Intuitive Physics: Understanding Dynamics in Real World Physical Scenes

Martin Andreev Asenov
s1247380

Robotics Research Report
School of Informatics
University of Edinburgh

2017

Abstract

Intuitive physics, or naive physics, is our ability to predict outcomes of a variety of basic physical interactions and to reason about the dynamics of a scene. It is an important ability people have, a crucial part of our intelligence, that helps us deal with the complexity of the surrounding world. Different fields have tried to answer parts of the many questions that one could ask - what is the underlying mechanism allowing us to do this type of inference, why do we experience certain illusions, how we can mimic our ability in software systems and robots. This report focuses on the third question by analyzing the newest approaches in machine learning and robotics.

Table of Contents

Chapter 1

Introduction

Understanding physics and dynamics is a fundamental part of our intelligence, ability to reason and perform complicated actions. Knowing where objects are going to be in a few moments time is central to evaluation of a situation and reacting accordingly. Examples of this include passing a busy street, catching a frisbee or hitting a tennis ball with a racket [21]. Thus if we want to build software systems or robots able to truly understand the world around them, instead of performing single vision tasks like recognition, segmentation and depth estimation, they need to able to understand the dynamics in the scene and perform inference for short and long term future moments.

Exactly how we do this type of inference about the world has been an active research topic in psychology, philosophy and, with developments in computation and hardware, robotics and machine learning. The ability of humans to predict the outcomes of different physical interactions like watching objects falling, colliding, sliding, etc. has been referred to as naive or intuitive physics. Different studies have argued how these computations could possibly be done in our brain, raising the question if they follow Newtonian mechanics. Due to errors humans experience in their judgment of different scenarios, it was suggested they do not. However recent studies suggest that our brain follows a "noisy Newtonian" framework, where Newton's laws and noisy observations are combined in a probabilistic model [10].

In the next few chapters, we examine different approaches and models developed, aimed to solve the problem of physical inference, based on single or sequences of images. With the development of the topic, two possible solutions emerged - trying to map a scene we observe to a physical engine and infer different latent variables or using purely data-driven approaches to predict a property or future state of the scene based on visual observation. Interesting recent work also tries to combine both of the approaches, limiting their weaknesses and exploiting their strengths.

Figure 1.1: **Dynamic scene from the volleyball finals at Rio 2016.** Even from a static image, people can do a remarkable amount of reasoning of where different people and objects are going to be is subsequent moments. We know that player 4 is going to hit the ball, players 4, 6 and 15 recently jumped and are going to fall now, player 12 is going to the left, player 1 is going to the right and so on.

Chapter 2

Background work

In this section we briefly comment on two developments which gave the necessary tools for conducting the work presented in the next chapter. First, we cover neural networks and more specifically Convolutional Neural Networks, which led to a revolution in image recognition. Secondly, we discuss game engines.

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [18] is a custom type of neural network, designed to mimic how our own visual system works. The first few layers of the CNN are a repeated sequence of convolutions followed by a max-pooling layer. After stacking a couple of these layers, a CNN is usually followed by a few fully connected layers. An example CNN for predicting handwritten digits can be seen in fig. 2.1. CNNs have come a long way since their first successful application in document recognition. By combining efficient GPU calculations [17] and dropout [14], CNN were applied to large-scale image recognition tasks such as ImageNet [8], beating previous results. It's been shown that no matter what output we train for if we are using images as an input to a CNN, the first few layers always seemed to be very similar, resembling Gabor filters and color blobs [27]. Thus taking pre-trained convolutions layers, and training just the fully connected layer on top for the specific task chosen, works well. The stacked convolution/max-pooling layers extract features from a given image, from edges and colors to increasingly more complicated patterns. Therefore we get an effective representation of an image, which can be used for training of any problem we want, using another neural network on top. This method of 'compressing' an image while preserving useful information about it has been used in different methods discussed in the next chapter.

