
THE UNIVERSITY *of* EDINBURGH

THE UNIVERSITY *of* EDINBURGH

ipab | Institute of Perception, Action and Behaviour

# Visual System Identification: Learning Physical Parameters and Latent Spaces from Pixels


**Author** Miguel Jaques[1]
**Submitted** Januray 7th 2022

**Supervisors** Prof. T. Hospedales[1]
Prof. C. K. Williams[1]
Dr. M. Burke[2]

[1]University of Edinburgh, UK.
[2]Monash University, Australia.


A thesis submitted in fulfilment of the requirements for the degree of *Philosophiae Doctor in Data Science*


Institute of Perception, Action and Behaviour
School of Informatics

University of Edinburgh


2022

Institute of Perception, Action and Behaviour,
School of Informatics,
University of Edinburgh,
10 Crichton Street,
Edinburgh,
EH8 9AB

*To my grandmother Zília.*

# Declaration

I declare that this thesis was composed by me, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

# Acknowledgements

I would like to start by thanking Tim Hospedales, my main supervisor, for guiding me through the process of becoming a (somewhat) competent researcher. He has taught me not how to think, but how to do research. This includes all those things that make ninety percent of our work, like how to frame an idea against the existing literature, how to write a paper that is pleasant to look at and read, how to write a rebuttal, or how to turn a mediocre idea into a published paper. Being a PhD student is a unique experience in the challenges it poses, and having an experienced and supportive mentor to guide me was key to get me to this point.

I would like to give a special thank to Michael Burke, who essentially carried me through most of this PhD. He was instrumental at every stage, being always available for a quick chat about new ideas, introducing me to the world of control and robotics, and helping on day-to-day research issues. I am sure I would not have produced the same quality and quantity of work had it not been for Michael, and I am grateful we got to collaborate in the last 3 years.

Naturally, a PhD is not bearable without friends, so I must thank those who kept me sane through this journey, in particular Melo, Cal, Capinha, Manel, Tomás, Fidelma, Filip, Diana, James, Mavi, Paul, Ben, Luke, Etienne, Todor, Nick, Juozas and Allie. I am lucky to be surrounded by the most wholesome group of people, who challenge me to be better and are always there to listen to my never ending rants or yet-another-project ideas. I guess the real treasure of the PhD is the friends we make along the way, and I can only hope I can continue to be there for you the same way you have been there for me.

Last but not least, I am forever indebted to my all family, whose effort and sacrifice is what has enabled me to follow my dreams and pursue my goals so freely throughout my life. They have supported me through all my adventures and misadventures, and I am certain I would not be where I am today without them.

# Abstract

*In this thesis, we develop machine learning systems that are able to leverage the knowledge of equations of motion (scene-specific or scene-agnostic) to perform object discovery, physical parameter estimation, position and velocity estimation, camera pose estimation, and learn structured latent spaces that satisfy physical dynamics rules. These systems are unsupervised, learning from unlabelled videos, and use as inductive biases the general equations of motion followed by objects of interest in the scene. This is an important task as in many complex real world environments ground-truth states are not available, although there is physical knowledge of the underlying system. Our goals with this approach, i.e. integration of physics knowledge with unsupervised learning models, are to improve vision-based prediction, enable new forms of control, increase data-efficiency and provide model interpretability, all of which are key areas of interest in machine learning. With the above goals in mind, we start by asking the following question: given a scene in which the objects' motions are known up to some physical parameters (e.g. a ball bouncing off the floor with unknown restitution coefficient), how do we build a model that uses such knowledge to discover the objects in the scene and estimate these physical parameters?*

*Our first model, PAIG (Physics-as-Inverse-Graphics), approaches this problem from a vision-as-inverse-graphics perspective, describing the visual scene as a composition of objects defined by their location and appearance, which are rendered onto the frame in a graphics manner. This is a known approach in the unsupervised learning literature, where the fundamental problem then becomes that of derendering, that is, inferring and discovering these locations and appearances for each object. In PAIG we introduce a key rendering component, the Coordinate-Consistent Decoder, which enables the integration of the known equations of motion with an inverse-graphics autoencoder architecture (trainable end-to-end), to perform simultaneous object discovery and physical parameter estimation. Although trained on simple simulated 2D scenes, we show that knowledge of the physical equations of motion of the objects in the scene can be used to greatly improve future prediction and provide physical scene interpretability.*

*Our second model, V-SysId, tackles the limitations shown by the PAIG architecture, namely the training difficulty, the restriction to simulated 2D scenes, and the need for noiseless scenes without distractors. Here, we approach the problem from first principles by asking the question: are neural networks a necessary component to solve this problem? Can we use simpler ideas from classical computer vision instead? With V-*

*SysId, we approach the problem of object discovery and physical parameter estimation from a keypoint extraction, tracking and selection perspective, composed of 3 separate stages: proposal keypoint extraction and tracking, 3D equation fitting and camera pose estimation from 2D trajectories, and entropy-based trajectory selection. Since all the stages use lightweight algorithms and optimisers, V-SysId is able to perform joint object discovery, physical parameter and camera pose estimation from even a single video, drastically improving data-efficiency. Additionally, due to the fact that it does not use a rendering/derendering approach, it can be used in real 3D scenes with many distractor objects. We show that this approach enables a number of interest applications, such as vision-based robot end-effector localisation and remote breath rate measurement.*

*Finally, we move into the area of structured recurrent variational models from vision, where we are motivated by the following observation: in existing models, applying a force in the direction from a start point and an end point (in latent space), does not result in a movement from the start point towards the end point, even on the simplest unconstrained environments. This means that the latent space learned by these models does not follow Newton's law, where the acceleration vector has the same direction as the force vector (in point-mass systems), and prevents the use of PID controllers, which are the simplest and most well understood type of controller. We solve this problem by building inductive biases from Newtonian physics into the latent variable model, which we call NewtonianVAE. Crucially, Newtonian correctness in the latent space brings about the ability to perform proportional (or PID) control, as opposed to the more computationally expensive model predictive control (MPC). PID controllers are ubiquitous in industrial applications, but had thus far lacked integration with unsupervised vision models. We show that the NewtonianVAE learns physically correct latent spaces in simulated 2D and 3D control systems, which can be used to perform goal-based discovery and control in imitation learning, and path following via Dynamic Motion Primitives.*

# Lay Summary

*Machine learning is a field at the intersection of computer science, statistics, and mathematics, that aims to allow computer algorithms to learn from data, rather than behaving according to human-specified rules. Due to the rapid increase in compute power and the availability of ever larger amounts of data, researchers have been able to develop machine learning models that are able to identify objects in an image, understand actions in a video, transcribe audio to text, translate text between languages, etc. We focus particularly on computer vision, an area of artificial intelligence that tries to model our visual cortex, more specifically, the ability to go from a raw visual input that hits our retina to object concepts, whose appearance and motion can be used to inform our actions. In computer vision, the retinal input equivalent is the data coming from cameras (photos or videos), which consist of grids of RGB pixels that are then passed to the machine learning models.*

*Despite rapid developments in the machine learning and computer vision literature, artificial intelligence systems still lack the ability to reason about the way objects move in the world the way we humans do: accurately and intuitively. Existing algorithms require thousands, if not millions of videos to understand how objects move, and even then they fail to perform accurately when they encounter scenarios that are unusual, but easy to reason about for humans. For example, while I have never seen a tree-sized volleyball being thrown in the air, I am confident that I would be able to predict its motion, because I know that objects thrown in the air follow a common trajectory. That is, I am able to predict not because I have seen a visual scenario like this before, but because I have an intuitive understanding of how objects move, regardless of their exact shape or size. This is a crucial distinction, as existing computer vision models tend to rely on the raw input they see ("I have never seen a round object this big, so I don't know what it is"), rather than higher level concepts ("I have never seen a round object this big, but I know round objects tend to follow a certain type of trajectory").*

*In this thesis, we incorporate knowledge of how objects move in the world (i.e. physics) with computer vision systems, in order to develop models that are able to predict, and act on, the physical world around us. We introduce 3 different models, aimed at different faculties of physics understanding from video. The first two models, PAIG and V-SysId, try to discover and learn physical characteristics of objects in a scene, e.g., "find and determined how elastic is the object bouncing on the floor in this video." The third model, NewtonianVAE, focuses on providing vision models with an intuitive understanding of*

*how objects move when acted upon in the real world, e.g., "how does this cube laying on the table move when I push it along this direction?" Both of these areas of development are crucial in order to obtain artificial systems that can learn more from less data, and whose actions we can rely on even on rare or unseen scenarios.*

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

**Thesis statement:** When we have knowledge about the underlying mechanics of a system, imposing physics-based constraints on unsupervised computer vision models improves interpretability, generalisation and data-efficiency.

## 1.1 Why integrate physics and vision?

The human brain has an extraordinary ability to navigate and interact with the visual world. Light stimulus in the eyes is processed by the visual cortex, which turns raw input into scene components that can be used by other parts of the brain to perform mental reasoning at the concept level. This higher level reasoning can be used to imagine hypotetical situations, such as when we ask ourselves *"what would happen if that boulder fell down the hill?"*, or as a means to perform dynamical tasks in the real world, like catching an incoming frisbee, riding a bicycle through traffic, or chopping vegetables in quick succession.

Endowing autonomous agents with the ability to perform such tasks involves creating accurate and reliable models of the world, which is a fundamental problem tackled by many scientific areas, including physics, applied mathematics, engineering, and machine learning. In machine learning, specifically, modelling physical systems has 2 main goals. Firstly, it allows us to make predictions about the system's evolution given some observations. This is crucial for being able to perform actions that lead to a desired outcome. This applies both to model-aided human actions, like planning for floods in riverside regions according to weather predictions, and autonomous agents' actions, like a self-driving car driving safely through a busy intersection. Secondly, it provides insights about the system being studied. If we model a system according to some physical equations, e.g., modelling the motion of a ball bouncing off the floor with unknown mass and restitution coefficient, fitting the equations to the observations will give us an understanding of the particular characteristics of the system.

From a computational point of view, performing actions that lead to a desired outcome in response to visual input involves, broadly, 3 steps: **low-level visual processing**, **object state and property inference**, and **control**[1]. Though both low-level visual processing and control are fascinating fields

[1]Even though in the human brain these processes are not necessarily performed in a rigid sequence - as prior knowledge of an object's location can help inform low-level visual processing - this conceptual decoupling allows us to study specific stages in greater depth while assuming the others are fixed. However, as shown by our NewtonianVAE model in Chapter 5, object inference can be tied to and optimised for specific forms of control.

with active research, the work presented in this thesis focuses on the learning and inference of object states and properties, with particular emphasis on physically interpretable properties such as position, velocity, mass, etc., using structured deep learning models. We see 3 important benefits derived from the integration of physical inductive biases[2] in dynamical systems from vision: improving interpretability, improving generalisation, and aiding remote measurement of physical quantities of interest in real systems.

**Improved interpretability** Despite the ample debate on the trade-off between interpretability and expressiveness of machine learning models, interpretability is a property most researchers would agree is of major importance, as it provides a human inspector with insight into the decision process of a particular agent. This applies not only to computer vision, but to other data modalities as well, as it is a topic of great interest to the community. In visual dynamical systems, adding appropriate physical inductive biases to typically black-box neural network-based models (which provide low interpretability) allows us to encourage explicit representations of object positions, velocities and other physical properties without sacrificing performance (see Chapters 3-5).

**Improved generalisation** Since objects in the world move and interact according to physical laws, it is only natural that building such inductive biases into machine learning models will allow generalisation far beyond what purely data-driven fitting of black-box models would allow. For example, we all have an intuitive understanding that all balls hitting the ground will bounce in a similar fashion, since the underlying interaction between the ball and the floor is always the same (ignoring noise factors). Even though I have never seen a tennis ball the size of a building hitting the ground, I am confident in my ability to predict its trajectory if I ever were to encounter such scenario. A completely general machine learning system (e.g. a vanilla neural network) that was fit on realistic balls and their respective trajectories would not be able to generalise to the thought experiment above because it is well out of the observed distribution. An explicit understanding of world physics (see Chapter 3) will aid models in generalising better to out of distribution scenarios, as they will not rely exclusively on the statistics of their training set, but rather will use physical inductive biases as a means of generalisation.

**Aiding remote measurement systems** Understanding real physical systems is inherently tied to understanding the physics behind them. The experimental lab work one does as a physics undergraduate, measuring rigid body coefficients, fluid viscosity, electromagnetic charges, etc., is a simple example of the process that happens in every field of engineering. While physics famously addresses the discovery of new laws, engineering applications are most often concerned with fitting parameters of known laws in a given scenario.

[2] In the context of this thesis, the term *inductive bias* encompasses the choices made in the model/neural network architecture that encourage the learned latent/hidden representations to have some set of desired characteristics. For example, imposing a bottleneck in a neural network architecture in order to encourage a set of latent variables to represent position and velocity in a disentangled manner would constitute an inductive bias. Crucially, here we do not consider the use of loss functions to impose such structure as an inductive bias.

A machine learning system with the ability to infer physical quantities of interest according to equations describing a system, from vision, would allow human operators to obtain insight directly from images (see Chapters 3 and 4), without having to first manually convert the signal found in images to physical coordinates (in the general sense of the word), and then fit the appropriate equations. Indeed, there is research in remote measuring of physical quantities from vision, either for educational or engineering purposes (e.g. Torres et al. (2016)), though not necessarily from a machine learning perspective.

**Motivation summary** While there has been a lot of recent research in integrating physics with machine learning models (see Willard et al. (2020) for a recent review of real world applications), the applications to and integration with vision models, particularly neural networks, have lagged behind. Advancing this line of research was the central tenet of the work developed by my collaborators and I, whose output, and historical context, is compiled in this thesis.

## 1.2 Papers included in this thesis

This thesis compiles the 3 major pieces of work done during this PhD:

- **Physics-as-Inverse-Graphics: Unsupervised Physical Parameter Estimation from Pixels**. Miguel Jaques, Michael Burke, Timothy Hospedales. Published at the International Conference of Learning Representations (ICLR), 2020.
- **NewtonianVAE: Proportional Control and Goal Identification from Pixels via Physical Latent Spaces**. Miguel Jaques, Michael Burke, Timothy Hospedales. Published at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, with Oral acceptance and shortlisted for Best Paper Award.
- **Vision-based System Identification and 3D Keypoint Discovery using Dynamics Constraints**. Miguel Jaques, Martin Asenov, Michael Burke, Timothy Hospedales. Accepted at the Neural Information Processing Systems (NeurIPS) Workshop on Physical Reasoning and Inductive Biases for the Real World, 2021, with Oral acceptance (Top 3 papers).

For simplicity, throughout this document we will refer to the papers above as PAIG, NewtonianVAE, and V-SysId, respectively.

## 1.3 Structure of this thesis

This thesis is written with a standard Introduction-Background-Method/Experiments-Discussion structure. The main body of contributions (corresponding to the Method/Experiments sections in a normal paper) is split into 2 parts. Part 1 is about physical parameter estimation from vision, and includes the 1st and 3rd paper of the list above. Part 2 is about the integration of physics-based

inductive biases in latent variable models of vision data, and includes the 2nd paper of the list above. The Discussion chapter includes a more comprehensive review of the impact of each work to the field of machine learning at large, and a review of advances in the field after paper publication. Additionally, we propose a number of promising future work directions, in a more comprehensive manner than discussed in the individual papers.

The papers above are presented here with minimal content modifications, each corresponding to a chapter. The main changes relative to the published versions are the incorporation of the appendices' content in the main text, inclusion of some paragraphs that had to be removed in the published version due to paper size constraints in conference venues, and differences in notation where needed. Even though this thesis can be read coherently as a uniform whole, the paper sections contain all the information needed to be read individually without further context.

## 1.4 Contributors

All the contributions presented in this thesis were done in collaboration with Michael Burke and Timothy Hospedales, who contributed with project ideas, guidance, paper writing, proof-reading, and conference rebuttal writing. In addition to project supervision, Michael Burke conducted the Reinforcement Learning baselines present in the papers and collected the data used in the real robot experiments on the V-SysId paper. Martin Asenov contributed to the V-SysId paper by discussing project ideas, developing and generating the simulated bouncing ball dataset, and writing/proof-reading sections of the paper. All other work, including formulation of the mathematical models, running experiments, making visualisations, and paper writing, was carried out by me.

Chapter 2

# Background

Endowing agents with the ability to reason about the behaviour of objects in the real world, and how these objects interact with agents and each other, requires machine learning models that are specifically designed to capture such patterns. Our hypothesis is that learning physical dynamics will benefit from specific architectural inductive biases, the same way that convolutional layers were key for the advancement of vision models, and LSTMs and Transformers were key for the advancement of language models.

Recent years have seen a rapid development of physical systems modelling. Intuitive physics models based on graph networks, differentiable simulators/physics engines, and neural ODEs have witnessed particularly fast progress, allowing ever more complex systems to be modelled with unprecedented accuracy.

As the work in this thesis aims to integrate physical models with computer vision, it is important to understand the developments of both the physical modelling *and* the computer vision literature, particularly around unsupervised object representations. This will allow us to better integrate existing physics models with existing computer vision models, or create new components that enable such integration. This Background chapter aims to provide both an understanding of what physical modelling entails in a machine learning framework, how it is currently used in computer vision, and a review of recent developments.

## Organization of the Background Chapter

The current chapter starts with a brief section establishing some basic definitions of training and inference in dynamical systems, as well as the distinction between *explicit* and *implicit* models in the context of our work.

This is followed by a Preliminaries section, which will introduce a number of foundational topics, including differential equations and numerical integration methods, recurrent latent variable models, and object representations in computer vision, with a focus on vision-as-inverse-graphics and keypoints. We choose to present these at the start, as the concepts therein are used and referenced throughout this thesis. The Preliminaries section is particularly useful for readers unfamiliar with our line of work.

We then provide a comprehensive review of explicit, implicit, and hybrid models. These will go into depth on the literature most relevant to our work within each area, and will be used to motivate and

contextualise the development of the main works in this thesis - PAIG, V-SysId, and NewtonianVAE - presented in later chapters.

### 2.0.1 Dynamical systems: learning, inference, and control

In its simplest formulation, a dynamical model can be defined as a parametric function $f$ that defines a family of discrete-time transitions:

$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \mathbf{u}_t; \boldsymbol{\theta}), \tag{2.1}$$

where $\mathbf{z}_t \in \mathbb{R}^{d_z}$ is the state of the system at time $t$, $\mathbf{u}_t \in \mathbb{R}^{d_u}$ is a possible input/action, and $\boldsymbol{\theta}$ are the model parameters. The state $\mathbf{z}_t$ can include any number of physical quantities, but for simplicity of exposition (and without loss of generality) we will assume it consists of a set of positions and velocities. Naturally, (2.1) can be used in a probabilistic framework, where a state transition distribution can be constructed as:

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_{t+1}|f_{\boldsymbol{\theta}}(\mathbf{z}_t, \mathbf{u}_t), Q) \tag{2.2}$$

where $Q$ is the transition covariance (constant in this example, for simplicity), under an assumption that the noise of the true process is Gaussian with zero mean. This transition distribution can then be used to perform learning and inference from data in a variety of settings. For example, we may want obtain an estimate of $\boldsymbol{\theta}$ by maximising the likelihood of a sequence of observed states $\mathbf{z}_{1:T}$:

$$\boldsymbol{\theta}_{MLE} = \arg\max_{\boldsymbol{\theta}} \log p(\mathbf{z}_{1:T}|\mathbf{u}_{1:T}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \log p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_{t-1}, \boldsymbol{\theta}), \tag{2.3}$$

or we may want estimate the posterior distribution $p(\boldsymbol{\theta}|\mathbf{z}_{1:T})$ using Markov Chain Monte-Carlo (MCMC) or variational methods (Murphy 2012)[1].

Having a trained dynamical model of an actuated system, we can perform continuous control (Lesort et al. 2018) tasks by solving the optimisation problem:

$$\mathbf{u}^*_{1:T} = \arg\min_{\mathbf{u}_{1:T}} \sum_{t=1}^{T} \mathcal{C}(\mathbf{z}_t, \mathbf{u}_t), \quad \text{s.t.} \quad \mathbf{z}_t \sim p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_{t-1}, \boldsymbol{\theta}), \tag{2.4}$$

where $\mathbf{u}^*_{1:T}$ is the optimal action sequence under the cost function $\mathcal{C}$. This problem is referred to as *model-predictive control* (MPC), and it can be applied to a variety of settings, from simulated environments, like video games, to real environments, like robots. The cost usually consists of a

[1] In this work, we assume that we can always write a likelihood function for any physical model $f$, as we focus mostly on physical object dynamics. However, in many complex/large scale physical systems (e.g. particle physics or epidemiology models) it is possible to simulate the system according to $f$ but there is no likelihood function. This requires the use of likelihood-free inference methods, which is outside the scope of this work.

distance to a goal state, with optional trajectory/action regularisation terms. A common choice is a quadratic cost of the form:

$$\mathcal{C}(\mathbf{z}_t, \mathbf{u}_t) = \mathbf{z}_t^T Q\, \mathbf{z}_t + \mathbf{u}_t^T R\, \mathbf{u}_t. \tag{2.5}$$

When the transition function is globally linear, the LQR (Kirk 1970) algorithm can be used to efficiently find a solution to the planning problem, whereas in non-linear settings other approaches like iLQR (Li et al. 2004) (which performs local linearisation) can be used instead. However, we often do not have locally linear transitions or quadratic costs, in which case more general black-box MPC methods have to be used, such as the Cross-Entropy Method (CEM, Rubinstein (1997)) or AICO (Ryder et al. 2018). In all the works in this thesis we make use of CEM to perform control tasks (either as the main method or as a baseline), so we provide a detailed description of it, along with implementation details, in Appendix A

When modelling scenes from vision, (2.2) is most often used as the transition distribution within a latent variable model. A latent variable model assumes an agent only has access to noisy and/or partial observations of the true state of the system, $\mathbf{x}_t \sim p_{\boldsymbol{\phi}}(\mathbf{x}_t|\mathbf{z}_t)$, where $\boldsymbol{\phi}$ are the parameters of the observation distribution. These observations can be either noisy linear transformations of ground-truth states or higher dimensional sensor information, e.g. images. We will refer to ground-truth state observations as *proprioception*[2], so as to not overload the term *state*. Given a set of observations $\mathbf{x}_{1:t}$, inference then consists of estimating the latent state distribution $p(\mathbf{z}_t|\mathbf{x}_{1:t})$. Estimating this distribution is known as *filtering*, though one can also estimate the full sequence of states $p(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})$, know as *smoothing*.

Inferring latent states $\mathbf{z}$ from images $\mathbf{x}$ is significantly harder as it involves an additional step of representation learning from a high-dimensional, unstructured pixel space, to a low-dimensional, latent space. Furthermore, we interested in not just *any* latent space, but rather in physically structured latent spaces. As such, it is important to understand how physical inductive biases are built into models from proprioception alone, so that these can then be integrated into large vision-based models. Both cases will be thoroughly explored throughout this chapter.

## Parameter interpretability

A key differentiator between types of models $f$ lies in the interpretability of the parameters $\boldsymbol{\theta}$. When $\boldsymbol{\theta}$ is interpretable (e.g. a physical quantity like mass or friction), we will say $f$ is an *explicit* physical model, and when $\boldsymbol{\theta}$ is not interpretable (e.g. the parameters of a neural network), we will say that $f$ is an *implicit* physical model. Grouping physical models into these two categories is useful for understanding the literature, as each comes with advantages and disadvantages, which must be considered by a researcher or engineer when deciding which type of model to use. Furthermore,

[2]The term proprioception is usually associated with robotic systems, but here we use it for any system whose components' positions and velocities we can inspect, be it a real or simulated enviornment.

interpretable and non-interpretable parameters impose different types of constraints on the model formulation, requiring different types of modelling and optimisation advances. It is worth noting that although the states $\mathbf{z}$ can also be interpretable (such as positions and velocities) or not (abstract latent vector), our distinction between explicit and implicit models only takes into account the interpretability of the parameters $\boldsymbol{\theta}$, as interpretable parameters imply interpretable states by construction.

Of the three main models that comprise this thesis, two are explicit and one is implicit. Explicit models have the advantage that they provide direct model interpretability and insight into a scene, but they typically involve tight information bottlenecks that make them harder to train when integrated with neural networks, as will be shown in Chapter 3. On the other hand, implicit models are easier to train and can be applied across datasets in a more straightforward manner, but interpretability requires post hoc analysis and inspection, often requiring ground-truth labels for the properties of interest. Naturally, these are simply two ends of the modelling spectrum, as there are also exist *hybrid* models, which attempt to integrate both interpretable and non-interpretable parameters with the aim of bridging the gap between implicit and explicit models. The extent of the "explicitness" of the representations to use is not a question of absolute right or wrong, but rather depends on the particular use-case and the goals of the model. The advantages and disadvantages of implicit and explicit models are succinctly described in Table 2.1, and discussed more thoroughly at the end of their respective sections.

| | Implicit Models | Explicit Models |
|---|---|---|
| Advantages | • scalability<br>• versatility<br>• transferability | • explicit physical quantities<br>• when correct, better extrapolation<br>• counterfactual reasoning<br>• data efficiency |
| Disadvantages | • hard to obtain insights<br>• may not extrapolate correctly<br>• data hungry | • hard to model complex systems<br>• requires a different set of equations for each system |

**Table 2.1:** Advantages and disadvantages of implicit and explicit physical models

# 2.1 Preliminaries

## 2.1.1 Differential Equations and Numerical Integration Methods

At the fundamental level, physical systems, and mechanical systems in particular, are often governed by ordinary differential equations (ODEs):

$$\frac{d\mathbf{z}_t}{dt} = h_{\boldsymbol{\theta}}(\mathbf{z}_t) \tag{2.6}$$

If $h_{\boldsymbol{\theta}}$ is Lipschitz continuous with respect to $\mathbf{z}$, a solution for the ODE exists and is unique for some initial conditions $\mathbf{z}_0$[3]. The solution for $\mathbf{z}_t$ can be obtained by integrating (2.6):

$$\mathbf{z}_t = \mathbf{z}_0 + \int_{t'=0}^{t} h_{\boldsymbol{\theta}}(\mathbf{z}_{t'})dt' \tag{2.7}$$

As an example, in the falling ball case above, we can write (2.6) using Newton's equations:

$$\begin{cases} \dot{\mathbf{x}}_t = \mathbf{v}_t \\ \dot{\mathbf{v}}_t = -g \end{cases}$$

where $\mathbf{x}_t$ is the position vector, $\mathbf{v}_t$ is the velocity vector, and $g$ is gravity (here $\mathbf{z}_t \equiv [\mathbf{x}_t, \mathbf{v}_t]$). The solution to (2.7) can be explicitly written as:

$$\begin{cases} \mathbf{x}_t = \mathbf{x}_0 + \mathbf{v}_0\, t - \frac{1}{2} g\, t^2 \\ \mathbf{v}_t = \mathbf{v}_0 - g\, t \end{cases}$$

This is not, however, a common scenario. For the vast majority of differential equations with applications of interest, (2.7) does not have analytical solution, so it must be solved numerically. For this reason, throughout this work we will ignore a system's analytical solution even if it exists, and we will always use the numerical solutions, so as to avoid artificially simplifying the problem down to unrealistic settings. We now describe the two most common methods for numerical integration: Euler and Runge-Kutta.

### Euler method

The simplest numerical integration method is the Euler method, which approximates (2.7) as:

$$\mathbf{z}_{t+\Delta t} = \mathbf{z}_t + \Delta t\, h(\mathbf{z}_t; \boldsymbol{\theta}) \tag{2.8}$$

[3] We will not consider stochastic differential equations that involve noisy perturbations of the inputs or parameters according to some Wiener process (or variation thereof), as it involves further mathematical tools and complexity that are outside the scope of this work.

for a time increment $\Delta t$. One glaring limitation of the Euler method is that it can easily result in unstable or diverging solutions unless $\Delta t$ is chosen to be very small (Lambert 1992). In order to remain precise while using this method, we can compute multiple Euler steps within a step $\Delta t$:

$$\mathbf{z}_{t+\Delta t \cdot m/M} = \mathbf{z}_{t+\Delta t \cdot (m-1)/M} + \frac{\Delta t}{M} h(\mathbf{z}_{t+\Delta t \cdot (m-1)/M}; \boldsymbol{\theta}), \quad \text{for } m \in 1..M, \tag{2.9}$$

which will make the approximation arbitrarily accurate for $\Delta t/M \to 0$.

### Runge-Kutta method

The gold standard for computing numerical integrals with high accuracy is the Runge-Kutta method (RK), particularly the 4th order Runge-Kutta (RK4), which we describe here. For the more general formulation of RK, of which the Euler method is a special case, see Isaacson et al. (1994).

RK4 approximates (2.7) as:

$$\mathbf{z}_{t+\Delta t} = \mathbf{z}_t + \frac{1}{6}\Delta t(k_1 + 2k_2 + 2k_3 + k_4), \tag{2.10}$$

where

$$k_1 = h(\mathbf{z}_t; \boldsymbol{\theta}) \quad , \quad k_2 = h(\mathbf{z}_t + \frac{\Delta t}{2} k_1; \boldsymbol{\theta})$$
$$k_3 = h(\mathbf{z}_t + \frac{\Delta t}{2} k_2; \boldsymbol{\theta}) \quad , \quad k_4 = h(\mathbf{z}_t + \Delta t\, k_3; \boldsymbol{\theta}).$$

### Further considerations

We note that if $h$ is differentiable w.r.t $\boldsymbol{\theta}$, we can compute derivatives of the integral w.r.t $\boldsymbol{\theta}$ with backpropagation, as the integration recurrence for the three methods above only involves arithmetic operations. This fact is particularly important when using differential equation integrators within deep learning models (c.f. Sec. 2.2.1)

Although the RK method is more accurate[4], the works presented in this thesis (Ch. 3 and 4) use the Euler method with small time steps, as it provided the simplest implementation while maintaining a sufficient level of accuracy for our use cases.

## 2.1.2 Deep Recurrent State-Space Models

Recall from earlier in this chapter that we often do not have access to the true state of the system, but rather to some noisy or partial high-dimensional observation. In a computer vision context, these observations are images, $\mathbf{I}_t$, which are received through a camera feed. In order to model a

[4]When solving Hamiltonian dynamics, there is also an alternative method called Symplectic integration, which is significantly more accurate than the Euler method (Leimkuhler et al. 2005), while being as computationally efficient.

physical system given only its images, it is necessary to learn or identify a latent space where it is easier to perform predictions and model object dynamics.

Due to the high-dimensionality of the visual observation space, neural networks are particularly well suited to the problem of learning dynamical scene representations (Bengio et al. 2013). The simplest example of such a neural system is perhaps a recurrent autoencoder (e.g. Srivastava et al. (2015)), where visual observations are reduced to a lower-dimensional latent space by an encoder $\mathbf{z}_t = enc_\theta(\mathbf{I}_t)$, the dynamics are rolled forward using a recurrent transition function, $\mathbf{z}_{t+1} = rec_\theta(\mathbf{z}_t)$, such as an LSTM (Hochreiter et al. 1997a), and the images are reconstructed using a decoder, $\hat{\mathbf{I}}_{t+1} = dec_\theta(\mathbf{z}_{t+1})$. This model can be trained by minimising the next-step reconstruction error using gradient-descent:

$$\boldsymbol{\theta}^* = \arg\min_\theta \sum_t \left\| \hat{\mathbf{I}}_{t+1} - \mathbf{I}_{t+1} \right\|^2. \tag{2.11}$$

Assuming the latent dimensionality is lower than the image dimensionality, this loss will encourage the encoder to learn a compact representation that can be used to predict the future state of the system. A number of works build on this base formulation, either by incorporating further structure into the recurrent process or adding regularisation terms to the loss[5], as we will see in later sections. However, the formulation above is deterministic, so it does not provide a probabilistic framework for inference, generation, and uncertainty estimation. As such, it is useful to formulate latent space models as probabilistic models, of which deterministic models can be seen as a special case when all distributions are Dirac-deltas[6].

State-space models formulate dynamical systems from observations as generative models of the form:

$$p_\theta(\mathbf{I}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{I}_t|\mathbf{z}_t) p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}), \tag{2.12}$$

where we use a Markovian transition function under the assumption that all the instantaneous system information is encoded in the latent state $\mathbf{z}_t$, and the decoder and transitions distributions are non-linear functions parametrised by neural networks. In this formulation, inference, i.e. estimating $p(\mathbf{z}_{1:t}|\mathbf{I}_{1:t})$ is not straightforward, as $p(\mathbf{z}_{1:T}|\mathbf{I}_{1:T}) = p(\mathbf{I}_{1:T}|\mathbf{z}_{1:T})p(\mathbf{z}_{1:T})/p(\mathbf{I}_{1:T})$ is intractable. Therefore, we must resort to approximate inference methods to estimate it. In the deep learning literature, a popular formulation to train deep generative models from videos is variational inference[7] (Jordan et al. 1999), which we now describe.

[5] Interestingly, Jonschkowski et al. (2017) skips the reconstruction loss altogether by relying only on regularisation terms.

[6] Although it is easy to turn a probabilistic model into a deterministic model, in our experience the opposite is not true, as probabilistic formulations usually involve noisier parameter updates, which can lead to training instability and poor calibration.

[7] This is not the case in classical systems, where MCMC and Expectation Propagation (Murphy 2012) are also common approaches.

### Variational formulation

In variational inference, the intractability of the posterior is overcome by defining a factorised approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T}) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{I}_t, )$, parametrised by $\boldsymbol{\phi}$, and minimising the KL-divergence between the approximate and the true posteriors:

$$\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T})\|p_{\theta}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T})) = \int q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T})}{p_{\theta}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T})} \, \mathrm{d}\mathbf{z}_{1:T}. \tag{2.13}$$

Using the property that $\mathrm{KL}(\cdot\|\cdot) \geq 0$, (2.13) can be rewritten as:

$$p(\mathbf{I}_{1:T}) \geq \int q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{I}_{1:T}|\mathbf{z}_{1:T}) p_{\boldsymbol{\theta}}(\mathbf{z}_{1:T})}{q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}|\mathbf{I}_{1:T})} \, \mathrm{d}\mathbf{z}_{1:T} \equiv \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}} \tag{2.14}$$

where $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}$ is referred to as the evidence lower-bound (ELBO). Therefore, minimising the KL divergence between the approximate and the true posterior is equivalent to maximising a lower-bound on the marginal likelihood of the data. Due to the factorisation of the approximate posterior, the integral above can be written as:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}} = \sum_{t=1}^{T} \int q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{I}_t) \log \frac{p_{\theta}(\mathbf{I}_t|\mathbf{z}_t) p_{\theta}(\mathbf{z}_t|\mathbf{z}_{1:t-1})}{q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{I}_t)} \, \mathrm{d}\mathbf{z}_t. \tag{2.15}$$

This is called a *Variational RNN* (VRNN, Chung et al. (2015)). Using the reparametrisation trick for Variational Autoencoders (VAEs, Kingma et al. (2014b),Rezende et al. (2014)) and backpropagation through time (Werbos 1990), it is possible to train the model end-to-end, that is, jointly optimise the encoder, decoder, and transition parameters with a single pass through a sequence of images by using, for each time-step, a Monte Carlo estimate of the integral above:

$$\int q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{I}_t) \log \frac{p_{\theta}(\mathbf{I}_t|\mathbf{z}_t) p_{\theta}(\mathbf{z}_t|\mathbf{z}_{1:t-1})}{q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{I}_t)} \, \mathrm{d}\mathbf{z}_t \approx \frac{1}{J} \sum_{j=1}^{J} \log \frac{p_{\theta}(\mathbf{I}_t^{(j)}|\mathbf{z}_t^{(j)}) p_{\theta}(\mathbf{z}_t^{(j)}|\mathbf{z}_{t-1}^{(j)})}{q_{\boldsymbol{\phi}}(\mathbf{z}_t^{(j)}|\mathbf{z}_{t-1}^{(j)}, \mathbf{I}_t)}, \tag{2.16}$$

where $\mathbf{z}_t^{(j)} = g(\epsilon^{(j)}, \psi_{\theta}(\mathbf{z}_{t-1}^{(j)}))$, $\epsilon^{(j)}$ is a sample from a base distribution, and $\psi_{\theta}(\mathbf{z}_{t-1}^{(j)})$ is a neural networks whose output are the distribution parameters. Throughout this work we use a Gaussian posterior, where $g(\epsilon^{(j)}, \psi_{\theta}(\mathbf{z}_{t-1}^{(j)})) = \mu_{\theta}(\mathbf{z}_{t-1}^{(j)}) + \epsilon \cdot \sigma_{\theta}(\mathbf{z}_{t-1}^{(j)})$ and $\epsilon^{(j)} \sim \mathcal{N}(0, 1)$. Similarly to the deterministic case, in the most general VRNN formulation the transition distribution $p_{\theta}(\mathbf{z}_t|\mathbf{z}_{1:t-1})$ is simply a recurrent neural network, such as an LSTM.

This approach has been very successful at solving a number of control tasks from vision (Wahlstrom et al. 2015; Ha et al. 2018; Hafner et al. 2019, 2020; Kobayashi 2020; Sekar et al. 2020), but they so with minimal use of physical or dynamical inductive biases. As we argue in this thesis, these naive approaches fail to exploit the regularities that exist in the object dynamics and their interactions, missing out on possible generalisation and control gains. In later sections we discuss works that

suggest improvements in this direction.

### 2.1.3 Visual object representations

When learning deep state-space models, it is important to build into the model inductive biases that will encourage latent representations that enable better dynamics modelling and provide an appropriate level of interpretability. This is particularly advantageous in settings where little data is available for training the model, which would result in poor generalisation by a highly flexible model. In the case of computer vision, these representations are visual object representations, which can include any number of physical properties that describe an element of a visual scene such as position, velocity and acceleration; shape and volume; density, mass, and other physical quantities; colour, reflectance, and shadows; pose relative to the camera. Ideally, the model should learn *disentangled* representations (Kulkarni et al. 2015; Matthey et al. 2017), such that each latent component (or group thereof) describes a particular property.

