixed dynamical models

We first consider the class of methods which rely upon the independent development of a physics engine in conjunction with a parameter estimation technique. In Wu et al., [55], for instance, a MCMC is utilized to determine physical parameters corresponding to a physics engine describing the motion of blocks sliding down an inclined plane. Figure 1 displays the schemetic demonstrating the joint use of the physics engine and a tracking algorithm to generate predictions of block velocities. Works such as those by Bhat et al., [6], Brubaker et al., [9], and Mottaghi et al., [37], similarly estimate parameters in pre-determined Newtonian equations from images and video data. Due to the inherent difficulty in producing comprehensive dynamical models for most systems, the use of fixed physics engines finds limited application with real-world tasks.

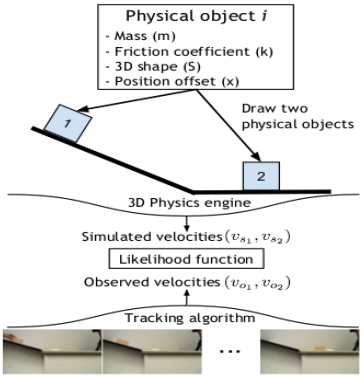

Figure 1. Schematic for integrated physics engine and tracking algorithm method to predict object motion. Image courtesy of [55].

In [1], Kyriazis et al. anticipate the trajectory of bouncing balls using a physics-based simulation. Given single 2D camera observations, the physical model serves as a prior to the position of the ball, even when the ball is occluded in certain frames. In this case, the physical parameters of the physics engine are determined by optimizing over a physical parameter space. Salzmann and Urtasun present a framework for handling a wider variety of tracking problems, and present their results on tasks such as articulated human motion tracking and billiards dynamics [44].

3. Approximate physical models

When the form of a physical model cannot be constructed, then we rely upon machine learning methods to learn approximations for the dynamical system as a whole. This might involve simplified techniques such as that presented in [48] by Urtasun et al., in which human motion paths are anticipated by using a linear dynamical model. In this section, we will examine a variety of methods for learning approximate dynamical models.

3.1. Dimensionality reduction

Autoencoder networks are designed such that the network is forced to learn a low-dimensional representation, or encoding, for a set of data. A number of works similarly seek to construct physical models by transforming raw visual data to lower-dimensional features, which are then used in conjunction with reinforcement learning or control methods.

We look to works such as Lampe & Riedmiller [27], Kietzmann & Riedmiller [26], Lange et al. [28], and Finn et al. [14] for examples of such encoding from visual input techniques applied in the robotics community. More advanced applications include policy learning for robotic arms. For example, in [49], Wahlström et al. use deep autoencoders to learn a closed-loop control policy from pixel information alone via reinforcement learning. The low-dimensional embedding is learned jointly with a predictive model, allowing both static and dynamic information to be accounted for, and thereby improving long-term predictions. Additionally, autoencoder-based techniques are commonly applied to toy controllers for comparison with classical literature, as in [51] of M. Watter et al. Here, a classical pendulum system is controlled using images of the pendulum alone, motivating more advanced applications of vision-based control.

However, as the field of deep learning evolves, autoencoders are being used in fewer and fewer applications. With the exception of such areas as data de-noising, compression, and dimensionality reduction for visualization, many of the benefits of autoencoders have been overtaken by supervised-learning approaches in practical applications. Autoencoders are likely to remain relevant in certain semi-supervised applications, while bottleneck architectures in general are likely to become more prevalent in computer vision methods, given the recent success of the Stacked Hourglass Networks [38] for the task of human pose estimation.

3.2. Kalman filters

Kalman filters, and their nonlinear extensions, extended Kalman filters (EKFs) are algorithms from classical control theory which optimally produce estimates of the state of a dynamic system. Classical Kalman filters were applied to computer vision problems with limited success; see, for instance, Weng et al.'s application of Kalman filters to video object tracking using classical techniques ([54]).

Krishnan et al. proposed deep Kalman filters in 2015 in [43]. In this early work, a generative model is fit to a

sequence of observations and actions, and the relationship between a latent state and the observations and actions is assumed to be nonlinear. That is, the underlying state transition matrices that control the dynamics of a hidden process state are replaced with deep neural networks. But, only the state space equations of the Kalman filter are used. Alternative approaches which combine deep neural networks and Kalman filters seek to replace the manual fine-tuning of noise parameters in robotic navigation tasks, as in [2].

A recent improvement to pose estimation in video builds off of [43]. In Krishnan et al.'s work, an LSTM network is trained which jointly learns to propagate the state, incorporate measurement updates and react to control inputs. By constrast, in [11], Coskun et al. utilize distinct prediction and update models and demonstrate that their approach produces superior state estimations, particularly in the absence of large-scale training data. Refer to Figure 2 for the schematic of the LSTM-KF architecture from [43], which obtained SOTA in three tasks spanning pose estimation in video, camera tracking, and object tracking.