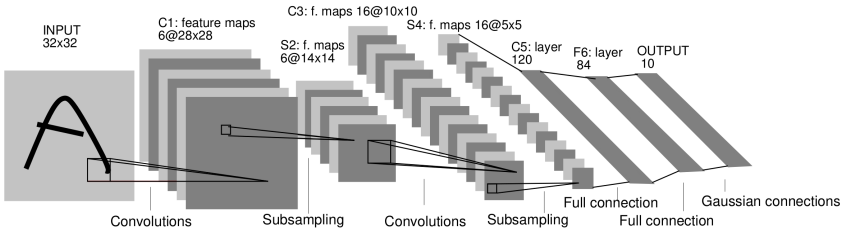
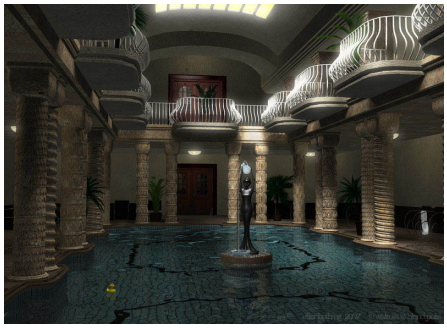


Figure 2.1: **Convolutional Neural Network.** One of the big successes of CNN [18] was automatic recognition of handwritten digits and letters. Applied in post offices all around the world, it is part of the reason why sending letters nowadays is so cheap. Since then CNNs have been applied to many problems in vision and natural languages.

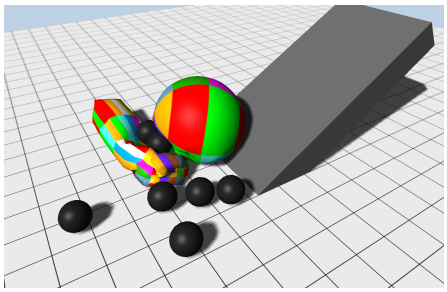
2.2 Game and physics engines

Game engines and computer animation have reached the state where they can be used for generation of lifelike images and videos. As a tool, they can be exploited in a couple of different ways. First of all, since we do the simulation, we have all the information we would want for the scene - current physical properties of all the objects, future and past states, etc. Secondly, we have unlimited exploration possibilities, by generating scenes with new physical properties. Generate a sequence of images as the simulations run, and having ground truth information, is a powerful tool, used in different methods. Examples of sample scenes, generated from different engines can be seen in fig.2.2.

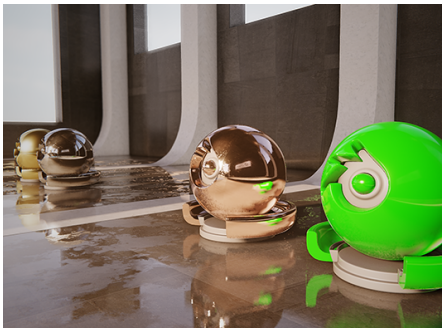


(a)

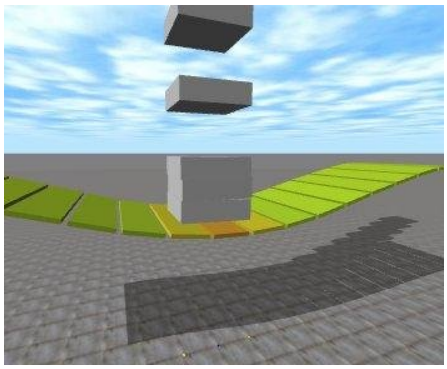
(b)



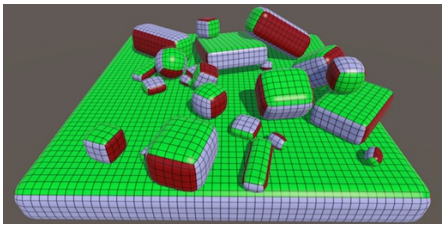
(c)



(d)



(e)



(f)

Figure 2.2: **Different game and physics engines.** (a) Blender [5] (b) Bullet Physics [7] (c) three.js [23] (d) Panda3D [12] (e) Open Dynamics Engine [22] (f) Unity [25]

Chapter 3

Current methods

In this section we explore the state of the art methods for physics understanding in trivial situations. Three methods are examined - inference using physics engines, data-driven approaches using neural networks and a combination between the two. Example problems on which different approaches were developed and tested can be seen in fig. 3.1.