A common paradigm for object representations, which we follow here, is the *what-where* separation. In this paradigm, properties related to appearance are represented in the $\mathbf{z}^{\text{what}}$ variables, and properties related to shape, location and dynamics are represented in the $\mathbf{z}^{\text{where}}$ variables. This separation is particularly useful when modelling dynamical systems, as the appearance of objects is usually constant and only their positions change, allowing the transition models to use a small subset of the full latent vector, and avoid modelling spurious factors of variation. In this section we focus on two types of object representation that differ primarily on the way the $\mathbf{z}^{\text{where}}$ vector is obtained, and served as the foundation for our works: vision-as-inverse-graphics and keypoints.

#### Vision-as-Inverse-Graphics

As the name implies, vision-as-inverse-graphics tackles the object representation problem from the point-of-view that a visual scene should be described by the parameters of a graphics engine. This analysis-by-synthesis approach uses a graphics renderer (learned or fixed) as the decoder $d_\theta(\mathbf{z})$, so the inference problem becomes that of finding the latent variables $\mathbf{z}$ whose rendering most closely matches a given image.

A classic example of the inverse-graphics approach is that of learning programs that use use known stroke/shape renderers to represent part primitives (Lake et al. 2015; Moreno et al. 2016; Romaszko et al. 2017; Wu et al. 2017b; Ellis et al. 2018), where a latent vector for a part, $\mathbf{z}^{i\text{where}}$, can include, for example, the start position, end position, and curvature of the stroke. These methods differ primarily in the way scene parameters are inferred, and they benefit from advances in differentiable rendering (Loper et al. 2014; Kato et al. 2018; Chen et al. 2019; Liu et al. 2019), which allow gradients to be propagated through an end-to-end model. By contrast, *transforming autoencoders* (Hinton et al. 2011; Tieleman 2014) and their successor, *capsules* (Sabour et al. 2017; Kosiorek et al. 2019), treat scenes as a set of part-whole relationships between object parts, where set of visual

primitives are *learned*. The fundamental problem then becomes how to construct the model such that correct primitives are discovered.

One of the most influential works in the learnable renderer literature is the Spatial Transformer (ST), which introduced a differentiable layer, ST($\mathbf{I}$, $\rho$), that takes an image $\mathbf{I}$ and applies an affine/mesh transformation with parameters $\rho$. Crucially, this layer is differentiable w.r.t both the input image and the transformation parameters, enabling its use as a differentiable rendering decoder within a deep learning architecture. The ST has enabled end-to-end variational models to discovery objects in an inverse-graphics framework for both static (Eslami et al. 2016; Huang et al. 2016; Rezende et al. 2016; Kosiorek et al. 2019; Engelcke et al. 2020) and dynamical (Hsieh et al. 2018; Kosiorek et al. 2018; Zhu et al. 2018) scenes[8].

### Keypoints

An alternative representation to object appearance, pose and position uses keypoints. Keypoints correspond to salient image coordinates, and can be used to describe $\mathbf{z}^{\text{where}}$ as a set of (possibly ordered) locations of interest, $\mathbf{z}^{\text{where}} = \{(x_j, y_j)\}_{j=1}^{J}$. Keypoint extractors are foundational methods in computer vision (Lowe 2004; Rosten et al. 2006; Rublee et al. 2011), and in recent years there has been increased interest in learning unsupervised object representations as keypoints (Ehrhardt et al. 2018; Jakab et al. 2018; Suwajanakorn et al. 2018; Jakab et al. 2019; Kulkarni et al. 2019; Das et al. 2020; Gopalakrishnan et al. 2020), including in deep variational recurrent models (Minderer et al. 2019).

Unsupervised keypoint models have been particularly successful at providing low-dimensional representations for multi-joint robot control from vision (Manuelli et al. 2019; Das et al. 2020; Manuelli et al. 2020), where vision-as-inverse-graphics models fall short, due to the inherent difficulty in learning visually complex scenes with unknown part interactions through differentiable renderers, fixed or learned. In fact, during this thesis, I have experimented with this idea multiple times and found it extremely challenging to perform unsupervised learning using an inverse-graphics approach in real scenes containing multi-joint objects, in such a way that the parts discovered corresponded to disentangled object components. Our work on the NewtonianVAE (Ch. 5), which deals with multi-joint robots, circumvents this difficulty by building the disentanglement prior into the latent dynamics as opposed to the rendering mechanism (we simply use a black-box decoder).

[8] As an additional note for the reader, we contrast these works with some related models that use the *what-where* paradigm *without* constituting vision-as-inverse-graphics. The DRAW model (Gregor et al. 2015), for example, uses a soft attention mechanism to represent images as a sequence of patches, but this isn't a graphics representations as the patches do not represent individual scene parts. Similarly, deep generative models that represent objects as a product of segmentation masks and content (corresponding to object location and appearance, respectively) can perform object discovery and relational reasoning (Greff et al. 2017; Steenkiste et al. 2018; Xu et al. 2019) but typically use a black-box decoder without rendering properties, so object locations have to be inferred via ad-hoc inspection of object masks.

## 2.2 Explicit Models

We define explicit models as models that aim to describe the data-generating process according to some *known* physics equations. That is, we have a set of equations that we expect to explain the data observed, and we want to find the values of the unknown parameters by minimising the difference between the outputs of the model and the observations using some optimisation algorithm. These parameters have well defined interpretations as physical quantities of interest. For example, fitting Newton's equations under constant acceleration to a falling ball in order to determine the value of gravity constitutes explicit modelling. Physicists and applied mathematicians have long sought to describe physical phenomena using equations whose parameters need to be determined from observations. A particularly topical example in these COVID times is that of measuring the value of $R$ in epidemiological models, using the transmission and contamination data collected by public authorities.

In the machine learning and robotics literature, this problem is usually referred to as *system identification*[9] (Söderström et al. 1992; Kozlowski 1998; Ljung 1998). An early example of system identification can be found in An et al. (1985), where physical parameters of rigid links of a robot, such as the mass and moment of inertia of each link, are estimated by fitting the Newton-Euler equations to manipulator data. Additionally, we might not know the exact form of these equations, and we want to find them together with their parameters.

Observations usually consist of discrete and uniformly spaced sequences, $\tilde{\mathbf{z}}_{1:t}$ (though there are exceptions), so the differential equations (2.7) are converted to the discrete form (2.1) by setting $f_{\boldsymbol{\theta}}(\mathbf{z}_t) \equiv \mathbf{z}_t + \int_{t'=t}^{t+1} h_{\boldsymbol{\theta}}(\mathbf{z}_t)\, dt$. Maximum likelihood estimation under a Gaussian likelihood and next-step prediction as per (2.3) then becomes the optimisation problem:

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T-1} \left\| \tilde{\mathbf{z}}_{t+1} - \tilde{\mathbf{z}}_t + \int_{t'=t}^{t+1} h_{\boldsymbol{\theta}}(\mathbf{z}_t)\, \mathrm{d}t \right\|^2 . \tag{2.17}$$

Naturally, we can also optimise for multi-step ahead prediction (as the differential equation can be rolled out arbitrarily ahead in time), though here we write next-step prediction for simplicity. Matching a putative differential model to observations by rolling it forward is sometimes referred to as *simulation alignment* (Romeres et al. 2016; Lopez-Guevara et al. 2017; Ramos et al. 2019).

### 2.2.1 Differentiable physics

Although (2.17) has a relatively simple form, the lack of an analytical integral in most cases means that solving this minimisation from data involves optimising through a numerical integration, which

[9] The term "system identification" has different meanings in different areas. It has been used to describe models that infer interpretable physical quantities or equations, as well as general black-box model fitting of dynamical system data. In the context of this thesis, only the former constitutes system identification.

can be done either with gradient-based or gradient-free optimisation methods. In this thesis we focus only on gradient-based methods, as they are the ones used in our works, but several approaches to gradient-free optimisation exist for physical parameter estimation, such as Bayesian Optimisation (Ramos et al. 2019); Bayesian inference, (Ramsay et al. 2007; Romeres et al. 2016); and Gaussian Processes (Raissi et al. 2017). Gradient-based methods are of particular interest to the deep learning research community, as they enable the combination of physical equations within deep learning architectures, such that is it possible to backpropagate through the whole computation graph.

Solving (2.17) with gradient-based methods boils down to computing $\nabla_{\boldsymbol{\theta}} \int_{t'=t}^{t+1} h_{\boldsymbol{\theta}}(\mathbf{z}_t)$. If the numerical integration method used involves only operations that are supported by automatic differentiation engines, then the gradients can be computed automatically using any standard deep learning framework, like Tensorflow (Abadi et al. 2016) or Pytorch (Paszke et al. 2019). This is the case for the integration methods in Section 2.1.1, which consist only of recurrent arithmetic computations. As long as the function $h_{\boldsymbol{\theta}}$ is differentiable everywhere w.r.t $\boldsymbol{\theta}$ and $\mathbf{z}_t$, (2.17) can be solved with gradient descent (or a variation thereof) by computing the derivatives with backpropagation. This very simple method is used effectively in our PAIG model (Ch. 3).

In many cases, however, state-of-the-art physics engines compute the integral above using more much more complex algorithms than the Euler, RK, or Symplectic integrators above, applying operations that are not supported by general purpose automatic differentiation engines. An example is the computation of rigid-body equations of motion through contact and friction forces, which involes solving a linear complementarity problem (Cline 2002; Cottle et al. 2009). One could compute such gradients naively using the method of finite-differences as:

$$\nabla_{\boldsymbol{\theta}} f(\mathbf{z}_t; \boldsymbol{\theta}) = \frac{f(\mathbf{z}_t; \boldsymbol{\theta} + \epsilon) - f(\mathbf{z}_t; \boldsymbol{\theta} - \epsilon)}{2\,\epsilon} \tag{2.18}$$

where $\epsilon > 0$ is a very small vector perturbation (which is the approach we take in our V-SysId model (Ch. 4)). Though very easy to implement, finite-differences are also not appropriate for use within a larger end-to-end deep learning model, as they are not easy to combined with other components whose gradients are computed by backpropagation. This has led researchers to formulate methods to compute the gradients of said physics engine operators, giving rise to *differentiable physics engines* (Degrave et al. 2016; Belbute-Peres et al. 2018; Toussaint et al. 2018; Qiao et al. 2020; Song et al. 2020), which implement layers with custom feed-forward and -backward computations to enable combination with general deep learning frameworks.

### 2.2.2 Symbolic Discovery

In addition to learning the physical parameters, we may want to discover the form of the differential equations governing a system. This poses an even greater challenge to system identification, as both the equations and the respective parameters have to be determined. Symbolic discov-

ery/regression is a topic of great interest as it could enable data-driven discovery of physical laws governing dynamical systems. Although none of main works in this thesis (Secs. 3-5) involves symbolic discovery, we believe this area warrants a brief review, as these models are prime candidates for integration with our physics+vision models in future work, and system identification and symbolic discovery areas often overlap.

While there are a number of methods and packages that enable symbolic discovery (see Section 2 of Cranmer et al. (2020a) for an exhaustive list), here we describe only regression-based methods based on linear regression or neural networks, as these are the most amenable for integration within larger end-to-end deep learning systems, particularly from vision.

Early models for data-driven discovery of physical equations Bongard et al. (2007) and Schmidt et al. (2009) compared numerical derivatives of observational data with analytic derivatives of a set of candidate functions, and applied an evolutionary algorithm to perform system identification. However, using gradient-free optimisation techniques can be computationally expensive, so attention has progressively turned to methods that regress candidate functions of interest directly and can be optimised through gradient-descent.

In symbolic regression with sparse optimisation (Brunton et al. 2016a; Rudy et al. 2017; Schaeffer 2017), we assume we can regress the true values of $\dot{\mathbf{z}}(t)$ through a sparse linear combination of non-linear input transformations. That is, for an input state $\mathbf{z}$, we create an augmented input vector with various common non-linear functions and feature interactions, e.g.:

$$\hat{\mathbf{z}} = [1, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_1 \cdot \mathbf{z}_2, \mathbf{z}_1^2, \cos(\mathbf{z}_1), \sin(\mathbf{z}_1), ...] \tag{2.19}$$

The optimisation problem is then one of regularised linear regression:

$$\boldsymbol{\Theta}_{MLE} = \arg\min_{\boldsymbol{\Theta}} \|\dot{\mathbf{z}} - \boldsymbol{\Theta}\hat{\mathbf{z}}\|^2 + \lambda R(\boldsymbol{\Theta}) \tag{2.20}$$

where $R(\boldsymbol{\Theta})$ is a sparsity regularization term to encourage a small number of non-zero components in $\Theta$. The standard example is the $L_1$ loss, $R(\boldsymbol{\Theta}) = \sum_{ij} |\Theta_{ij}|$. We note that this approach can be used to model both ODEs and PDEs, although PDEs are of limited use in mechanical systems.

Using only one level of linear combinations greatly restricts the types of equations that can be modelled. More complex equations with higher level feature interactions can be obtained by stacking multiple such layers as shown by the Equation Learner model (EQL, Martius et al. (2016)), and Neural Arithmetic Logic Units (NALU, Trask et al. (2018)), which enables application outside of toy models. Though symbolic regression is typically restricted to learning simulated physical systems, Sahoo et al. (2018) uses EQL to model the dynamics of controllable systems with noisy data, showing promising applicability to control problems. Symbolic models are useful in control settings be-

cause they provide high expressiveness while maintaining functional regularity at the output, which is important in systems where limited data is available and the model must extrapolate correctly outside the training domain.

Zheng et al. (2018) uses an Interaction Network-based model (Sec. 2.3.1) to build a dynamics model from state observations, which enables, in a downstream step, inference of the physical properties of systems of interacting point masses. These properties include mass, charge, and coefficient of restitution. Cranmer et al. (2020a) goes a step further, by performing symbolic discovery on the learned dynamics model.

### 2.2.3 Learning Physical Parameters from Video

Most of the works discussed in this section so far work from state observations directly. This reduces the problem to "pure" system identification, where we assume that we have access to system states. This assumptions holds for simulated environments and many robotics settings, where proprioception data is available. In some cases, however, only a camera feed is available, so the aforementioned methods are no longer directly applicable.

Inferring physical parameters from video involves an additional level of complexity relative to physical parameter estimation from states, as these are not available and must therefore be estimated from visual observations. That is, besides solving (2.17) to determine the physical parameters, we must also determine the states $\hat{\mathbf{z}}_t$ from image observations, $\mathbf{I}_t$. As we will see in this section, state and physical parameter estimation can be done either a) separately, where latent states are first inferred from images, and then passed to a physical parameter estimation algorithm as state observations; or b) jointly, where the putative equations of motion are used to inform learning of the object detector from images, while at the same time learning the correct physical parameters. We will refer to the problem of inferring objects states and the parameters of the equations of motion from video as *visual system identification*.

An early example of physical parameter estimation from unlabelled videos was the work of Bhat et al. (2002). There, the authors use the free-fall differential equations of motion for a rigid body in order to jointly discover object locations from static silhouette observations and physical parameters of the ODE (initial rigid-body position and velocity, in this case). However, this initial approach was limited to a single T-shaped object, whose object properties like shape and inertia moments were assumed to be known. Monszpart et al. (2016) follows a similar line of work, using rigid-body free fall and collision equations to estimate the physical parameters of multiple objects from complex rigid-body collisions.

The work that had perhaps the greatest impact on the development of our own research was Galileo (Wu et al. 2015) and the accompanying dataset, Physics101 (Wu et al. 2016). In the Galileo model, physical parameters are partly inferred by a neural network trained in a supervised manner (for mass, volume and material properties), and partly inferred via MCMC (for bounce height, accel-

erations and velocities). However, the authors obtain object locations via a hand-crafted, scene-specific KLT tracker, which reduces the applicability of this method in general scenes. Ehrhardt et al. (2017) used a convolutional neural network to regress physical parameters for trajectory rollout, but they relied on ground-truth locations for supervision. In PAIG and V-SysId, we improve upon these limitations by proposing more general unsupervised object detectors that can be used by the physics engines without any state supervision.

In one of the most recent applications of visual system identification to real robotic control tasks, Asenov et al. (2019) propose the Vid2Param model, where bouncing ball trajectories are used to train a VRNN that predicts the ball's physical parameters (including restitution coefficient, rolling resistance, and air drag). The use of a deep variational model enables uncertainty-aware control of a robot arm whose goal is to intercept the ball, even under temporary occlusion. Here, a segmentation-based method is used to extract ping-pong ball locations from a video.

Contrary to the works above, which use mostly hand-crafted object detectors to obtain position and velocity vectors, Stewart et al. (2017) uses physical equations of motion with known physical parameters to learn an object detector. This was one of the early models that used physics as supervision, which is an essential part of joint learning of objects *and* physics. In more recent work, Runia et al. (2020) and Murthy et al. (2021) integrate differentiable physics engines with differentiable renderers to perform visual system identifications on scenes that are simultaneously visually and physically complex.

**Relevance to our work** Two of the works presented in this thesis address learning physical parameters from video. In PAIG (Ch. 3), we bring together differentiable physics (Sec. 2.2.1) and vision-as-inverse-graphics (Sec. 2.1.3) to enable joint learning of objects locations and physical parameters. In V-SysId (Ch. 4), we explore an alternative approach, using off-the-shelf keypoint detectors to propose a large number of possible object trajectories, and fitting each of these trajectories to jointly identify the object/region of interest, the physical parameters of the equation of motion, and the camera pose.

### 2.2.4 Advantanges and Disadvantages

Explicit models provide two major benefits besides the determination of physical quantities. Firstly, if the equations that govern the system of interest are known (up to noise), a correctly fit model will be able to extrapolate correctly even to regions far outside the training domain, as the learned model accurately represents the data generating process. Secondly, having parameters in the model that control known characteristics of the system allows for counterfactual reasoning, i.e. answering questions like *"how would the trajectory have changed if this objected weighed twice as much?"* (in the case of mass being a learned physical parameter of the system).

There are, however, two major disadvantages with explicit models. Firstly, this type of modelling is

naturally limited by our ability to describe the system with known equations. While this is straightforward for simple systems, it quickly becomes impractical or infeasible for more complex systems. Secondly, there is limited transferability of models across systems, as different equations have to be used to describe the dynamics of each system.

## 2.3 Implicit Models

Unlike explicit models, implicit models do not try to fit a set of physically equations to the data in order to explain the data generating process in a human-interpretable form. Instead, they use more general computational components, such as neural networks, to fit observations from systems whose complexity prevents (or makes very difficult) the formulation of physical equations describing its behaviour. These components can be combined with physics-inspired losses and inductive biases that reflect the behaviour of real-world objects and their interactions, in order to obtain algorithms that are able to model a wide variety of scenes, regardless of the exact types of motion present. This is in contrast with explicit models, which tend to be scene specific.

In this section, we dive into three areas of implicit physical modelling that we find particularly relevant, not only for our existing, but also future work: neural physics engines, deep lagrangian/hamiltonian models, and koopman operators.

### 2.3.1 Neural Physics Engines

Early attempts to model dynamical systems from state observations (Chen et al. 1990; Narendra et al. 1990; Moore 1991) fit state transitions with a standard feed-forward neural network by minimising the prediction error. This simple learning paradigm essentially aims to use neural networks to model the system evolution, hence constituting a neural physics engine rather than a hardcoded (explicit) physics engine, as described in Section 2.2. This is still the base for most newer models, with advances being driven, in large part, by new architectures or prediction objectives that incorporate more relevant dynamical inductive biases. Such inductive biases aim to improve long-term prediction, extrapolation, and learning symmetries from system observations.

Neural physics engines are strongly motivated by the intuitive physics literature (Grzeszczuk et al. 1998; Hamrick et al. 2011; Battaglia et al. 2013; Ullman et al. 2014; Hamrick et al. 2016), which argues that our brains use previous observations of the world, and interactions with it, to construct an "intuitive" understanding of object dynamics and cause-effect relationships, enabling us to reason, predict and act in the real world. For example, a young child will be able to predict that a ball thrown in the air will follow a roughly parabolic trajectory, even though they do not know what the equation of a parabola is.

Some works model intuitive physics by predicting qualitative scene outcomes. For example, Iten et al. (2020) learns a neural network capable of answering numerical questions about a physical

system from observations, e.g. "*where will the particle be at time $t$?*" or "*what is the mass of this particle?*", and Mottaghi et al. (2016) is able to identify the category and orientation of an object's trajectory from a single image. However, these models are trained on ground-truth answers, without any physical inductive bias, whereas we are interested in unsupervised models that learn from observations alone, without additional supervisory labels.

In terms of unsupervised modelling of object relations and interactions, one of the areas of the literature that has seen the most development has been that of Graph Networks (GNs, Battaglia et al. (2018)), which encompasses various sub-classes of neural network structures such as Interaction Networks (INs, Battaglia et al. (2016) and Chang et al. (2017)), Relation Networks (RNs, Santoro et al. (2017)), and Graph Neural Networks (GNNs, Scarselli et al. (2009) and Li et al. (2016)). These networks treat objects/components of a scene as nodes in a graph, whose pairwise interactions are represented by the graph edges. They can then be stacked or integrated with other models.

The focus on better object interaction models is crucial, as most systems of interest contain multiple interacting objects. How these objects act on each other, either through contact or forces at a distance, is one of the hardest challenges in physical systems modelling. Building on the GN framework, a number of works have tried to tackle this problem. For instance, Sanchez-Gonzalez et al. (2018) represents the causal relationship between interacting components of a robotic system as a graph, using a recurrent graph network to model latent state dynamics; CLEVERER (Yi et al. 2020) and CoPhy (Baradel et al. 2020) extend the relational framework to perform counterfactual reasoning; and Sanchez-Gonzalez et al. (2020) uses GNs to learn and accurately simulate fluid and soft-object dynamics composed of thousands of particles, a task that was out of reach for non-GN neural systems.

### Neural physics engines from vision

The integration of neural physics engines with vision models is of particular interest, as it enables end-to-end discovery of objects and physics, which can be used to perform control in real-world settings (Fragkiadaki et al. 2016; Mrowca et al. 2018; Wang et al. 2018). Such integration will naturally benefit from developments in both the dynamics models literature and unsupervised object discovery literature.

Graph networks have also been integrated with dynamical vision models. Visual Interaction Networks (VIN, Watters et al. (2017)) extend INs to the visual domain with a recurrent autoencoder architecture, although they provide ground-truth state data as supervision in addition to images. On the other hand, Kipf et al. (2018) propose a variational formulation of GNNs, such that graph structures can be inferred and generated as part of a VAE-like model trained from unsupervised video. Other works focus on probabilistic formulations of joint object discovery and learning latent dynamics (Hsieh et al. 2018; Kosiorek et al. 2018; Steenkiste et al. 2018). Closer to the intuitive

physics theme, Lerer et al. (2016), Wu et al. (2017a), Groth et al. (2018), and Janner et al. (2019) learn to predict the stability of block towers from vision, by integrating neural physics engines with vision-as-inverse-graphics.

### 2.3.2 Deep Lagrangian/Hamiltonian models

Although the neural physics engines described in the previous section use architectural components (such as graph networks) geared towards learning interactions and dynamics, learning a system with a general black-box model ignores a large body of physics knowledge about how mechanical systems evolve, which could provide valuable structured priors. This prevents such models from learning conservation laws and invariant quantities, which can improve long-term prediction and generalisation ability.

In this section, we see how ideas from Hamiltonian and Lagrangian mechanics can be used to improve trajectory prediction and generalisation in physical systems, thereby getting closer to their "true" description. This will help motivate the use of Newtonian mechanics in a deep latent variable model in Chapter 5.

#### Hamiltonian models

In Hamiltonian mechanics, a system is described by the coordinates $(\mathbf{q}_t, \mathbf{p}_t)$, which typically represent positions and momenta, respectively. In a conservative mechanical system, the Hamiltonian, $\mathcal{H}(\mathbf{q}, \mathbf{p})$, represents the total energy of the system, and its evolution follows the differential equations:

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}}, \quad \frac{\partial \mathbf{p}}{\partial t} = -\frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}}. \tag{2.21}$$

Recent works have sought to learn a systems Hamiltonian from observations, as opposed to known system equations. Greydanus et al. (2019), Bertalan et al. (2019) and Zhu et al. (2020) parametrize the Hamiltonian with a neural network, $\mathcal{H}_\theta(\mathbf{q}, \mathbf{p})$, whose gradients w.r.t to the input are matched to the time evolution of the system as per (2.21):

$$L^{\text{HNN}} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|^2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|^2 \tag{2.22}$$

Bondesan et al. (2019) and Mattheakis et al. (2020) propose similar models, but with greater focus on learning system symmetries. In the same research direction, Zhong et al. (2020a) formulated system evolution by modelling Hamiltonian dynamics with a NeuralODE (Chen et al. 2018). This allows the model to be trained via one- or multi-step ahead prediction by backpropagating through the Hamiltonian derivatives:

$$(\mathbf{q}, \mathbf{p})_{t+1} = \text{ODEIntegrator}\left( \frac{\mathcal{H}_\theta(\mathbf{q}_t, \mathbf{p}_t)}{\partial \mathbf{p}}, -\frac{\mathcal{H}_\theta(\mathbf{q}_t, \mathbf{p}_t)}{\partial \mathbf{q}} \right). \tag{2.23}$$

It also generalises Greydanus et al. (2019) to arbitrary coordinates and external inputs. Similarly, Chen et al. (2020) models the Hamiltonian dynamics with a NeuralODE, but uses a leapfrog symplectic integrator (Section 2.1.1), which is as computationally efficient as the simpler Euler method, but more accurate when applied to Hamiltonian systems. Sanchez-Gonzalez et al. (2019) goes one step further by combining ODE integrators and GNNs as used by Sanchez-Gonzalez et al. (2018) to obtain a richer model of the Hamiltonian. Modelling the Hamiltonian as a GNN enables learning of complex interactions between multiple objects. Moving in a very recent and promising direction, Lee et al. (2021) integrate HNNs within a Model Agnostic Meta-Learning framework (MAML, Finn et al. (2017)), enabling few-shot learning of the Hamiltonian governing a system.

### Lagrangian models

Although the Hamiltonian formulation provides a useful inductive bias for data-driven system modelling, it often assumes a convervative system, where the effect of non-conservative forces has to be modelled indirectly via changes in total energy. This framework is therefore not the most appropriate for control tasks.

The Lagrangian formulation has recently been explored as an alternative for modelling non-conservative systems (Gupta et al. 2019; Lutter et al. 2019; Cranmer et al. 2020b). Describing a parametrized Lagrangian as:

$$\mathcal{L}_\theta(\mathbf{q}, \dot{\mathbf{q}}) = T_\theta(\mathbf{q}, \dot{\mathbf{q}}) - V_\theta(\mathbf{q}), \tag{2.24}$$

where $T$ is the kinectic energy and $V$ is the potential energy, the system dynamics follow the Euler-Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}_\theta}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial \mathcal{L}_\theta}{\partial \mathbf{q}} = F(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}), \tag{2.25}$$

where $F$ is the generalized force that acts on the system, and $\mathbf{u}$ is an actuation input. As an added benefit, the Lagrangian formulation enables the use of arbitrary coordinates $\mathbf{q}$, since we do not have to know the form of $\mathbf{p}$, which may not always be equal to $m \cdot \dot{\mathbf{q}}$.

Similarly to the Hamiltonian systems, the Lagrangian positions and velocities can be predicted by integrating the expression for the acceleration,

$$\ddot{\mathbf{q}}_\theta = \left(\frac{\partial^2 \mathcal{L}_\theta}{\partial \dot{\mathbf{q}}^2}\right)^{-1} \left(F - \frac{\partial \mathcal{L}_\theta}{\partial \mathbf{q}} - \frac{\partial^2 \mathcal{L}_\theta}{\partial \mathbf{q} \partial \dot{\mathbf{q}}} \dot{\mathbf{q}}\right). \tag{2.26}$$

The works cited above have recently proposed learning neural Lagrangian models from data using with slightly different optimisation objectives similar to (2.22). Gupta et al. (2019) minimies the difference between observed and predicted positions and velocities, whereas (Cranmer et al. 2020b) minimises the difference between observed accelerations and (2.26) directly, and (Lutter et al. 2019) minimises the difference between applied forces and inverse system model.

While most of the aforementioned works (both Hamiltonian and Lagrangian) use generalised coordinates, making them Cartesian or polar as needed depending of the system, Finzi et al. (2020) embed all systems using Cartesian coordinates (in both Hamiltonian and Lagrangian formulation), and enforce explicit holonomic constraints using Lagrange multipliers. The use of explicit constraints simplifies the learning process and increases data efficiency for complex system, though it requires knowledge of the exact constraints to be applied to each system.

#### Learning Lagrangian/Hamiltonian models from vision

Although deep Lagrangian/Hamiltonian models have seen fast progress, they are still seldom applied to unlabeled vision data. Two early attempts in this direction include Greydanus et al. (2019) and Bertalan et al. (2019), where a very simple simulated pendulum is modelled from video. A more recent model that focuses exclusively on learning from vision is the Hamiltonian Generative Network (HGN, Toth et al. (2020)), where the neural symplectic approach outlined in Section 2.3.2 is used to learn a generative model of a physical system from unlabeled video. The HGN is able to learn energy-presenting dynamics, resulting in highly accurate visual system predictions. However, this model is only applied to simple simulated 2D scenes, opening the door for future work to extend this formulation to real-world scenarios, particularly those involving objects with high visual diversity in 3D, energy dissipation, and noisy multi-object interactions.

#### Other applications

There is a concurrent literature on applying the same Hamiltonian, ODE/PDE, and GNN methods to fluid dynamics, but we do not describe it in depth here, as our work focuses exclusively on mechanics. The interested reader can refer to Long et al. (2017), Long et al. (2018), Schenck et al. (2018), Belbute-Peres et al. (2020), and Mohan et al. (2020) for some recent works.

### 2.3.3 Locally-linear models and Koopman operators

The idea of using a representation where a system evolves linearly in time is not a new one. For example, a standard Kalman filter assumes a locally linear model, and there is a wide range of motions that can be described with such a model. For instance, a falling object can be described as a linear dynamical system:

$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \cdot g \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ 1 \end{pmatrix}, \tag{2.27}$$

where $g$ is the value of gravity. However, as noted in earlier sections, the majority of real systems do not follow linear dynamics. It would therefore be of great interest to map non-linear systems to a latent space where dynamics were linear. The seminal work of Koopman (1931) was the first

to formulate this idea, which has wide applicability to machine learning, particularly for control. Using a Koopman representation allows us to apply the simpler and better understood linear control techniques, such as LQR, whereas non-linear models requires the use of more general and computationally expensive MPC controllers. As we will see in Chapter 5, imposing locally-linear state-space transitions can be used to encourage physically correct latent spaces, which can enable the use of the even simpler PID controllers.

Let $x_t \in \mathcal{X}$ be an observable of a system, that evolves according to a non-linear transition function $x_{t+1} = F(x_t)$[10]. A Koopman operator (Koopman 1931), $\mathcal{K} : \mathcal{Z} \to \mathcal{Z}$, is a linear operator that acts on a latent representation $g(x_t) : \mathcal{X} \to \mathcal{Z}$ as:

$$g(x_{t+1}) = K\, g(x_t) = g(F(x_t)). \tag{2.28}$$

The Koopman operator theory guarantees the existence of an infinite-dimensional space $\mathcal{Z}$, but in practice a finite-dimensional subspace is used. Although the original formulation does not cover actionable systems, it is common to extend (2.28) with an external action $u_t$ as:

$$z_{t+1} = Kz_t + Lu_t, \tag{2.29}$$

where $z_t = g(x_t)$. The reader familiar with state-space models will recognise this as a standard linear transition model, which is arguably the simplest, most common, and most well studied form of state-space model, lying at the base of Kalman Filters and Hidden Markov Models (in the discrete state case). Due to its linearity, the Koopman operator enables analysis of many properties of the system. As described by Arbabi et al. (2018), *"the spectral properties of the Koopman operator can be used to characterise the state space dynamics; for example, the Koopman eigenvalues determine the stability of the system and the level sets of certain Koopman eigenfunctions carve out the invariant manifolds and isochrons. Moreover, for smooth dynamical systems with simple nonlinear dynamics,e.g., systems that possess hyperbolic fixed points, limit cycles and tori, the evolution of observables can be described as a linear expansion in Koopman eigenfunctions"* (c.f. Budišić et al. (2012) and Arbabi et al. (2017)).

In a machine learning setting, the central problem is that of determining the Koopman encoder $g$ and matrix $K$. Early models used hand-crafted basis functions $g_i$, $i \in \{1...N\}$, to map the observables $x_t$ into a physical space $z_t$ with known linear evolution. However, this approach is limited, as the exact underlying dynamics may not be known in advance. Therefore, there has been an increased interest in learning these functions and operators directly from data. In its simplest form, we can use a dataset of system observations $X = \{x^j_{1:T}\}^J_{j=1}$ to train a parametrised func-

[10] Here we assume discrete transitions, but the formalism is analogous for continuous transitions. For consistency with the rest of this thesis, we follow a latent variable model nomenclature, where an 'observable' is a vector that results from sensory input from the system, e.g. robot proprioception or camera images. This is in contrast with the Koopman operator literature, where the mapping $g$ is the observable.

tion $g_\theta$ (e.g. a neural network) and $[K, L]$ by minimising the next-step prediction error in latent space:

$$\|g_\theta(x_{t+1}) - Kg_\theta(x_t) - Lu_t\|, \tag{2.30}$$

with additional regularisation terms to prevent trivial solutions (Li et al. 2020a), such as an autoencoding loss $\|x_t - h_\phi(g_\theta(x_t))\|$, where $h_\phi$ is a parametrised decoder, or a distance preserving loss $\Big|\|x_{t+1} - x_t\| - \|g_\theta(x_{t+1}) - g_\theta(x_t)\|\Big|$. Naturally, one can also parametrise the Koopman matrix itself with a neural network whose input is the current state, making the model locally linear instead of globally linear.

A number of works have proposed learning of Koopman operators for control (Brunton et al. 2016b; Abraham et al. 2017; Takeishi et al. 2017; Lusch et al. 2018; Kaiser et al. 2021) with notable applications to soft robot control (Bruder et al. 2019; Mamakoukas et al. 2020), and fluid flow analysis (Arbabi et al. 2018). Morton et al. (2019) formulates Koopman operator learning in a deep Variational framework (Kingma et al. 2014b; Rezende et al. 2014), enabling uncertainty estimation in the linearised latent space. Li et al. (2020a) extends the Koopman formalism to a multi-object setting, using a block structured $K$ to impose a relational inductive bias on the interactions between system components, while (Pan et al. 2019) extends the deep Koopman operator models to continuous time with stability guarantees.

### Locally-linear deep state-space models

The formulation of Eq. (2.28) applies equally to the case when the observed vector is in state space (e.g. proprioception) or in image space (e.g. video inputs). This allows the Koopman operator formalism to be extended to vision problems, where we want to learn a low dimensional, locally linear mapping in state space from unlabaled videos, in order to harness the benefits provided by the locally-linear formalism.

An early attempt to incorporate linear transition biases into deep variational models was by Watter et al. (2015), with the Embed to Control model (E2C). The authors parametrise the encoder, transition, decoder distributions, respectively, as:

$$\begin{aligned} q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{I}_t, \mathbf{I}_{t-1}) &= \mathcal{N}(\mathbf{z}_t|\mu_t, \sigma_t^2) \\ p_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{u}_t) &= \mathcal{N}(\mathbf{z}_{t+1}|A_t \cdot \mu_t + B_t \cdot \mathbf{u}_t, C_t) + \mathbf{o}_t) \\ p_{\boldsymbol{\theta}}(\mathbf{I}_t, \mathbf{I}_{t-1}|\mathbf{z}_t) &= \mathcal{B}(\mathbf{p}_t), \end{aligned}$$

where $\mu_t$, $\sigma_t^2$, $\mathbf{o}_t$, $A_t$, $B_t$, and $\mathbf{p}_t$ are parametrised by neural networks with the respective inputs. The matrix $A_t$ is further factorised as $(\mathbf{I} + \mathbf{a}_t \cdot \mathbf{b}_t)$ in order to reduce the total parameter count. Furthermore, outputs neural networks belonging to the same distribution share a common backbone up to the last layer. The conditioning on two frames is so that the latent vector $\mathbf{z}_t$ encodes both position and velocity information, so that the transition distribution can be Markovian.

Learning is done by optimising a modified ELBO:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{I}_t,\mathbf{I}_{t-1})p_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{u}_t)} \big[ &- \log p_{\boldsymbol{\theta}}(\mathbf{I}_{t+1},\mathbf{I}_t|\mathbf{z}_{t+1}) - \log p_{\boldsymbol{\theta}}(\mathbf{I}_t,\mathbf{I}_{t-1}|\mathbf{z}_t)\big] \\ &+ \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{I}_t,\mathbf{I}_{t-1})\|p(\mathbf{z}_t)\right) \\ &+ \lambda\,\mathrm{KL}\left(p_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{u}_t)\|q_{\boldsymbol{\phi}}(\mathbf{z}_{t+1}|\mathbf{I}_{t+1},\mathbf{I}_t)\right), \end{aligned} \tag{2.31}$$

where $p(\mathbf{z}_t)$ is a standard unit Gaussian prior. The second KL term is used to encourage generated trajectories in the latent space to remain in the manifold of encoder outputs, so that during generation the system does not evolve to regions in the latent space where no vector from $q_{\boldsymbol{\phi}}(\mathbf{z}_t|\mathbf{I}_t,\mathbf{I}_{t-1})$ would be produced, for any $\mathbf{I}_t$. Control is done by encoding a target image pair, $\mathbf{z}^* = \text{mean}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{I}^*,\mathbf{I}^*))$, and using iLQR (Li et al. 2004) to minimise the cost function (2.4) in latent space.