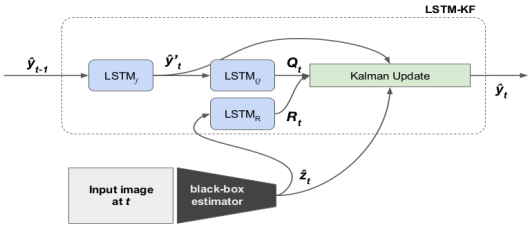

Figure 2. Schematic of the LSTM-KF architecture. Classical components of Kalman filters have been replaced by LSTM networks, such as the nonlinear state-transition model, f, and the covariance models of the process and observation noise, Q and R, respectively. Image courtesy of [43].

3.3. Hidden Markov Models

A hidden Markov model (HMM) consists of a finite set of states, each of which is associated with a probability distribution, and which are "hidden" in the sense that only the outcome of the model is visible to an external observer. The process is assumed to be closely approximated by a Markov process. A hidden Markov model can be considered to be a generalization of a mixture model in which the latent variables are related to one another via a Markov process.

Traditionally, hidden Markov models have been applied to a number of computer vision tasks. In 1997, HMMs were being used for motion prediction for visual tracking. For example, Brand et al. utilized HMMs for action recognition using 3D hand tracking data [8], while Ghahramani & Jordan extend the unconstrained HMM structure to handle a distributed hidden state representation [18].

In November of 2017, Bacciu proposed Hidden Tree Markov Models, which combine a tree-based generalization of HMMs with LSTM-based neural networks [4]. This work may have laid the foundation for future extensions of HMMs to modern computer vision tasks.

3.4. Causality models

Causal-effect reasoning has not yet become widely studied in the context of computer vision, but the current applications suggest its promise in future tasks. This set of methods attempt to explore how hidden states cause observed actions. Bayesian networks, for instance, represent sets of variables and their conditional dependencies using a directed acyclic graph [19], [40]. By contrast, grammar models generate hypothesis spaces of possible causal networks at a more abstract level [20].

In [30], a grammar model structure is used to learn containment relations in videos: for example, where objects are relative to "container" objects, such as a pizza in a box. Xu et al. utilize grammar models to re-identify humans in images based on reference images of the individual [56]. Here, the grammar model structure is used in the sense that human bodies are decomposed into body-part templates, and candidate matches are compared across images. Earlier this year, Fang et al. proposed a more physics-oriented pose grammar to obtain improved results on the task of 3D pose estimation [12]. In this, the pose grammar network encodes human body dependencies and relations with respect to kinematics, symmetry, and motor coordination, with each grammar itself represented as a bi-directional RNN. In the future, grammar models in the form of hierarchical recurrent transformer networks as applied in [57] may be utilized for more physics-based tasks.

A successful, recent method for object tracking is [42], published in March 2018. The motivation for this work comes largely from the work of Fire & Zhu, who in [15] describe a causal grammar model to learn the status of an object from video data, such as inferring whether a light is turned on or off. Fire & Zhu describe Causal And-Or Graphs for object status, noting that a light can be in the "on" state because it was already on, or because an agent in the video turned the light on. Refer to Figure 3 for a visual representation of a Causal And-Or Graph.

Qin et al. were the first to apply Causal And-Or Graph models to the task of tracking humans in video, when the humans are interacting with other objects and subjects in complex scenes [42]. The complete trajectory of an object is a function of the subject's surroundings. Thus it is feasible that physical effects on an object can be represented in a similar fashion.

3.5. Graph-based models

More generally, one might consider complex systems which are represented by a broader range of graphs-whether or not they are acyclic or cyclic, undirected or di-

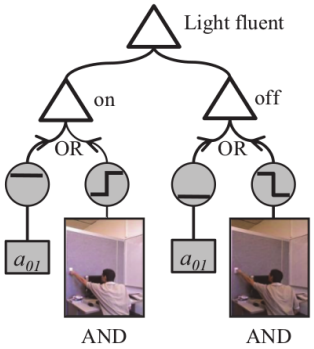

Figure 3. A basic Causal And-Or Graph. The light can be in the "on" state because it was already on, or because an agent in the video turned the light on. Image courtesy of [15].

rected, and so on. The Graph Neural Network model was initially introduced in [45], and ultimately inspired the Interactions Networks model [5], which has proven to be a natural framework for expressing physical relationships between objects. "Graph neural networks" are those frameworks that reason about graphs using neural networks by sharing learning across nodes and edges.

In [5], Battaglia et al. suggest that humans learn to interact with their physical environments by decomposing the setting into distinct objects and relations between the objects, and then reasoning about the consequences of their interactions and dynamics. Thus, the interaction networks are designed such that the reasoning about objects and the reasoning about the relations are assigned to distinct models. Interaction networks are also designed such that learning is automatically generalized across a variable number of objects and relations. Consider Figure 4 for a schematic and description of two simple interaction networks.