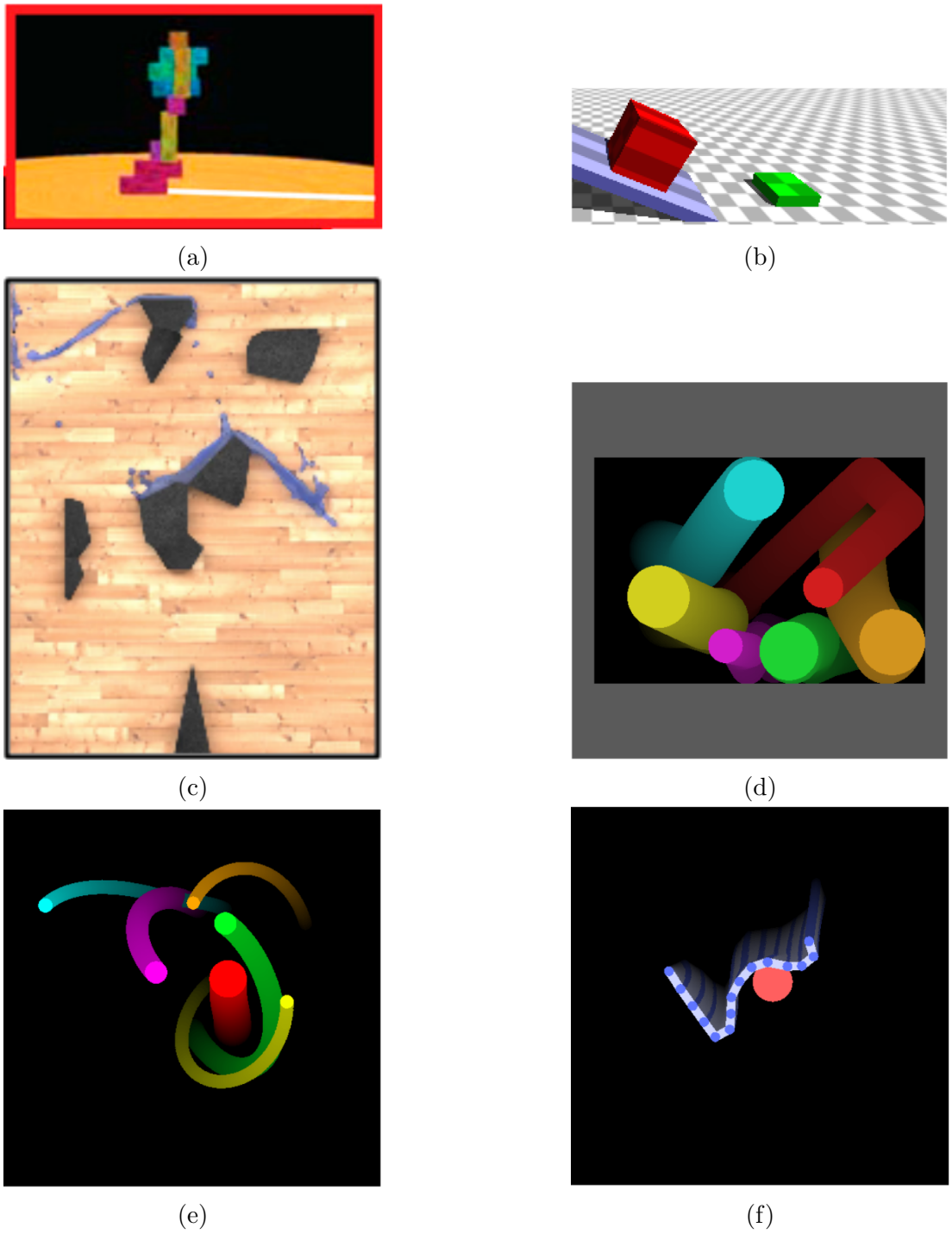


Figure 3.1: **Physical interactions scenarios.** A list of some of the experiments performed for testing different algorithms (a) A stack of 10 cubes - predict if the cubes will fall [4] (b) Sliding objects on inclined plane - infer physical properties and final position of the object [26] (c) Dropping liquid over different obstacles - predict the amount that is going to fall on the left and on the right [2] (d) Bouncing balls in a box - predict the movement of the balls [3] (e) n-body problem - predict the movement of the bodies[3] (f) Springs collide with a rigid body - predict the movement of the spring [3]

3.1 Inference using physics engines

People have a remarkably good intuition about physics and dynamics in different situations. Yet, we often experience different illusions and inaccuracies in our judgment. This lead to the idea that although we follow Newtons' laws, we do it so in a noisy framework [10]. Following that idea and with the development of modern graphics engines, the "intuitive physics engine" (IPE) was introduced in [4]. This work presents a biologically plausible way of doing fast physical inference, based on graphics and simulations tools and Bayesian modeling. Their work shows that even though this differs from ground truth physics, it is remarkably consistent with the intuition people have. It is claimed that probabilistic simulations are key to our efficiency in reason about the dynamic world we live in, even when encountering unknown situation.