Even though the modified ELBO proposed is an *ad hoc* solution to the distribution shift problem, the authors show that the E2C model obtains more structured and interpretable latent space than that produced by equivalent models without structured transition functions. This was shown to improve control from vision in simple simulated environments, and sparked interest in the use of locally-linear models. A number of related follow-up/concurrent works to E2C formulate locally-linear models in a deep variational framework, most notably Deep Kalman Filters (DKFs, Krishnan et al. (2015)), Deep Variational Bayes Filters (DVBFs, Karl et al. (2017), KalmanVAEs (Fraccaro et al. 2017), and Robust Controllable Embeddings (RCEs, Banijamali et al. (2018)), providing increasing levels of expressiveness and control performance, and more principled ELBO derivations.

**Relevance to our work** Hamiltonian models and Koopman theory served as a strong inspiration for the development of our NewtonianVAE model, presented in Chapter 5, where we bring ideas from Newtonian mechanics (which can be seen as a special case of Lagrangian mechanics) together with locally-linear variational models to obtain improved representations that enable the use of proportional controllers. Furthermore, in the NewtonianVAE we formulate a principled ELBO that enables transition distribution consistency like in E2C, as well as position and velocity information separation by construction, without the use of extra regularisation terms or image pair stacking.

### 2.3.4 Advantages and Disadvantages

Implicit models are often highly scalable, being able to model large, complex systems, for which explicit equations would be too hard to formulate. They are also highly transferable and versatile, as the use of general compute blocks does not place restrictions on the types of dynamics that can be modelled. The advantages are particularly clear in real-world systems, where unknown and unmodelable sources of noisy and dissipation are taken into account by construction.

One problem with implicit models is that due to the high number of trainable parameters, these

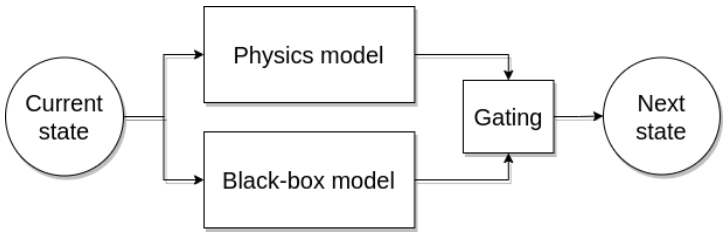



**Figure 2.1:** General architecture of hybrid/residual models. The current state is passed to both physics and black-box models, whose predictions are combined through a gating mechanism (e.g. weighted sum) to produce a predicted state.

models require large amounts of data in order to correctly learn the system dynamics. Additionally, like any machine learning system with weak inductive biases, they tend to have difficulty extrapolating to regions outside the training domain. Nonetheless, some recent works have made progress in this area, such as Sanchez-Gonzalez et al. (2018), which showed that a deep GNN trained to model the dynamics of multi-joint simulated agents is able to generalise to agents with a higher number of joints than seen at training time.

## 2.4 Hybrid models

An interesting area of research that tries to bring together the best of implicit and explicit models is that of *hybrid* or *residual* physics. These models use known equations to model the average, or part, of the trajectories, with more flexible, implicit models being responsible for predicting the residuals, i.e. the system dynamics that the explicit equations fail to model (either by design or due to noise).

The hybrid formulation makes intuitive sense because in any real world environment, even a well isolated system will not perfectly follow any set of equations. There are always factors that can perturb the idealised dynamics, so modelling them become necessary to avoid making explicit models inapplicable to real scenarios. Additionally, it might allow the inclusion of explicit models even in very large scale systems which otherwise would have to be modelled purely implicitly. This would increase our ability to extract insights from implicit models. Figure 2.1 shows the general architecture of these hybrid models. In order to predict the future state of the system, the current system (known or inferred from vision) is passed to both the explicit (physics) and implicit (black-box) models, with their predictions being combined through a gating mechanism. The design of the gating mechanism determines the components of the object's motion that each model predicts.

While hybrid models are still recent within machine learning, some notable examples include:

- Long et al. (2019) integrate symbolic computation modules with the convolutional PDE structure from Long et al. (2017), thereby increasing expressiveness of the full network.

- Seo et al. (2019) extend graph-based simulation engines (c.f. Section 2.3.1) by supplementing the recurrent GNN with a symbolic module that encourages correct PDE evolution in the form of e.g. diffusion or wave operators.
- Park et al. (2019) uses a GNN model with wind-dynamics priors for wind power generation estimation.
- Read et al. (2019) uses a deep learning model with process specific equations for predicting lake water temperature.
- Zeng et al. (2020) learn a neural model of the trajectory noisy around a physics-based ballistic trajectory. The physical model enables provide consistent estimates position and velocity estimates that generalise across landing locations, while the neural noise model produces adjustments to the ideal trajectory based on the properties of each object.

Further real-world applications can be found in Willard et al. (2020).

Having given an overview of the areas at the intersection of physics modelling and unsupervised vision models in machine learning, we now move into the main chapters of this thesis, where we present the three major models developed during this PhD: PAIG, V-SysId, and Newtonian-VAE.

# Part I

# Physical Parameter Estimation from Vision

Chapter 3

# Physics-as-Inverse-Graphics: Unsupervised Physical Parameter Estimation from Pixels

This chapter corresponds to the paper:

Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-Inverse-Graphics: Unsupervised Physical Parameter Estimation from Pixels. In *ICLR*, 2020.

In this chapter we propose a model that is able to perform unsupervised physical parameter estimation of systems from video, where the differential equations governing the scene dynamics are known, but labelled states or objects are not available. Existing physical scene understanding methods require either object state supervision, or do not integrate with differentiable physics to learn interpretable system parameters and states. We address this problem through a *physics-as-inverse-graphics* approach that brings together vision-as-inverse-graphics and differentiable physics engines, enabling objects and explicit state and velocity representations to be discovered. This framework allows us to perform long term extrapolative video prediction, as well as vision-based model-predictive control. Our approach significantly outperforms related unsupervised methods in long-term future frame prediction of systems with interacting objects (such as ball-spring or 3-body gravitational systems), due to its ability to build dynamics into the model as an inductive bias. We further show the value of this tight vision-physics integration by demonstrating data-efficient learning of vision-actuated model-based control for a pendulum system. We also show that the controller's interpretability provides unique capabilities in goal-driven control and physical reasoning for zero-data adaptation.

## 3.1 Introduction

System identification or physical parameter estimation is commonly required for control or state estimation for physical modelling, and typically relies on dedicated sensing equipment and carefully constructed experiments. Current machine learning approaches to physical modelling from video either require training by supervised regression from video to object coordinates before estimating explicit physics (Watters et al. 2017; Wu et al. 2017a; Belbute-Peres et al. 2018), or are able to discover and segment objects from video in an unsupervised manner, but do not naturally integrate

with a physics engine for long-term predictions or generation of interpretable locations and physical parameters for physical reasoning (Steenkiste et al. 2018; Xu et al. 2019). In this work, we bridge the gap between unsupervised discovery of objects from video and learning the physical dynamics of a system, by learning unknown physical parameters and explicit trajectory coordinates.

Our approach, called *physics-as-inverse-graphics*, solves the physical modelling problem via a novel vision-as-inverse-graphics encoder-decoder system that can render and de-render image components using Spatial Transformers (ST) (Jaderberg et al. 2015) in a way that makes it possible for the latent representation to generate disentangled interpretable states (position/velocity). These can be used directly by a differentiable physics engine (Degrave et al. 2016; Belbute-Peres et al. 2018) to learn the parameters of a scene where the family of differential equations governing the system are known (e.g. objects connected by a spring), but the corresponding parameters are not (e.g. spring constant). This allows us to to identify physical parameters and learn vision components of the model jointly in an end-to-end fashion. Our contribution is a solution to unsupervised learning of physical parameters from video, without having access to ground-truth appearance, position or velocities of the objects, a task that had so far remained unsolved (Wu et al. 2015; Belbute-Peres et al. 2018).

In addition to showing that our model can learn physical parameters without object or state supervision (a task with intrinsic scientific interest in and of itself), we show that incorporating dynamics priors in the form of known physical equations of motion with learnable parameters together with learnable vision and graphics can improve model performance in two challenging tasks: long term video prediction and visual model predictive control. We first evaluate physical parameter estimation accuracy and future video frame prediction on 4 datasets with different non-linear interactions and visual difficulty. We then demonstrate the value of our method by applying it for data-efficient learning of vision-based control of an under-actuated pendulum. Notably our unique ability to extract interpretable states and parameters from pixels without supervision enables end-to-end vision-based control to exploit goal-paramaterized policies and physical reasoning for zero-shot adaptation.

## 3.2 Related Work

The ability to build inductive bias into models through structure is a key factor behind the success of modern neural architectures. Convolutional operations capture spatial correlations (Fukushima 1980) in images, recurrency allows for temporal reasoning (Hochreiter et al. 1997b), and spatial transformers (Jaderberg et al. 2015) provide spatial invariance in learning. However, many aspects of common data generation processes are not yet considered by these simple inductive biases. Importantly, they typically ignore the physical interactions underpinning data generation. For example, it is often the case that the underlying physics of a dynamic visual scene is known, even if specific parameters and objects are not. Incorporation of this information would be benefi-

cial for learning, predicting the future of the visual scene, or control. Physics-as-inverse graphics introduces a framework that allows such high-level physical interaction knowledge to be incorporated into learning, even when ground-truth object appearance, positions and velocities are not available.

In recent years there has been increased interest in physical scene understanding from video (Finn et al. 2016b; Fragkiadaki et al. 2016; Chang et al. 2017; Fraccaro et al. 2017; Jonschkowski et al. 2017; Zheng et al. 2018; Janner et al. 2019). In order to learn explicit physical dynamics from video our system must discover and model the objects in a scene, having position as an explicit latent variable. Here we build on the long literature of neural vision-as-inverse-graphics (Hinton et al. 2011; Kulkarni et al. 2015; Huang et al. 2016; Romaszko et al. 2017; Wu et al. 2017b; Ellis et al. 2018), particularly on the use of spatial transformers (ST) for rendering (Eslami et al. 2016; Rezende et al. 2016; Zhu et al. 2018).

There are several models that assume knowledge of the family of equations governing system dynamics, but where the individual objects are either pre-segmented or their ground-truth positions/velocities are known (Stewart et al. 2017; Wu et al. 2017a; Belbute-Peres et al. 2018). In terms of learning physical parameters, our work is directly inspired by the Galileo model and Physics 101 dataset (Wu et al. 2015, 2016), which fits the dynamics equations to a scene with interacting objects. However, the Galileo model makes use of custom trackers which estimate the position and velocity of each object of interest, and is incapable of end-to-end learning from video, thus bypasses the difficulty of recognizing and tracking objects from video using a neural system. To the best of our knowledge, our model is the first to offer end-to-end unsupervised physical parameter and state estimation.

Within the differentiable physics literature (Degrave et al. 2016), Belbute-Peres et al. (2018) observed that a multi-layer perceptron (MLP) encoder-decoder architecture with a physics engine was not able to learn without supervising the physics engine's output with position/velocity labels (c.f. Fig. 4 in Belbute-Peres et al. (2018)). While in their case 2% labeled data is enough to allow learning, the transition to *no* labels causes the model to not learn at all. The key contribution of our work is the incorporation of vision-as-inverse-graphics with physics, which makes the transition possible.

Another related area of increasing interest is unsupervised discovery of objects and/or dynamics from video (Steenkiste et al. 2018; Burgess et al. 2019; Greff et al. 2019; Xu et al. 2019). Though powerful, such models do not typically use interpretable latent representations that can be directly used by a physics engine, reasoned about for physical problem solving, or that are of explicit interest to model users. For example, Kosiorek et al. (2018) and Hsieh et al. (2018) use ST's to locate/place objects in a scene and predict their motion, but this work differs from ours in that our coordinate-consistent design obtains explicit cartesian, angular or scale coordinates, allowing us to feed state vectors directly into a differentiable physics engine. Under a similar motivation as our

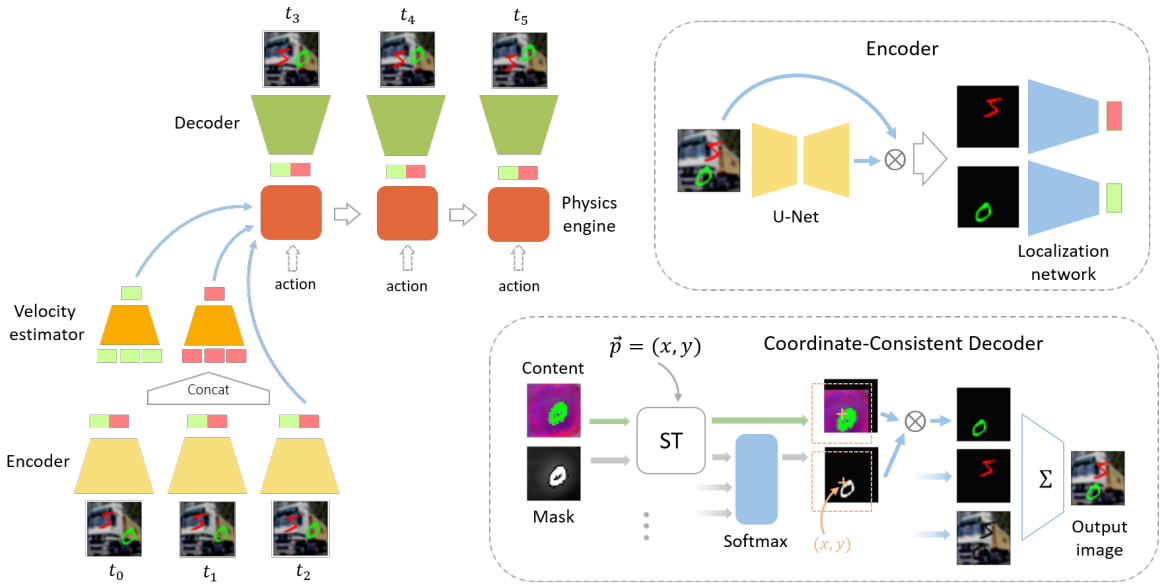



**Figure 3.1: Left:** High-level view of our architecture. The encoder (**top-right**) estimates the position of $N$ objects in each input frame. These are passed to the velocity estimator which estimates objects' velocities at the last input frame. The positions and velocities of the last input frame are passed as initial conditions to the physics engine. At every time-step, the physics engine outputs a set of positions, which are used by the decoder (**bottom-right**) to output a predicted image. If the system is actuated, an input action is passed to the physics engine at every time-step. See Section 3 for detailed descriptions of the encoder and decoder architectures.

work, but without an inverse-graphics approach, Ehrhardt et al. (2018) developed an unsupervised model to obtain consistent object locations. However, this only applies to cartesian coordinates, not angles or scale.

Despite recent interest in model-free reinforcement learning, model-based control systems have repeatedly shown to be more robust and sample efficient (Deisenroth et al. 2011; Mania et al. 2018; Watters et al. 2019a). Hafner et al. (2019) learn a latent dynamics model (PlaNet) that allows for planning from pixels, which is significantly more sample efficient than model-free learning strategies A3C (Mnih et al. 2016) and D4PG (Barth-Maron et al. 2018). However, when used for control, there is often a desire for visually grounded controllers operating under known dynamics, as these are verifiable and interpretable (Burke et al. 2019a), and provide transferability and generality. However, system identification is challenging in vision-based control settings. Byravan et al. (2018) use supervised learning to segment objects, controlling these using known rigid body dynamics. Penkov et al. (2019) learn feedforward models with REINFORCE (Williams 1992) to predict physical states used by a known controller and dynamical model, but this is extremely sample inefficient. In contrast, we learn parameter and state estimation modules jointly to perform unsupervised system identification from pixels, enabling data-efficient vision-actuated model-based control.

## 3.3 Learning Physical Parameters from Video via Inverse Graphics

In order to learn explicit physics from video, several components have to be in place. First, the model must be able to learn to identify and represent the objects in an image. In order to perform dynamics prediction with a physics engine, the position and velocity of the object must be represented as explicit latent states (whereas appearance can be represented through some latent vector or, in our case, as a set of learned object templates). Our sequence-to-sequence video prediction architecture consists of 4 modules trained jointly: an encoder, a velocity estimator, a differentiable physics engine, and a graphics decoder. The architecture is shown in Figure 3.1.

**Encoder** The encoder net takes a single frame $I_t$ as input and outputs a vector $\mathbf{p}_t \in \mathbb{R}^{N\times D}$ corresponding to the $D$-dimensional coordinates of each of $N$ objects in the scene, $\mathbf{p}_t = [\mathbf{p}_t^1, ..., \mathbf{p}_t^N]$. For example, when modelling position in 2D space we have $D = 2$ and $\mathbf{p}_t^n = [x, y]_t^n$; when modelling object angle we have $D = 1$ and $\mathbf{p}_t^n = [\theta_t^n]$. The encoder architecture is shown in Figure 3.1(top right).

To extract each object's coordinates we use a 2-stage localization approach[1]. First, the input frame is passed through a U-Net (Ronneberger et al. 2015) to produce $N$ unnormalized masks. These masks (plus a learnable background mask) are stacked and passed through a softmax to produce $N+1$ masks, where each input pixel is softly assigned to a mask. The input image is then multiplied by each mask, and a 2-layer location network produces coordinate outputs from each masked input component. For a 2D system where the coordinates of each object are its $(x, y)$ position (the polar coordinates case is analogous) and the images have dimensions $H \times H$, the encoder output represents $(x, y)$ coordinates with values in $[0, H]$. To do this, the activation of the encoder's output layer is a saturating non-linearity $H/2 \cdot tanh(\cdot) + H/2$.

**Velocity estimator** The velocity estimator computes the velocity vector of each object at the $L$-th input frame given the coordinates produced by the encoder for this object at the first $L$ input frames, $\mathbf{v}_L^n = f(\mathbf{p}_1^n, ..., \mathbf{p}_L^n)$. We implement this as a 3 hidden layer MLP with 100 tanh activated units.

**Differentiable physics engine** The physics engine contains the differential equations governing the system, with unknown physical parameters to be learned – such as spring constants, gravity, mass, etc. Given initial positions and velocities produced by the encoder and velocity estimator, the physics engine rolls out the objects' trajectories. In this work we use a simple physics engine with Euler integration, where $\mathbf{p}_t, \mathbf{v}_t$ is computed from $\mathbf{p}_{t-1}, \mathbf{v}_{t-1}$ by repeating for $i \in [1..M]$:

$$\mathbf{p}_{t+\frac{i}{M}} = \mathbf{p}_{t+\frac{i-1}{M}} + \frac{\Delta t}{M} \cdot \mathbf{v}_{t+\frac{i}{M}}; \quad \mathbf{v}_{t+\frac{i}{M}} = \mathbf{v}_{t+\frac{i-1}{M}} + \frac{\Delta t}{M} \cdot \mathbf{F}(\mathbf{p}_{t+\frac{i-1}{M}}, \mathbf{v}_{t+\frac{i-1}{M}}; \theta)\,, \tag{3.1}$$

[1] Though any other architecture capable of effectively extracting object locations from images would work.

where $\Delta t$ is the integration step, $\theta$ are the physical parameters and $\mathbf{F}$ is the force applied to each object, according to the equations in Appendix A. We use $M = 5$ in all experiments. In principle, more complex physics engines could be used (Belbute-Peres et al. 2018; Chen et al. 2018).

**Coordinate-Consistent Decoder** The decoder takes as input the positions given by the encoder or physics engine, and outputs a predicted image $\tilde{I}_t$. The decoder is the most critical part of this system, and is what allows the encoder, velocity estimator and physics engine to train correctly in a fully unsupervised manner. We therefore describe its design and motivation in greater detail.

While an encoder with outputs in the range $[0, H]$ *can* represent coordinates in pixel space, it does not mean that the decoder *will* learn to correctly associate an input vector $(x, y)$ with an object located at pixel $(x, y)$. If the decoder is unconstrained, like a standard MLP, it can very easily learn erroneous, non-linear representations of this Cartesian space. For example, given two different inputs, $(x_1, y_1)$ and $(x_1, y_2)$, with $y_1 \neq y_2$, the decoder may render those two objects at different horizontal positions in the image. While having a correct Cartesian coordinate representation is not strictly necessary to allow physical parameters of the physics engine to be learned from video, it is critical to ensure correct future predictions. This is because the relationship between position vector and pixel space position must be fixed: if the position vector changes by $(\Delta x, \Delta y)$, the object's position in the output image must change by $(\Delta x, \Delta y)$. This is the key concept that allows us to improve on Belbute-Peres et al. (2018), in order to learn an encoder, decoder and physics engine without state labels.

In order to impose a correct latent-coordinate to pixel-coordinate correspondence, we use spatial transformers (ST) with inverse parameters as the decoder's writing attention mechanism. We want transformer parameters $\omega$ to be such that a decoder input of $\mathbf{p}_t^n = [x, y]_t^n$, places the center of the writing attention window at $(x, y)$ in the image, or that a decoder input of $\mathbf{p}_t^n = \theta_t^n$ rotates the attention window by $\theta$. In the original ST formulation (Jaderberg et al. 2015), the matrix $\omega$ represents the affine transformation applied to the *output* image to obtain the *source* image. This means that the elements of $\omega$ in Eq. 1 of Jaderberg et al. (2015) do not directly represent translation, scale or angle of the writing attention window. To achieve this representation, we use a ST with inverse transformation parameters. For a general affine transformation with translation $(x, y)$, angle $\theta$ and scale $s$, we want to modify the source image coordinates according to:

$$\begin{pmatrix} x_o \\ y_o \\ 1 \end{pmatrix} = \begin{pmatrix} s \cdot \cos\theta & s \cdot \sin\theta & x \\ -s \cdot \sin\theta & s \cdot \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix} \tag{3.2}$$

This transformation can be obtained with a ST by inverting (3.2):

$$\begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix} = \frac{1}{s} \begin{pmatrix} \cos\theta & -\sin\theta & -x\cos\theta + y\sin\theta \\ \sin\theta & \cos\theta & -x\sin\theta - y\cos\theta \\ 0 & 0 & s \end{pmatrix} \begin{pmatrix} x_o \\ y_o \\ 1 \end{pmatrix} \tag{3.3}$$

Therefore, to obtain a decoder with coordinate-consistent outputs, we simply use a ST with parameters $\omega$ as given in (3.3)

Each object is represented by a learnable content $\mathbf{c}^n \in [0,1]^{H\times H\times C}$ and mask $\mathbf{m}^n \in \mathbb{R}^{H\times H\times 1}$ tensor, $n = 1..N$. Additionally, we learn background content $\mathbf{c}^{bkg} \in [0,1]^{H\times H\times C}$ and mask $\mathbf{m}^{bkg} \in \mathbb{R}^{H\times H\times 1}$, that do not undergo spatial transformation. One may think of the content as an RGB image containing the texture of an object and the mask as a grayscale image containing the shape and z-order of the object. In order to produce an output image, the content and mask are transformed according to $[\hat{\mathbf{c}}_t^n, \hat{\mathbf{m}}_t^n] = \text{ST}([\mathbf{c}^n, \mathbf{m}^n], \omega_{\mathbf{p}_t^n})$ and the resulting logit masks are combined via a softmax across channels, $[\tilde{\mathbf{m}}_t^1, ..., \tilde{\mathbf{m}}_t^N, \tilde{\mathbf{m}}_t^{bkg}] = \text{softmax}(\hat{\mathbf{m}}_t^1, ..., \hat{\mathbf{m}}_t^N, \mathbf{m}^{bkg})$. The output image is obtained by computing the channel-wise inner product of the masks with the contents:

$$\tilde{I}_t = \tilde{\mathbf{m}}_t^{bkg} \odot \mathbf{c}^{bkg} + \sum_{n=1}^{N} \tilde{\mathbf{m}}_t^n \odot \hat{\mathbf{c}}_t^n. \tag{3.4}$$

The decoder architecture is shown in Fig. 3.1, bottom-right. The combined use of ST's and masks provides a natural way to model depth ordering, where the object with the highest logit mask value ($\tilde{\mathbf{m}}_t^i$, channel-wise) becomes the foreground object. This allows us to capture occlusions between objects in a continuously differentiable manner.

**Auxiliary autoencoder loss** Using a constrained decoder ensures the encoder and decoder produces objects in consistent locations. However, it is hard to learn the full model from future frame prediction alone, since the encoder's training signal comes exclusively from the physics engine. To alleviate this and quickly build a good encoder/decoder representation, we add a static per-frame autoencoder loss.

**Training** During training we use $L$ input frames and predict the next $T_{pred}$ frames. Defining the frames produced by the decoder via the physics engine as $\tilde{I}_t^{\text{pred}}$ and the frames produced by the decoder using the output of the encoder directly as $\tilde{I}_t^{\text{ae}}$, the total loss is:

$$\mathcal{L}_{total} = \mathcal{L}_{\text{pred}} + \alpha\mathcal{L}_{\text{rec}} = \sum_{t=L+1}^{L+T_{pred}} \mathcal{L}(\tilde{I}_t^{\text{pred}}, I_t) + \alpha \sum_{t=1}^{L+T_{pred}} \mathcal{L}(\tilde{I}_t^{\text{ae}}, I_t) \tag{3.5}$$

where $\alpha$ is a hyper-parameter. We use mean-squared error loss throughout. During testing we predict an additional $T_{ext}$ frames in order to evaluate long term prediction beyond the length seen for training.

# 3.4 Experiments

## 3.4.1 Physical parameter learning and future prediction

**Setup** To explore learning physical parameters and evaluate long-term prediction we train our model on scenes with 5 different settings: two colored balls bouncing off the image edges; two colored balls connected by a spring; three colored balls with gravitational pull – all on a black background; and to test greater visual complexity, we also use 2 MNSIT digits connected by a spring, on a CIFAR background. We train using values of $(L, T_{pred}, T_{ext})$ set to $(3, 7, 20)$, $(3, 7, 20)$, $(3, 7, 20)$, $(4, 12, 24)$ and $(3, 7, 20)$, respectively. For the spring systems the physical parameters to be learned are the spring constant $k$ and equilibrium distance $l$, and for the gravitational system it is the gravity constant $g$ or mass of the objects $m$ (when learning gravity the mass if fixed, and vice-versa). In all cases we use objects with mass $m = 1$. The equations of motion used in these systems were:

- **2-balls and 2-digits spring:** The force applied on object $i$ by object $j$ follows Hooke's law:

$$\vec{F}_{i,j} = -k\,(\vec{p}_i - \vec{p}_j) - l\frac{\vec{p}_i - \vec{p}_j}{|\vec{p}_i - \vec{p}_j|}. \tag{3.6}$$

  Each step corresponds to an interval $\Delta t = 0.3$.

- **3-balls gravity:** The force applied on object $i$ by object $j$ follows Newton's law of gravity:

$$\vec{F}_{i,j} = -g\,m_i m_j \frac{\vec{p}_i - \vec{p}_j}{|\vec{p}_i - \vec{p}_j|^3} \tag{3.7}$$

  where the masses are set to 1. Each step corresponds to an interval $\Delta t = 0.5$.

- **Pendulum:** The pendulum follows the equations used by the OpenAI Gym environment:

$$\vec{F} = -\frac{3}{2}g\sin(\theta + \pi) + 3u \tag{3.8}$$

  where $u$ is the action. Each step corresponds to an interval $\Delta t = 0.05$. In the physics engine used by the model we introduce an extra actuation coefficient $a$ to be learned along with $g$:

$$\vec{F} = -\frac{3}{2}g\sin(\theta + \pi) + a \cdot u \tag{3.9}$$

**Training details** All datasets consist of 5000 sequences for training, 500 for validation, and 500 for testing. We use a learnable ST scale parameter initialized at $s = 2$ in the balls datasets and $s = 1$ in the digits dataset. In these datasets we set $\theta = 0$. For all datasets we use RMSProp Hinton et al. (2012) with an initial learning rate of $3 \times 10^{-4}$. For the balls and digits datasets we train

for 500 epochs with $\alpha = 2$, and divide the learning rate by 5 after 375 epochs. For the pendulum data we train for 1000 epochs using $\alpha = 3$, but divide the learning rate by 5 after 500 epochs. The image sizes are $32 \times 32$ for the 2-balls bouncing and spring, $36 \times 36$ for the 3-balls gravity, $64 \times 64$ for the 2-digits spring, and $64 \times 64$ grayscale for the pendulum.

The content and mask variables are the output of a neural network with a constant array of 1s as input and 1 hidden layer with 200 units and tanh activation. We found this easier to train rather than having the contents and masks as trainable variables themselves.

**Baselines** We compare our model to 3 strong baselines: DDPAE (Hsieh et al. 2018)[2], which is a generative model that uses an inverse-graphics model with black-box dynamics; VideoLSTM (Srivastava et al. 2015), which uses black-box encoding, decoding and dynamics; Interaction Network + Inverse-Graphics, which uses the same encoder and decoder as our Physics-as-Inverse-Graphics model, but where the dynamics module is an Interaction Network (Battaglia et al. 2016). The latter model allows us to compare explicit physics with relational dynamics networks, in terms of their ability to correctly capture object interactions[3]. All the models and their components are trained from scratch on our datasets.

**Results** Table 3.1 shows that our model finds physical parameters close to the ground-truth values used to generate the datasets, and Figure 3.2 shows the contents and masks learned by the decoder. This highlights the fact that the proposed model can successfully perform unsupervised system identification from pixels.

| **Dataset** | **2-balls spring** | **2-digits spring** | **3-balls gravity** | |
|---|---|---|---|---|
| Parameters | $(k,\ l)$ | $(k,\ l)$ | $g$ | $m$ |
| Learned value | $(4.26,\ 6.17)$ | $(2.18,\ 12.24)$ | $65.7$ | $0.95$ |
| Ground-truth value | $(4.0,\ 6.0)$ | $(2.0,\ 12.0)$ | $60.0$ | $1.0$ |

**Table 3.1:** Physical parameters learned from video are within 10% of true system parameters.

Future frame predictions for two of the systems are shown in Figure 3.3, and per-step Structural Similarity Index (SSI) [4] of the models on the prediction and extrapolation range are shown in Figure 3.4. While all models obtain low error in the prediction range (left of the green dashed line), our model is significantly better in the extrapolation range. Even many steps into the future, our model's predictions are still highly accurate, unlike those of other black-box models (Figure 3.3). This shows the value of using an explicit physics model in systems where the dynamics are non-linear yet well defined. Further rollouts are shown in Appendix B, and we encourage the reader to watch the videos for all the datasets at https://sites.google.com/view/physicsasinversegraphics.

[2] Using the code provided by the authors.

[3] This baseline also serves as strong proxy for comparison with recent relational models (Watters et al. 2017; Steenkiste et al. 2018), which due to their supervision method or input-output space cannot be directly compared our model.

[4] We choose SSI over MSE as an evaluation metric as it is more robust to pixel-level differences and alignment.

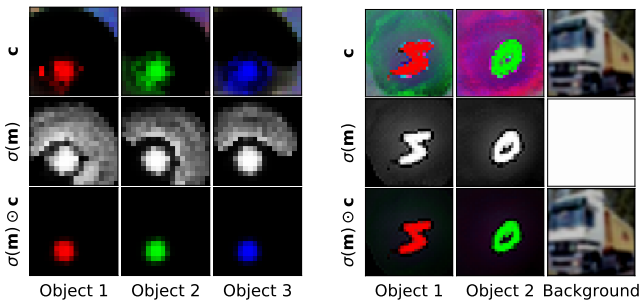



**Figure 3.2:** Contents and masks learned by the decoder. Object masks: $\sigma(\mathbf{m})$. Objects for rendering: $\sigma(\mathbf{m}) \odot \mathbf{c}$. Contents and masks correctly capture each part of the scene: colored balls, MNIST digits and CIFAR background. We omit the black background learned on the balls dataset.

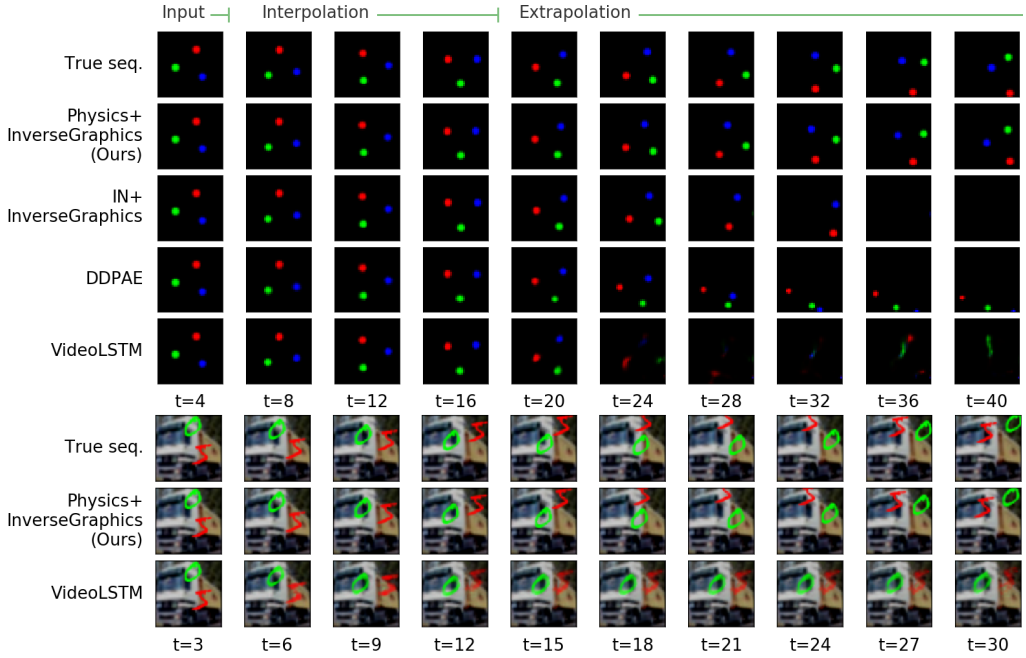



**Figure 3.3:** Future frame predictions for 3-ball gravitational system (**top**) and 2-digit spring system (**bottom**). IN: Interaction Network. Only the combination of Physics and Inverse-Graphics maintains object integrity and correct dynamics many steps into the future.

This difference in performance is explained in part by the fact that in some of these systems the harder-to-predict parts of the dynamics do not appear during training. For example, in the gravitational system, whiplash from objects coming in close contact is seldom present in the first $K + T_{pred}$ steps given in the training set, but it happens frequently in the $T_{ext}$ extrapolation steps evaluated during testing. We do not consider this to be a failure of black-box model, but rather a consequence of the generality *vs* specificity tradeoff: a model without a sufficiently strong inductive bias on the dynamics is simply not able to correctly infer close distance dynamics from long distance dynamics.

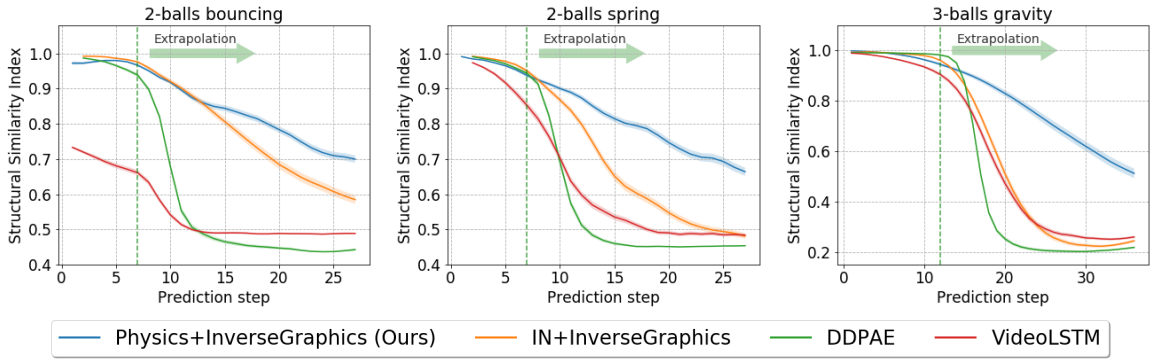



**Figure 3.4:** Frame prediction accuracy (SSI, higher is better) for the balls datasets. Left of the green dashed line corresponds to the training range, $T_{pred}$, right corresponds to extrapolation, $T_{ext}$. We outperform Interaction Networks (IN) (Watters et al. 2017), DDPAE (Hsieh et al. 2018) and VideoLSTM (Srivastava et al. 2015) in extrapolation due to incorporating explicit physics.

### 3.4.2 Vision-based model-predictive control (MPC)

**Tasks** One of the main applications of our method is to identify the (actuated) dynamical parameters and states of a physical system from video, which enables vision-based planning and control. Here we apply it to the pendulum from OpenAI Gym (Brockman et al. 2016) – one typically solved from proprioceptive state, not pixels. For training we collect 5000 sequences of 14 frames with random initialization ($\dot{\theta}_0 \sim \text{Unif}(-6, 6)$) and actions ($u_t \sim \text{Unif}(-2, 2)$). The physical parameters to learn are gravity $g = 10.0$ and actuation coefficient $a = 1.0$. We use $K = 4$ and $T_{pred} = 10$. We use the trained MPC model as follows. At every step, the previous 4 frames are passed to the encoder and velocity nets to estimate $[\theta_t, \dot{\theta}_t]$. This is passed to the physics engine with learned parameters $g$ and $a$. We perform 100-step model-predictive control using the cross entropy method (Rubinstein 1997), exactly as described in Hafner et al. (2019), setting vertical position and zero velocity as the goal.

**Baselines** We compare our model to an oracle model, which has the true physical parameters and access to the true pendulum position and velocity (not vision-based), as well as a concurrent state-of-the art model-based RL method (PlaNet (Hafner et al. 2019)), and a model-free[5] deep deterministic policy gradient (DDPG) agent (Lillicrap et al. 2016). To provide an equivalent comparison to our model, we train PlaNet on random episodes.

**Results** In terms of system identification, our model recovers the correct gravity ($g = 9.95$) and force coefficient ($a = 0.99$) values from vision alone, which is a prerequisite to perform correct planning and control. Figure 3.5 (top-left) highlights the data efficiency of our method, which is comparable to PlaNet, while being dramatically faster than DDPG from pixels. Importantly, the interpretability of the explicit physics in our model provides some unique capabilities. We can

[5]DDPG, TRPO and PPO learned from pixels failed to solve the pendulum, highlighting the complexity of the vision-based pendulum control task and brittleness of model-free reinforcement learning strategies.