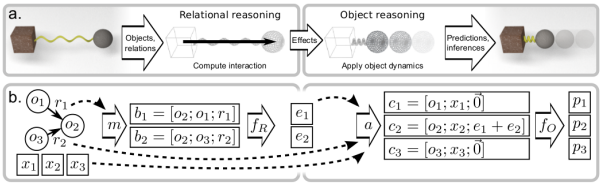

Figure 4. Schematic of the interaction network architecture. *a.* In a simple system, the objects and relations are passed to the network as input. The network first reasons about their interactions, and then applies this information and physical dynamics to make predictions of the system. *b.* For more complex systems, the network takes in systems of objects and relations between them. For more details, refer to [5].

The natural extension of interaction networks to computer vision tasks is the visual interaction network (VIN), as first described by Watters et al. in [53]. Visual interaction networks consist of a perceptual front-end based on convolutional neural networks followed by the interaction networks for discovering the system dynamics and producing state predictions. Through joint training, the raw visual input to the system is converted into the object-relations systems which can be utilized by the interaction network, as described previously. Watters et al. demonstrate that using only a few frames of input data, the VIN can generate accurate trajectories across hundreds of future timesteps for physical systems involving springs, gravity, billiard dynamics, drift, variable mass, and more. Refer to Figure 5 to view examples of predicted billiard ball trajectories produced by the VIN across a range of billiard table geometries.

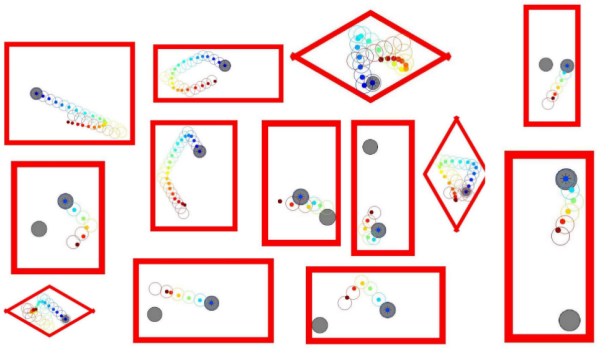

Figure 5. Predictions of billiard ball positions produced by the Visual Interaction Network method. Billiard physics are described by a rich dynamical model, and the restriction of the motion to a 2D table enables the direct application of convolutional neural networks. Image courtesy of [53].

3.6. Constraining search space using physics simulators

An alternative method for capturing the approximate physics of a system involves utilizing existing physical models to constrain the learned physical model. Such methods share a similar set of limitations to the parameter estimation methods described previously, because it is unlikely that a physical model is available for general tasks. However, some flexibility is gained in the sense that a wider constraint space may be specified than in the case where the simulation itself is wholly governed by the physics engine.

For a nice introduction to constraint learning, we refer the reader to [46] by Stewart & Ermon. Here, the constraints that are defined are those that hold over the output space of the neural network in the context of computer vision tasks. If the objective is to track a falling pillow across several video frames, as in one of their examples, then the equation for an object acting under gravity may be encoded, and a special loss function is fixed to ensure that the (convolutional) neural network is optimized over a space of possible functions specifying a hypothesis class related to this dynamic equation. That is, the network is trained in the direction of better satisfying this physical equation. The method is also applied to tasks such as tracking human motion in

video and detecting objects with causal relationships. Constraint learning is especially beneficial for tasks with scarce labeled data.

4. Visual predictive models

In the previous sections, we have summarized a number of ways for constructing the internal physical model of dynamical systems. We now consider the application of such models to visual prediction: that is, using raw visual input to predict future states of the system. This is a challenging task: in complex environments, models of both the agents and their surrounding environments are necessary. Thus, modern visual predictive methods rely upon deep neural networks for their ability to generalize more broadly to a wide variety of environments.

We have already briefly introduced the visual predictive method known as Visual Interaction Networks [52] by Watters et al. Recall that VINs discover systems of objects and relations between the objects using raw visual input, and combines models of relational and object reasoning to predict the future states of the overall system. Such a method might adequately capture the dynamics of the agents, yet fail to generalize well to novel environments. The challenge in developing new visual predictive models arises due to the necessity of simultaneously learning both specific physical relationships and the ways in which broad classes of environments may act on agents.

One recent method which seeks to overcome this dual challenge is presented in [16] by Fragkiadaki et al. In order to handle a variety of environments and situations, the visual predictive models presented here present an *object-centric* approach, rather than a *frame-centric* approach, as is commonly employed in standard computer vision applications (e.g. typical object detection methods). This line of reasoning closely resembles Battaglia's object-oriented approach described in [5], or Xu et al.'s object-oriented grammar model described in [56], though neither was explicitly referenced by Fragkiadaki et al. The object-centric approach, in contrast with the frame-centric approach, is depicted in Figure 6. The future states of the world are determined by individually modeling the temporal evolution of each object in the system from object-centric glimpses. In this way, the model is able to handle translation invariance of the objects as well as model sharing, spanning novel environments and object instances.