The experimental setup in [4] is a tower of 10 randomly stacked blocks, simulated using [22]. The IPE takes as an input a scene configuration, as well as three parameters - σ, noise in the 3D location of the blocks, ϕ, the magnitude of external forces like breeze, vibration or bump and μ, physical properties of the object like mass, elasticity or surface roughness. The output of the IPE for a given scene is calculated by running a number of simulations based on those hyperparameters and averaging their outcome to produce the final result. The IPE performance is compared to people's intuition on two questions about the tower of blocks - is it going to fall and in which direction. Further experiments include varying object masses, object shapes, physical obstacles and applied forces. Several observations were drawn from performing those experiments. The output of the IPE (under the best-fit parameters, $\sigma = 0.2$, $\phi = 0.2$) and people's judgments were highly correlated. On the other side when compared to ground truth physics ($\sigma = 0$, $\phi = 0$) the correlation was much lower (Pearson coefficient of $\rho = 0.64$ for the ground truth physics, against $\rho = 0.92$ for the IPE). This coincides with illusions people experience of surprisingly balanced objects. If we use the ground truth physics model we will predict that the objects will stay balanced. However, both IPE and people's intuition classify those scenarios as highly likely to fall due to the high number of similar unbalanced configuration. The performance of the IPE is compared to a purely model-free method based on regression of 22 geometric features like tower's height, height of the tower's center of mass, average height of a block, etc. This data-driven approach has been shown to perform consistently worse than the IPE model. While the authors don't argue that geometric features are important, from their experiment they can't provide a general-purpose alternative to IPE.

A general disclaimer is made that each physics engine suffers from three fundamental weakness. It is based on simulation, generating every scene from the previous one, rather than calculating analytic solutions. Secondly, the IPE is probabilistic, running a stochastic simulation to represent uncertainty about the scene. Finally, it is also approximate, trading between precision and accuracy.

Building upon this idea, similar methods were used to model the people's intuition

of liquid dynamics [2]. The work made two contributions. They gathered experimental data of how humans reason about liquid dynamics. The work extended the model presented in [4] by adding models of fluids. By simulating liquids the paper was able to test physical properties like stickiness and viscosity hold true within the IFE model. Participants in the experiment with randomly generated 3D scenes with liquid position above randomly generated obstacles. The task was to predict how much liquid would fall in each of the two basins located at the bottom of the obstacles. Two types of liquid were tested - water like and high-viscosity, honey-like. From the experiments conducted, it was shown that simulation models drastically outperform non-simulation models. For the purpose, the authors implemented a simple heuristics model as well as CNN trained on thousand generated scenes. IFE is shown to have the highest correlation with the intuition of the participants in the experiments compared to both of those models, as well as the ground truth physics. Moreover, the stickiness is shown to improve the performance of the model, suggesting that people are sensitive to those properties of the objects when observing them.

Another work aims to develop a generative model for solving the problem of inferring different physical properties of objects and understanding dynamics in videos and static images [26]. Their model consists of three parts. Every object in their model is represented as a rigid body with shape, position in space, mass and friction. Using these properties the object is simulated in a physics engine, in this case, Bullet [7], generating velocities and positions for the object. Finally, objects in real-world videos are tracked and their velocities compared to the ones from the model using a likelihood function. The method is shown to work well for the example problem of sliding objects on an inclined surface. However, the question of how this would scale up remains unanswered.

3.2 Data-driven approaches using deep learning

Methods presented in the previous section try to model the world using a physics engine and try to infer the different parameters through simulation. However, some physical properties could be a lot easier inferred from simple observations. As an example, it might be easier to estimate that a feather will fall slower than a stone, just based on texture, instead of estimating different physical properties. This idea is developed in [1]. The paper discusses an approach for deriving model of intuitive physics by manipulation tasks and learning. The authors use CNNs to predict a poking action based on start and goal state image of objects. The model is learned from repeated simulations beforehand. The authors argue that this is a better approach than "simulator-based" model as in [16], due to the parameter estimation errors and different functional forms of the parameterizations. It is possible that infants learn to do similar inference, based on all the experience they gathered playing with random objects. Finally, the recent successes of neural nets might make this training possible. The two major issues, with a model based on interaction and learning, are the large amounts of experiments required to gather

the necessary data, as well as the difficulty of future visual state predictions. This work tackles these problems by collecting 400 hours of videos of 50K pokes, used for training of two models - for forward and inverse kinematics. Two neural networks are trained together, the first having I_t and I_{t+1} and outputting a_t, the second taking I_{t+1} and a_t and outputting I_t. The network follows a similar approach to [11], where one network tries to fake an image, the other tries to discriminate it. In our case one of the networks tries to come up with the results of an action given current scene, the other tries to predict the action given initial and end state of an image.