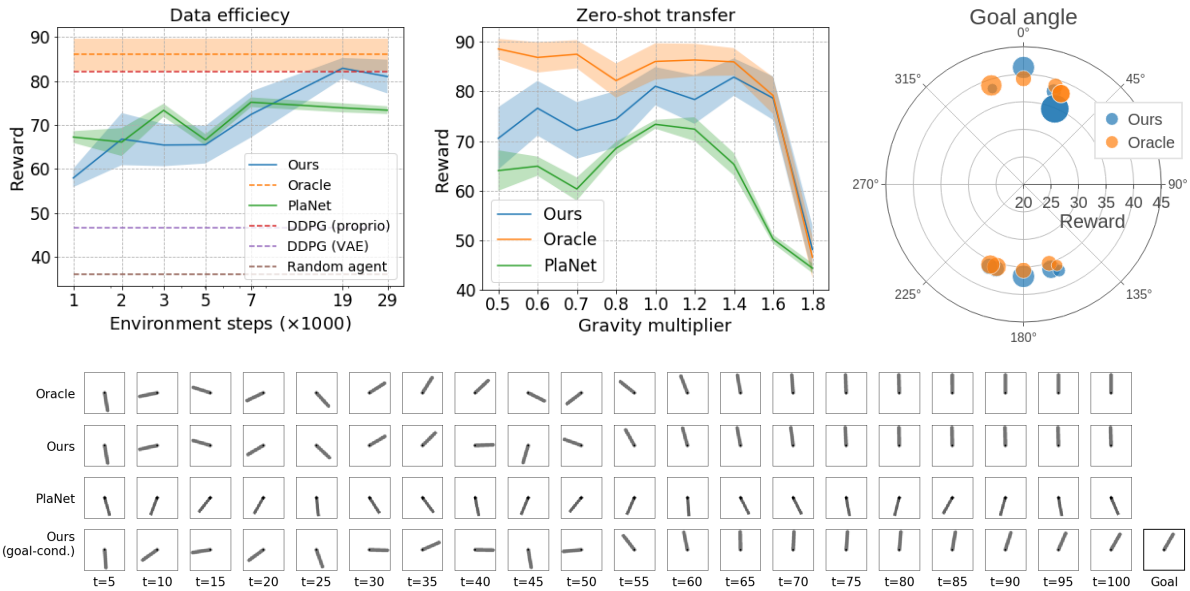



**Figure 3.5: Top:** Comparison between our model and PlaNet Hafner et al. (2019) in terms of learning sample efficiency **(left)**. Explicit physics allows reasoning for zero-shot adaptation to domain-shift in gravity (**center**) and goal-driven control to balance the pendulum in any position (**right**). DDPG (VAE) corresponds to a DDPG agent trained on the latent space of an autoencoder (trained with 320k images) after 80k steps. DDPG (proprio) corresponds to an agent trained from proprioception after 30k steps. **Bottom:** The first 3 rows show a zero-shot counterfactual episode with a gravity multiplier of 1.4 for an oracle, our model and planet, with vertical as the target position (as trained). The last row shows an episode using a goal image to infer the non-vertical goal state.

perform simple counter-factual physical reasoning such as '*How should I adapt my control policy if gravity was increased?*', which enables zero-shot adaptation to new environmental parameters. Figure 3.5 (top-middle) shows that our model can exploit such reasoning to succeed immediately over a wide range of gravities with no re-training. Similarly, while the typical inverted pendulum goal is to balance the pendulum upright, interpretable physics means that this is only one point in a space of potential goals. Figure 3.5 (top-right) evaluates the goal-paramaterized control enabled by our model. Any feasible target angle specified can be directly reached by the controller. There is extrapolative generalisation across the space of goals even though only one goal (vertical) was seen during training. Importantly, these last two capabilities are provided automatically by our model due to its disentangled interpretable representation, but cannot be achieved without further adaptive learning by alternatives that are reward-based (Mnih et al. 2016) or rely on implicit physics (Hafner et al. 2019).

# 3.5 Ablation studies

## 3.5.1 Loss and training ablations

Since the encoder and decoder must discover the objects present in the image and the corresponding locations, one might assume that the velocity estimator and physics engine could be learned using only the prediction loss, and encoder/decoder using only the static autoencoder loss, i.e., without joint training. In Table 3.2 we compare the performance of four variants on the 3-ball gravity dataset: joint training using only the prediction loss; joint training using the prediction and autoencoder losses; training the encoder/decoder on the autoencoder loss and the velocity estimator and physics engine on the prediction loss; and joint training, but using an MLP black-box decoder.

| Train using | $\mathcal{L}_{\text{pred}}$ | $\mathcal{L}_{\text{rec}}$ |
|---|---|---|
| only $\mathcal{L}_{\text{pred}}$ | 31.4 | 20.5 |
| separate gradients | 28.1 | 0.22 |
| joint $\mathcal{L}_{\text{pred}} + \alpha\mathcal{L}_{\text{rec}}$ | **1.39** | 0.63 |
| black-box decoder, joint | 30.9 | 2.87 |

**Table 3.2:** Test loss under different training conditions. Separate gradients: Train encoder/decoder on $\mathcal{L}_{\text{rec}}$, and velocity estimator and physics engine on $\mathcal{L}_{\text{pred}}$. Black-box decoder, joint: Joint training using a standard MLP network as the decoder. Only joint training using our coordinate-consistent decoder succeeds.

We can see that only joint prediction and autoencoder loss obtain satisfactory performance, and that the use of the proposed coordinate-consistent decoder is critical. The prediction loss is essential in order for the model to learn encoders/decoders whose content and masks can be correctly used by the physics engine. This can be understood by considering how object interaction influences the decoder. In the gravitational system, the forces between objects depend on their distances, so if the objects swap locations, the forces must be the same. If the content/mask learned for each object are centered differently relative to its template center, rendering the objects at positions $[x, y]$ and $[w, z]$, or $[w, z]$ and $[x, y]$ will produce different distances between these two objects in image space. This violates the permutation invariance property of the system. Learning the encoder/decoder along with the velocity estimator and physics engine on the prediction loss allows the encoder and decoder to learn locations and contents/masks that satisfy the characteristics of the system and allows the physics to be learned correctly. In the next section we perform further ablations on the decoder architecture and its ability to correctly render objects in regions of the image not seen during training.

### 3.5.2 Decoder extrapolation to unseen image regions

One limitation of standard fully-connected or deconvolutional decoders is the inability to decode states corresponding to object poses or locations not seen during training. For example, if in the training set no objects appear in the bottom half of the image, a fully-connected decoder will simply learn to output zeros in that region. If in the test set objects move into the bottom half of the image, the decoder lacks the inductive bias necessary to correctly extrapolate in image space.

To test this hypothesis, we replaced our model's decoder with a Deconv and Spatial Broadcast (Watters et al. 2019b) decoder, and compared them in a spatial extrapolation experiment. In this experiments, objects never enter the bottom half of the image in the input and prediction range, though in the extrapolation range in the test set objects move to this region of the scene. In the rollouts shown in Fig. 3.6, Broadcast performs better than Deconv, but they both fail to maintain object integrity when the balls move to the bottom half of the image in the extrapolation steps, validating our hypothesis that a black-box decoder has insufficient extrapolation ability. In contrast, our rendering decoder is be able to correctly decode states not seen during training.

In the limit that our renderer corresponds to a full-blown graphics-engine, any pose, location, color, etc. not seen during training can still be rendered correctly. This property gives models using rendering decoders, such as ours and Hsieh et al. (2018), an important advantage in terms of data-efficiency. We note, however, that in general this advantage does not apply to correctly inferring the states from images whose objects are located in regions not seen during training. This is because the encoders used are typically composed simply of convolutional and fully-connected layers, with limited de-rendering inductive biases.

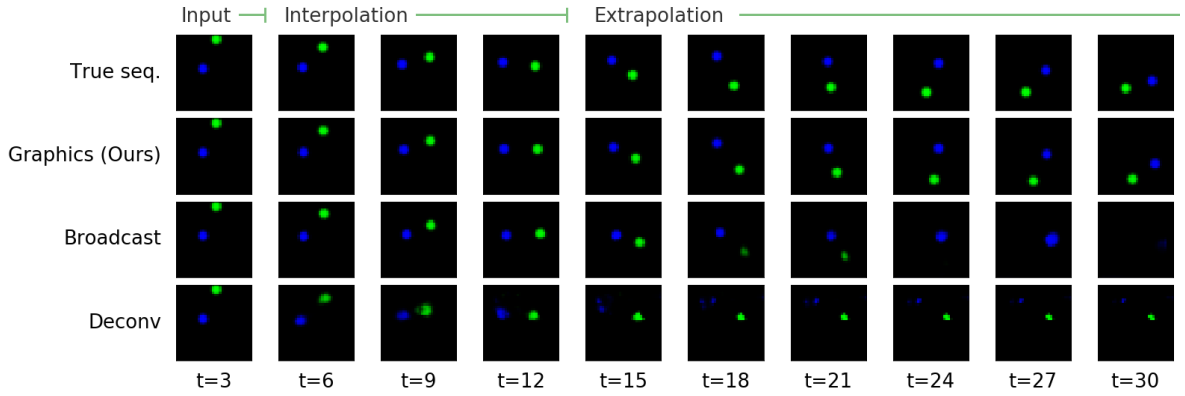



**Figure 3.6:** Comparison between graphics decoder and two black-box decoders, trained on data where objects only appear in the top half of the scene. Only the graphics decoder is able to correctly render the objects in the bottom half of the scene at test time. Broadcast: spatial broadcast decoder (Watters et al. 2019b); Deconv: standard deconvolutional network.

### 3.5.3 Incorrect number of object slots

The model proposed assumes we know the number of objects present in the scene. Here we briefly explore how to the model behaves when we use an incorrect number of slots $N$. We use

the gravitational system, since interaction forces between objects are easy to generalize for any $N$. Fig. 3.7, left, shows that when using only 2 object slots, two of the objects are found, since the model does not have capacity to find more. Fig. 3.7, right, shows that when using more slots than the number of objects in the scene, all objects are discovered, and extra slots are left empty. However, in both cases we found predictive performance to be subpar, since in one case there are objects missing to correctly infer interactions, and in the other there are interactions between object slots and empty slots, confusing the dynamics.

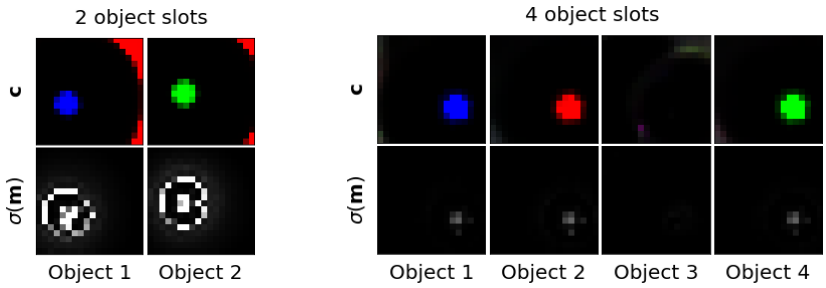



**Figure 3.7:** Results for incorrect number of object slots in the physics engine for the 3-body gravitational system **Left:** Contents and masks learned for 2 object slots. **Right:** Contents and objects learned for 4 object slots.

## 3.6 Limitations

Alhough the approach presented here shows promising results in terms of physical parameter estimation, long-term video prediction and MPC, a number of limitations need to be overcome for real-world application.

**Templates as object representation** Though the assumption that every scene in a dataset is a combination of learnable templates is a common one in the literature (c.f. Tieleman (2014) for an extensive study on this), this is insufficient to model real-world scenes. For example, applying physics-as-inverse-graphics to the Physics101 dataset would require representing objects using a latent appearance representation that could be used by the decoder (Eslami et al. 2016). This would introduce new modelling challenges, requiring object tracking to keep correct object identity associations (Kosiorek et al. 2018). In this work we simplify this problem by assuming that objects are visually distinct throughout the dataset, though this does not detract from the essential contributions of the paper.

**Rigid sequence to sequence architecture** In this work we used a sequence-to-sequence architecture, with a fixed number of input steps. This architectural choice (inspired by Watters et al. (2017)), prevents the model from updating its state beliefs if given additional input frames later in the sequence. Formulating the current model in a probabilistic manner that would allow for state/parameter filtering and smoothing at inference time is a promising direction of future work.

**Static background assumption** Many scenes of interest do not follow the assumption that the only moving objects in the scene are the objects of interest (even though this assumption is widely used). Adapting our model to varying scene backgrounds would require additional components to discern which parts of the scene follow the dynamics assumed by the physics engine, in order to correctly perform object discovery. This is a challenging problem, but we believe it would greatly increase the range of applications of the ideas presented here.

## 3.7 Conclusion

Physics-as-inverse graphics provides a valuable mechanism to integrate inductive bias about physical data generating processes into learning. This allows unsupervised object tracking and system identification, in addition to sample efficient, generalisable and flexible control. However, incorporating this structure into lightly supervised deep learning models has proven challenging to date. We introduced a model that accomplishes this, relying on a coordinate-consistent decoder that enables image reconstruction from physics. We have shown that our model is able to perform accurate long term prediction and that it can be used to learn the dynamics of an actuated system, allowing us to perform vision-based model-predictive control.

Chapter 4

# Vision-based System Identification and 3D Keypoint Discovery using Dynamics Constraints

This chapter corresponds to the paper:

Miguel Jaques, Martin Asenov, Michael Burke, and Timothy Hospedales. Vision-based System Identification and 3D Keypoint Discovery using Dynamics Constraints. *arXiv preprint arXiv:2109.05928*, 2021.

This chapter introduces V-SysId, a novel method that enables simultaneous keypoint discovery, 3D system identification, and extrinsic camera calibration from an unlabeled video taken from a static camera, using only the family of equations of motion of the object of interest as weak supervision. V-SysId takes keypoint trajectory proposals and alternates between maximum likelihood parameter estimation and extrinsic camera calibration, before applying a suitable selection criterion to identify the track of interest. This is then used to train a keypoint tracking model using supervised learning. Results on a range of settings (robotics, physics, physiology) highlight the utility of this approach.

## 4.1 Introduction

An understanding of the motion and physics of objects in the real world is a hallmark of the human visual system. Humans have the ability to identify objects and their properties (e.g. mass, friction, elasticity) as they move and interact in the world, due to our intuitive understanding of common trajectories, object interactions, and outcomes. This ability is typically studied under the umbrella of *intuitive physics* (Battaglia et al. 2013; Ullman et al. 2014; Hamrick et al. 2016; Baker et al. 2017), and often considered a critical component for machines to be able to think more like humans.

In the context of machine learning systems, this ability can be distilled to a requirement for *unsupervised* 3D object localization and physical parameter estimation (also known as system identification) from a sensory stream, subject to some inductive bias or intuitive physics prior.

Taking inspiration from this view, this paper introduces V-SysId, a novel method that enables simultaneous keypoint discovery, 3D system identification, and extrinsic camera calibration from a

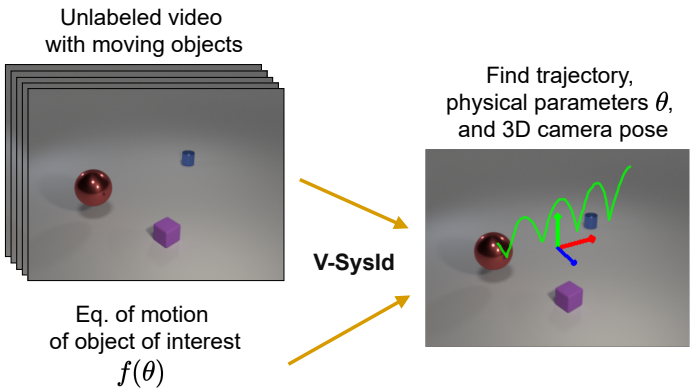



**Figure 4.1:** Problem statement. Given an unlabeled video containing moving objects and an equation of motion, our model (V-SysId) identifies the trajectory corresponding to the object of interest, along with its physical parameters (e.g. restitution coefficient, initial height), and 3D pose relative to the camera.

single unlabeled video taken from a static camera, using only the family of equations of motion of the object of interest as weak supervision. Crucially, our approach is able to identify the correct object(s) in a scene even in the presence of other moving objects or distractors. This property is key, as it greatly increases applicability to cluttered real world environments. This allows us to perform queries such as "*find the 3D location of the bouncing ball, and determine its restitution coefficient*" (Fig. 4.1).

V-SysId follows a 3-stage process of keypoint track proposal, optimization, and selection (Fig. 4.2). The optimisation process alternates between maximum likelihood extrinsic camera calibration and maximum likelihood physical parameter estimation for motion tracks detected in video. This joint optimisation can be unstable, which we address through the inclusion of a curriculum-based optimisation strategy, alongside a maximum entropy criterion for keypoint identification. A key benefit of V-SysId is that a neural network is *not* needed for discovery or system identification in our pipeline. This means that V-SysId enables keypoint discovery with high-resolution images; and can also perform system identification in *single* videos, without the need to obtain large datasets, which is particularly useful in robotics applications, where data collection for neural network training can be laborious and time-consuming. The keypoints discovered by V-SysId can then be used as pseudo-labels to train a supervised keypoint detector, to provide keypoint inference at test time for tracking or control.

These properties provide significant flexibility to V-SysId, enabling its use in real world environments with important applications for control, physics understanding, and health monitoring. Specifically, we show that the V-SysId can be applied to end-effector localization and extrinsic camera calibration, bouncing ball discovery and physical property estimation, and breathing frequency estimation from chest videos - all unlabeled and without regions of interest provided a priori. This is made possible by the fact that V-SysId identifies keypoints belonging to objects of interest present in scenes, while ignoring any other moving objects or artifacts that do not follow the expected

dynamical constraints. This alleviates the need for hand-crafted object segmentation methods or tricks to selectively remove parts of the image that may contain moving distractors; and allows keypoint discovery at a fraction of the computational expense of unsupervised neural methods that learn to identify and model every moving object in an image.

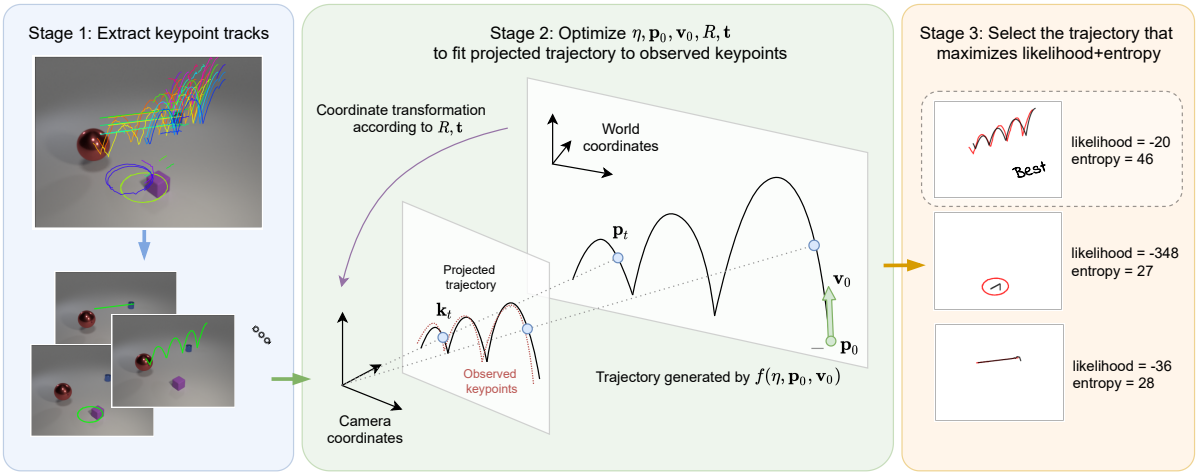



**Figure 4.2:** Our V-SysId comprises 3 stages. Stage 1 extracts keypoint tracks from a video using a grid keypoint detector + KLT tracking. Each of these 2D tracks is passed to Stage 2, where the physical parameters $\theta = \{\boldsymbol{\eta}, \mathbf{p}_0, \mathbf{v}_0\}$ of the 3D equation of motion $f$, and the camera pose parameters $R, \mathbf{t}$ are optimized in order to minimize the difference between the projected 3D trajectory (black, Stage 2) and the 2D keypoint track observed (red, Stage 2). Stage 3 chooses the best trajectory and corresponding parameters as those which maximize the sum of projected likelihood and a trajectory entropy criterion. Here, a bouncing ball scene with 2 moving distractors is shown, where the bouncing ball is correctly discovered as the object that corresponds to the highest entropy motion that fits the equation of motion $f$.

## 4.2 Related Work

**System identification** and physics understanding are key to allow machine learning agents to interact with the real world. System identification is typically performed using proprioceptive trajectory data directly, and there has been extensive research across a range of fields (Juang et al. 1985; Brincker et al. 2001; Wu et al. 2015; Brunton et al. 2016a; Wu et al. 2017a; Li et al. 2020b) in support of this. Recent contributions include developments in physical parameter estimation (Belbute-Peres et al. 2018; Cranmer et al. 2020a), simulator learning (Qiao et al. 2020; Sanchez-Gonzalez et al. 2020), simulation alignment for robot interaction (Asenov et al. 2019), trajectory generation (Jegorova et al. 2020) and compositionality (Abraham et al. 2017; Li et al. 2020a).

Unsupervised system identification from vision is a recent area of research that removes the requirements for trajectory data, with approaches including unsupervised physical parameter estimation (Jaques et al. 2020), structured latent space learning (Karl et al. 2017; Guen et al. 2020; Jaques et al. 2021), and hamiltonian or lagrangian learning (Greydanus et al. 2019; Toth et al. 2020;

Zhong et al. 2020b). Unfortunately, these approaches are still relatively limited in the complexity of scene they can model (both visually and dynamically), and typically restricted to toy problems and simulated environments, often only in 2D.

The seminal GALILEO model (Wu et al. 2015) demonstrated physical system identification and simulation alignment using the Physics101 dataset (Wu et al. 2016). A key shortcoming of Galileo is that it assumes that the camera is parallel to the plane of motion, and relies on manually identified object tracks to lift the visual scenes onto object positions. In contrast V-SysId is able to simultaneously estimate 3D trajectories and camera pose relative to the scene from arbitrary camera angles, greatly increasing its applicability to real world scenes. Furthermore, V-SysId automatically identifies object tracks from keypoint proposals without needing human intervention, allowing us to automatically discover the objects of interest in video that are governed by the relevant equations of motion.

**Keypoint discovery** Keypoints are a natural representation for object parts, with keypoint detection and tracking one of the earliest and most studied areas of computer vision. Approaches like SIFT (Lowe 2004), FAST (Rosten et al. 2006) and ORB (Rublee et al. 2011) are still widely used to perform SLAM, SFM, VO[1] and other tracking tasks (using, e.g. a KLT tracker (Tomasi et al. 1991)). Given keypoint trajectories, the problem of inferring the 3D structure of a 2D trajectory using assumptions about the dynamics has been coined "trajectory triangulation" by (Avidan et al. 2000; Kaminski et al. 2002), who assume that objects follow a straight-line or conic-section trajectory in 3D space, and that physical parameters can be uniquely identified using multiple cameras. In contrast, our method assumes only a single static monocular view. Other approaches to infer moving object structure using motion constraints include (Fitzgibbon et al. 2000; Han et al. 2003; David et al. 2004; Scaramuzza et al. 2009).

When it comes to 2D keypoint discovery, several recent works have proposed neural network based methods that use a regularized reconstruction objective to discover objects of interest in an image (Jakab et al. 2018, 2019; Kulkarni et al. 2019; Minderer et al. 2019; Das et al. 2020; Gopalakrishnan et al. 2020), which can be used for downstream control tasks. However, these approaches lack the ability to estimate keypoint depth, limiting their application in realistic control scenarios. Even though these approaches obtain semantically meaningful keypoints (and in some instances are able to ignore scene objects with unpredictable motion (Gopalakrishnan et al. 2020)), they require visual inspection in order to obtain interpretability. In contrast, V-SysId provides equation-driven keypoint discovery, ensuring a known semantic meaning for learned keypoints.

A parallel stream of research tackles this from a geometric perspective, where 3D keypoints are inferred using camera motion cues or geometric constraints (Vijayanarasimhan et al. 2017; Suwajanakorn et al. 2018; Jau et al. 2020; Wei et al. 2020). Even though this approach has been used in

[1] Simultaneous Localisation and Mapping, Structure-from-Motion, Visual Odometry.

complex real world settings, these keypoints lack semantic meaning, making these unsuitable for semantic discovery queries (eg. "find the bouncing ball following these dynamics").

The use of dynamics as a learning constraint in this way has not been explored in keypoint discovery literature to date. This work proposes a method to integrate dynamical inductive biases into the keypoint discovery process, enabling extrinsic camera calibration and physics-guided discovery of objects of interest alongside the corresponding physical parameter estimation.

# 4.3 Method

Our goal is to discover the 3D trajectory of an object of interest in a video with possibly many moving objects, given only its family of motion dynamics, $f$. To this end, we must estimate: a) 2D keypoint locations $\mathbf{k}_t$ of the object of interest in each frame $\mathbf{I}_t$; b) physical parameters and initial conditions $\boldsymbol{\theta}$, of the equation of motion $f(\boldsymbol{\theta})$; and c) camera rotation and translation relative to the scene $[R, \mathbf{t}]$.

Joint estimation of these quantities would be intractable, so we split the objective into multiple tractable components. Our method, V-SysId, can be broken down into 3 stages:

1. Keypoint track proposal;
2. Joint physical parameter and camera pose estimation;
3. Trajectory selection.

These stages are depicted in Fig. 1. We start by describing the physical parameter+camera pose estimation stage, as it is helpful for understanding the trajectory proposal stage.

## 4.3.1 Physical parameter and camera pose estimation

**Setup** Let us assume we have a set of N 2D keypoint tracks $\mathbf{K} = \{\tilde{\mathbf{k}}^n_{1:T}\}_{n=1}^N$ across the video $\mathbf{I}_{1:T}$, and a family of 3D equations of motion $f$ with unknown physical parameters $\boldsymbol{\eta}$ and initial position and velocity $\mathbf{p}_0$ and $\mathbf{v}_0$, respectively. The equation $f$ can be rolled out over $T$ time steps using a standard integration method in order to obtain a 3D trajectory $\mathbf{p}_{1:T} = f(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\eta}, \mathbf{p}_0, \mathbf{v}_0\}$.

**Objective** Our goal is to maximize the likelihood of the observed keypoint trajectory $\tilde{\mathbf{k}}_{1:T}$ w.r.t. the physical parameters and initial conditions, $\boldsymbol{\theta}$, and the extrinsic camera rotation and translation, $[R\ \mathbf{t}]$:

$$\boldsymbol{\theta}^*, R^*, \mathbf{t}^* = \underset{\boldsymbol{\theta}, R, \mathbf{t}}{\arg\max}\ p(\tilde{\mathbf{k}}_{1:T} | \boldsymbol{\theta}, R, \mathbf{t}), \tag{4.1}$$

where we factorize the trajectory likelihood as:

$$\begin{aligned} p(\tilde{\mathbf{k}}_{1:T}|\boldsymbol{\theta}, R, \mathbf{t}) &= \prod_t p(\tilde{\mathbf{k}}_t|\boldsymbol{\theta}, R, \mathbf{t}) \\ &= \prod_t \mathcal{N}(\tilde{\mathbf{k}}_t|\mathbf{k}_t(\boldsymbol{\theta}, R, \mathbf{t}), \sigma^2), \end{aligned} \tag{4.2}$$

and $\mathbf{k}_t(\boldsymbol{\theta}, R, \mathbf{t})$ are the 2D projection of the simulated 3D trajectory *(given by $f(\theta)$)*:

$$\begin{aligned} \mathbf{k}_t(\boldsymbol{\theta}, R, \mathbf{t}) &= [\tilde{\mathbf{p}}_{x,t}/\tilde{\mathbf{p}}_{z,t}, \tilde{\mathbf{p}}_{y,t}/\tilde{\mathbf{p}}_{z,t}] \\ \tilde{\mathbf{p}}_t &= \mathbf{M}\,[R\ \mathbf{t}]\,\mathbf{p}_t \end{aligned} \tag{4.3}$$

with $\mathbf{M}$ being the intrinsic camera matrix. In this work we assume the camera intrinsics are known.

In order to reduce the space of possible solutions (and therefore local minima) of Step 1 above, we restrict the camera rotation matrix $R$ to have roll $= 0$. This means the camera cannot rotate about its projection axis, which is the case in the vast majority of settings. Using the projection plane in camera coordinates as $xy$ and the projection axis as $z$, we parametrize $R$ as $R(\alpha, \beta) =$ EulerRotationMatrix$(\alpha, \beta, 0)$, where $\alpha$, $\beta$ and $\gamma = 0$ correspond to the pitch, yaw and roll Euler angles, respectively. We found that this parametrisation greatly improves results and optimisation stability.

**Optimization** To maximise (4.2) we apply an iterative optimisation procedure. Given an initial estimate for $\boldsymbol{\theta}$, $R$ and $\mathbf{t}$, we alternate the following steps until convergence:

1. Keeping $\boldsymbol{\theta}$ fixed, maximise (4.2) w.r.t. $R$ and $\mathbf{t}$ using gradient descent;
2. Keeping $R$ and $\mathbf{t}$ fixed, maximise (4.2) wrt $\boldsymbol{\theta}$ using gradient descent (with numerical or analytical (Belbute-Peres et al. 2018) differentiation) or global optimiser (e.g. Cross-Entropy Maximisation (Rubinstein 1997); Bayesian Optimisation (Mockus 1989)).

Estimation of the physical parameters over the full sequence (possibly hundreds of timesteps) is prone to local minima, as the dependency on the parameters can be highly non-linear[2]. This is further affected by the use of a non-optimised camera pose at the first iteration. In order to address this, we start by performing a step of physical parameter and pose estimation on a small initial trajectory interval, $T_0$, adding $m$ points to the trajectory at each iteration, as described in the appendix, Algorithm 1.

[2]Global optimisers have a slight advantage in this case, although they require very many iterations to find a good minimum.

### 4.3.2 Trajectory proposal

In an unlabelled video, ground-truth 2D keypoints are not available, but keypoint trajectories are required to maximise the likelihood in (4.2). Joint estimation of physical parameters and a neural network-based keypoint detector would be hard to optimise due to the difficulty of backpropagating through physics rollouts and camera projection into a CNN in a stable manner (as suggested by (Jaques et al. 2020)). Therefore, we propose a simpler, more robust approach: We extract keypoints from the first frame of the video using a keypoint detector, and track them using an optical-flow-based tracker. This produces a set of 2D keypoint tracks $\tilde{\mathbf{k}}_{1:T}$, and allows physical parameter+pose estimation to be performed for each track independently.

### 4.3.3 Trajectory selection

Once the physical parameters and pose are estimated for each keypoint track, the best tracklet can be identified by isolating the highest projection likelihood (4.2). However, in order to prevent trivial keypoint tracklets from being chosen (since a static keypoint will easily attain maximal likelihood), we add a temporal entropy term to the likelihood, such as the temporal standard deviation of the observed trajectory, resulting in the following selection criterion:

$$\begin{aligned} n^{\text{best}} &= \underset{n \in 1..N}{\arg\max}\ \text{SelectionCriterion}(\tilde{\mathbf{k}}^n_{1:T} | \boldsymbol{\theta}, R, \mathbf{t}) \\ &= \underset{n \in 1..N}{\arg\max}\ p(\tilde{\mathbf{k}}^n_{1:T} | \boldsymbol{\theta}, R, \mathbf{t}) + \text{Stddev}_t(\tilde{\mathbf{k}}^n_{1:T}). \end{aligned} \tag{4.4}$$

Intuitively, this criterion (which had been proposed by Stewart et al. (2017) in a similar but simplified setting) finds the highest entropy trajectory that satisfies the physical motion constraints.

The full V-SysId procedure is depicted in Fig. 4.2 and pseudocode is shown in Algorithm 1.

**Algorithm 1** V-SysId

**Input:** Video $V$ of length $T$
**Input:** Equation of motion $f$ of the object of interest
**Input:** KeypointTrackExtractor # function that outputs a set of keypoint track proposals
**Output:** Trajectory, physical parameters, and camera pose of the object of interest

```
# Get N keypoint track proposals
Tracks ← KeypointTrackExtractor(V)

# Fit physical parameters and camera pose to trajectory
SelectionCriterion ← [ ]
Params ← [ ]
for n ∈ {1...N} do
    k̃_{1:T} ← Tracks[n]
    Initialize α ← 0, β ← 0, t ← [0,0,0], v_0 ← [0,0,0];
    Initialize p_0 as the projection of k̃_0 onto the z = 5 plane in world coordinates;
    Initialize η to some sensible initial values (setting dependent);
    for t ∈ {1...T} do
        θ ← arg max_θ  p(k̃_{1:t} | θ, R, t)
        R, t ← arg max_{R,t}  p(k̃_{1:t} | θ, R, t)
    end for
    Append {θ, R, t} to Params
    Append the scalar p(k̃_{1:t} | θ, R, t) + H(k̃_{1:T}) to SelectionCriterion
end for

# Trajectory selection
n* ← arg max SelectionCriterion
k̃* = Tracks[n*]
θ*, R*, t* = Params[n*]
return k̃*, θ*, R*, t*
```

### 4.3.4 Inference at test-time

Once the V-SysId procedure is complete, keypoints are available for the objects of interest in each frame in the video. These can be treated as pseudo-ground truth keypoints, and used to train a neural network (or another visual object detector) by supervised learning, in order to perform fast keypoint detection at test-time.

### 4.3.5 Challenges

**Scale unidentifiability** Due to the projection operation $\mathbf{k}_t(\boldsymbol{\theta}, R, \mathbf{t}) = [\tilde{\mathbf{p}}_{x,t}/\tilde{\mathbf{p}}_{z,t}, \tilde{\mathbf{p}}_{y,t}/\tilde{\mathbf{p}}_{z,t}]$, the 3D trajectory $\mathbf{p}_{1:T}$ can only be determined up to a scale parameter. For this reason, we evaluate the correlation between the true- and learned parameters, not the error. This is also the metric used by GALILEO and Physics101 when doing physical parameter estimation from visual trajectories. Although scale unidentifiability leads to the existence of infinitely many solutions for $\boldsymbol{\theta}$ and $\mathbf{t}$, our use

of alternate optimization steps guarantees that a single solution is found, as $\boldsymbol{\theta}$ and $\mathbf{t}$ are optimized conditioned on one another, not jointly.

**Broken trajectories and occlusions** In settings where classical keypoint detectors are unreliable, one can either use state-of-the-art pretrained keypoint networks, like SuperPoint (DeTone et al. 2018) and LF-Net (Ono et al. 2018), or pretrain an unsupervised keypoint discovery network (Jakab et al. 2018; Kulkarni et al. 2019; Minderer et al. 2019). However, we found show that a simple grid keypoint detector yielded more reliable tracks than using classic (SIFT, ORB, FAST), or modern (SuperPoint, LF-Net) keypoint detectors.

In settings where standard optical flow computation is unreliable, more recent models (such as FlowNet (Fischer et al. 2015)) could be used to provide flow estimates to the KLT tracker. More recent improvements to the KLT tracker like the CoMaL (Ramakrishnan et al. 2016) algorithm could also be used. The multi-stage, modular nature of our pipeline allows for the easy replacement of individual components, although we found standard optical flow computation to work very well in practice.

**Camera roll set to zero:** We found that setting the camera roll angle to zero greatly stabilized the optimization procedure. While this might be perceived as too strong of a constraint on the model, in the vast majority of real settings the camera has zero roll (i.e. it's rare for the camera to rotate around its projection axis). Therefore, imposing this constraint does not reduce the applicability of our method in the vast majority of cases, while providing improved results. Naturally, allowing roll optimization would make the model more general, but this is left as future work.

## 4.4 Experiments

**Keypoint detection and tracking** We detect keypoints in the first frame by using taking the locations of a 10x10 grid across the frame, and use the KLT algorithm to track these across the video. We show comparisons between grid, ORB, SuperPoint and LF-Net keypoint detectors in Section 4.5.

**Track filtering** Since the grid keypoint detector extracts hundreds of keypoints, we remove tracks whose length is less than 60% of the full video, and whose temporal stddev (4.4) is less than 10 pixel, prior to optimization. This reduces computation, as physical parameter + pose estimation is performed on only the most feasible tracks.

**Physical parameter estimation** The gradient-based method BFGS (Fletcher 2000)[3] is used with numerical derivatives for the physical parameter optimization step. Although (Belbute-Peres et al. 2018) provides an elegant method for analytical differentiation through contacts, we found it much

[3] As implemented by Scipy (Virtanen et al. 2020).

harder to implement, and ultimately slower, than simply using BFGS. Additionally, (Belbute-Peres et al. 2018) has only been applied to noiseless simulated environments and assumes knowledge of the true initial position and velocity of the objects, which is not available here. Since the equations of motion considered here are planar, the $z$ component of $\mathbf{v}_0$ is constrained to $0$. The remaining parameters are learnable.

On the first iteration, the initial position $\mathbf{p}_0$ is set to be the reprojection of the first 2D keypoint $\tilde{\mathbf{k}}_0$ onto the $z = 5$ plane in world coordinates. This results in an initial position whose camera projection is the first keypoint. The initial velocity is $\mathbf{v}_0 = [0, 0, 0]$. We found these settings essential to avoid local minima in the incremental optimization.

**Camera pose estimation** BFGS is also used with finite differencing for the camera pose optimization step. The parametrization of $R$ on pitch and yaw provides a smooth objective that is easy to optimize, whereas we found the PnP algorithm to result in large and not necessarily optimal jumps between steps. We initialize the camera pose parameters as $\alpha = 0$, $\beta = 0$, and $\mathbf{t} = [0, 0, 0]$.