In[16], the authors test their method on a simulated billiards game, which is a dynamical system that exhibits relatively complex dynamics yet is constrained to a 2D surface. Different numbers of balls and wall geometries are considered, ensuring generalization to novel settings. The visual predictive model itself takes in as input four previous glimpses of the table centered at each object's position, applied forces, and the previous LSTM hidden states. The

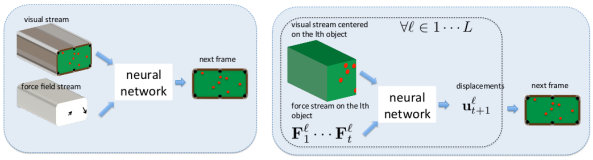

Figure 6. Frame-centric versus object-centric prediction for modeling billiard dynamics. *Left:* The frame-centric approach, which requires the image of the billiards table as a whole, plus forces on the ball objects, to be taken as input. *Right:* The object-centric approach, which models future states of the table by individually modeling the temporal evolution of each ball. In the billiards world with L objects, it is necessary to discover the velocities, u, for making predictions of future states. For more details, refer to [16].

network outputs the displacement of each object. Refer to Figure 7 for a schematic of the network architecture. Experimental results demonstrate good generalization of the method to varying numbers of billiard balls and novel geometries of the billiard table.

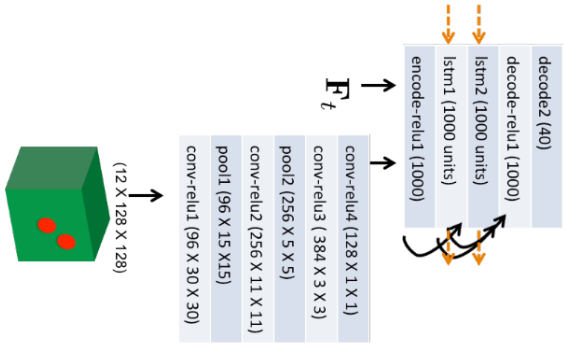

Figure 7. Schematic of the visual predictive model as applied to billiard dynamics. The model takes as input four previous time-step glimpses centered on the object position, the forces acting on the object, and the previous LSTM hidden states. For more details, refer to [16].

The network uses an AlexNet-style architecture, with layer 1 adapted to process a stack of four frames, and conv4 modified such that the output matches the value range of the applied forces, which are joined in via a fully-connected layer. Long-range temporal dynamics are then modeled by two LSTM units, and finally the output is passed into a decoder to predict billiard ball velocities. A Euclidean loss between ground-truth and predicted ball velocity over h timesteps is minimized. We study how well the model generalizes by comparing performance on a variety of billiard table geometries and number of balls. Some results are displayed in Figures 8 and 9, which were taken from [16].

The same research group proposed the Encoder-Recurrent-Decoder (ERD) model for learning even more complex phenomena, such as the movement of a person,

Time	Overall Error			Error Near Collisions		
	CV	FC	OC	CV	FC	OC
t+1	3.0^o/0.00	6.2^o/0.04	5.1^o/0.03	23.2^o/0.00	11.4^o/0.06	9.8^o/0.04
t+5	11.8^o/0.01	8.7^o/0.05	7.2^o/0.04	56.6^o/0.05	21.1^o/0.12	17.9^o/0.10
t+20	45.3^o/0.01	16.3^o/0.09	14.8^o/0.09	123.0^o/0.04	54.8^o/0.20	54.8^o/0.20

Figure 8. Errors in magnitude and angle of the velocity, as reported in a°/b, with a the mean of angular error in degrees, and b the relative error in the magnitude of predicted velocity. The Overall Error is averaged across all frames, while the Error Near Collisions is reported within $\pm$ frames of a collision event. Constant Velocity (CV), Frame Centric (FC), and Object Centric (OC) models are considered. For more details, refer to [16].

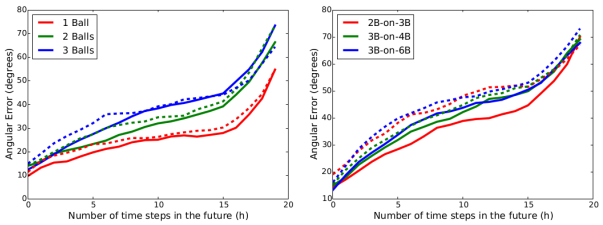

Figure 9. The angular error near collisions is measured for Frame-Centric (dashed line) and Object-Centric (solid line) methods, with h=20. *Left.* Models are both trained and tested on 1, 2, and 3 ball worlds. *Right.* Models trained on 2- and 3- ball worlds and tested on worlds with larger numbers of balls. This demonstrates that the object-centric model generalizes better than the frame-centric model. For more details, refer to [16].

compared to simpler phenomena such as the motion of physical objects[17]. This brings an additional degree of complexity to the model: now, it becomes necessary to anticipate a wide variety of future behavior. However, the ERDs do capture system dynamics in an object-centric fashion: future outcomes are conditioned on an object tracklet, rather than variations at a particular pixel location. Thus for predictions of human kinematics, the visual predictive model is said to be "Lagrangian" rather than "Eulerian" in nature, and realistic human motions were predicted for longer periods of time than a standard LSTM-based method.