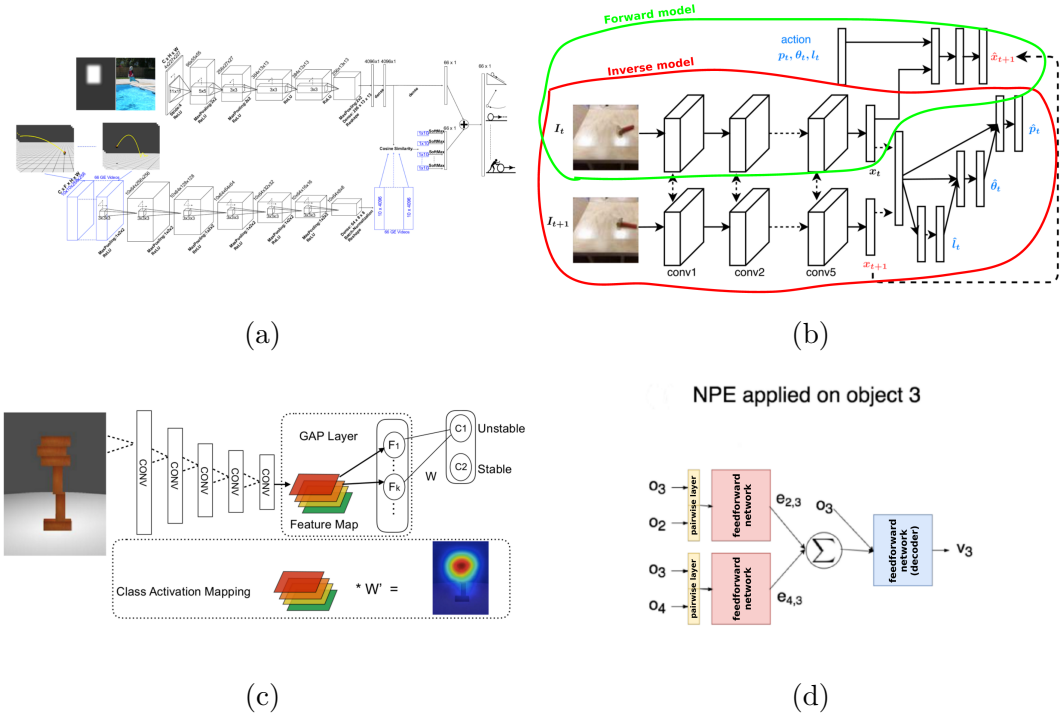


Figure 3.2: **Using neural networks for intuitive physics models.** (a) Mapping between static image (top part of the network) and one of 12 Newtonian scenarios (bottom part of the network) (b) Forward and inverse kinematics. Two neural networks are trained simultaneously. The forward model predicts an image on the next time step, given the image at the current time step and the applied action. The inverse model predicts the applied action based on the images for the current and next time step (c) Predicting stability of tower of blocks only based on the observed image. The network can also show the pixels in the image that contributed the most for making the prediction (d) Modeling physical interactions in a generic way

Another data-driven, non-simulation approach was presented in [19]. Here the task was similar to the one [4] - participants had to predict the stability of stacked blocks. Custom scenes were generated using graphics engines, Bullet [7] and Panda3D [13]. This provided the train and test images for the experiments as well as the ground truth labels (is the scene stable or not). A CNN is trained, on the generated images as an input and the stability labels as an output. A

Global Average Pooling (GAP) layer is added to the feature maps generated by the last convolutional layer, instead of traditional fully-connected layer [20]. This allows for extracting the Class Activation Map (CAM), visualizing regions from the image where an unstable prediction is being made [28]. Those focused regions appear to be correlated to where the fall begins to happen, although not any specific details are given in the paper. Using the classifier it is possible to generate new images, test if they are stable, and use a robot (in this case Baxter) to stake cubes while keeping the balance.