**Curriculum-based optimization** We use 25 input frames to start the optimization, adding 10 frames per iteration until reaching the full length of the sequence.

### 4.4.1 Environments

**Franka Emika Panda Robot:** This sequence consists of a multi-joint robot arm (Franka Emika Panda) in a laboratory setting, where the goal is to find the end-effector's 3D location and the camera pose relative to this. The end-effector was programmed to do an archimedes spiral along an unknown 2D plane. The spiral is described by:

$$r = a + b \cdot t \quad ; \quad \theta = \theta_0 + \omega \cdot t \tag{4.5}$$

where $r$, $a$, $b$, $\theta_0$, $\omega$ are unknown parameters, to be learned by V-SysId, and $t$ is the time in seconds. A sequence of frames for this environment can be seen on Fig. 4.4, bottom. The video is 250 frames long, with a resolution of $640 \times 480$.

**Simulated bouncing ball:** This environment consists of a simulated bouncing ball with moving distractor objects. The bouncing ball follows the equation of motion:

$$\begin{cases} a_y = -g \ , \ \text{if } y > \text{floor} \\ v_y = -\epsilon \, v_y \ , \ \text{if } y = \text{floor} \\ v_x = v_{x_0} \\ v_z = 0 \end{cases} \tag{4.6}$$

where $a$ is the acceleration, $v$ is the velocity, $y$ is the ball height, $\epsilon \in [0, 1]$ is the restitution coefficient, and $g = 9.8$ is the gravity. The ball moves in the $z = z_0$ plane with constant horizontal velocity, with the pose parameters $R, \mathbf{t}$ being responsible for correctly inferring the location of this plane relative to the camera. Photorealistic scenes are rendered in Blender following the Clevr protocol (Johnson et al. 2017), and trajectories are rolled out using Euler integration.

There are two distractor objects on the floor scene, one moving in a circle, and another in a straight line. This environment is used to obtain thorough quantitative results regarding the physical parameter and camera pose estimation abilities of V-SysId. To this end we generate 108 sequences along the following factors of variation: initial height; initial horizontal velocity; restitution coefficient; camera location; moving/static distractor objects. The physical parameters $y_0$, $v_{y_0}$, $v_{x_0}$, $\eta$, and floor height are unknown, and discovered by the optimization process of V-SysId. The sequences are 120 frames long, with a resolution of $320 \times 240$.

**Breathing videos:** To further demonstrate the applicability of V-SysId to real world scenarios, we collected 8 videos of people breathing under different pose, lightning, clothing and distractor settings, with the goal of discovering the relevant region region of the image and using it for breathing rate identification. The true breathing rate was obtained by manual annotation. The videos contain between 150 and 300 frames, at 30 fps and 480x640 resolution. Example videos containing moving distractors can be seen in Fig. 4.3.

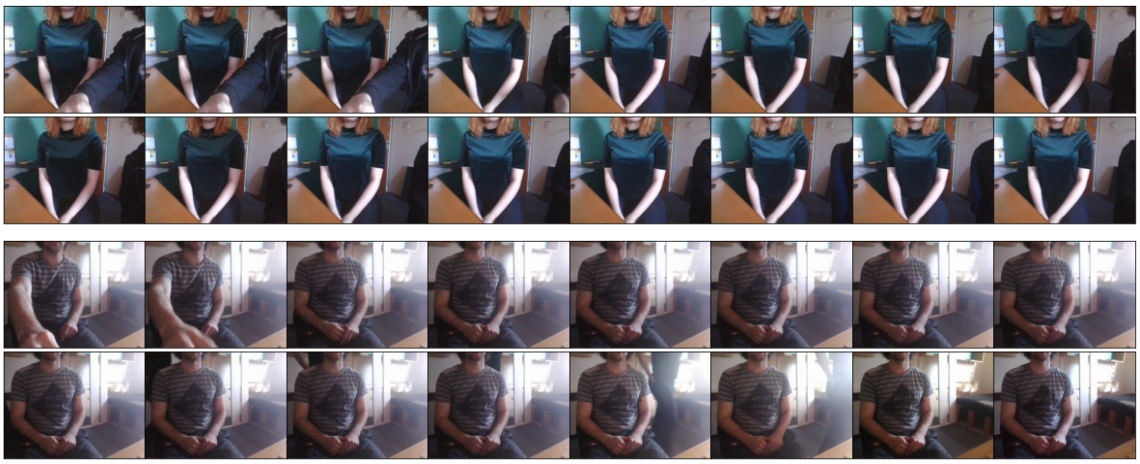


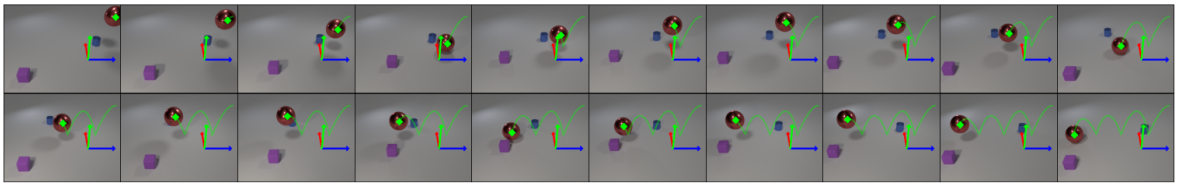
**Figure 4.3:** Frames of breathing scenes containing distractors.

Unlike seminal work in video-based physiology and plethysmography (Boer et al. 2010) V-SysId does *not* require careful hand selection of the regions of interest and is robust to the existence of distractor motions in the scene, which vastly increases its real world applicability. V-SysId simultaneously identifies the region of interest (here, the set of relevant keypoints, rather than a single one) corresponding to sinusoidal motion, and the underlying breathing rate. The ROI discovery process is described in more detail in Sec. 4.4.6.

### 4.4.2 Visualizing keypoint proposal and optimization

We start by visually exploring the results obtained by V-SysId on the spiral robot and bouncing ball datasets. Fig. 4.4 shows the keypoints discovered for two of the scenes. These show that V-SysId correctly identifies the object of interest according to the given equation of motion.

Bouncing ball with unknown velocity, initial height, and restitution coefficient.



Archimedes spiral with unknown radius, radius increase rate, and angular velocity.

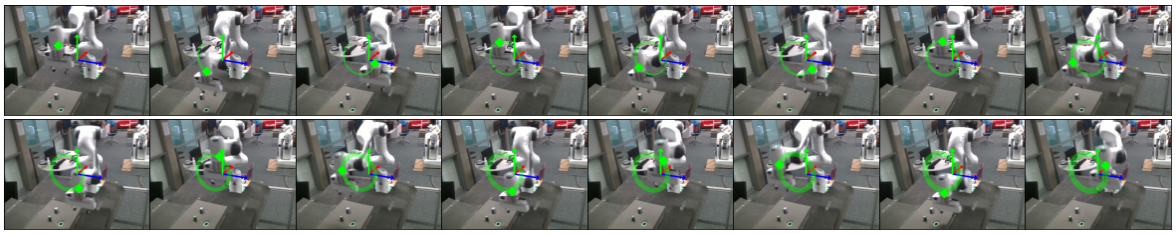


**Figure 4.4:** Discovered object and 3D perspective given the only the family of equations above as weak supervision. **Top:** Example bouncing ball scene. More scenes can be found in Fig. 4.5. **Bottom:** Spiral robot arm end-effector in a real lab setting.

The keypoint proposal and selection process is visualized further in Fig. 4.6. Fig. 4.6 (left) shows the proposed keypoint tracks extracted at the proposal stage, and Fig. 4.6 (right) shows the results obtained by the optimization process on a subset of these, ordered by their selection criterion score (the third number above each plot). The trajectory chosen by V-SysId according to the maximum entropy criterion (Sec 4.3.3) is labeled as "Best". These figures highlight several important points: Firstly, V-SysId is successful despite the large number of distractor keypoints from the various moving parts of the scene (most notable in the robot arm sequence). Secondly and crucially, the optimization process and the maximum entropy criterion are able to fit and identify the best trajectory, correctly discovering the object corresponding to the motion of interest.

In order to further understand the curriculum-based optimization process, we visualize the optimization iterations of two keypoint tracks selected by V-SysId in Fig. 4.7. We can see that upon completion (2nd column), the orientation of the trajectory in 3D space is correctly identified by the model, and that each iteration progressively adjusts both the trajectory's shape (parametrized by the physical parameters) and the camera pose. This leads to a stable optimization procedure where both physical parameters and camera pose are correctly identified.

### 4.4.3 Evaluating parameter estimation

Even though the scale is generally unidentifiable (as discussed in Sec. 4.3.5), in the case of a bouncing ball both the initial height and the restitution coefficient are exactly identifiable. This allows us

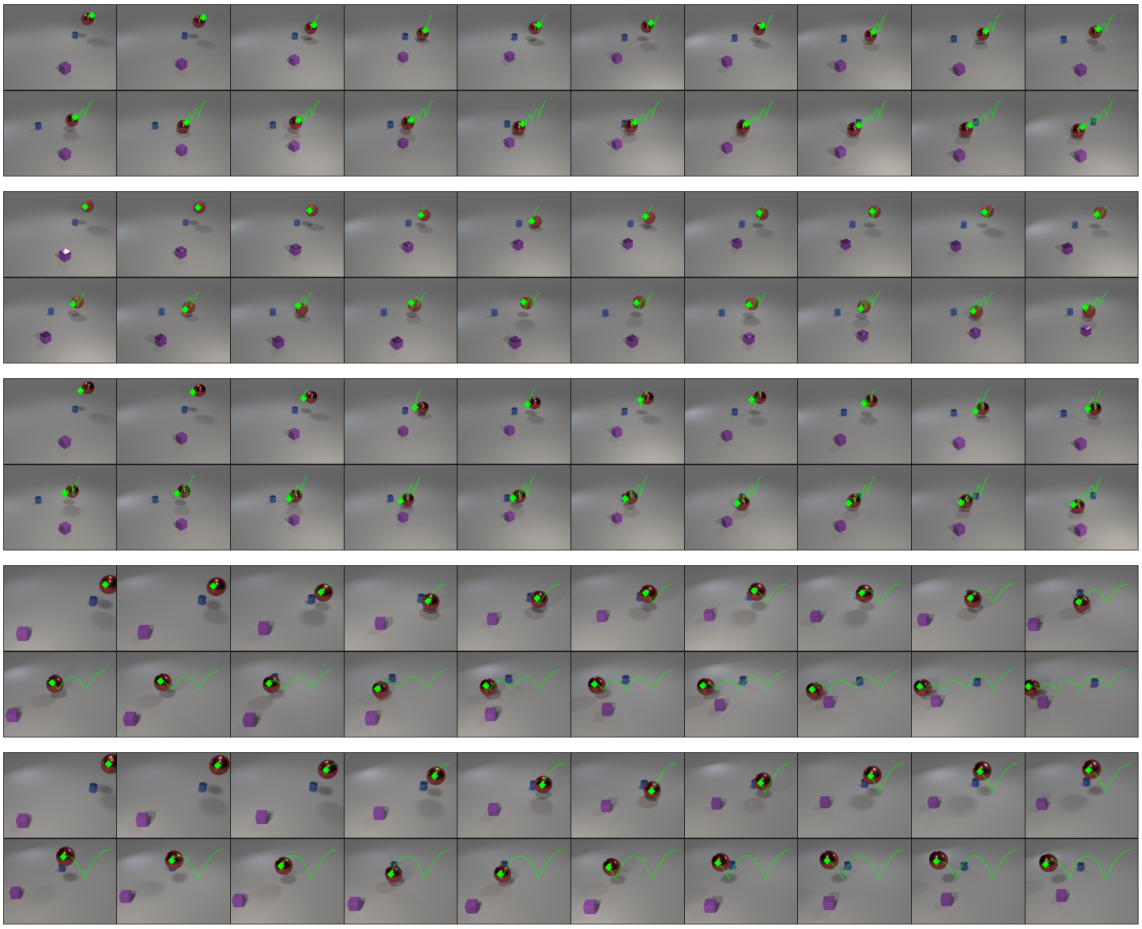


**Figure 4.5:** More visualisations of the discovered object in various bouncing ball scenes.

to compare their learned values to the ground truth values used for the simulations. In addition, we can compare the camera angles identified to those used in simulation in order to evaluate the quality of the extrinsic camera calibration.

The percentage error in restitution coefficient, initial height (distance to floor), and camera angle relative to the simulation ground-truth can be seen in Table 4.1. We can see that all parameters are found with a good degree of accuracy, with physical parameters being slightly more accurate than the camera pose. Notably, the errors are similar with and without moving distractors (within 95% confidence intervals), validating the claim that V-SysId is able to correctly identify the object of interest according to the equations given, even in the presence of distractor objects.

In order to highlight the importance of the curriculum-based optimization strategy, we compare the projection likelihood using our incremental alternate optimization with alternate optimization using the full sequence at every step. Averaging over the bouncing ball scenes, we obtain projection RMSE (pixels) of $-9.31$ and $-109.35$, respectively. This shows that incrementing from a small input segment is key to convergence in the optimization stage of V-SysId.

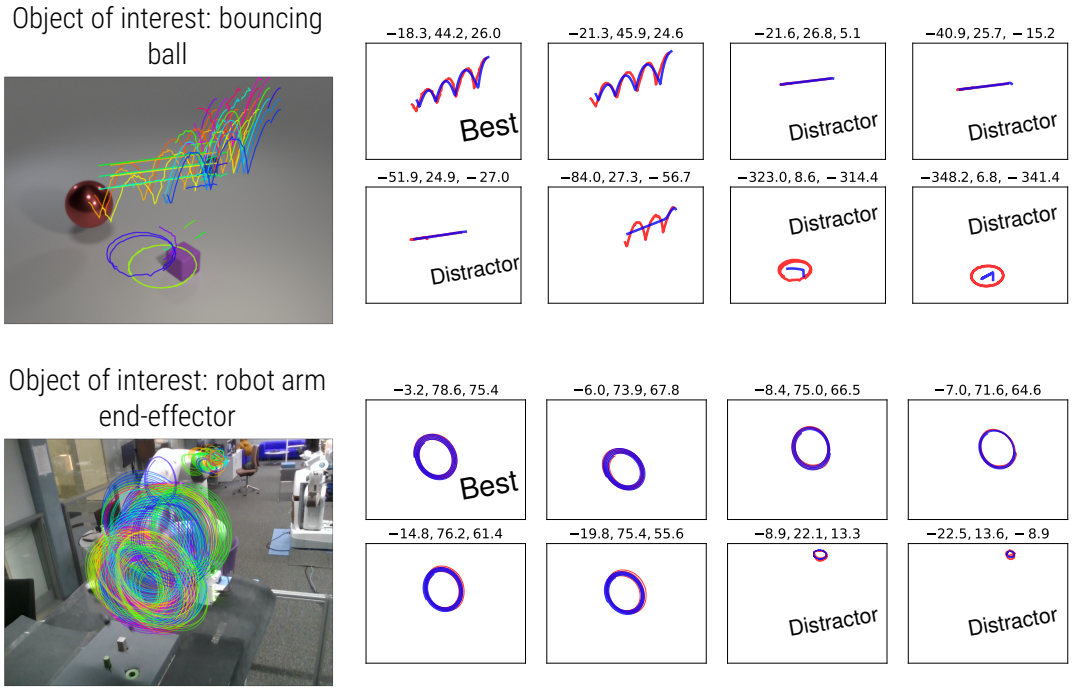



**Figure 4.6: Left:** Keypoint tracks propsed by a grid keypoint detector + KLT tracker (short or static tracks not shown here for improved visualization). **Right:** Subset of the extracted keypoint tracks (red) and projected fitted trajectories (blue), with the corresponding projection loglikelihood, entropy, and their sum, over each plot.

| Distractors | Restitution coefficient (%) | Initial height in 3D (%) | Camera angle (°) |
|---|---|---|---|
| With | $3.8 \pm 1.5$ | $9.7 \pm 4.0$ | $8.0 \pm 1.8$ |
| Without | $2.7 \pm 0.8$ | $6.7 \pm 3.0$ | $9.9 \pm 2.6$ |

**Table 4.1:** Relative error (percentage) between the ground-truth simulation physical parameters and camera pose, and those estimated by V-SysId, for the bouncing ball scene. Error bounds correspond to a 95% confidence interval.

### 4.4.4 Evaluating future trajectory prediction

We now evaluate the ability of V-SysId to perform accurate tracking and prediction given a sequence of correct keypoints. For a video $V$ of length $T$, let the pseudo-ground truth track be $\tilde{\mathbf{k}}^*_{1:T} = \text{V-SysId}(V_{1:T})$. Then, for each $t \in 1..T$, we use the parameters $\boldsymbol{\theta}$, $R$ and $\mathbf{t}$ inferred by V-SysId on the $\tilde{\mathbf{k}}^*_{1:t}$ subset to predict the rest of the track, $\tilde{\mathbf{k}}^{\text{pred}}_{t+1:T}$. We then measure the RMSE between this predicted track and $\tilde{\mathbf{k}}^*_{t+1:T}$. This allows us measure how quickly the trajectory fitting process converges as more keypoints are given. Although here we use subsets of a pre-computed keypoint track, we could also use a keypoint track given by the output of an inference neural network (Sec. 4.4.5).

The results for the bouncing ball and spiral robot are shown in Fig. 4.8. The curves show that the optimization process quickly converges to correct system identification, leading to correct trajectory prediction after only 2 seconds of input.

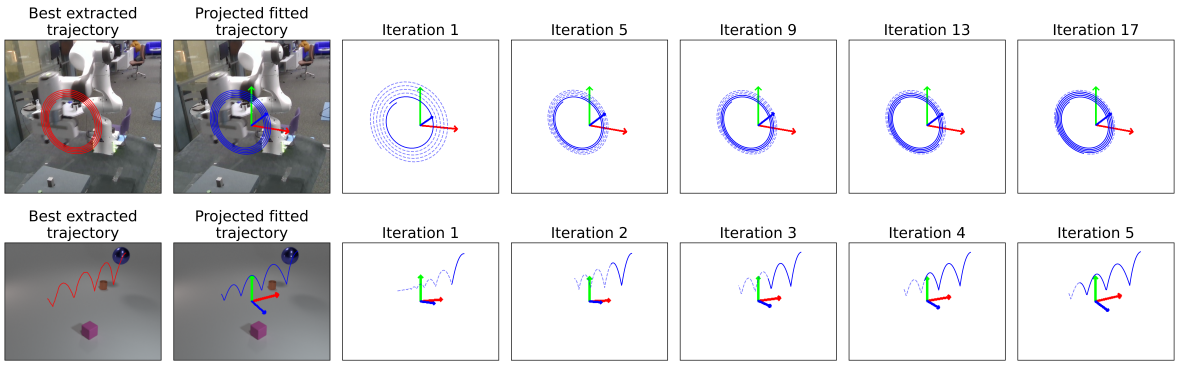



**Figure 4.7:** Visualization of the curriculum-based optimization iterations for the spiral robot (top) and bouncing ball (bottom) scenes. The red line corresponds to the extracted keypoint track and the solid blue line corresponds to the trajectory with parameters estimated so far. The dashed blue line corresponds to the predicted trajectory over the full length of the sequence, under the parameters estimated so far. We can see that the curriculum-based optimization progressively improves the physical parameter and pose estimates.

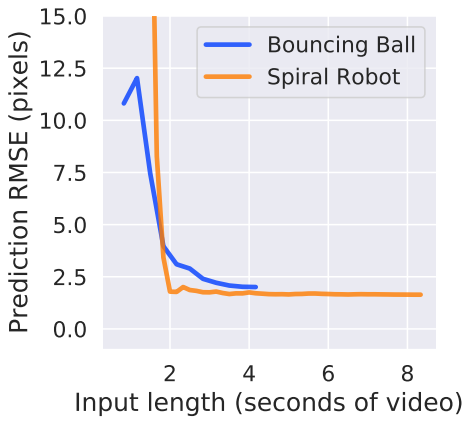



**Figure 4.8:** Future trajectory prediction error under estimated parameters as a function of input length.

### 4.4.5 Tracking by supervised keypoint detection

Once detected, the keypoints discovered by V-SysId can be used as pseudo-ground-truth to train a supervised keypoint detector. For the bouncing ball dataset, the training set consists of 2838 pseudo-labeled frames, and the test set consists of 948 hand-labeled frames from unseen scene configurations. For the robot dataset, the training set consists of 250 pseudo-labeled frames, and the test set consists of 150 hand-labeled frames from unseen end-effector positions.

For the supervised keypoint detector, we use a fully convolutional neural network with 6 ReLU layers with 32 channels, with stride 2 on the 3rd layer, and 2 output channels with 2D softmax activation. These maps are converted to $[x, y]$ coordinates by taking the softmax-weighted mean over the output coordinate grid, as per (Jakab et al. 2018). The input images have a downsampling factor of 4 relative to the original frame resolutions, but we report the keypoint error in the original image space. We train the networks for 20 epochs with batch size 16, and Adam (Kingma et al. 2014a)

(learning rate $3 \times 10^{-4}$).

Results are shown in Table 4.2. The supervised keypoint detector produces highly accurate detections, confirming the quality and usability of the keypoints discovered by V-SysId even on small datasets of high-resolution scenes.

| Environment | RMSE (pixel distance) |
|---|---|
| Simulated bouncing ball ($240 \times 320$) | $8.41 \pm 1.50$ |
| Spiral robot ($480 \times 640$) | $3.89 \pm 0.45$ |

**Table 4.2:** Detection error on the held-out test set of the keypoints extracted by the inference neural network, after training using the keypoints discovered by V-SysId as supervision. Bounds correspond to 95% confidence interval.

### 4.4.6 ROI discovery in chest videos using RANSAC

We have seen how single keypoint discovery can be achieved using V-SysId, but the algorithm can be easily modified to allow discovery of sets of keypoints constituting a region-of-interest. We use the chest video dataset as a prototypical application. The goal is to discover the keypoints in the video corresponding to sinusoidal motion. We start by extracting keypoint tracks as in Stage 1 of V-SysId (filtering out any tracks with a temporal stddev less than 0.7), and transform these 2D tracks into 1D timeseries by taking the projection onto the 1st PCA component of the timeseries (i.e. the 2D direction of highest variance). Each timeseries is standardised, and fit to a sinusoid as per Stage 2 of V-SysId (without the 3D component). In order to identify the best set of tracks, we use a RANSAC inlier count, by measuring the error between a track's fitted sinusoid and all the other extracted tracks, and considering a track an inlier if the MSE is below 0.75. The best track is chosen according to a modified maximum entropy criterion in Stage 3, where the likelihood term is replaced by the inlier count. The ROI is defined as the set of inlier tracks of the best track.

Fig. 4.9 (top) shows the keypoints discovered for the 8 videos, with Fig. 4.9 (bottom) showing the timeseries and its sinusoidal fit for one of the keypoints in the ROI. We can see that the model correctly identifies keypoints corresponding to the chest area, while ignoring distractor and lower-body keypoints. Comparing the respiratory periods identified with V-SysId with the annotated values results in an MSE of 0.016 (in seconds/breath). In contrast, a baseline that uses the mean of the true rates for all videos obtains an MSE of 0.085. These results demonstrate the accuracy of V-SysId for physical parameter estimation from an unknown region of interest, using only the knowledge that the motion of interest is sinusoidal as supervision.

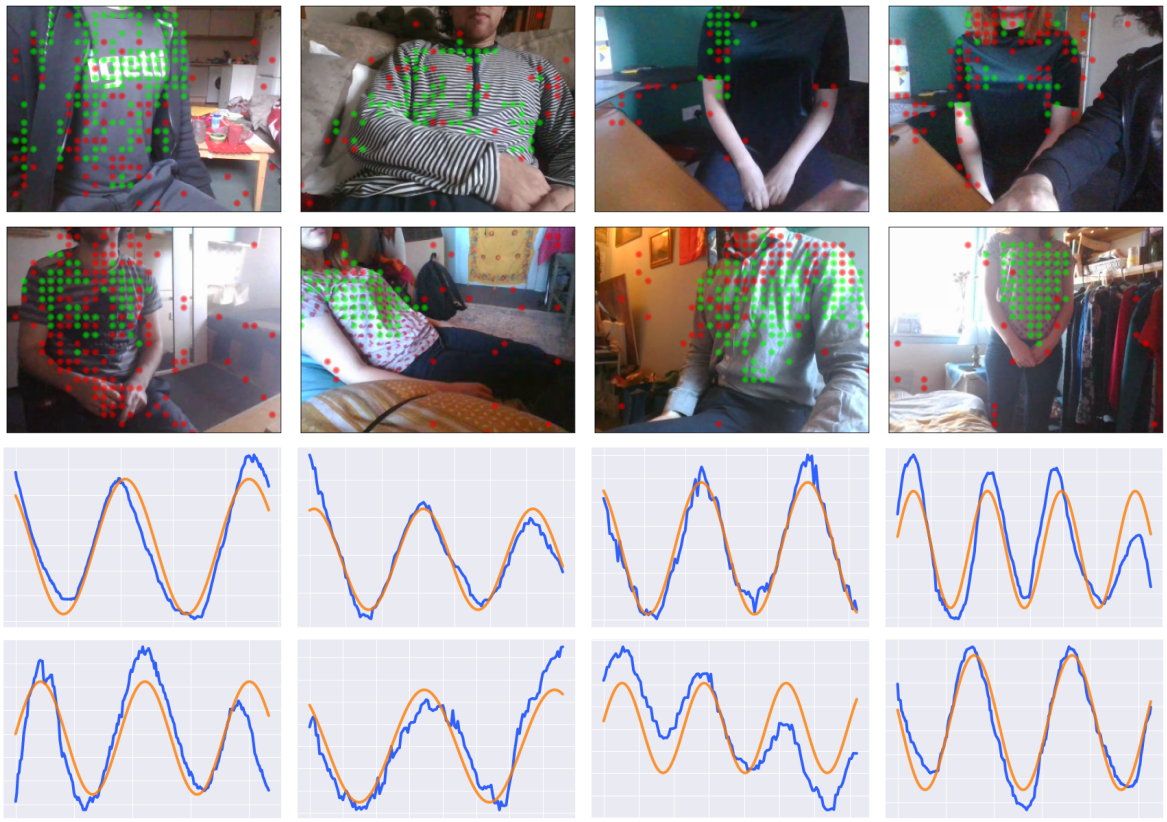


**Figure 4.9: Top:** Green dots correspond to keypoints identified by V-SysId as relevant for determining the breathing rate. The red dots are discarded keypoints. Note that some the videos contain distractors that move in the scene (rollouts of scenes with distractors are shown in Fig. 4.3). V-SysId with RANSAC is able to automatically discover regions of interest. **Bottom:** Timeseries (blue) and sinusoidal fit (orange) of one keypoint in the ROI for each of the scenes (same position in the $2 \times 4$ grid)

## 4.5 Comparison of keypoint detectors

Here we provide a visual comparison of the trajectory proposals obtained using grid, ORB, LF-Net and SuperPoint keypoint extractors, in conjunction with a KLT tracker. Fig. 4.10 shows this comparison for the bouncing ball dataset (after filtering for short and static tracks). It can be seen that despite its simplicity, the grid extractor performs just as well as the more modern keypoint detectors, while running over an order of magnitude faster.

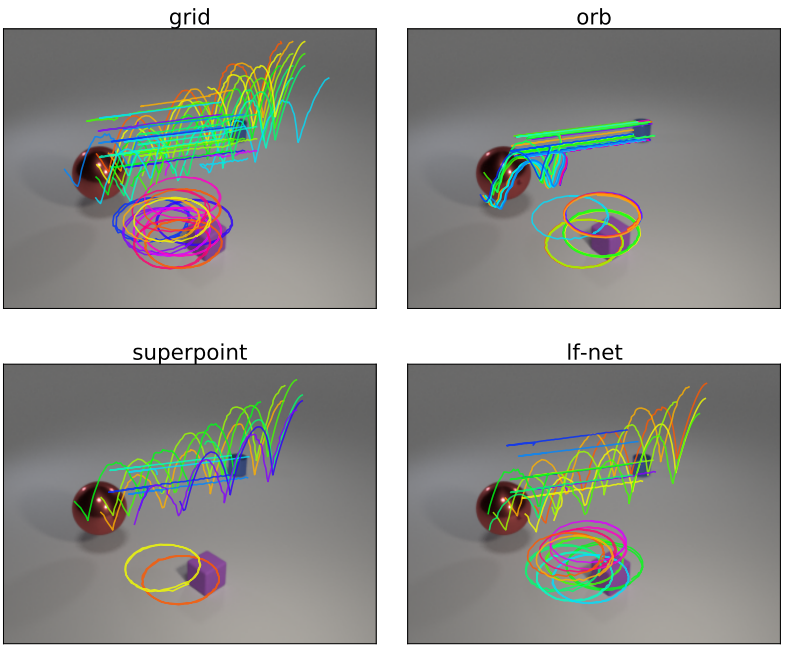



**Figure 4.10:** Comparison of various keypoints extractor and trackers on a bouncing ball scene.

## 4.6 Conclusion and future work

This paper has introduced V-SysId, a 3-stage method for dynamics-constrained keypoint discovery and system identification, which alternates between maximum likelihood extrinsic camera calibration and maximum likelihood physical parameter estimation for motion tracks detected in video. We enhance the stability of this optimization through the inclusion of a curriculum-based optimisation strategy, alongside a maximum entropy selection criterion for keypoint identification. Future avenues of work include extensions to multiple interacting objects, rigid or fluid body dynamics from video, and incorporation with a neural network for material and volume inference from vision.

# Part II

# Physical Inductive Biases for Deep Latent Variable Models

Chapter 5

# NewtonianVAE: Proportional Control and Goal Identification from Pixels via Physical Latent Spaces

This chapter corresponds to the paper:

Miguel Jaques, Michael Burke, and Timothy Hospedales. NewtonianVAE: Propotional Control and Goal Identification from Pixels via Physical Latent Spaces. In *CVPR*, 2021.

Learning low-dimensional latent state space dynamics models has proven powerful for enabling vision-based planning and learning for control. In this chapter introduce a latent dynamics learning framework that is uniquely designed to induce *proportional* controlability in the latent space, thus enabling the use of simple and well-known PID controllers. We show that our learned dynamics model enables proportional control from pixels, dramatically simplifies and accelerates behavioural cloning of vision-based controllers, and provides interpretable goal discovery when applied to imitation learning of switching controllers from demonstration. Notably, such proportional controlability also allows for robust path following from visual demonstrations using Dynamic Movement Primitives in the learned latent space.

## 5.1 Introduction

Vision-based control is highly desirable across numerous industrial applications, both in robotics and process control. At present, much practical vision-based control relies on supervised learning to build bespoke perception modules, prior to downstream dynamics modelling and controller design. This can be expensive and time consuming, and as a result there is growing interest in developing model-based approaches for direct vision-based control.

Model-based approaches for visual control tend to learn latent dynamics models that are subsequently used within suitable planning or model predictive control (MPC) frameworks, or to train policies for later use. We argue that this decoupling of dynamics and control is computationally expensive and often unnecessary. Instead we learn a structured latent dynamical model that directly allows for simple proportional control to be applied. Proportional-Integral-Derivative (PID) feedback control produces commands that are proportional to an error or cost term between cur-

rent system state $\mathbf{x}$ and a (potentially dynamic) target state $\mathbf{x}^{goal}$:

$$\mathbf{u}_t = K_p \left(\mathbf{x}_t^{goal} - \mathbf{x}_t\right) + K_i \sum_{t'} (\mathbf{x}_{t'}^{goal} - \mathbf{x}_{t'}) + K_d \frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\Delta t} \tag{5.1}$$

Gain terms $(K_p, K_i, K_d)$ shape the controller response to errors. PID control is ubiquitous in industry, and broadly applicable across numerous domains, providing a simple and reliable off-the-shelf mechanism for stabilising systems. PID control is also the basis of a wide range of more powerful control strategies, including the more flexible dynamic movement primitives (Schaal 2006; Ijspeert et al. 2013) that augment PD control laws with a forcing function for trajectory following. Essentially we learn the state encoding $\mathbf{x}(I)$ from images $I$ for which robots can be trivially controlled from pixels according to Eq 5.1.

We structure latent dynamics so that that PID control can be applied to move between latent states, to remove the requirement for complex planning or reinforcement learning strategies. Moreover, we show that imitation learning from demonstrations becomes a simple goal inference problem under a proportional control model in this latent space, and can even be extended to sequential tasks comprising multiple sub-goals.

Imitation learning from high dimensional visual data is particularly challenging (Bagnell 2015). Behaviour cloning, which seeks to reproduce demonstrations, is particularly vulnerable to generalisation failures for high dimensional visual inputs, while inverse reinforcement learning (IRL) (Ng et al. 2000) strategies are hard to train and extremely sample inefficient. By learning a structured dynamics model, we allow for more robust control in the presence of noise and simplify the inverse reward inference process. In summary, the primary contributions of this work are:

**Embedding for proportional controllability** We induce a latent space where taking an action in the direction between the current position and some target position, $\mathbf{u} \propto \mathbf{x}^{target} - \mathbf{x}$, moves the system towards the target position. Uniquely, this enables simple proportional control from pixels.

**Imitation learning using latent switching proportional control laws** We leverage the properties of this embedding to frame imitation learning as a goal inference problem under a switching proportional control law model in the structured latent space for sequential goal reaching problems. This enables one-shot interpretable imitation learning of switching controllers from high-dimensional pixel observations.

**Imitation learning using dynamic movement primitives (DMPs)** We also leverage the properties of our embedding to fit dynamic movement primitives in the structured latent space for trajectory tracking problems. This enables one-shot imitation learning of trajectory following controllers from pixels.

Results show that embedding for proportional controllability produces more interpretable latent

spaces, allows for the use of simple and efficient controllers that cannot be applied with less structured latent dynamical models, and enables one-shot learning of control and interpretable goal identification in sequential multi-task imitation learning settings.

## 5.2 Related Work

This paper takes a model-based approach to visual control, using variational autoencoding (VAE) (Kingma et al. 2014b). Latent dynamical systems modelling using autoencoding is widely used (Lesort et al. 2018), and has been proposed for Bayesian filtering (Krishnan et al. 2015; Fraccaro et al. 2017; Karl et al. 2018), and as inverse graphics for improved video prediction and vision-based control (Jaques et al. 2020). Ha et al. (2018) train a latent dynamics model using a variational recurrent neural network (VRNN) in the latent space of a VAE, and then learn a controller that acts in this space using a known reward model. Hafner et al. (2019) extend this approach to allow planning from pixels. Unfortunately, because these approaches decouple dynamics modelling and control, they place an unnecessary computational burden on control, either requiring sampling-based planning or further RL policy optimisation. We argue that this burden can be alleviated by imposing additional structure on the latent space such that proportional control becomes feasible.

In doing so, we build on the control hypothesis advocated by Full et al. (1999), which seeks to model complex phenonoma and systems through simple template models and controllers, using anchor networks to abstract the complexity away from control. This also simplifies the challenges of imitation learning, allowing for sequential task composition (Burridge et al. 1999).

The addition of structural inductive biases into neural models has become increasingly important for generalisation. Injecting knowledge of known physical equations (Guen et al. 2020; Jaques et al. 2020) has been shown to improve dynamics modelling, while the inclusion of structured transition matrices was essential to learn Koopman operators (Abraham et al. 2017) that model dynamical systems with compositional properties (Li et al. 2020a). Here, a block-wise structure with shared blocks was used to learn transition dynamics, which highlighted the importance of added structure in linear state space models, but this was not applied to visual settings. Models like embed to control (E2C) (Watter et al. 2015) or deep variational Bayes filters (DVBF) (Karl et al. 2018) recover structured conditionally linear latent spaces which can be used for control, but, as will be demonstrated later, are still unsuitable for direct proportional control. PVEs (Jonschkowski et al. 2017) learn an explicit positional representation, but do so by minimizing a combination of several heuristic loss functions. Since these models do not use a decoder, it is not possible to visually inspect the learned representations in image space.

NewtonianVAE not only provides latent space interpretability, but also simplifies imitation learning. Inverse reinforcement learning (IRL) strategies for imitation learning typically struggle to learn from high dimensional observation traces as they tend to be based on the principle of feature counting

and observation frequency matching (Ng et al. 2000), as in maximum entropy IRL (Ziebart et al. 2008). Maximum entropy IRL has been extended to use a deep neural network feature extractor (Ziebart et al. 2008), but this is highly vulnerable to overfitting and has extensive data requirements. Recent adversarial IRL approaches (Ho et al. 2016; Fu et al. 2018; Ghasemipour et al. 2019) avoid the challenge of learning a global reward function by training policies directly, but these have yet to be successfully scaled to high dimensional problems. As a result, most imitation learning approaches tend to assume access to low dimensional states, avoiding the challenge of learning from pixels.

Behaviour cloning approaches using dynamic movement primitives (DMP) (Schaal 2006; Ijspeert et al. 2013) have proven particularly powerful for trajectory following control, but are typically applied to low-dimensional proprioceptive states directly as they require proportionally controllable state spaces. Deep DMPs (Pervez et al. 2017) learn visually task parametrised DMPs, but the DMP itself still requires low dimensional state measurements. Chen et al. (2016) propose VAE-DMPs, which impose DMP dynamics in the latent space of a variational auto-encoder, allowing for direct imitation learning. In contrast, this work learns dynamics models independently of tasks, which allows for more flexible downstream applications, including DMP fitting for trajectory following and switching multi-goal imitation learning from pixels (unlike Chen et al. (2016), which use proprioception observations).

Standard imitation learning learning strategies can fail in multi-goal settings or on more complex tasks. In order to address this, many approaches frame the problem of imitation learning from these lower level states as one of skill or options (Sutton et al. 1999; Konidaris et al. 2009) learning using switching state space models. These switching models include linear dynamical attractor systems (Dixon et al. 2004), conditionally linear Gaussian models (Chiappa et al. 2010; Levine et al. 2014), Bayesian non-parametrics (Niekum et al. 2011; Ranchod et al. 2015), and neural variational models (Kipf et al. 2019). Kipf et al. (2019) learn task segmentations to infer compositional policies, but the model uses environment states directly instead of images. Burke et al. (2019a,b) use a switching controller formulation for control law identification from image, proprioceptive state and control action observations. This work applies a similar strategy for goal inference, but, unlike the approaches above, makes use of a learned latent state representation and does not require proprioceptive or low level state information.