Finn et al. tackle the problem of predicting physical motion without labels in [13]. Given videos of agents interacting with various objects, long-term predictions are made by using variations of a dynamic neural advection (DNA) model, which outputs a distribution over locations in the previous frame for each pixel in the new frame. Extended versions of the DNA method utilize convolutional operations and spatial transformers to handle separate objects, like the object-centric methods described previously. It remains to be seen whether more recent methods for handling long-range interactions across frames, such as non-local neural networks [50], will improve performance on prediction tasks involving dynamical systems.

Surprisingly, despite the recent emphasis on object-centric dynamics, no vision-based methods have been published which explicitly handle the tasks of predicting or controlling swarm behavior. Collective dynamics can be observed across nature and technology: for example, crowds of pedestrians, swimming shoals of fish, the migration of cells (see Figure 10), swarm robotics, and even seismic networks. Swarm behavior is well studied in the mathematical sciences, and models are typically object-centric, with relatively simple relations fixed between individual agents to control the group behavior. Thus, object-centric visual prediction methods seem to provide the most natural framework for learning swarming dynamics. Just as in the visual interaction network method, one could utilize convolutional neural networks to identify agents from raw visual input (which in this case could be pedestrians, fish, cells, etc.); then, the internal interaction network could learn the relation dynamics between individual objects. We see many potential future applications of swarm engineering, especially in combination with reinforcement learning techniques, which will be described in the following section.

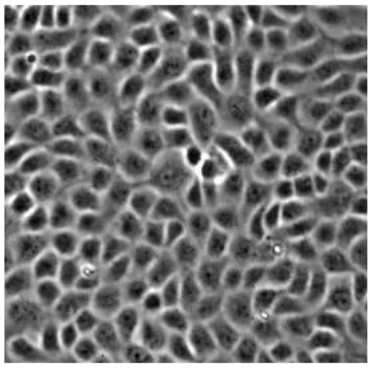

Figure 10. Bioelectric herding of collective cell dynamics is an example of swarm engineering. The ability to more accurately control collective cell migration using DC electric fields, as is done in [10], has potential applications to speeding up wound healing. Swarm dynamics are well-studied in the mathematical community and are well-suited to the object-centric framework which is becoming increasingly utilized for visual prediction. However, we have found no examples of visual prediction or control which address swarm dynamics. Shown here are kidney epithelial cells which can be induced to move back and forth under a DC electric field. Image courtesy of Daniel Cohen, *https://cohengroup.princeton.edu/projects/*.

5. Vision-based control

We have described a number of methods which utilize the feature representations learned by deep neural networks to predict future states of a dynamic system from visual data. There is also considerable interest in developing methods to then *control* such dynamical systems directly from visual input. The earliest example of vision-based control was presented by Mnih et al. in [36], in which a convolutional network used in conjunction with deep Q-learning-

a type of reinforcement learning (RL)- was first utilized to learn to play Atari games. The system architecture was relatively simple: the input to the system is raw pixel values, and the output is a value function estimating future rewards in the game, with the outputs from the convolutional layers taken as inputs to the Q-function. Even with such a straightforward approach, the algorithm was able to outperform human experts in three out of the six games.

The joint implementation of convolutional neural networks and reinforcement learning became widely applied in the robotics community. Lillicrap et al. demonstrated the use of CNNs and Q-learning to simulated robotics environments, and robust performance on 20 simulated physics tasks [31]. The authors note that the Deep-Q Network (DQN) which was utilized by Mnih et al. could only handle discrete and low-dimensional action spaces, even though it could handle high-dimensional observation spaces. Rather than discretize the action space, which would be intractable for problems of high dimensionality, Lillicrap et al. instead utilize an off-policy actor-critic algorithm with deep function approximators. Competitive control policies are found on such physical systems as the driven pendulum, 2D legged locomotion, cartpole swing-up, and so on.

Beyond simulated environments, the CNN/RL combination has also been applied to real-world robotics applications. Levine et al. learn a method which directly maps raw visual observations to torques at a robot's motors [29]. By using a guided policy search method, they are able to transform the policy learning task into a supervised method, and jointly train the perception and control systems in an end-to-end fashion. The robot learns to perform such tasks as hanging a coat hanger on a rack, screwing on a bottle top, and using a hammer to remove a nail.