Some work even tackled the complicated problem of approximating dynamics based on a single static image [21]. The authors use 12 Newtonian scenarios (fig.3.3) to represent all dynamics that could happen to an object - sliding, gliding, hanging, lifting, etc. Given an object in an image, the object is mapped into one of the scenarios, after which a physical simulation using game engine is performed. The result of the simulation gives us information how the object in the image would move in the nearby moments. The mapping between an object in an image and one of the Newtonian scenarios is done with a custom neural network architecture, trained on a dataset, VIsual Newtonian Dynamics (VIND), collected by the authors. The data consists of 6000 videos aligned with the Newtonian scenarios and 4500 still images with their accompanying ground truth dynamics. A neural network based on the AlexNet architecture [17] is used for processing the input image and a mask showing the object of interest. Another network based on [24] is used for the videos generated from a game engine based on the different Newtonian scenarios. Both of the networks, through series of convolutions and max pooling, followed by fully-connected layers, generate a latent representation of the query image and the different frames from the game engine videos. Those latent representations are compared using cosine similarity, which maps the image to the most probable Newtonian scenario. This approach is compared to another network, which is trained directly on the images as input and their trajectories in the 3D space. Mapping to Newtonian scenarios and simulating them to predict dynamics is shown to significantly outperform direct mapping between images and trajectories. However, the paper leaves some doubt as not enough details are given for the training of the second network. Although solving the same problem, the two networks have different architectures, using different data and loss function. Thus the comparison between them is somewhat weak, as there are simply too many differences between them and it is not clear what the reason for lower accuracy is. The paper also does not address the problem what happens in more complicated scenarios involving more objects and a different scenario from one of the 12.

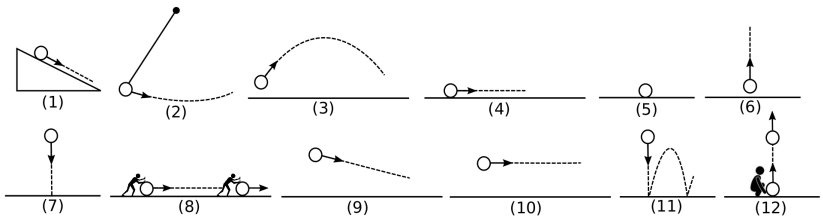


Figure 3.3: **12 Newtonian scenarios.** A lot of basic physics interactions can be represented as one of those 12 scenarios. Using a neural network we can learn a mapping from an image, with specified query object, to one of those scenarios. Then running a short physical simulation, we can predict the future trajectory of the queried object.

Predicting dynamics can be treated as a time series problem and some work using recurrent neural networks addresses this problem [9], in the context of playing billiards. The goal is to predict the future n time steps, based on the last 4 frames of the visual scene of the table together with all the applied forces. A custom CNN is used again, and three different models are compared. The simplest one, constant velocity (CV) model is based on the velocity from the last known time step. The authors argue that since collisions with other balls or hitting a wall are relatively infrequent this is a good baseline. However this does not appear to be the case from the examples given, where we can see the hitting a wall or ball, are indeed fairly often even in the simplest scenarios. Two neural network models are implemented - frame-centric and object-centric. The frame-centric (FC) model takes as an input the visual and force field stream and outputs next frame. The object-centric (OC) model on the other hand individually models every single object, taking visual stream with a mask covering the object and the forces that act on it, for every object on the table. Then the displacements for each object are computed based on which the next frame is generated. The structure of the neural network used is a combination of AlexNet [17] and LSTM [15]. The first few layers are convolutional/max-pooling, following by a fully-connected (encoder) layer and finally two layers of LSTM units. As expected the frame-centric and object-centric neural networks perform significantly better than the constant velocity model, the difference between the two increasing as we predict a further time step. In this model again we can predict desirable actions by generating forces to the different objects, feeding them through the neural network to, generating resulting visual stream, comparing it to the one we desired and picking the best action based on that.

3.3 Learnable physics engines

As [6] summarizes the two commonly used approaches when dealing with problems in intuitive physics. The top-down approach formulates the problem as an inference of physics parameters in a symbolic physics engine. The bottom down

approach, on the other hand, aims to directly match observations to motion predictions or physical properties. Both of the approaches come with their strengths and weaknesses. Top-down approaches can generalize over different scenarios, as long as the physical event observed can fit within the framework. If they don't however, we either might get inaccurate results or modify the physics engine in order to accommodate the new cases. On the other hand, bottom-up scenarios often don't require any structure to be predefined. However, this comes at the cost of reduced generality and difficulty of transfer learning.