Despite this reliance on proprioceptive state information, there is a growing interest in direct visual imitation learning and control. Nair et al. (2018) train a variational autoencoder (VAE) on image observations of an environment, and subsequently sample from this latent space in order to train goal-conditioned policies that can be used to move between different goal states. In contrast, we propose a latent dynamics model that allows for latent proportional controllability and eliminates the need to train a policy to move between goal states.

In addition to the works discussed above, a research area in the unsupervised learning literature of

particular interest is that of learning physically plausible representations (from video) by enforcing temporal evolution according to explicit or implicit physical dynamics (Belbute-Peres et al. 2018; Greydanus et al. 2019; Jaques et al. 2020; Toth et al. 2020). Though promising, these approaches have only been applied to very simple toy environments where dynamics are well known, and are still to be scaled up to real world scenes.

## 5.3 Variational models for visual control

In order to learn a compact latent representation of videos that can be used for planning and control we use the variational autoencoder framework (VAE) (Kingma et al. 2014b; Rezende et al. 2014) and its recurrent formulation (VRNN), (Chung et al. 2015). In this section we briefly present a general formulation of the VRNN, of which many recent models are particular cases or variations (Krishnan et al. 2015; Watter et al. 2015; Fraccaro et al. 2017; Karl et al. 2018; Hafner et al. 2019). For derivation details please refer to (Chung et al. 2015).

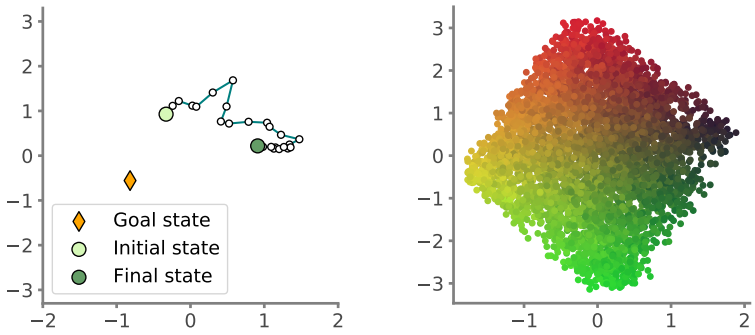



**Figure 5.1:** Trajectory of a point mass actuated using $\mathbf{u}_t \propto (\mathbf{x}^{goal} - \mathbf{x}_t)$ (left) in the latent space learned by an E2C model (right).

Given a sequence of $T$ images, $\mathbf{I}_{1:T}$, and actuations $\mathbf{u}_{1:T} \in \mathbb{R}^{d_u}$ and the corresponding latent representations, $\mathbf{z}_{1:T} \in \mathbb{R}^{d_z}$, the marginal image likelihood is given by:

$$p(\mathbf{I}_{1:T}|\mathbf{u}_{1:T}) = \int p(\mathbf{I}_{1:T}|\mathbf{z}_{1:T}, \mathbf{u}_{1:T}) p(\mathbf{z}_{1:T}|\mathbf{u}_{1:T}) \mathrm{d}\mathbf{z}_{1:T} \tag{5.2}$$

where we factorize the terms above as:

$$p(\mathbf{I}_{1:T}|\mathbf{z}_{1:T}, \mathbf{u}_{1:T}) = \prod p(\mathbf{I}_t|\mathbf{z}_t)$$
$$p(\mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \prod p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_{t-1}),$$

with an approximate positerior given by:

$$q(\mathbf{z}_{1:T}|\mathbf{I}_{1:T}) = \prod q(\mathbf{z}_t|\mathbf{I}_t, \mathbf{z}_{t-1}, \mathbf{u}_{t-1}). \tag{5.3}$$

The model components are trained jointly by maximizing the lower bound on (5.2):

$$\mathcal{L} = \sum_t \mathbb{E}_{q(\mathbf{z}_t|\mathbf{I}_t,\mathbf{z}_{t-1},\mathbf{u}_{t-1})} \big[ p(\mathbf{I}_t|\mathbf{z}_t) + \\ + \mathrm{KL}\left( q(\mathbf{z}_{t+1}|\mathbf{I}_{t+1},\mathbf{z}_t,\mathbf{u}_t) \| p(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{u}_t) \right) \big] , \tag{5.4}$$

via the reparametrization trick, by drawing samples from the posterior distributions, $q(\mathbf{z}_t|\mathbf{I}_t, \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$. Under this framework, the various desired inductive biases are usually built into the structure of the transition prior $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{u}_t)$. In this work we will build on the formulation that uses a linear dynamical system as latent dynamics:

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{u}_t) = A(\mathbf{z}_t) \cdot \mathbf{z}_t + B(\mathbf{z}_t) \cdot \mathbf{u}_t + \mathbf{c}(\mathbf{z}_t) \tag{5.5}$$

which has been studied extensively in the context of deep probabilistic models (Krishnan et al. 2015; Fraccaro et al. 2017; Linderman et al. 2017; Karl et al. 2018; Becker-Ehmck et al. 2019).

## 5.4 Newtonian Variational Autoencoder

**Motivation** To motivate our model, we begin by examining the properties of an existing latent variable model used for control. We train an E2C model (Watter et al. 2015), since it applies a locally linear latent transition as in (5.5) and is highly representative of properties obtained in these types of model. We use a simple point mass system that can move in the $[x, y]$ plane and train the model on random transitions in image space (more details in the experiments section). Since the environment is 2D with 2D controls, we use a 4D latent space (2 dimensions for position and 2 for velocity). Our goal is to explore how the E2C model behaves when a basic proportional control law $\mathbf{u}_t \propto (\mathbf{x}^{goal} - \mathbf{x}_t)$ is applied, where $\mathbf{x}$ is the latent system configuration.

An immediate problem is that even though the latent coordinates corresponding to position are correctly learned (Fig. 5.1(right)), it is necessary to plot *every coordinate pair* and their correlation with ground truth positions in order to visually determine which 2 coordinates correspond to the position $\mathbf{x}$. Having determined such $\mathbf{x}$, we can use a random target position $\mathbf{x}^{goal}$ and see if successively applying an action $\mathbf{u}_t \propto (\mathbf{x}^{goal} - \mathbf{x}_t)$ will guide the system towards $\mathbf{x}^{goal}$ (which we term *proportional controllability*). Note that PID control is trivially achievable given a P-controllable system, so we focus on P-control for simplicity of exposition, without loss of generality. Fig. 5.1(left) shows that this simple control law fails to guide the system towards the goal state, even though the latent space is seemingly well structured. These problems are present in existing variational models for controllable systems, including E2C (Watter et al. 2015), DVBF (Karl et al. 2018) and the Kalman VAE (Fraccaro et al. 2017).

To avoid the need for ground truth data and visual inspection, we construct a model that explicitly

treats position and velocity as separate latent variables $\mathbf{x}$ and $\mathbf{v}$. To ensure correct behaviour under a proportional control law[1] the change in position and velocity should be directly related to the force applied. I.e. given an external action $\mathbf{u}$ representing the force (=acceleration) acting on a system, $\mathbf{x}$ and $\mathbf{v}$ should follow Newton's second law, $d^2\mathbf{x}/dt^2 = \mathbf{F}/m$. Although this might seem like a trivial statement from a physical standpoint, this type of behaviour is not built into existing neural models, where the relationship between action and latent states can be arbitrary. This arbitrary relationship in turn complicates control, and it becomes necessary to learn downstream controllers or policies to compensate for these dynamics while meeting a control objective.

We make one additional observation: in many cases the external action $\mathbf{u}$ is applied along disentangled dimensions of the system. For example, for a 2-arm robot, actions correspond to torques on the angles of each arm relative to its origin[2]. These action dimensions correspond to the polar coordinates $[\theta_1, \theta_2]$, which are the ideal disentangled coordinates to describe such a robot. We use this fact to formulate a model that not only provides an interpretable and P-controllable latent space, but also the correct disentanglement by construction.

**Formulation** We now formulate a model satisfying the above desiderata. For an actuated rigid body systems with $D$ degrees of freedom, we model the system configuration (positions or angles) by a set of coordinates $\mathbf{x} \in \mathbb{R}^D$ with double integrator dynamics, inspired by Newton's equations of motion:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}\,, \quad \frac{d\mathbf{v}}{dt} = A(\mathbf{x}, \mathbf{v}, \mathbf{u}) \cdot \mathbf{x} + B(\mathbf{x}, \mathbf{v}, \mathbf{u}) \cdot \mathbf{v} + C(\mathbf{x}, \mathbf{v}, \mathbf{u}) \cdot \mathbf{u} \tag{5.6}$$

To build a discrete form of (5.6) into a VAE formulation, we use the instantaneous system configuration (or position) $\mathbf{x}$ as the stochastic variable that is inferred by the approximate posterior, $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{I}_t)$, with velocity a deterministic variable that is simply the finite difference of positions, $\mathbf{v}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$. The generative model is now given by

$$p(\mathbf{I}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod p(\mathbf{I}_t|\mathbf{x}_t) \tag{5.7}$$

$$p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) = \prod p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}; \mathbf{v}_t) \tag{5.8}$$

where the transition prior is:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}; \mathbf{v}_t) = \mathcal{N}(\mathbf{x}_t|\mathbf{x}_{t-1} + \Delta t \cdot \mathbf{v}_t,\ \sigma^2) \tag{5.9}$$

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \Delta t \cdot (A\mathbf{x}_{t-1} + B\mathbf{v}_{t-1} + C\mathbf{u}_{t-1}) \tag{5.10}$$

with $[A, \log(-B), \log C] = \text{diag}(f(\mathbf{x}_t, \mathbf{v}_t, \mathbf{u}_t))$, where $f$ is a neural network with linear output activation. Using diagonal transition matrices encourages correct coordinate relations between

[1] For further analysis of the convergence and stability of PID controllers, see (Duc et al. 2006; Dorf et al. 2011).
[2] The example also applies more generally to any robot actuated with torques along its joints.

$\mathbf{u}$, $\mathbf{x}$ and $\mathbf{v}$, since linear combinations of dimensions are eliminated. In order to obtain the correct directional relation between $\mathbf{u}$ and $\mathbf{x}$, required for interpretable controllability, we set $C$ to be strictly positive (in addition to diagonal). $B$ is strictly negative to provide a correct interpretation of the term in $\mathbf{v}$ as friction, which aids trajectory stability. During inference, $\mathbf{v}_t$ is computed as $\mathbf{v}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$, with $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{I}_t)$ and $\mathbf{x}_{t-1} \sim q(\mathbf{x}_{t-1}|\mathbf{I}_{t-1})$. This inference model provides a principled way to infer velocities from consecutive positions, similarly to (Jonschkowski et al. 2017). We use Gaussian $p(\mathbf{I}_t|\mathbf{x}_t)$ and $q(\mathbf{x}_t|\mathbf{I}_t)$ parametrized by a neural network throughout.

We train all model components using the following ELBO (full derivation in Appendix C):

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{I}_t)q(\mathbf{x}_{t-1}|\mathbf{I}_{t-1})}[\mathbb{E}_{p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t;\mathbf{v}_t)}p(\mathbf{I}_{t+1}|\mathbf{x}_{t+1}) + \\ + \mathrm{KL}\left(q(\mathbf{x}_{t+1}|\mathbf{I}_{t+1})\|p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t;\mathbf{v}_t)\right)]\end{aligned} \tag{5.11}$$

A crucial component of this ELBO is performing future- rather than current-step reconstruction through the generative process (first term above). This is known to encourage the use of the transition prior when learning the latent representation (Watter et al. 2015; Karl et al. 2018; Hafner et al. 2019).

**Further considerations** Another key difference between a simple LDS and our Newtonian model is the fact that we consider velocity to be a deterministic latent variable that is uniquely determined by the stochastic positions. In contrast, independent inference through $\mathbf{z}$ means that position and velocity might not have the direct relation that is present in the physical world (velocity as the derivative of position). Both of these contribute to a lack of physical plausability, in the Newtonian sense, in existing models. Though technically our transition prior is a special case of the LDS (5.5), these added structural constraints are crucial in order to induce a Newtonian latent space that directly allows for PID control of latent image states.

## 5.5 Efficient Imitiation with P-Control

A key benefit of the Newtonian latent space is that it dramatically simplifies image-based imitation learning. Given a visual demonstration sequence $D_\mathbf{I} = \{(\mathbf{I}_1, \mathbf{u}_1), ..., (\mathbf{I}_T, \mathbf{u}_T)\}$, we encode the frames using the inference network $q(\mathbf{x}|\mathbf{I})$ described above in order to produce demonstrations in latent space, $D_\mathbf{x} = \{(\mathbf{x}_1, \mathbf{u}_1), ..., (\mathbf{x}_T, \mathbf{u}_T)\}$.

### 5.5.1 Learning Vision-Driven Switching P-Control

We can fit a switching P-controller[3] to a set of demonstration sequences in latent space using a Mixture Density Network (MDN), where the action likelihood given a state is a mixture of $N$ propor-

[3] We use a P-controller instead of a PID-controller for simplicity of exposition and without loss of generality.

tional controllers:

$$P(\mathbf{u}_t|\mathbf{x}_t) = \sum_{n=1}^{N} \pi_n(\mathbf{x}_t)\,\mathcal{N}\left(\mathbf{u}_t|K_n(\mathbf{x}_n^{goal} - \mathbf{x}_t),\, \sigma_n^2\right) \tag{5.12}$$

where $K_n$, $\mathbf{x}_n^{goal}$ and $\sigma_n^2$, $\forall n \in 1..N$, are learnable parameters, and $\boldsymbol{\pi}(\mathbf{z})$ is a parametric function like a neural network. Intuitively, fitting this MDN to the latent demonstrations splits the demonstrations into regions where a specific proportional controller would correctly fit that part of the trajectory. If the latent space is P-controllable (such as the one produced by the NewtonianVAE), the vectors $\mathbf{x}_n^{goal}$ will correspond to the intermediate goals or bottleneck states in the demonstration sequence. As an added benefit, we can pass the learned goals through NewtonianVAE's decoder in order to obtain their visual representation, providing an interpretable control policy.

**Learning a finite-state machine** Having identified the latent vectors corresponding to the goals, we determine the order in which they must be reached by analysing their visits during the demonstrations, directly extracting initiation sets and termination conditions. This produces a simple finite-state machine (FSM) that determines goal state transitions. The FSM and extracted P-controllers can then be used to reproduce demonstrated behaviours by driving the robot to each goal in succession, but could also be used within an options framework (Sutton et al. 1999) for reinforcement learning.

## 5.5.2 Learning Visual Path Following with DMPs

It is clear that the latent space of a NewtonianVAE can be used for switching goal-based imitation learning, but proportionality is also a precursor for trajectory following using DMPs. A DMP (Ijspeert et al. 2013) is a proportional-derivative controller with a learned forcing function

$$\tau\ddot{\mathbf{x}} = \alpha\left(\beta\left(\mathbf{x}^{\text{goal}} - \mathbf{x}\right) - \dot{\mathbf{x}}\right) + \mathbf{f}. \tag{5.13}$$

Here, $\tau$ is a time scaling constant, and $\alpha, \beta$ are proportional control gain terms. The forcing function

$$\mathbf{f}_t = \sum_{i=1}^{N} \frac{\Phi(t)\mathbf{w}_i}{\sum_{i=1}^{N}\Phi(t)}(\mathbf{x} - \mathbf{x}^{\text{goal}}) \tag{5.14}$$

captures trajectory dynamics, using a weighted linear combination of radial basis functions, $\Phi(t) = \exp(-\frac{1}{\sigma_i^2}(y - c_i)^2)$, with centres $c_i$ and variances $\sigma_i^2$.

The canonical system $\dot{y} = -\alpha_y y$ gently decays over time, smoothly modulating the forcing function until reaching an end goal, $\mathbf{x}^{\text{goal}}$. Basis functions and parameters are fit to demonstration trajectories using weighted linear regression. Since the NewtonianVAE embeds for proportionality, DMPs can be fit directly to the latent space from demonstration data, allowing for vision-based trajectory control and path following.

## 5.6 Experiments

**Environments** We validate our model on 3 simulated continuous control environments, to allow for better evaluation and ablations, and on data collected from a real PR2 robot.

- **Point mass:** A simple point mass system adapted from the `PointMass` environment from `dm_control`. The mass is linearly actuated in the 2D plane and its movement bounded by the edges of the frame.
- **Reacher-2D:** A 2D reacher robot adapted from the `Reacher` environment in `dm_control` and inspired by (Kipf et al. 2019). We alter the environment so that the robot's middle joint can only bend in one direction, in order to prevent the existence of two possible arm configurations for every end effector position. We also limit the origin joint angle range to $[-160, 160]$ so that the system configuration can be described in polar coordinates by two variables corresponding to the angle of each arm, avoiding a discontinuity in case of full circular motion.
- **Fetch-3D:** The 3D reacher environment `FetchReachEnv` from OpenAI Gym. We use this to show that our model learns the desirable representations even in visually rich 3D environments of multi-joint robots with partial occlusions.

To train the models, we generate 1000 random sequences with 100 time-steps for the point mass and reacher-2D systems, and 30 time-steps for the fetch-3D system. More implementation details for each of the environments can be found in Appendix D.

**Baseline models** We compare our model to E2C[4] and a static VAE (each frame encoded individually). Additionally, in order to better understand the effect of diagonality and positivity of the transition matrices in (5.10), we test Full-NewtonianVAE, where the matrices $A$,$B$,$C$ are unbounded and full rank.

**Training details** All models used the encoder and decoder from Ha et al. (2018), except for the point mass environment, where we use a spatial broadcast decoder (Watters et al. 2019b). All temporal models were trained using 2-step ahead prediction in the ELBO (instead of single step), which is a straightforward extension of (5.11), as done in latent overshooting (Hafner et al. 2019). All experiments use $64 \times 64$ RGB frames as input to the encoder.

To compute the transition matrices as a function of the state we use a fully connected network with 2 hidden layers with 16 units and ReLU activation, with the appropriate input and output dimensionality. In the NewtonianVAE variants, $\Delta t$ was set to the known environment time step. All models were trained using Adam (Kingma et al. 2014a) with a learning rate of $3 \cdot 10^{-4}$ and batch size 1 (a single sequence per batch) for 300 epochs. In the point mass experiments we found it

[4] DBVF (Karl et al. 2018) and E2C learn similar latent spaces, as both rely on an unstructured conditionally linear dynamical system.

useful to anneal the KL term in the ELBO, starting with a value of 0.001 and increasing it linearly to 1.0 between epochs 30 and 60.

### 5.6.1 Visualizing latent spaces and P-controllability

In this section we compare the latent space and P-controllability properties of the NewtonianVAE and baseline models on the simulated enviroments: point mass, reacher-2D and fetch-3D.

**Comparing latent spaces** We start by visualizing the latent spaces learned by each models on all the environments. Fig. 5.2 shows that only the NewtonianVAE is able to learn a representation corresponding to the natural disentangled coordinates in both environments (e.g. $[x, y]$ in the point mass and $[\theta_1, \theta_2]$ in the reacher-2D), and that these are correctly correlated with ground-truth values, coded in the red-green spectrum. This shows that the structure imposed on the transition matrices in (5.10) is key to learning correct latent spaces in both Cartesian and polar coordinates.

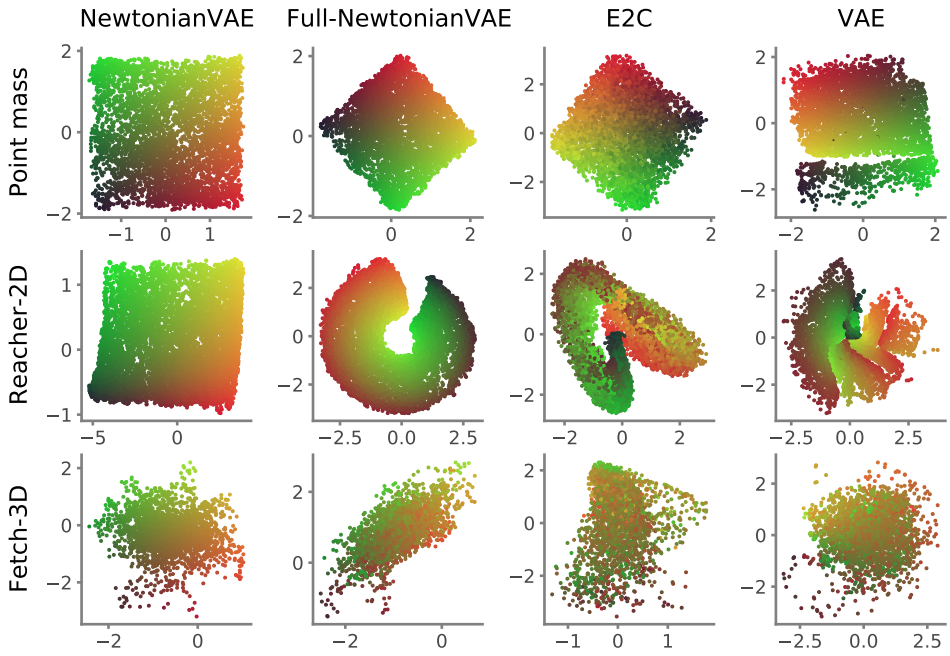



**Figure 5.2:** Latent spaces of various models in the point mass, reacher-2D and fetch-3D environments. Each dot corresponds to the latent representation of a test frame, and the red-to-green color coding encodes the true 2D position/angle values. For E2C (Watter et al. 2015), we plot the two latent dimensions that best correlated with the true positions. Since the configuration space of the fetch-3D env is 4D, we visualize only the first two coordinates. Only for our NewtonianVAE does latent space (position) and true space (color) correlate perfectly.

**P-controllability** Even though the models above produce different latent spaces, most are well structured and show a clear correlation with the ground truth state (color coded). Although their structure is visually appealing, we are primarily interested in verifying is whether they satisfy P-controllability. To do this, we sample random starting and goal states, and successively apply the

control law $\mathbf{u}_t \propto (\mathbf{x}(\mathbf{I}^{goal}) - \mathbf{x}_t(\mathbf{I}_t))$. A space is deemed P-controllable if the system moves to $\mathbf{x}(\mathbf{I}^{goal})$ in the limit of many time-steps. For reference, we also apply model-predictive control to E2C.

Convergence curves in the true state space are shown in Fig. 5.3, and example rollouts in the learned latent space are shows in Fig. 5.4 (more examples in Appendix E). We can see that only NewtonianVAE produces P-controllable latent states, as all the remaining models diverge under a P-controller. This highlights the fact that even though the latent spaces learned by the Full-NetwtonianVAE and E2C are seemingly well structured for the point mass system, they fail to provide P-controllability. While these systems can still be stabilised using more complex control schemes such as MPC, this is entirely unnecessary with a P-controllable latent space, where trivial control laws can be applied directly.

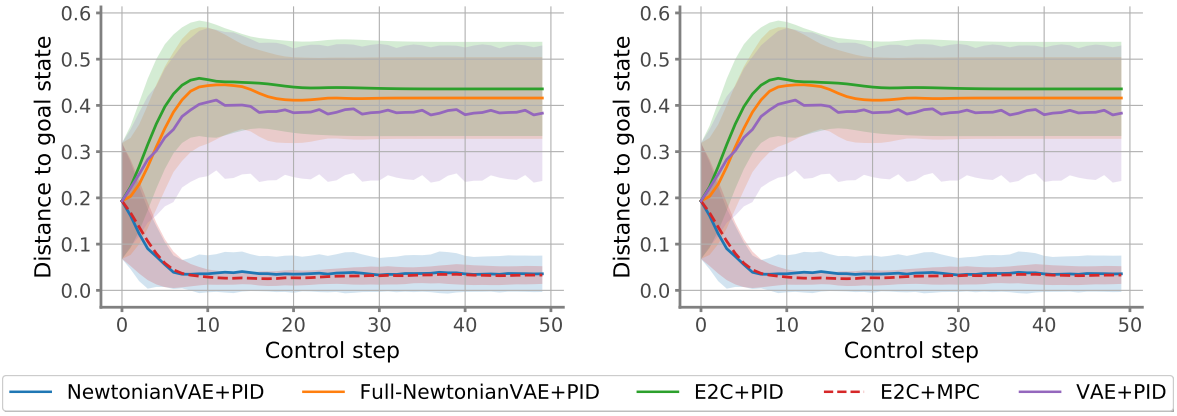



**Figure 5.3:** Convergence rates of PID control using various latent embeddings for the point mass (left) and reacher-2D (right) systems, over 50 episodes. We use gain parameters $K_p = 8$, $K_i = 2$, $K_d = 0.5$. For contrast, we show Model Predictive Control (MPC, using CEM planning as per (Hafner et al. 2019)).

## 5.6.2 MDN goal and boundary visualization

Having trained a NewtonianVAE on a dataset of random transitions we can use the learned representations to fit the mixture of P-controllers in (5.12) to the few-shot demonstration sequences.

**Reacher-2D** In this environment there are three colored balls in the scene and the task is reaching the three balls in succession, where the arm's starting location varies across demonstration sequences. We used the true reacher model with a custom controller to generate demonstration images. A full demonstration sequence is shown in Appendix B. For this experiment we use a linear $\boldsymbol{\pi}(\mathbf{x})$, though a MLP yields similar results.

After fitting (5.12) on *a single* demonstration sequence, we visualize the goals $\mathbf{x}^{goal}$ and the decision boundaries of the switching network $\boldsymbol{\pi}(\mathbf{x})$ in Fig. 5.5 (left). The figure shows that goal states are correctly identified (diamond markers), and that the three sub-task regimes are correctly seg-

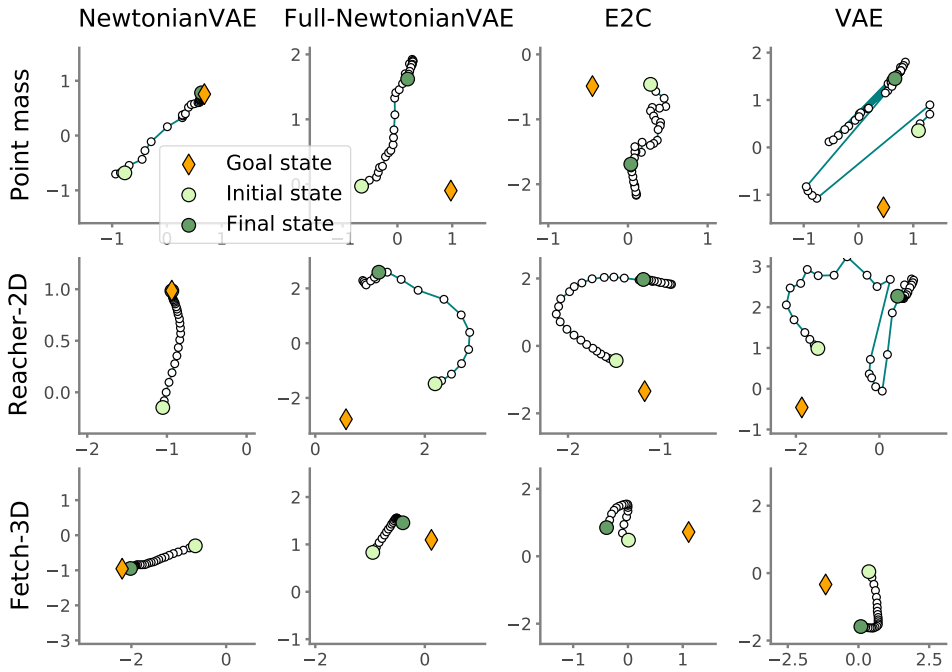



**Figure 5.4:** P-control trajectories for point mass, reacher-2D and fetch-3D environments. Plots are in the latent space of Fig. 2. We can see that only NewtonianVAE produces a latent space where a P-controller correctly leads the systems from the initial to goal state.

mented. Decoding $\mathbf{x}^{goal}$, confirms that the goals are correctly represented in image space, adding a layer of interpretability to an upstream control policy.

**Imitation learning performance** We now compare various imitation learning methods in the simulated task described above. A reward of 1.0 is given when the system reaches a neighborhood of each target (as measured in the true system state), but the targets must be reached in sequence. Our method (switching P-controller) uses a finite-state machine inferred from the MDN trained on latent demonstrations (Fig. 5.5(left)). We compare it to behaviour cloning with an LSTM with 50 recurrent units, in the NewtonianVAE's latent space, and GAIL (Ho et al. 2016), a state-of-the-art IRL method trained on ground truth proprioceptive states. Table 5.1 shows the imitation efficiency for increasing numbers of demonstration sequences, with example rollouts shown in Fig. 5.5 (right). The results show that goal-driven P-control in a hybrid control policy is significantly more data efficient and robust to noise than a standard behaviour cloning policy. Additionally, switching controllers dramatically outperform GAIL[5], even though this was trained on 5 times the number of environment interactions used by the NewtonianVAE.

**Real multi-object reacher** We now apply our model to real robot data. Here, we record a 7-DoF PR2 robot arm that moves between 6 objects in succession in a hexagon pattern. A full sequence comprises approximately 100 frames. We use 636 frames to train the NewtonianVAE and an ad-

[5]Maximum Entropy IRL performed equally poorly, failing to reach a single goal. This is unsurprising, due to the connections between this and adversarial imitation learning (Finn et al. 2016a).

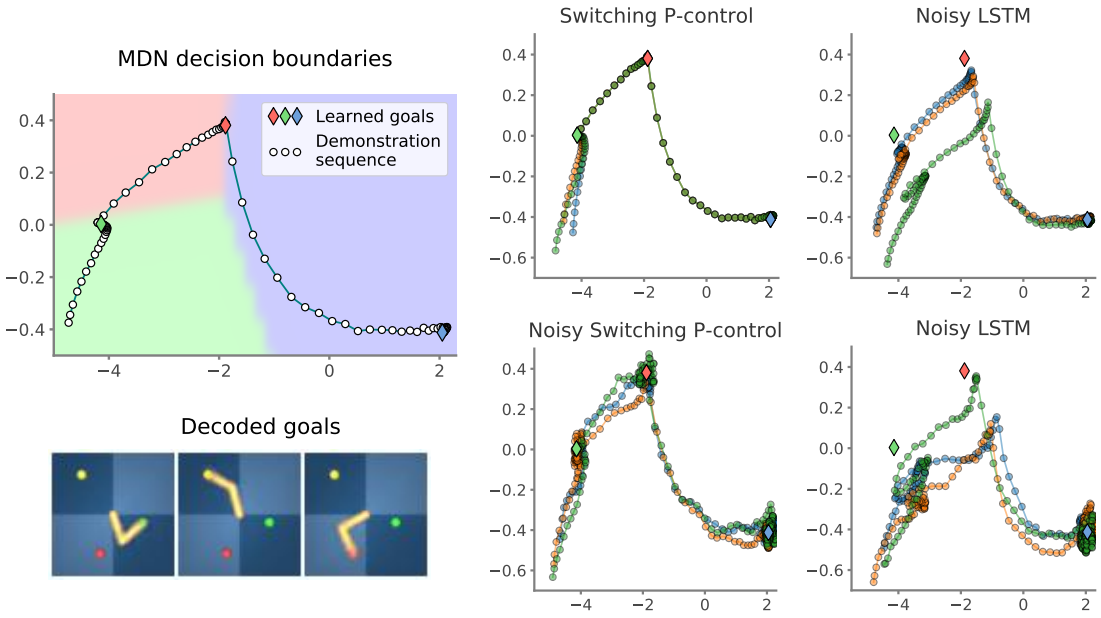



**Figure 5.5: Left:** Demonstration sequence and learned mixture of P-controllers (MDN). Each background color and corresponding diamond correspond to a component $\pi_n(\mathbf{x})$ and $\mathbf{x}_n^{goal}$, $\forall n \in \{1, 2, 3\}$, respectively. **Right:** Rollouts after imitation learning using switching P-controllers and LSTM policy, with a single demonstration sequence. In the noisy regime each action has an added noise $\mathcal{N}(0, 0.25^2)$. All plots are in the NewtonianVAE's latent space.

| Demonstration sequences | Switching P-controller | | LSTM | | GAIL from proprioception |
|---|---|---|---|---|---|
| | Clean | Noisy | Clean | Noisy | |
| 1 | $3.0 \pm 0.0$ | $2.17 \pm 0.32$ | $0.81 \pm 0.35$ | $0.27 \pm 0.20$ | – |
| 10 | $3.0 \pm 0.0$ | $2.01 \pm 0.34$ | $3.00 \pm 0.00$ | $1.42 \pm 0.34$ | – |
| 100 | $3.0 \pm 0.0$ | $2.06 \pm 0.30$ | $3.00 \pm 0.00$ | $1.23 \pm 0.30$ | 0.62 |

**Table 5.1:** Efficiency of imitation learning methods for vision-based sequential multi-task control. Metric: Environment Reward (max = 3.0). The NewtonianVAE is used to encode the frames. 'Noisy': Added action noise $\mathcal{N}(0, 0.25^2)$ during the rollouts. Error ranges: 95% confidence interval across 100 rollouts. GAIL is trained for 5000 episodes.

ditional 100 held-out frames to train the MDN. Further model and dataset details can be found in Appendix D.

Fig. 5.6 shows the image representations of the learned goals (left) and the mode $\boldsymbol{\pi}(\mathbf{x})$ that is active for each frame in the demonstration sequence (right). We can see that the six goals are correctly identified by the MDN, and that segmentations are correct in the sense that a frame is assigned to the learned goal to which the robot is moving at that time step. Note that the model is able to recover correct goals and segmentations even though not all of the joints are visible in every frame.

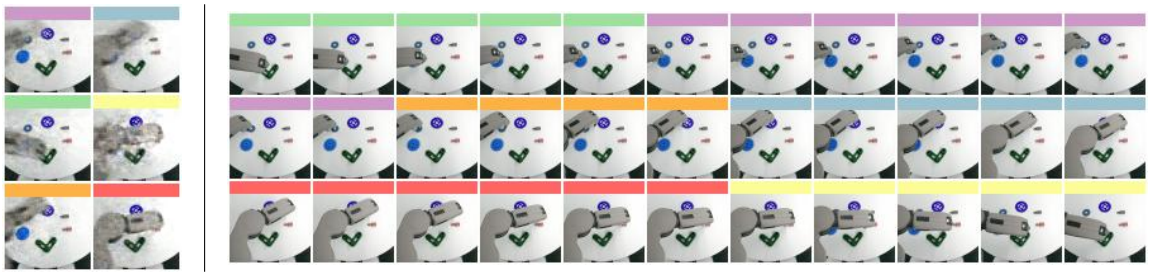


**Figure 5.6:** Decoded goals (left) and sequence segmentation (right) learned for a 6-goal visual trajectory of a PR2 robot. The sequence shows 33 equally spaced frames of a 100-frame demonstration.

### 5.6.3 Fitting DMPs for path following in latent space

We show how the NewtonianVAE can be used to enable a robot to learn a vision-driven controller to follow a demonstration trajectory, using the fetch-3D environment. To this end, we draw a 'G'-shaped trajectory in the first 2 dimensions of the latent space and fit a DMP. The DMP runs in 100 time-steps, spanning 4 seconds of execution, where we feed the acceleration output by the DMP as the action to the environment, and the new state and velocity is inferred by the NewtonianVAE.

Fig. 5.7 shows that the robot correctly follows the demonstration trajectory, showing that the latent space induced by the NewtonianVAE enables path following using a DMP just by virtue of its P-controlability property, without needing to be explicitly trained to perform well under a DMP, as done by (Chen et al. 2016).

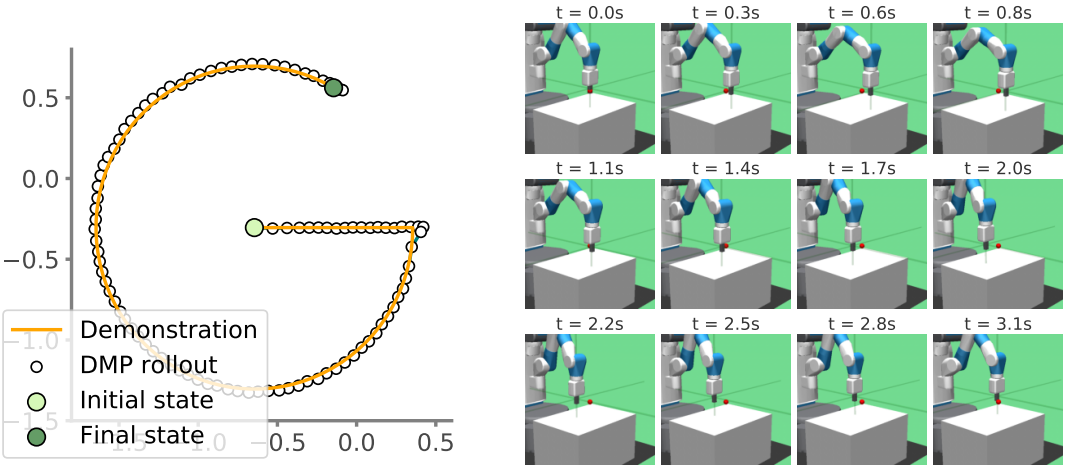



**Figure 5.7: Left:** Overhead view of demonstration and trajectory produced by the DMP in the fetch-3D environment. The first 2 dimensions of the NewtonianVAE's latent space are shown. **Right:** Frames seen by the NewtonianVAE during this rollout.

## 5.7 Limitations and Future Work

This work assumes that underlying systems are proportional controllable, and follow Newtonian dynamics. Moreover, it should be noted that vision-based torque control of high dimensional robot manipulators requires high speed vision. However, in our opinion, the most notable limitation is the fact that the imitation learning model only learns a fixed set of goals. Ideally, the agent would learn a semantic goal, which would represent a command "fetch the yellow ball", for a variable position of the yellow ball and not a fixed state. However, this would require demonstration data with substantially more variety than considered here. We have also avoided multi-modal demonstrations for simplicity, though we believe it would be of interest to integrate our method with approaches like InfoGAIL (Li et al. 2017).