One major application of joint prediction and control in the field of robotics is the task of navigation in novel environments. Gupta et al. utilize camera input from the perspective of a robot to build a belief map of the robot's environment and to plan a sequence of actions towards goals in that environment [21]. Figure 11 displays the general network architecture: by structuring the network as a sequence of mapping and planning modules, the latent space is able to infer a map of the robot's environment and plan actions accordingly. Even more recent papers, such as [24] by Hong et al., jointly train the perceptive and control networks but link them via a meta-representation; in this case, via visual semantic segmentation.

As research has progressed in vision-based control, we see a consistent use of reinforcement learning, with modifications to the specific algorithm used. The current state-of-the-art RL strategy for vision-based control tasks seems to be A3C, asynchronous advantage actor-critic, which has been proven to be data-efficient and is described in [35]. Meta-representations connecting the perceptive and control

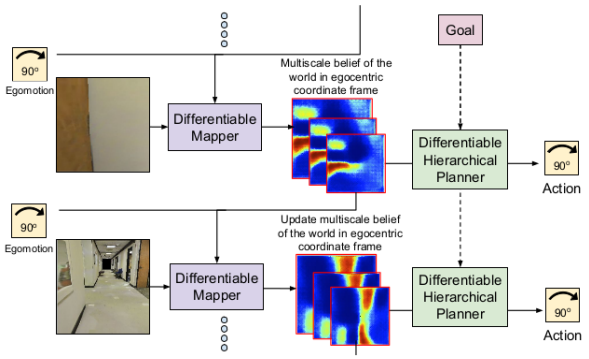

Figure 11. Schematic of the network architecture from [21] consisting of multiple mapping and planning modules. The latent space corresponds to a map of the environment from the robot's perspective. Image courtesy of the authors.

networks will likely continue to be explored, particularly to ease the use of simulators as training platforms for real-world control systems.

6. Multi-modal extensions

In practical applications, it is often the case that multi-modal data is available: for instance, the accessibility of both audio and video data. Having additional "touch" sensory data is essential to robots if they are to interact with physical objects and build an internal representation of the natural world. We here focus on one example of such a multi-modal extension. Agrawal et al. investigate an experiential learning paradigm in which robots are able to freely poke objects in order to learn to displace objects to target locations [3]. This paper is the intellectual heir of earlier papers which utilize convolutional neural networks to teach robots how to interact with physical objects, such as [41] by Pinto & Gupta, in which the robot learned to grasp objects following 700 hours of trial-and-error grasping attempts. However, this work relies upon self-supervision to develop policies to accomplish a single fixed task, while the work in [3] learns a more general predictive model of the physics which can be utilized later to accomplish a variety of tasks. In this case, the robot is confronted with objects on a table, and records a visual state of the world before and after randomly performing an action (see Figure 12). In order to learn the mapping from one visual state to the next, *the forward and inverse dynamics models are jointly learned*. That is, a forward model predicts the next state given the current state and an action, while the inverse model predicts the action given the initial and target states. Thus the two models provide supervision for one another.

A schematic of the network architecture for learning to poke by poking is shown in Figure 13. One training sample is composed of the initial snapshot of the table, I_t, the snapshot of the table after the action is performed, I_{t+1}, and the

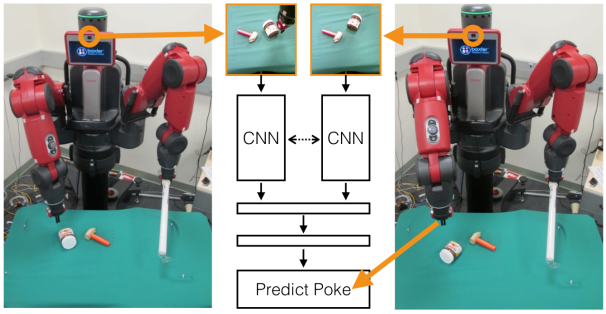

Figure 12. Schematic of the neural network setup for learning to poke by poking using a Baxter robot. The robot pokes objects on a table and records the visual state of the table before and after performing the action. A mapping is learned between the applied poke and the corresponding change in visual state. After 400 hours of random pokes, the robot learns to push objects to their desired locations. Image courtesy of [3].

action, u_t. The images (I_t, I_{t+1}) are passed into a Siamese convolutional network in order to learn their latent feature representations (x_t, x_{t+1}), which are concatenated and passed through fully convolutional layers in order to predict the poke location, angle, and length conditionally. Thus the inverse model can be learned. Additionally, the poke location, angle and length are discretized into a number of bins for modeling multimodal poke distributions. For building the forward model, the action and initial table state snapshot are passed through a series of fully connected layers to predict the post-action table state.