The work presented in [3] aims to combine the two approaches by introducing a general approach for a learnable physics engine. The model presented, named *interaction network*, takes graphs as an input and performs object and relation reasoning using neural nets. It is a very general approach, shown to work well in n-body problems, rigid-body collisions and non-rigid dynamics. However, results were only shown in 2D scenarios. The main idea behind the work is to mimic a physical simulator - to generate a sequence of states by repeatedly applying physics laws, taking into account the interaction between the objects, approximating the effect of dynamics over time. Unlike a physics engine, however, those rules are not specified a priori but learned from data as we observe the objects. The core of the network consists of two functions which are learned from training data. ϕ_R, which given all the sets of sender object $o_{1,t}$ and receiver object $o_{2,t}$ and their relation r at time step t, outputs all the effects e_{t+1}. ϕ_O takes an input the receiver object $o_{2,t}$ all computer effects e_{t+1} on him and outputs $o_{2,t+1}$. The network was trained on synthetically generated data representing n-body problems, bouncing balls in a grid and strings. Results were compared against baseline neural net with two 300-length hidden units, taking as in input a flatten vector of all the input data. While alternative models were mentioned, no benchmark comparisons were made with any of them, instead of only using an in-house developed algorithm. Moreover, even for the fairly simple 2D problems posed, the network was trained on fast amounts of data - 2000 scenes over 1000 time steps. No estimates of the amount of data needed for more dynamic interactions and 3D scenarios were made, or if it's going to be possible to extend the model to accommodate the increased complexity. Two suggestions are made for transforming the *IN* into a recurrent neural network or probabilistic generative model to accommodate more accurate long-term predictions or inference about unknown parameters of the objects and relations between them. But again no specific details about how this can be accomplished are given.

Some of those issues were addressed in [6]. Objects are defined in a general way and the relations between them trained from data. In this way we get the modularity of a physics engine and ability to completely train the model from data, without building a physics model a priori. To more formally specify the model, we have all the objects o_1, o_2, o_3, o_4... and they change their state based on some physics program. If we capture all the information about the object such as position, velocity, mass, gravitational and pairwise forces, etc. we can treat the process as Markovian (fig.3.4).

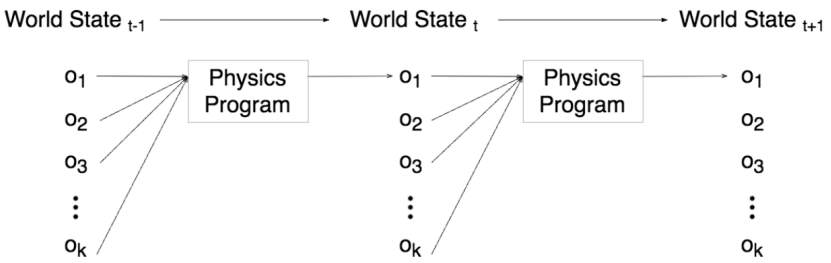


Figure 3.4: **Markovian representation of a physics engine.** [6]

Structuring and learning the interactions between the objects is one of the main contributions in [6]. Three different models are benchmarked - a Neural Physics Engine (NPE), No-Pairwise (NP) and LSTM [15]. Comparison between their structure can be seen in fig.3.5. The NPE model makes three contributions - pairwise factorization, neighborhood mask and function composition. All these contributions aim to make the model general, but learnable at the same time, as well as modular with scalability in mind. The only difference between the NPE and NP models is the removed pairwise factorization in the NP, for benchmarking purposes and evaluation. The networks are compared on a dataset of bouncing balls with obstacles. NPE is shown to outperform both LSTM and NP for tasks like predicting future states and inferring latent properties like mass. The network also deals well with unseen worlds, where it still makes an accurate prediction even when we increase the number of balls or change the obstacles. However, it is worth pointing out that this is still the same scenario, just with an increased number of objects. It will be interesting to see transfer learning between different objects and completely different scenarios.