## 5.8 Conclusion

We introduced NewtonianVAE, a structured latent dynamics model designed to allow P-controllability from pixels. Results show that this structured latent space allows for trivial, robust control in the presence of noise and dramatically simplifies and improves imitation learning, which can be framed either as a switching goal-inference or as a path following problem in the latent space. Additionally, our model provides visually interpretable goal discovery and task segmentation under both simulated and real environments, without any labelled or proprioception data.

Chapter 6

# Discussion

## 6.1 Impact

The work developed during my PhD, presented in this thesis, has contributed to the wider literature by showing the value (and the feasibility) of integrating physics priors into visual models. A range of frameworks to do so have been developed, and we have shown that inductive biases for physics a) allow for system identification in support of predictive modelling and control from pixels; b) provide a supervisory signal that aids in unsupervised object discovery or keypoint selection; c) improve extrapolation in long term, out of domain vision-based prediction tasks; and d) can be used to simplify the design of downstream control and imitation learning algorithms. The work focused particularly on the ways visual inputs are transformed into latent states such that these can be used, learned, or modified to enable integration with physics engines, and their respective optimisers. We now summarise the key contributions of each model.

### PAIG

The main contribution of the PAIG model was the observation that in order to integrate physics engines with end-to-end, unsupervised deep learning models from vision (in mechanical settings), it is necessary to use graphics rendering decoders. While previous works had resorted to providing ground-truth positions and velocities, PAIG was the first to show that using only the family of equations of motion as dynamical constraint was sufficient to identify objects in the scene and their dynamics from unlabelled videos, provided that the encoder and decoder used differentiable graphics representations. This paradigm was adopted by a number of follow-up works, such as:

- Guen et al. (2020) extended the PAIG architecture to a hybrid latent model;
- Kandukuri et al. (2020) extended PAIG to rigid body motion;
- Zhong et al. (2020b) extended the PAIG architecture to model more general systems using the Lagrangian formulation (Sec. 2.3.2);
- Murthy et al. (2021) used a more complex physics and rendering engine in order to model arbitrary objects.

In work concurrent with PAIG, Heiden et al. (2019) used rigid body equations to learn multi-link models from vision. Though close in spirit (both use a physical motion model as the dynamics bottleneck in an autoencoder architecture), they regress position and velocity coordinates from

ground-truth values. This is unlike PAIG, where there is no state supervision.

Despite its contributions, PAIG has a number of serious limitations which should be improved upon, and should be kept in mind by anyone trying to build on it. Firstly, and like other autoencoder models based on inverse-graphics frameworks, PAIG tends to be very hard to train in the sense that convergence to a correct minimum (i.e. discovering the correct objects) depends strongly on the model initialisation, and if objects are not correctly discovered determining the physics becomes impossible. Secondly, it is limited to 2D scenes where the family of motion of all the objects in the scene is known. Thirdly, the use of object templates to describe object appearance prevents modelling objects whose appearance can change throughout the video. These last two points greatly limit PAIG's application to real world scenes as they are inherently 3D, objects' appearances change due to changes in lightning, rotation, and perspective, and many scenes contain moving objects that are *not* of interest. Though our V-SysId model addresses some of these issues using keypoint representations, it would be of interest to develop models that tackle them from an inverse-graphics perspective.

### V-SysId

With the V-SysId model, we tried to break away from the end-to-end neural network paradigm in order to obtain representational flexibility and not be bound by the difficult convergence and differentiability properties experienced with PAIG. This resulted in a very simple and data-efficient model, that we were able to apply to real-world settings without difficulty, and whose outputs can be used to supervise a neural network for downstream tasks. This is in stark contrast with PAIG and the NewtonianVAE (and related models), which are often limited to very simple and/or simulated scenes without distractors. The 3-stage approach of V-SysId enabled principled and easy-to-analyse solutions at every stage, which ultimately resulted in a model able to simultaneously identify keypoint of interest, estimate physical parameters, and estimate camera pose from unlabelled video, using only a family of equations of motion as a constraint. Therefore, we see the use of dynamical constraints instead of geometrical view constraints for scene analysis as the main contribution of V-SysId to the wider literature. This is an important contribution, as it breaks from the simulated scene limitations of PAIG and NewtonianVAE, being applicable to complex real-world scenes.

Nevertheless, it is not without its limitations. By construction, V-SysId is not built to detect multiple objects under different types of motion - only one object or regions under the same family of equations. This limits its application to scenes with interacting objects, although this is likely a straightforward extension to the current model. Due to the representation of objects as keypoints, V-SysId is only able to estimate centre-of-mass motion, rather than rigid-body motion, which limits its ability to estimate physical properties that depend on object rotation or compression.

### NewtonianVAE

The main contribution of the NewtonianVAE model was the formulation of a latent transition structure that promotes the learning of a latent space that respects Newton's second law, $\mathbf{F} = m\,\mathbf{a}$, in deep variational models. Such a latent space enables control from vision using very simple and well understood PID controllers, unlike related models thus far, which require the use of model-predictive controllers. While most related works are evaluated on a small set of usual tasks (generative/reconstruction ability; latent space structure; performance under MPC), the compatibility with PID controllers enabled us to tackle a wider variety of tasks under one model, such as goal discovery and segmentation in imitation learning and path following using dynamic movement primitives.

From a peer review perspective, the NewtonianVAE was our most successful work, having received an Oral acceptance at CVPR2020 and shortlisted for Best Paper Award. As of this writing, one paper has adopted a similar approach to proportional control in latent spaces from vision (Wang et al. 2021), though instead of imposing proportional controllability via latent structure, the authors impose it via a novel Lyapunov risk loss. I believe our approach and Wang et al. (2021)'s can be complimentary rather than mutually exclusive, so combining both ideas is an interesting avenue of future work. I am confident that the adoption of PID-compatible latent spaces in future works can lead to many more exciting and useful applications which have traditionally been out of reach for unsupervised models from vision.

Like PAIG, the NewtonianVAE model was only applied to simulated scenes, though of much greater complexity, where exact object motion is not known, such as a 3D robot arm. Though at first glance the NewtonianVAE does not have any inherent limitations that might prevent its application to real-world scenes, experiments might highlight issues with the model and need for improvement. Furthermore, the model has only been applied to single-object scenes, though we believe that adapting it to multi-object scenes will require few modifications.

## 6.2 Future Work

Our works open the door to a number of promising extensions and applications to, and combinations with, different areas of machine learning. Besides addressing the limitations described in the previous section, all of which could constitute sources of novel work, there are a number of short- and long-term directions that could be investigated.

**Probabilistic PAIG/V-SysId** Framing PAIG and V-SysId in a probabilistic framework would enable probabilistic estimation of object states and physical parameters, allowing for improved control and prediction in noisy environments via robust uncertainty estimation. For PAIG, a variational autoencoder formulation could be used, though we expect convergence difficulties as the original

formulation is already unstable, and adding probabilistic bottlenecks would only worsen it. For V-SysId, on the other hand, it would be possible to use particle filters or other MCMC methods, which would better capture the posterior distribution of parameters given a video.

**Symbolic discovery with PAIG/V-SysId** Integrating PAIG or V-SysId with symbolic discovery method, as discussed in Section 2.2.2, would lift the limitation of having to know the exact family of equations of motion of the objects of interest in advance, while maintaining strong physical inductive biases. However, this approach would bring about object and motion discovery difficulties, as a broad motion prior (in the form of a set of motion primitives) would likely be too general to induce the correct trajectories from vision in an unsupervised manner. In that case, it might make sense to perform object discovery and equation of motion discovery in separate steps (for example, feeding the outputs of Kosiorek et al. (2018) to Cranmer et al. (2020a)), rather than end-to-end, which would likely also improve convergence.

**Few-shot learning for PAIG/NewtonianVAE** Integrating PAIG with a meta-learning framework where not only the parameters, but also the form of the equations of motion (i.e. symbolic discovery) can be quickly determined from small amounts of data, as done by Lee et al. (2021) with Hamiltonian models, would provide large training speed-ups and applicability across scenes. This is particularly relevant in models with tight information bottlenecks such as PAIG, where it takes a long time to reach convergence on the physical parameters. A model akin to a "Neural Physicist" (borrowing from the concept of the Neural Statistician (Edwards et al. 2017)) could, for example, provide approximate estimates of the objects and parameters in an amortised manner, which could then be fine-tuned through direct optimisation.

**3D PAIG** Integrating unsupervised 3D vision-as-inverse-graphics models such as Henderson et al. (2020) is a natural extension of PAIG, although it will likely involve major work and improvements in order to enable coordinate-consistent rendering, which is needed for integration with a physics engine.

**Multi-object NewtonianVAE** A Newtonian latent structure could be built into existing variational object discovery models (e.g. Engelcke et al. (2020)) in order to provide the benefits of physically consistent dynamics with multi-object scene modelling.

**Model-based RL** Though in our works we only used the trained models to solve control tasks where an objective is known *a priori* (i.e. we can define a goal in state or image space), hence where the task can be solved with model-predictive control, it is valuable to consider the possible applications in model-based reinforcement learning (MBRL, e.g. Hafner et al. (2019)), where the control reward is unknown and given by the environment. In particular, seeing how the NewtonianVAE enables interesting control applications such as goal sequence discovery and path following, we presume that a PID-controllable latent space would enable learning of MBRL policies that are

significantly more efficient and interpretable than current methods, which would be of interest in real world scenarios where it is difficult to perform data collection and environment rollouts at scale.

**Towards decoderless models** Models like Jonschkowski et al. (2017) and (Kipf et al. 2020) are able to learn physical state representations without relying on a reconstruction loss, so there is no rendering or decoding step. This would be of particular benefit to inverse-graphics models, as the reconstruction step is based on differentiable rendering, which, in our experience, is a tight bottleneck that hampers end-to-end model learning. Throughout this PhD, we tried to formulate object and keypoint discovery in a contrastive predictive coding framework (CPC, Oord et al. (2018)) several times, unsuccessfully. We found that while CPC is very good at learning unconstrained latent spaces, it often does not converge to the correct solution in the case when there is additional structure built into the model (e.g. explicit object positions/velocities or physical parameters). Nevertheless, I believe that this direction is worth exploring further, as there are significant gains to be obtained in terms of training speed and versatility if encoder-only models with structured latent priors become possible.

## Closing thoughts

Looking at the general trends in the field, we observe a clear movement towards integration of known or learned simulation models with visual or proprioception data, which can be seen by the increasing number of conference workshops dedicated to this topic. Developments in the neural physics engine and self-supervised learning areas in particular should be followed closely as they are experiencing very fast progress, and such advances are likely to provide valuable inspiration for models that aim to integrate physics with vision. While applications of such models to real world environments are still limited, a continued effort in this direction is bound to result in more efficient and widely applicable models. Lying at the intersection of several areas, I am confident physics+vision will remain an exciting area to follow in the coming years.

# References


Koopman, B. O. (1931). "Hamiltonian Systems and Transformation in Hilbert Space". In: *PNAS*.

Kirk, D. E. (1970). *Optimal control theory: An Introduction*. Prentice-Hall.

Fukushima, K. (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics*.

An, C. H. et al. (1985). "Estimation of Inertial Parameters of Rigid Body Links of Manipulators". In: *IEEE Conference on Decision and Control*. IEEE.

Juang, J.-N. et al. (1985). "An eigensystem realization algorithm for modal parameter identification and model reduction". In: *JGCD*.

Mockus, J. (1989). *Bayesian Approach to Global Optimization*. Springer.

Chen, S. et al. (1990). "Non-linear system identification using neural networks". In: *International Journal of Control*.

Narendra, K. S. et al. (1990). "Identification and Control of Dynamical Systems Using Neural Networks". In: *IEEE Transactions on Neural Networks*.

Werbos, P. J. (1990). "Backpropagation Through Time: What It Does and How to Do It". In: *Proceedings of the IEEE*.

Moore, A. (1991). "Fast, Robust Adaptive Control by Learning only Forward Models". In: *NIPS*.

Tomasi, C. et al. (1991). *Detection and Tracking of Point Features*. Tech. rep.

Lambert, J. D. (1992). *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Wiley.

Söderström, T. et al. (1992). "System Identification". In: *Automatica*.

Williams, R. J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning*.

Isaacson, E. et al. (1994). *Analysis of numerical methods*. Dover Publications.

Hochreiter, S. et al. (1997a). "Long Short-Term Memory". In: *Neural Computation*.

Hochreiter, S. et al. (1997b). "Long short-term memory". In: *Neural computation*.

Rubinstein, R. Y. (1997). "Optimization of computer simulation models with rare events". In: *EJOR*.

Grzeszczuk, R. et al. (1998). "NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models". In: *Conference on Computer Graphics and Interactive Techniques*.

Kozlowski, K. (1998). *Modelling and Identification in Robotics*. Springer.

Ljung, L. (1998). "System Identification". In: *Signal Analysis and Prediction*.

Burridge, R. R. et al. (1999). "Sequential composition of dynamically dexterous robot behaviors". In: *The International Journal of Robotics Research*.

Full, R. J. et al. (1999). "Templates and anchors: neuromechanical hypotheses of legged locomotion on land". In: *Journal of Experimental Biology*.

Jordan, M. I. et al. (1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning*.

Sutton, R. S. et al. (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial Intelligence*.

Avidan, S. et al. (2000). "Trajectory triangulation: 3D reconstruction of moving points from a monocular image sequence". In: *PAMI*.

Fitzgibbon, A. W. et al. (2000). "Multibody Structure and Motion: 3-D Reconstruction of Independently Moving Objects". In: ECCV.

Fletcher, R. (2000). *Practical Methods of Optimization*. John Wiley & Sons, Ltd.

Ng, A. Y. et al. (2000). "Algorithms for Inverse Reinforcement Learning". In: *ICML*.
Brincker, R. et al. (2001). "Modal identification of output-only systems using frequency domain decomposition". In: *Smart Materials and Structures*.
Bhat, K. S. et al. (2002). "Computing the Physical Parameters of Rigid-body Motion from Video". In: *ECCV*.
Cline, M. B. (2002). "Rigid body simulation with contact and constraints". In: *PhD Thesis*.
Kaminski, J. Y. et al. (2002). "General trajectory triangulation". In: *ECCV*.
De Boer, P.-T. et al. (2003). *A Tutorial on the Cross-Entropy Method*. Tech. rep.
Han, M. et al. (2003). "Multiple Motion Scene Reconstruction with Uncalibrated Cameras". In: *PAMI*.
Mannor, S. et al. (2003). "The Cross Entropy method for Fast Policy Search". In: *ICML*.
David, P. et al. (2004). "SoftPOSIT: Simultaneous Pose and Correspondence Determination". In: *IJCV*.
Dixon, K. R. et al. (2004). "Trajectory representation using sequenced linear dynamical systems". In: *ICRA*.
Li, W. et al. (2004). "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems". In: *ICINCO*.
Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *IJCV*.
Leimkuhler, B. et al. (2005). *Simulating Hamiltonian Dynamics*. Cambridge University Press.
Duc, L. et al. (2006). "On stability of linear time-varying second-order differential equations". In: *Quarterly of Applied Mathematics*.
Rosten, E. et al. (2006). "Machine learning for high-speed corner detection". In: *ECCV*.
Schaal, S. (2006). "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics". In: *Adaptive motion of animals and machines*. Springer.
Bongard, J. et al. (June 2007). "Automated reverse engineering of nonlinear dynamical systems". In: *PNAS*.
Ramsay, J. O. et al. (2007). "Parameter Estimation for Differential Equations: A Generalized Smoothing Approach". In: *Journal of the Royal Statistical Society Series B*.
Ziebart, B. D. et al. (2008). "Maximum entropy inverse reinforcement learning". In: *AAAI*.
Cottle, R. W. et al. (2009). *The Linear Complementarity Problem*. SIAM.
Konidaris, G. et al. (2009). "Skill discovery in continuous reinforcement learning domains using skill chaining". In: *NIPS*.
Scaramuzza, D. et al. (2009). "Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints". In: *ICCV*.
Scarselli, F. et al. (2009). "The graph neural network model". In: *IEEE Transactions on Neural Networks*.
Schmidt, M. et al. (2009). "Distilling free-form natural laws from experimental data". In: *Science*.
Boer, W. de et al. (2010). "SLP: A Zero-Contact Non-Invasive Method for Pulmonary Function Testing". In: *BMVC*.
Chiappa, S. et al. (2010). "Movement extraction by detecting dynamics switches and repetitions". In: *NIPS*.
Deisenroth, M. et al. (2011). "PILCO: A model-based and data-efficient approach to policy search". In: *ICML*.
Dorf, R. C. et al. (2011). *Modern Control Systems*. Pearson.
Hamrick, J. et al. (2011). "Internal physics models guide probabilistic judgments about object dynamics". In: *Annual Meeting of the Cognitive Science Society*.
Hinton, G. E. et al. (2011). "Transforming auto-encoders". In: *ICANN*.
Niekum, S. et al. (2011). "Clustering via Dirichlet Process Mixture Models for Portable Skill Discovery". In: *NIPS*.

Rublee, E. et al. (2011). "ORB: An efficient alternative to SIFT or SURF". In: *ICCV*.
Budišić, M. et al. (2012). "Applied Koopmanism". In: *Chaos*.
Hinton, G. et al. (2012). "Lecture 6a: Overview of mini-batch gradient descent". In: *Neural Networks for Machine Learning*.
Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.
Battaglia, P. W. et al. (2013). "Simulation as an engine of physical scene understanding". In: *PNAS*.
Bengio, Y. et al. (2013). "Representation learning: A review and new perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
Botev, Z. I. et al. (2013). "The Cross-Entropy Method for Optimization". In: *Handbook of Statistics*.
Ijspeert, A. J. et al. (2013). "Dynamical movement primitives: learning attractor models for motor behaviors". In: *Neural Computation*.
Kingma, D. et al. (2014a). "Adam: A Method for Stochastic Optimization". In: *ICLR*.
Kingma, D. et al. (2014b). "Auto-Encoding Variational Bayes". In: *ICLR*.
Levine, S. et al. (2014). "Learning neural network policies with guided policy search under unknown dynamics". In: *NIPS*.
Loper, M. M. et al. (2014). "OpenDR: An Approximate Differentiable Renderer". In: *ECCV*.
Rezende, D. J. et al. (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *International Conference on Machine Learning*.
Tieleman, T. (2014). "Optimizing Neural Networks that Generate Images". In: *PhD thesis*.
Ullman, T. et al. (2014). "Learning Physics from Dynamical Scenes". In: *CogSci*.
Bagnell, J. A. (2015). *An Invitation to Imitation*. Tech. rep.
Chung, J. et al. (2015). "A Recurrent Latent Variable Model for Sequential Data". In: *NIPS*.
Fischer, P. et al. (2015). "FlowNet: Learning Optical Flow with Convolutional Networks". In: *ICCV*.
Gregor, K. et al. (2015). "DRAW: A Recurrent Neural Network For Image Generation". In: *ICML*.
Jaderberg, M. et al. (2015). "Spatial Transformer Networks". In: *NIPS*.
Krishnan, R. G. et al. (2015). "Deep Kalman Filters". In: *arXiv preprin arXiv:1511.05121*.
Kulkarni, T. D. et al. (2015). "Deep Convolutional Inverse Graphics Network". In: *NIPS*.
Lake, B. M. et al. (2015). "Human-level concept learning through probabilistic program induction". In: *Science*.
Ranchod, P. et al. (2015). "Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning". In: *IROS*.
Ronneberger, O. et al. (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI*.
Srivastava, N. et al. (2015). "Unsupervised Learning of Video Representations using LSTMs". In: *ICML*.
Wahlstrom, N. et al. (2015). "From Pixels to Torques: Policy Learning with Deep Dynamical Models". In: *arXiv preprint arXiv:1502.02251*.
Watter, M. et al. (2015). "Embed to control: A locally linear latent dynamics model for control from raw images". In: *NIPS*.
Wu, J. et al. (2015). "Galileo : Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning". In: *NIPS*.
Abadi, M. et al. (2016). "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *Conference on Operating Systems Design and Implementation*.
Battaglia, P. W. et al. (2016). "Interaction Networks for Learning about Objects, Relations and Physics". In: *NIPS*.
Brockman, G. et al. (2016). *OpenAI Gym*. Tech. rep.
Brunton, S. L. et al. (2016a). "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems". In: *PNAS*.

Brunton, S. L. et al. (2016b). "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control". In: *PloS one*.
Chen, N. et al. (2016). "Dynamic movement primitives in latent space of time-dependent variational autoencoders". In: *IEEE-RAS International Conference on Humanoid Robots*.
Degrave, J. et al. (Nov. 2016). "A Differentiable Physics Engine for Deep Learning in Robotics". In:
Eslami, S. M. A. et al. (2016). "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models". In: *NIPS*.
Finn, C. et al. (2016a). "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models". In: *arXiv preprint arXiv:1611.03852*.
Finn, C. et al. (2016b). "Unsupervised Learning for Physical Interaction through Video Prediction". In: *NIPS*.
Fragkiadaki, K. et al. (2016). "Learning Visual Predictive Models of Physics for Playing Billiards". In: *ICLR*.
Hamrick, J. B. et al. (2016). "Inferring mass in complex scenes by mental simulation". In: *Cognition*.
Ho, J. et al. (2016). "Generative adversarial imitation learning". In: *NIPS*.
Huang, J. et al. (2016). "Efficient Inference in Occlusion-Aware Generative Models of Images". In: *ICLR Workshop*.
Lerer, A. et al. (2016). "Learning Physical Intuition of Block Towers by Example". In: *arXiv preprint arXiv:1603.01312*.
Li, Y. et al. (2016). "Gated Graph Sequence Neural Networks". In: *ICLR*.
Lillicrap, T. P. et al. (2016). "Continuous control with deep reinforcement learning". In: *ICLR*.
Martius, G. et al. (2016). "Extrapolation and learning equations". In: *arXiv preprint arXiv:1610.02995*.
Mnih, V. et al. (2016). "Asynchronous methods for deep reinforcement learning". In: *ICML*.
Monszpart, A. et al. (2016). "SMASH: Physics-guided Reconstruction of Collisions from Videos". In: *SIGGRAPH Asia*.
Moreno, P. et al. (2016). "Overcoming Occlusion with Inverse Graphics". In: *ECCV Workshops*.
Mottaghi, R. et al. (2016). "Newtonian Image Understanding: Unfolding the Dynamics of Objects in Static Images". In: *CVPR*.
Ramakrishnan, S. K. et al. (2016). "CoMaL Tracking: Tracking Points at the Object Boundaries". In: *CVPR*.
Rezende, D. J. et al. (2016). "One-Shot Generalization in Deep Generative Models". In: *ICML*.
Romeres, D. et al. (2016). "On-line Bayesian System Identification". In: *ECC*.
Torres, A. et al. (2016). "Turning the internet of (my) things into a remote controlled laboratory". In: *International Conference on Remote Engineering and Virtual Instrumentation*.
Wu, J. et al. (2016). "Physics 101: Learning physical object properties from unlabeled videos". In: *BMVC*.
Abraham, I. et al. (2017). "Model-Based Control Using Koopman Operators". In: *RSS*.
Arbabi, H. et al. (2017). "Ergodic theory, Dynamic Mode Decomposition and Computation of Spectral Properties of the Koopman operator". In: *SIAM Journal on Applied Dynamical Systems*.
Baker, C. L. et al. (2017). "Rational quantitative attribution of beliefs, desires and percepts in human mentalizing". In: *Nature Human Behaviour*.
Chang, M. B. et al. (2017). "A Compositional Object-Based Approach to Learning Physical Dynamics". In: *ICLR*.
Edwards, H. et al. (June 2017). "Towards a Neural Statistician". In: *ICLR*.
Ehrhardt, S. et al. (2017). "Learning A Physical Long-term Predictor". In: *arXiv preprint arXiv:1703.00247*.
Finn, C. et al. (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *ICML*.

Fraccaro, M. et al. (2017). "A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning". In: *NIPS*.
Greff, K. et al. (2017). "Neural Expectation Maximization". In: *ICLR*.
Johnson, J. et al. (2017). "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning". In: *CVPR*.
Jonschkowski, R. et al. (2017). "PVEs: Position-Velocity Encoders for Unsupervised Learning of Structured State Representations". In: *arXiv preprint arXiv:1705.09805*.
Karl, M. et al. (2017). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data". In: *ICLR*.
Li, Y. et al. (2017). "InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations". In: *NIPS*.
Linderman, S. W. et al. (2017). "Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems". In: *AISTATS*.
Long, Z. et al. (2017). "PDE-Net: Learning PDEs from Data". In: *ICML*.
Lopez-Guevara, T. et al. (2017). "Adaptable Pouring: Teaching Robots Not to Spill using Fast but Approximate Fluid Simulation". In: *CoRL*. PMLR.
Matthey, L. et al. (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*.
Pervez, A. et al. (2017). "Learning deep movement primitives using convolutional neural networks". In: *IEEE-RAS International Conference on Humanoid Robotics*.
Raissi, M. et al. (Aug. 2017). "Hidden Physics Models: Machine Learning of Nonlinear Partial Differential Equations". In: *Journal of Computational Physics*.
Romaszko, L. et al. (2017). "Vision-as-Inverse-Graphics: Obtaining a Rich 3D Explanation of a Scene from a Single Image". In: *ICCV*.
Rudy, S. H. et al. (Apr. 2017). "Data-driven discovery of partial differential equations". In: *Science Advances*.
Sabour, S. et al. (2017). "Dynamic Routing Between Capsules". In: *NIPS*.
Santoro, A. et al. (2017). "A Simple Neural Network Module for Relational Reasoning". In: *NIPS*.
Schaeffer, H. (2017). "Learning partial differential equations via data discovery and sparse optimization". In: *Proc. R. Soc. A*.
Stewart, R. et al. (2017). "Label-Free Supervision of Neural Networks with Physics and Domain Knowledge". In: *AAAI*.
Takeishi, N. et al. (2017). "Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition". In: *NIPS*.
Vijayanarasimhan, S. et al. (2017). "SfM-Net: Learning of Structure and Motion from Video". In: *arXiv preprint arXiv:1704.07804*.
Watters, N. et al. (2017). "Visual Interaction Networks: Learning a Physics Simulator from Video". In: *NIPS*.
Wu, J. et al. (2017a). "Learning to See Physics via Visual De-animation". In: *NIPS*.
Wu, J. et al. (2017b). "Neural Scene De-rendering". In: *CVPR*.
Arbabi, H. et al. (2018). "A data-driven Koopman model predictive control framework for nonlinear flows". In: *CDC*.
Banijamali, E. et al. (2018). "Robust Locally-Linear Controllable Embedding". In: *ICML*.
Barth-Maron, G. et al. (2018). "Distributed distributional deterministic policy gradients". In: *ICLR*.
Battaglia, P. W. et al. (2018). "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261*.
Belbute-Peres, F. D. A. et al. (2018). "End-to-End Differentiable Physics for Learning and Control". In: *NIPS*.

Byravan, A. et al. (2018). “SE3-Pose-Nets: Structured Deep Dynamics Models for Visuomotor Planning and Control”. In: *ICRA*.
Chen, R. T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In: *NIPS*.
Chua, K. et al. (2018). “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”. In: *NeurIPS*.
DeTone, D. et al. (2018). “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *CVPR*.
Ehrhardt, S. et al. (2018). “Unsupervised Intuitive Physics from Visual Observations”. In: *arXiv preprint arXiv:1805.08095*.
Ellis, K. et al. (2018). “Learning to Infer Graphics Programs from Hand-Drawn Images”. In: *NIPS*.
Fu, J. et al. (2018). “Learning robust rewards with adversarial inverse reinforcement learning”. In:
Groth, O. et al. (2018). “ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking”. In: *ECCV*.
Ha, D. et al. (2018). “World Models”. In: *NIPS*.
Hsieh, J.-T. et al. (2018). “Learning to Decompose and Disentangle Representations for Video Prediction”. In: *NIPS*.
Jakab, T. et al. (2018). “Unsupervised Learning of Object Landmarks through Conditional Image Generation”. In: *NIPS*.
Karl, M. et al. (2018). “Deep variational Bayes filters: Unsupervised learning of state space models from raw data”. In: *ICLR*.
Kato, H. et al. (2018). “Neural 3D Mesh Renderer”. In: *CVPR*.
Kipf, T. et al. (2018). “Neural Relational Inference for Interacting Systems”. In: *ICML*.
Kosiorek, A. R. et al. (2018). “Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects”. In: *NIPS*.
Lesort, T. et al. (2018). “State representation learning for control: An overview”. In: *Neural Networks*.
Long, Y. et al. (2018). “HybridNet: Integrating Model-based and Data-driven Learning to Predict Evolution of Dynamical Systems”. In: *CoRL*.
Lusch, B. et al. (2018). “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature Communications*.
Mania, H. et al. (2018). “Simple random search provides a competitive approach to reinforcement learning”. In: *NIPS*.
Mrowca, D. et al. (2018). “Flexible Neural Representation for Physics Prediction”. In: *NIPS*.
Nair, A. V. et al. (2018). “Visual reinforcement learning with imagined goals”. In: *NeurIPS*.
Ono, Y. et al. (2018). “LF-Net: Learning Local Features from Images”. In: *NeurIPS*.
Oord, A. van den et al. (2018). “Representation Learning with Contrastive Predictive Coding”. In: *arXiv preprint arXiv:1807.03748*.
Ryder, T. et al. (2018). “Black-box Variational Inference for Stochastic Differential Equations”. In: *ICML*.
Sahoo, S. S. et al. (2018). “Learning Equations for Extrapolation and Control”. In: *ICML*.
Sanchez-Gonzalez, A. et al. (2018). “Graph Networks as Learnable Physics Engines for Inference and Control”. In: *ICML*.
Schenck, C. et al. (2018). “SPNets: Differentiable Fluid Dynamics for Deep Neural Networks”. In: *CoRL*.
Steenkiste, S. van et al. (2018). “Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions”. In: *ICLR*.
Suwajanakorn, S. et al. (2018). “Discovery of Latent 3D Keypoints via End-to-end Geometric Reasoning”. In: *NIPS*.

Toussaint, M. et al. (2018). "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning". In: *RSS*.
Trask, A. et al. (2018). "Neural Arithmetic Logic Units". In: *NIPS*.
Wang, Z. et al. (2018). "Neural Allocentric Intuitive Physics Prediction from Real Videos". In: *arXiv preprint arXiv:1809.03330*.
Zheng, D. et al. (2018). "Unsupervised Learning of Latent Physical Properties Using Perception-Prediction Networks". In: *UAI*.
Zhu, G. et al. (2018). "Object-Oriented Dynamics Predictor". In: *NIPS*.
Asenov, M. et al. (2019). "Vid2Param: Modelling of Dynamics Parameters from Video". In: *ICRA*.
Becker-Ehmck, P. et al. (2019). "Switching Linear Dynamics for Variational Bayes Filtering". In: *ICML*.
Bertalan, T. et al. (Dec. 2019). "On learning Hamiltonian systems from data". In: *Chaos*.
Bondesan, R. et al. (2019). "Learning Symmetries of Classical Integrable Systems". In: *arXiv preprint arXiv:1906.04645*.
Bruder, D. et al. (2019). "Modeling and Control of Soft Robots Using the Koopman Operator and Model Predictive Control". In: *RSS*.
Burgess, C. P. et al. (2019). "MONet: Unsupervised Scene Decomposition and Representation". In: *arXiv preprint arXiv:1901.11390*.
Burke, M. et al. (2019a). "From Explanation to Synthesis: Compositional Program Induction for Learning From Demonstration". In: *RSS*.
Burke, M. et al. (2019b). "Hybrid system identification using switching density networks". In: *CoRL*.
Chen, W. et al. (2019). "Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer". In: *NeurIPS*.
Ghasemipour, S. K. S. et al. (2019). "A Divergence Minimization Perspective on Imitation Learning Methods". In: *CoRL*.
Greff, K. et al. (2019). "Multi-Object Representation Learning with Iterative Variational Inference". In: *ICML*.
Greydanus, S. et al. (2019). "Hamiltonian Neural Networks". In: *NeurIPS*.
Gupta, J. K. et al. (2019). "A General Framework for Structured Learning of Mechanical Systems". In: *arXiv preprint arXiv:1902.08705*.
Hafner, D. et al. (2019). "Learning Latent Dynamics for Planning from Pixels". In: *ICML*.
Heiden, E. et al. (2019). "Interactive Differentiable Simulation". In: *arXiv preprint arXiv:1905.10706*.
Jakab, T. et al. (July 2019). "Learning Landmarks from Unaligned Data using Image Translation". In: *arXiv preprint arXiv:1907.02055*.
Janner, M. et al. (2019). "Reasoning About Physical Interactions with Object-Oriented Prediction and Planning". In: *ICLR*.
Kipf, T. et al. (2019). "CompILE: Compositional Imitation Learning and Execution". In: *ICML*.
Kosiorek, A. R. et al. (2019). "Stacked Capsule Autoencoders". In: *NeurIPS*.
Kulkarni, T. et al. (2019). "Unsupervised Learning of Object Keypoints for Perception and Control". In: *NIPS*.
Liu, S. et al. (2019). "Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning". In: *ICCV*.
Long, Z. et al. (Dec. 2019). "PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network". In: *Journal of Computational Physics*.
Lutter, M. et al. (2019). "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning". In: *ICLR*.
Manuelli, L. et al. (Mar. 2019). "kPAM: KeyPoint Affordances for Category-Level Robotic Manipulation". In: *ISRR*.

Minderer, M. et al. (2019). "Unsupervised Learning of Object Structure and Dynamics from Videos". In: *NIPS*.
Morton, J. et al. (2019). "Deep Variational Koopman Models: Inferring Koopman Observations for Uncertainty-Aware Dynamics Modeling and Control". In: *IJCAI*.
Pan, S. et al. (2019). "Physics-Informed Probabilistic Learning of Linear Embeddings of Non-linear Dynamics With Guaranteed Stability". In: *SIAM Journal on Applied Dynamical Systems*.
Park, J. et al. (Nov. 2019). "Physics-induced graph neural network: An application to wind-farm power estimation". In: *Energy*.
Paszke, A. et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *NeurIPS*.
Penkov, S. et al. (2019). "Learning Programmatically Structured Representations with Perceptor Gradients". In: *ICLR*.
Ramos, F. et al. (2019). "BayesSim: adaptive domain randomization via probabilistic inference for robotics simulators". In: *RSS*.
Read, J. S. et al. (Nov. 2019). "Process-Guided Deep Learning Predictions of Lake Water Temperature". In: *Water Resources Research*.
Sanchez-Gonzalez, A. et al. (2019). "Hamiltonian Graph Networks with ODE Integrators". In: *arXiv preprint arXiv:1909.12790*.
Seo, S. et al. (2019). "Differentiable Physics-informed Graph Networks". In: *arXiv preprint arXiv:1902.02950*.
Watters, N. et al. (2019a). "COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration". In: *arXiv preprint arXiv:1905.09275*.
Watters, N. et al. (2019b). "Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs". In: *ICLR Workshop*.
Xu, Z. et al. (2019). "Unsupervised Discovery of Parts, Structure, and Dynamics". In: *ICLR*.
Baradel, F. et al. (Sept. 2020). "CoPhy: Counterfactual Learning of Physical Dynamics". In: *ICLR*.
Belbute-Peres, F. D. A. et al. (2020). *Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction*. Tech. rep.
Chen, Z. et al. (2020). "Symplectic Recurrent Neural Networks". In: *ICLR*.
Cranmer, M. et al. (2020a). "Discovering Symbolic Models from Deep Learning with Inductive Biases". In: *NeurIPS*.
Cranmer, M. et al. (2020b). "Lagrangian Neural Networks". In: *ICLR Workshop*.
Das, N. et al. (2020). "Model-Based Inverse Reinforcement Learning from Visual Demonstrations". In: *CoRL*.
Engelcke, M. et al. (2020). "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *ICLR*.
Finzi, M. et al. (Oct. 2020). "Simplifying Hamiltonian and Lagrangian Neural Networks via Explicit Constraints". In: *NeurIPS*.
Gopalakrishnan, A. et al. (2020). "Usupervised Object Keypoint Learning using Local Spatial Predictability". In: *arXiv preprint arXiv:2011.12930*.
Guen, V. L. et al. (2020). "Disentangling Physical Dynamics from Unknown Factors for Unsupervised Video Prediction". In: *CVPR*.
Hafner, D. et al. (2020). "Dream to Control: Learning Behaviors by Latent Imagination". In: *ICLR*.
Henderson, P. et al. (2020). "Unsupervised object-centric video generation and decomposition in 3D". In: *NeurIPS*.
Iten, R. et al. (2020). "Discovering physical concepts with neural networks". In: *PRL*.
Jaques, M. et al. (2020). "Physics-as-Inverse-Graphics: Unsupervised Physical Parameter Estimation from Video". In: *ICLR*.