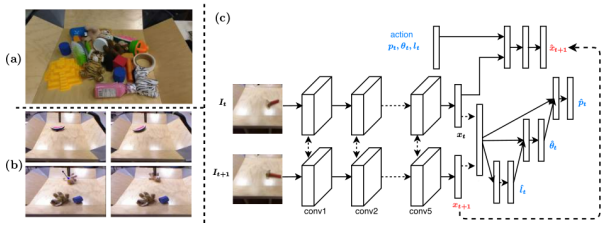

Figure 13. (a) The set of objects which were poked by the Baxter robot. (b) Pairs of images of the table state before and after the poking action was performed. (c) The schematic of the neural network architecture. The forward and inverse models of the system dynamics are simultaneously learned via a Siamese convolutional network and a series of fully convolutional layers. Image courtesy of [3].

On the task of displacing novel objects to desired locations, the joint modeling approach outperforms alternative methods. Likewise, given only limited training data, the joint method is shown to get closer to the goal in fewer steps than competitive methods (see Figure 14). The ability of this model to generalize well is even more important than small reductions in position error. For most real-world object manipulation tasks, small amounts of error in performed actions is manageable. However, it is essential that performance is robust to a variety of environmental conditions. Thus, the utilization of an "intuitive" model of physics, rather than a simulator-based model, provides benefit in this domain.

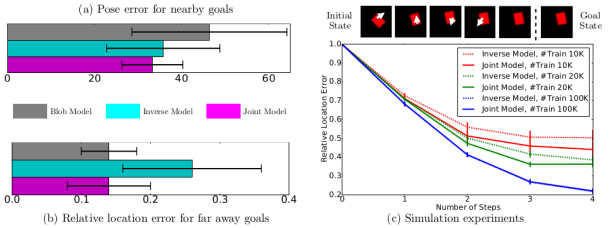

Figure 14. (a) and (b) compare the joint learning model as described in [3] to a baseline blob model, which estimates object locations using a template-based object detector and then uses the vector difference between the before-after image pairs to compute action parameters (poke location, angle, and length). The inverse model alone was also considered. (c) Simulation studies reveal that when fewer training examples are available, the joint model outperforms the inverse model. We infer that the forward model is able to regularize the inverse model. Image courtesy of the authors.

7. Prospective trends and conclusions

In this paper, we have reviewed a number of modeling techniques for learning dynamical systems in isolation, as well as several research directions with respect to visual prediction and control. Convolutional operations and deep neural networks remain essential in this domain.

Prediction. The trend in visual-predictive modeling is to mimic the object-centric manner in which humans identify objects and then construct internal models of relationships between them. The recent Visual Interaction Networks paper [53] illustrates the application of convolutional neural networks for identifying meaningful objects, and then utilizing an internal interaction network for discovering the physical relationships between those objects. We anticipate the utility of recent techniques such as dynamic neural advection models, which ease the prediction of physical motion without labels, or even non-local operations. Furthermore, we anticipate the future study of swarm dynamics using visual predictive models, given the suitability of this application to the recent object-centric frameworks.

Control. With respect to control, reinforcement learning remains the dominant method, with some variation in the specific RL algorithm used. The current SOTA RL strategy seems to be the asynchronous advantage actor-critic method (A3C). Additionally, more effort is being spent on enabling self-supervision. We anticipate improved meta-representations of physical dynamics to enable the substitution of simulated data for expensive real-world interaction, as is emphasized in [3].

References

[1] Binding computer vision to physics based simulation: The case study of a bouncing ball.

[2] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun. Discriminative training of kalman filters. In *Robotics: Science and systems*, volume 2, page 1, 2005.

[3] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016.

[4] D. Bacciu. Hidden tree markov networks: Deep and wide learning for structured data. *CoRR*, abs/1711.07784, 2017.

[5] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016.

[6] K. S. Bhat, S. M. Seitz, and J. Popovic. Computing the physical parameters of rigid-body motion from video. In *Lecture Notes in Computer Science, Heyden, Anders, Sparr, Gunnar, Nielsen, Mads, and Johansen, Peter (eds.), volume 2350*, ECCV (1), pages 551–565. Springer, 2002.

[7] B. Boots, A. Byravan, and D. Fox. Learning predictive models of a depth camera amp; manipulator from raw execution traces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4021–4028, May 2014.

[8] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999, Jun 1997.

[9] M. A. Brubaker, L. Sigal, and D. Fleet. Estimating contact dynamics. In *Computer Vision, 2009 IEEE 12th International Conference*, IEEE, pages 2389–2396, 2009.

[10] D. Cohen, M. M. Maharbiz, and W. James Nelson. Galvanotactic control of collective cell migration in epithelial monolayers. 13, 04 2014.

[11] H. Coskun, F. Achilles, R. S. DiPietro, N. Navab, and F. Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. *CoRR*, abs/1708.01885, 2017.

[12] H. Fang, Y. Xu, W. Wang, X. Liu, and S. Zhu. Learning knowledge-guided pose grammar machine for 3d human pose estimation. *CoRR*, abs/1710.06513, 2017.

[13] C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016.