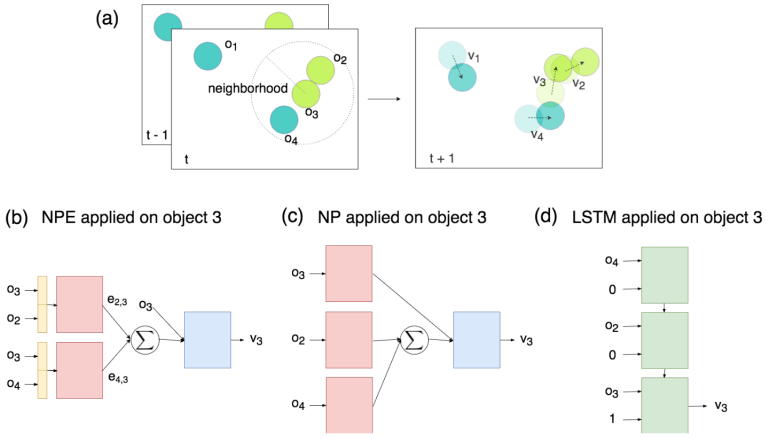


Figure 3.5: **NPE vs NP vs LSTM.**

Chapter 4

Future work

In the previous chapter we described some of the state of the art methods for reasoning about physical scenes. The field has still not managed to address some critical issues. A lot of the work compares the performance to somewhat weak baselines - constant velocity model or simple heuristics. A lot of work claims that method A is better than method B, but when a simulation approach is compared to a neural network, for example, it is not feasible to argue which is better due to their fundamental differences. A neural network may perform more poorly due to the need of more data, a simulation approach may perform better because we are in the domain where those exact scenarios are defined in the engine. Moreover, different papers compare their results to different problems and datasets, making it difficult to have an across-the-board comparison. A lot of the work shows promising results but on small and trivial datasets. The question still remains how this would work in real life scenarios, where dynamics and interactions between objects are often more complicated. A valuable alternative in the future might using some of the newest developed games (fig.4.1), where the scenes presented a lot more life like and complex. It will be interesting to see more hierarchical models, were interactions are learned in an increasingly more complicated fashion, with real knowledge transfer between objects.

(a) (b)

(c) (d)

Figure 4.1: **Gameplay screenshots of GTA V.** The development of computer graphics lead to the development of lifelike games with thousand of objects and possible interactions. Using this games as a basis for running more complicated experiments, might be an interesting area of exploration in the future.

Chapter 5

Summary

Intuitive physics is an exciting area of research with a lot of advances happening in recent years, based on developments from other fields like computer graphics and neural networks. In this report, we explore how we can model naive physics in a computer system. We start by motivating the problem and then proceed to explore current methods and their effectiveness. Three broad approaches are reviewed - inference using a physical engine, data-driven algorithms and recent work combining both. All three of the methods are shown to work well on simple scenarios like bouncing balls and n-body problems. Different implementation for those methods were proposed and tested on variety of problems and datasets. It appears that both structured models with inference of physical properties as well as data-driven approaches could work. However certain some problems still remain open - knowledge transferability, hierarchical representations, scaling up to real-world problems.

Bibliography

[1] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

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

[3] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016.

[4] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam,

[6] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[7] Erwin Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 15, 2013.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[9] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

[10] Tobias Gerstenberg, Noah Goodman, David A Lagnado, and Joshua B Tenenbaum. Noisy newtons: Unifying process and dependency accounts of causal attribution. In *In proceedings of the 34th*. Citeseer, 2012.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Mike Goslin and Mark R. Mine. The panda3d graphics engine. *IEEE Computer*, 37(10):112–114, 2004.

[13] Mike Goslin and Mark R Mine. The panda3d graphics engine. *Computer*, 37(10):112–114, 2004.

[14] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Svetoslav Kolev and Emanuel Todorov. Physically consistent state estimation and system identification for contacts. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 1036–1043. IEEE, 2015.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Wenbin Li, Aleš Leonardis, and Mario Fritz. Visual stability prediction and its application to manipulation. *arXiv preprint arXiv:1609.04861*, 2016.

[20] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[21] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3521–3529, 2016.

[22] Russell Smith et al. Open dynamics engine. 2005.

[23] three.js. three.js / editor, 2015.

[24] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[25] Unity Technologies. *Unity: The leading global game industry software*. Unity Technologies, Unity Technologies. 30 3rd Street. San Francisco, CA 94103. United States.,

[26] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in neural information processing systems*, pages 127–135, 2015.

[27] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[28] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.