Jau, Y.-Y. et al. (2020). "Deep Keypoint-Based Camera Pose Estimation with Geometric Constraints". In: *IROS*.
Jegorova, M. et al. (2020). "Adversarial Generation of Informative Trajectories for Dynamics System Identification". In: *IROS*.
Kandukuri, R. et al. (2020). "Learning to Identify Physical Parameters from Video Using Differentiable Physics". In: *GCPR*.
Kipf, T. et al. (2020). "Contrastive Learning of Structured World Models". In: *ICLR*.
Kobayashi, T. (2020). "q-VAE for Disentangled Representation Learning and Latent Dynamical Systems". In: *arXiv preprint arXiv:2003.01852*.
Li, Y. et al. (2020a). "Learning Compositional Koopman Operators for Model-Based Control". In: *ICLR*.
Li, Y. et al. (2020b). "Visual Grounding of Learned Physical Models". In: *ICML*.
Mamakoukas, G. et al. (2020). "Learning Data-Driven Stable Koopman Operators". In: *IEEE Trans. Robot.*
Manuelli, L. et al. (2020). "Keypoints into the Future: Self-Supervised Correspondence in Model-Based Reinforcement Learning". In: *CoRL*.
Mattheakis, M et al. (2020). "Physical Symmetries Embedded in Neural Networks". In: *arXiv preprint arXiv:1904.08991*.
Mohan, A. T. et al. (2020). "Embedding Hard Physical Constraints in Convolutional Neural Networks for 3D Turbulence". In: *ICLR Workshop on DeepDiffEq*.
Pinneri, C. et al. (Aug. 2020). "Sample-efficient Cross-Entropy Method for Real-time Planning". In: *CoRL*.
Qiao, Y.-L. et al. (2020). "Scalable Differentiable Physics for Learning and Control". In: *ICML*.
Runia, T. F. H. et al. (2020). "Cloth in the Wind: A Case Study of Physical Measurement through Simulation". In: *CVPR*.
Sanchez-Gonzalez, A. et al. (2020). "Learning to Simulate Complex Physics with Graph Networks". In: *ICML*.
Sekar, R. et al. (2020). "Planning to Explore via Self-Supervised World Models". In: *ICML*.
Song, C. et al. (2020). "Identifying Mechanical Models through Differentiable Simulations". In: *L4DC*.
Toth, P. et al. (2020). "Hamiltonian Generative Networks". In: *ICLR*.
Virtanen, P. et al. (2020). *SciPy 1.0: fundamental algorithms for scientific computing in Python*.
Wei, X. et al. (2020). "DeepSFM: Structure From Motion Via Deep Bundle Adjustment". In: *ECCV*.
Willard, J. et al. (2020). "Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems". In: *ACM*.
Yi, K. et al. (2020). "CLEVERER: Collision Events for Video Representation and Reasoning". In: *ICLR*.
Zeng, A. et al. (2020). "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics". In: *IEEE Transactions on Robotics*.
Zhong, Y. D. et al. (2020a). "Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control". In: *ICLR*.
Zhong, Y. D. et al. (2020b). "Unsupervised Learning of Lagrangian Dynamics from Images for Prediction and Control". In: *NeurIPS*.
Zhu, A. et al. (2020). "Deep Hamiltonian Networks Based on Symplectic Integrators". In:
Jaques, M. et al. (2021). "NewtonianVAE: Proportional Control and Goal Identification from Pixels via Physical Latent Spaces". In: *CVPR*.
Kaiser, E. et al. (Sept. 2021). "Data-driven discovery of Koopman eigenfunctions for control". In: *Machine Learning: Science and Technology*.
Lee, S. et al. (2021). "Identifying Physical Law of Hamiltonian Systems via Meta-Learning". In: *ICLR*.

Murthy, J. K. et al. (2021). “gradSim: Differentiable simulation for system identification and visuomotor control”. In: *ICLR*.
Wang, W. et al. (2021). “Learn Proportional Derivative Controllable Latent Space from Pixels”. In: *arXiv preprint arXiv:2110.08239*.

# Appendices

## A Cross-Entropy Method for Continuous Control

The cross-entropy method (CEM, Rubinstein (1997)) is a simple and elegant population-based algorithm used to solve many types of optimization problems such as combinatorial optimization, traveling salesman, and quadratic assignment (De Boer et al. 2003; Botev et al. 2013). It was originally developed for rare-event simulation/estimation (Rubinstein 1997), where we want to accurately estimate very small probabilities.

In the machine learning and robotics literature, it is a common tool used to solve the model-predictive control problem in (2.4) (e.g. Chua et al. (2018) and Hafner et al. (2019)), where the transition distributions $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ is learned from unlabeled data, as done by most of the models discussed in this thesis. In CEM, choosing an optimal action $\mathbf{u}_t^*$ for a state $\mathbf{z}_t$ consists of 5 simple steps:

1. Sample a set of action sequences from the proposal distribution, $\{\mathbf{u}_{t:t+n}^i\}_{i=1}^I \sim p_\phi(\mathbf{u}_{t:t+n})$ (if not defined, choose an initial distribution);
2. Pick the $J$ sequences with lowest cost under (2.4), $\{\hat{\mathbf{u}}_{t:t+n}^j\}_{j=1}^J$;
3. Reestimate the proposal distribution as $\phi = \arg\max_\phi \sum_{j=1}^J \log p_\phi(\mathbf{u}_{t:t+n}^j)$.
4. Repeat steps 1-3 $K$ times or until convergence;
5. Pick the lowest cost sequence, $\mathbf{u}_{t:t+n}^*$ and apply the first action on the environment, $\mathbf{u}_t^*$.

We can see this algorithm is very simple to implement, as it involves only a sampling step, cost estimation step, and parameter estimation steps. Moreover, it is very general, allowing for many design choices, including the form of the proposal distribution, the hyperparameters $I$ and $J$, and cost function used. Since it is a sampling based method, it can be used both with dense quadratic cost fuctions, such as (2.5), or sparse cost functions, such as $\mathcal{C}(\mathbf{z}_t, \mathbf{u}_t) = -c \cdot \mathbb{I}(\|\mathbf{z}_t - \mathbf{z}^*\| < \epsilon)$, where $\mathbb{I}$ is the indicator function, and $\epsilon$ is a neighbourhood of the target state $\mathbf{z}^*$.

Most often, the proposal function is parametrized by a set of independent diagonal Gaussians, $p_\phi(\mathbf{u}_{t:t+n}) = \prod_{t'=t}^{t+n} \mathcal{N}(\mathbf{u}; \mu_{t'}, \sigma_{t'}^2)$, in which case Step 3 is simply a mean and variance calculation.

A pseudo-code for this case, with more complete implementation details, is show in Algorithm 2. This is the CEM version we use in the papers presented in Sections 3 and 5.

**Algorithm 2** CEM with Gaussian proposal distributions

**Input:** Environment $\mathcal{E}$
**Input:** Cost function $\mathcal{C}$
**Input:** Learned or known transition model $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_{t-1})$
**Input:** Initial and target states, $\mathbf{z}^{\text{init}}$ and $\mathbf{z}^*$
**Input:** Target state reached criterion distance, $\epsilon$.
**Input:** Hyperparameters $I$, $J$, $K$, $H$
**Input:** Initial proposal distribution parameters $\mu'_t$ and $\sigma'_t$, $t' \in 0...H$
**Output:** Trajectory, physical parameters, and camera pose of the object of interest

$\mathbf{z}_t \leftarrow \mathbf{z}^{\text{init}}$
**while** $\|\mathbf{z}_t - \mathbf{z}^*\| > \epsilon$ **do**
  **for** $k \in \{1...K\}$ **do**
    # Sample action sequences
    $\{\mathbf{u}^i_{t:t+n}\}^I_{i=1} \sim p_\phi(\mathbf{u}_{t:t+n})$

    # Compute cost of each sequence
    Costs $\leftarrow$ []
    **for** $i \in \{1...I\}$ **do**
      Costs[$i$] $\leftarrow \sum_{t'=t}^{H} \mathcal{C}(\mathbf{z}_{t'}, \mathbf{u}^i_{t'})$, where $\mathbf{z}_{t'} \sim p(\mathbf{z}_{t'}|\mathbf{z}_{t'-1}, \mathbf{u}^i_{t'-1})$
    **end for**

    # Pick lowest cost sequences
    $\{\hat{\mathbf{u}}^j_{t:t+H}\}^J_{j=1} \leftarrow [\mathbf{u}^i_{t:t+H}$ **for** $i$ **in** arg-bottom-$K$(Costs)$]$

    # Estimate proposal distribution parameters
    $\mu_{t:t+H} \leftarrow \frac{1}{J} \sum_j \hat{\mathbf{u}}^j_{t:t+H}$
    $\sigma^2_{t:t+H} \leftarrow \frac{1}{J} \sum_j \left(\hat{\mathbf{u}}^j_{t:t+H} - \mu_{t:t+H}\right)^2$
  **end for**

  # Take chosen action on environment
  $\mathbf{u}^*_t \leftarrow \mu_t$
  $\mathbf{z}_t \leftarrow \mathcal{E}(\mathbf{z}_t, \mathbf{u}^*_t)$
**end while**

**Benefits** CEM is our go-to algorithm for MPC with unsupervised models, as it is easy to implement, inherently paralelizable, and only requires forward passes (gradient-free). Notably, it is generally insensitive to the exact hyperparameters used, converging correcly as long as the hyperparameters are within reasonable ranges (according to our experience, and as noted by Mannor et al. (2003)).

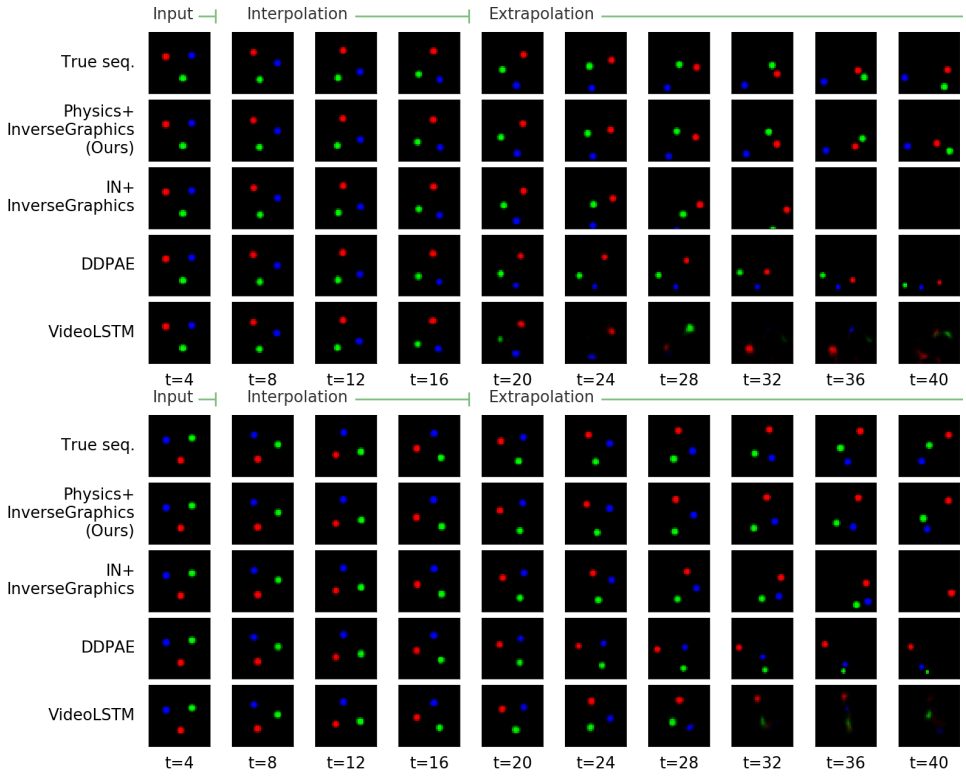
**Limitations** CEM's flexibility comes with a lack of theoretical guarantees, particularly when used with complex learned environment models and sparse rewards (as with any black-box optimization method). Furthermore, the standard CEM version as described here can be sample inefficient, as the action proposals are usually independently drawn with every iteration. To address this issue,

Pinneri et al. (2020) recently proposed an improved version that makes use of temporally-correlated actions and memory, increasing sample efficiency by an order of magnitude.

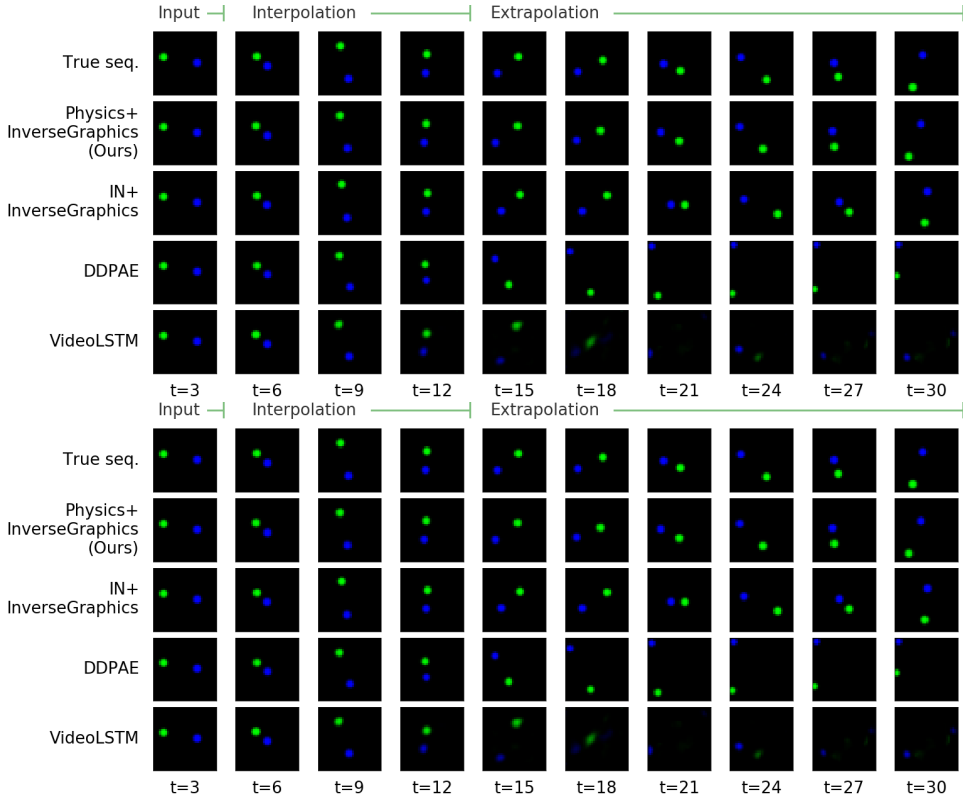
**Related methods** There is a number of alternative population-based optimization methods that can be used for MPC, such as Genetic Algorithms [cite], Neural Evolution Strategies (NES, [cite]), and Covariance Matrix Adaptation (CMA, [cite]), but detailed comparisons are outside the scope of this work. [add a review paper citation here]

# B Additional rollout comparisons for PAIG model

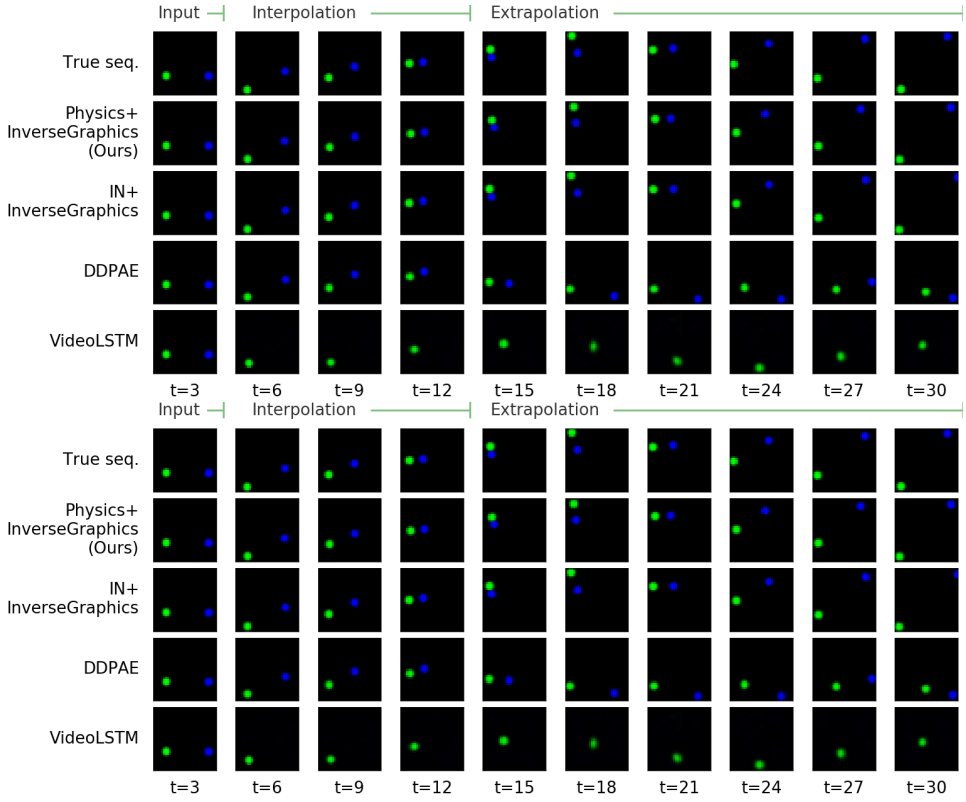
## 3-balls gravity



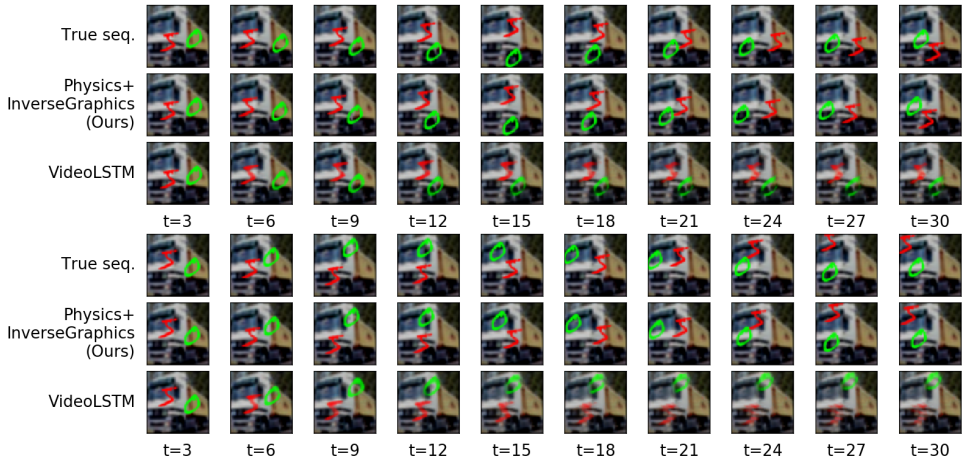
## 2-balls spring



### 2-balls bouncing




### 2-digits spring




# C NewtonianVAE ELBO derivation

We want to maximize the sequence marginal likelihood:

$$p(\mathbf{I}_{1:T}|\mathbf{u}_{1:T}) = \int p(\mathbf{I}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) \, \mathrm{d}\mathbf{x}_{1:T}. \tag{6.1}$$

We factorize the above terms as follows:

$$p(\mathbf{I}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod_t p(\mathbf{I}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \prod_t \int p(\mathbf{I}_t|\hat{\mathbf{x}}_t) p(\hat{\mathbf{x}}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}; \mathbf{v}_{t-1}) \, \mathrm{d}\hat{\mathbf{x}}_t \tag{6.2}$$

$$p(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) = \prod_t p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}; \mathbf{v}_{t-1}), \tag{6.3}$$

where $\mathbf{v}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$. Hence, $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}; \mathbf{v}_{t-1})$ depends on $\mathbf{x}_{t-2}$ through $\mathbf{v}_{t-1}$, but we will simply use $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}; \mathbf{v}_{t-1}) \equiv p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ for ease of readability. We use an approximate posterior factorized as:

$$q(\mathbf{x}_{1:T}|\mathbf{I}_{1:T}) = \prod_t q(\mathbf{x}_t|\mathbf{I}_t) \tag{6.4}$$

Using Jensen's inequality we can write (6.1) as:

$$\log p(\mathbf{I}_{1:T}|\mathbf{u}_{1:T}) = \tag{6.5}$$

$$= \log\Big( \int \frac{\prod_t q(\mathbf{x}_t|\mathbf{I}_t)}{\prod_t q(\mathbf{x}_t|\mathbf{I}_t)} \prod_t \int p(\mathbf{I}_t|\hat{\mathbf{x}}_t) p(\hat{\mathbf{x}}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \, \mathrm{d}\hat{\mathbf{x}}_t \prod_t p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \mathrm{d}\mathbf{x}_{1:T} \Big) \tag{6.6}$$

$$\geq \int \prod_t q(\mathbf{x}_t|\mathbf{I}_t) \left( \sum_t \log \left[ \int p(\mathbf{I}_t|\hat{\mathbf{x}}_t) p(\hat{\mathbf{x}}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \, \mathrm{d}\hat{\mathbf{x}}_t \right] + \sum_t \log \frac{p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})}{q(\mathbf{x}_t|\mathbf{I}_t)} \right) \mathrm{d}\mathbf{x}_{1:T} \tag{6.7}$$

$$= \sum_t \int q(\mathbf{x}_{t-1}|\mathbf{I}_{t-1}) q(\mathbf{x}_{t-2}|\mathbf{I}_{t-2}) \left( \log \left[ \int p(\mathbf{I}_t|\hat{\mathbf{x}}_t) p(\hat{\mathbf{x}}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \, \mathrm{d}\hat{\mathbf{x}}_t \right] \mathrm{d}\mathbf{x}_{t-1} + \right.$$
$$\left. \mathrm{KL}\left(q(\mathbf{x}_t|\mathbf{I}_t) \| p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\right) \right) \tag{6.8}$$

$$\geq \sum_t \mathbb{E}_{q(\mathbf{x}_{t-1}|\mathbf{I}_{t-1}) q(\mathbf{x}_{t-2}|\mathbf{I}_{t-2})} \Big( \mathbb{E}_{p(\hat{\mathbf{x}}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})} \log p(\mathbf{I}_t|\hat{\mathbf{x}}_t) + \mathrm{KL}\left(q(\mathbf{x}_t|\mathbf{I}_t) \| p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\right) \Big) \tag{6.9}$$

# D Simulated environment details

## D.1 Simulated point mass environment

The point mass environment is adapted from the `PointMass` environment from the `dm_control` library. The mass is linearly actuated in the 2D plane and its movement bounded by the edges of the frame. The simulator uses a time-step $\Delta t = 0.5$ and the $[x, y]$ forces are in the range $[-1, 1]^2$.

## D.2 Simulated reacher environment

The reacher-2D environment is a adapted from the `Reacher` environment and inspired by the simulated reacher task in Kipf et al. (2019). We limit the rotation of the shoulder joint to the $[-160, 160]$ range, and the wrist joint to $[0, 160]$. The simulator uses a time-step of $\Delta t = 0.1$ and the torques are in the range $[-1, 1]$. When generating random rollouts we sample shoulder and wrist angles in the whole range, and when generating demonstrations these angles are sampling according to `0.5+(np.random.rand()-0.5)` and `-np.pi+0.3+np.random.rand()*0.5` (in radians), respectively. A full 100-step demonstrations sequence is shown in Fig. 6.1.

To evaluate the trained control policies in the simulator, we compute a sparse reward as follows. When the distance between the end affector and the initial target is lower than 0.015 and the joint velocity is lower than 0.2, the agent earns a reward of 1. The targets must be reached in sequence, i.e., if the agent goes straight for the second target without stopping at the first target, the reward is still 0. The optimal agent will thus have a maximum reward of 3. We use 120- and 220-step rollouts when evaluating the noiseless and noisy settings, respectively.

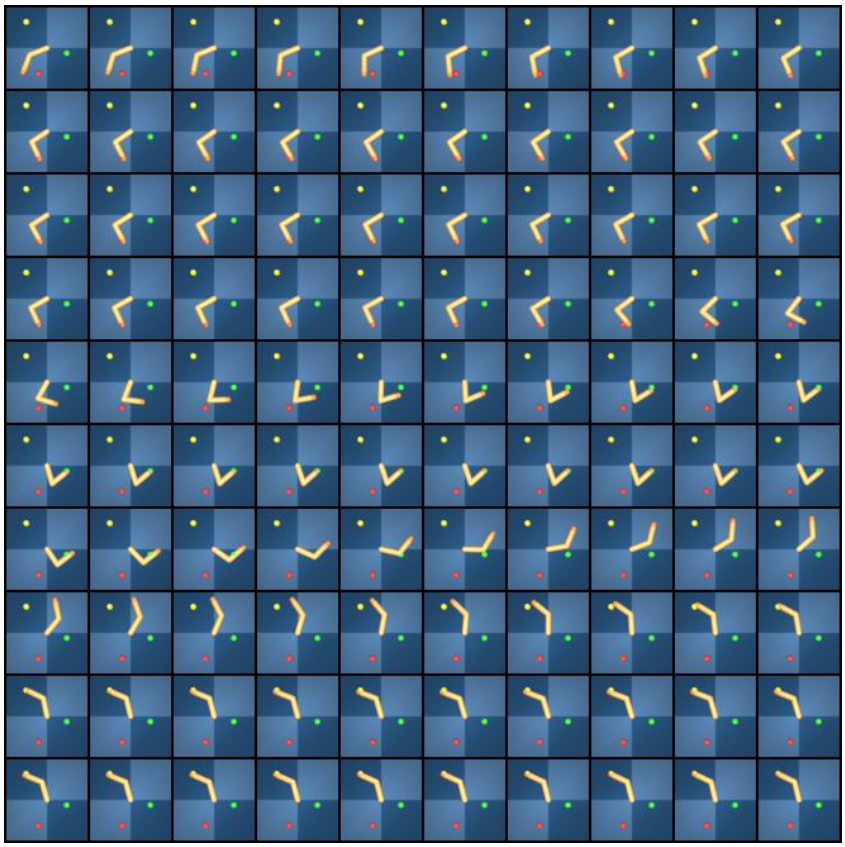


**Figure 6.1:** Full demonstration sequence for simulated reacher (progression left to right, top to bottom).

### D.3 PR2 robot arm

Real robot experiments were conducted using the left arm of a PR2 robot, with images recorded using a downward facing Kinect 2 camera mounted on the PR2 head. Arm motion demonstrations were obtained by pre-programming the robot to move to various objects in the scene using the MoveIt! motion planning library in the robot operating system (ROS). The robot arm is actuated using 8 torque commands (7 for the joints in the robot arm and one for the robot torso height), which were recorded alongside images.

After preprocessing the images (rescaling to $64 \times 64$ and cropping to the region of interest), we are left with 836 frames, which we split into 636 training, 100 validation, and 100 testing frames. Training used a batch size of 20 frames.

Due to the very small amount of training data available, we had to impose further constraints on the model to allow for correct learning. Firstly, the transition matrices were set to $A = 0, B = 0, C = 1$. Secondly, we added an additional regularization term to the latent space, $KL(q(\mathbf{x}|\mathbf{I})\|\mathcal{N}(0,1))$, to improve visualization of the goals (though this term was not necessary for obtaining correct sequence segmentations). Finally, we added a batch-wise entropy term in $\boldsymbol{\pi}(\mathbf{x})$ to encourage the use of all modes, as proposed by (Burke et al. 2019b):

$$\mathcal{L}^{\text{ENT}} = -\frac{1}{J}\sum_{j=1}^{J}\log\left(\frac{1}{T}\sum_{t=1}^{T}\pi_{j,t}\right). \tag{6.10}$$

# E Additional P-control trajectory comparisons for the NewtonianVAE model

Additional P-control trajectories for the point mass, reacher-2D, and fetch-3D systems are shown in Figs. 6.2, 6.3 and 6.4, respectively.

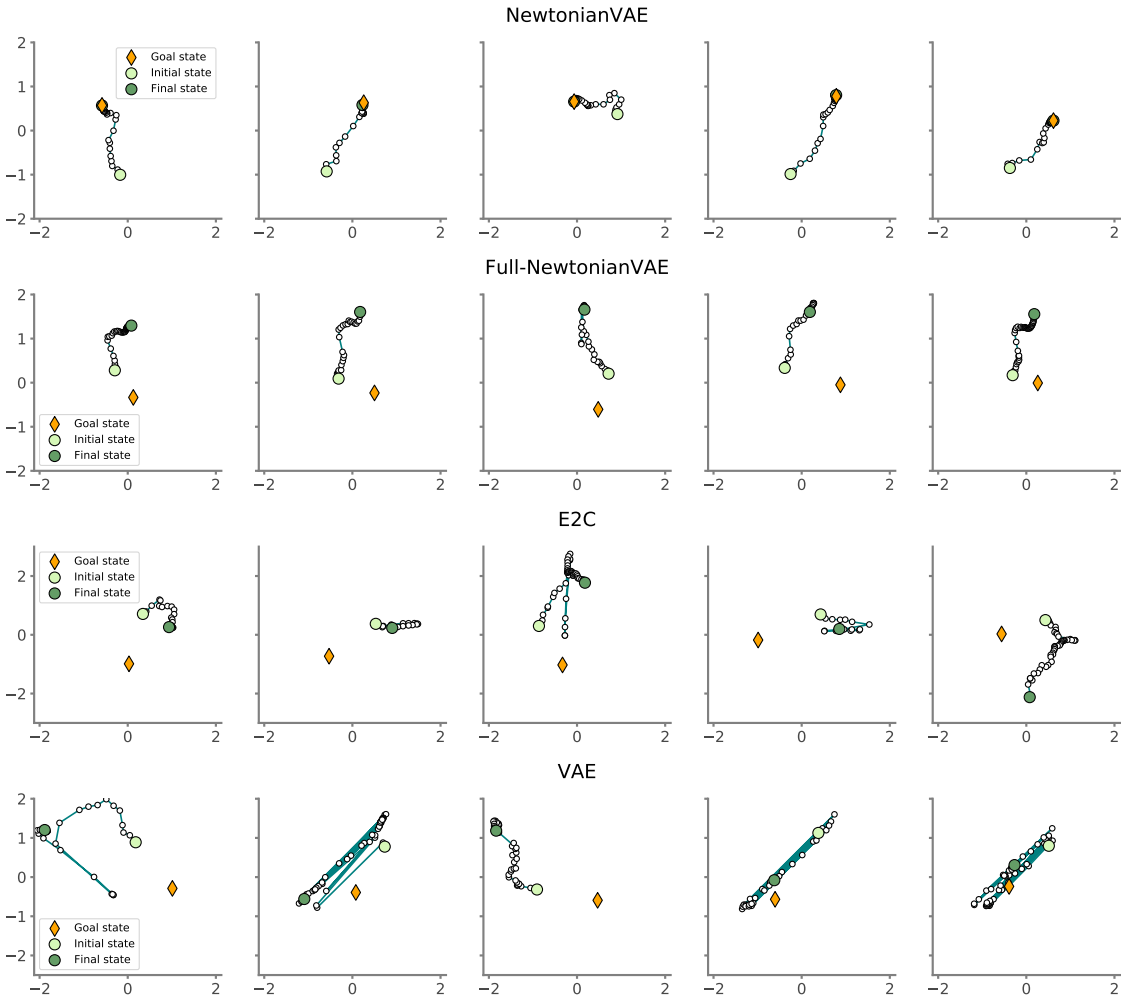



**Figure 6.2:** P-controllability in point mass system.

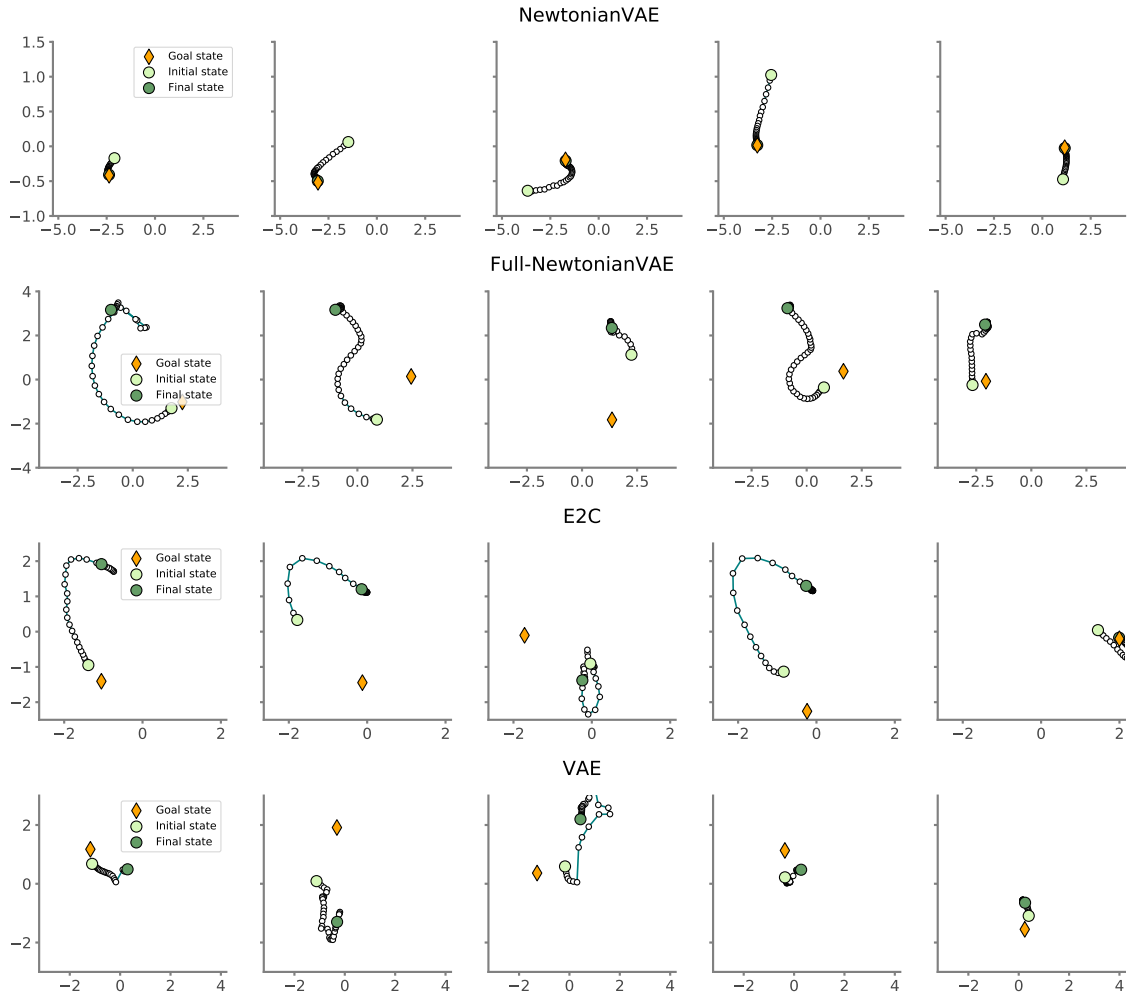



**Figure 6.3:** P-controllability in reacher system.

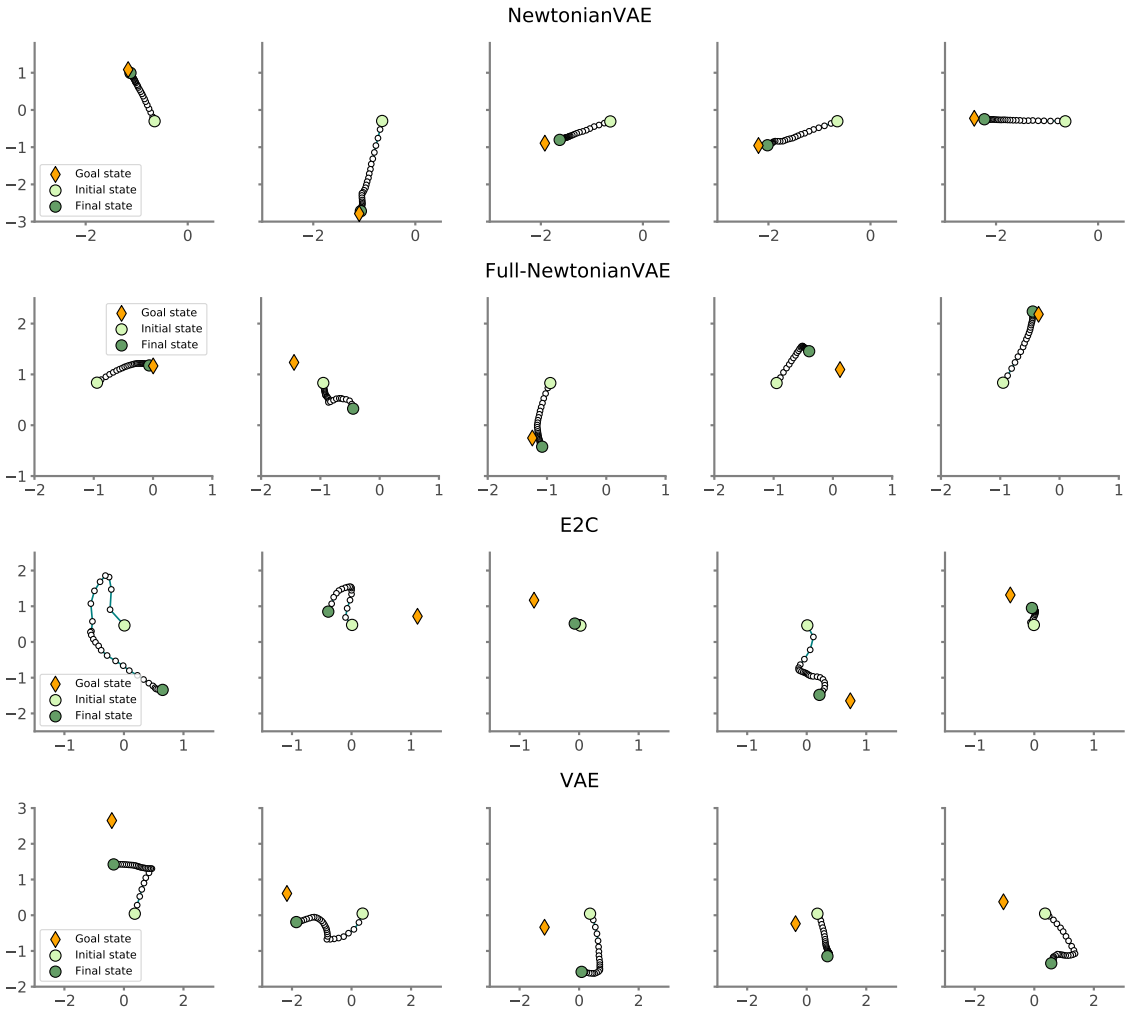



**Figure 6.4:** P-controllability in reacher system.