[14] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR*, abs/1509.06113, 2015.

[15] A. Fire and S.-C. Zhu. Learning perceptual causality from video. *ACM Trans. Intell. Syst. Technol.*, 7(2):23:1–23:22, Nov. 2015.

[16] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. *CoRR*, abs/1511.07404, 2015.

[17] K. Fragkiadaki, S. Levine, and J. Malik. Recurrent network models for kinematic tracking. *CoRR*, abs/1508.00271, 2015.

[18] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2):245–273, Nov 1997.

[19] T. L. Griffiths and J. B. Tenenbaum. Structure and strength in causal induction. *Cognitive psychology*, 51 4:334–84, 2005.

[20] T. L. Griffiths and J. B. Tenenbaum. Two proposals for causal grammars. In A. Gopnik and L. Schulz, editors, *Causal Learning: Psychology, Philosophy, and Computation*, pages 323–345. Oxford University Press, 2007.

[21] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *CoRR*, abs/1702.03920, 2017.

[22] J. Hamrick. Internal physics models guide probabilistic judgments about object dynamics. 01 2011.

[23] M. Haruno, D. Wolpert, and M. Kawato. Mosaic model for sensorimotor learning and control. 13:2201–2220, 10 2001.

[24] Z. Hong, C. Yu-Ming, S. Su, T. Shann, Y. Chang, H. Yang, B. H. Ho, C. Tu, Y. Chang, T. Hsiao, H. Hsiao, S. Lai, and C. Lee. Virtual-to-real: Learning to control in visual semantic segmentation. *CoRR*, abs/1802.00285, 2018.

[25] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354.

[26] T. C. Kietzmann and M. Riedmiller. The neuro slot car racer: Reinforcement learning in a real world setting. In *2009 International Conference on Machine Learning and Applications*, pages 311–316, Dec 2009.

[27] T. Lampe and M. Riedmiller. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Aug 2013.

[28] S. Lange, M. Riedmiller, and A. Voigtlnder. Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, June 2012.

[29] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015.

[30] W. Liang, Y. Zhao, Y. Zhu, and S.-C. Zhu. What is where: Inferring containment relations from videos. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3418–3424. AAAI Press, 2016.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[32] D. M Wolpert, Z. Ghahramani, and M. Jordan. An internal model for sensorimotor integration. 269:1880–2, 10 1995.

[33] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967 – 2986, 2014.

[34] V. Michalski, R. Memisevic, and K. Konda. Modeling deep temporal dependencies with recurrent grammar cells"". In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1925–1933. Curran Associates, Inc., 2014.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[37] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian image understanding: Unfolding the dynamics of objects in static images. *CoRR*, abs/1511.04048, 2015.

[38] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[39] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. P. Singh. Action-conditional video prediction using deep networks in atari games. *CoRR*, abs/1507.08750, 2015.

[40] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

[41] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, abs/1509.06825, 2015.

[42] L. Qin, Y. Xu, X. Liu, and S. Zhu. A causal and-or graph model for visibility fluent reasoning in human-object interactions. *CoRR*, abs/1709.05437, 2017.

[43] U. S. R. G. Krishnan and D. Sontag. Deep kalman filters. *NIPS Workshop on Advances in Approximate Bayesian Inference and Black Box Inference*, abs/1511.05121, 2015.

[44] M. Salzmann and R. Urtasun. Physically-based motion models for 3d tracking: A convex formulation. In *2011 International Conference on Computer Vision*, pages 2064–2071, Nov 2011.

[45] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, Jan 2009.

[46] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. *CoRR*, abs/1609.05566, 2016.

[47] E. Todorov and Z. Ghahramani. Unsupervised learning of sensory-motor primitives. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*, volume 2, pages 1750–1753 Vol.2, Sept 2003.

[48] R. Urtasun, D. J. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 238–245, June 2006.

[49] N. Wahlström, T. B. Schn., and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *CoRR*, abs/1502.02251, 2015.

[50] X. Wang, R. B. Girshick, A. Gupta, and K. He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

[51] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *CoRR*, abs/1506.07365, 2015.

[52] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran. Visual interaction networks. *CoRR*, abs/1706.01433, 2017.

[53] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4539–4547. Curran Associates, Inc., 2017.

[54] S.-K. Weng, C.-M. Kuo, and S.-K. Tu. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190 – 1208, 2006.

[55] J. Wu, I. Yildirim, J. J. Lim, W. T. Freeman, and J. B. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 127–135, Cambridge, MA, USA, 2015. MIT Press.

[56] Y. Xu, L. Lin, W. S. Zheng, and X. Liu. Human re-identification by matching compositional template with cluster sampling. In *2013 IEEE International Conference on Computer Vision*, pages 3152–3159, Dec 2013.

[57] S. Yan, Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Unconstrained fashion landmark detection via hierarchical recurrent transformer networks. *CoRR*, abs/1708.02044, 2017.