Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

forward together
sonke siya phambili
saam vorentoe

On the Limitations of Proportional Control from Pixels via Physical Latent Spaces

by Thomas Michael Marshall

Research report submitted for the
MSc in Machine Learning and Artificial Intelligence

Supervisor: Prof. Steve Kroon and Dr Michael Burke

December 2022

Abstract

Low-dimensional state representation learning lends itself to the development of techniques for robotic control from vision. Contemporary methods produce embedding spaces that require reinforcement learning policy search or model predictive control for downstream control tasks. To simplify the task of control, we consider a formulation that designs the embedding for compatibility with established control theory [JBH21]. We reproduce key results obtained by the authors of the original paper and identify various ways in which the results fail to extend to more complex scenarios. We systematically investigate the failure modes and limitations of this model. Our results suggest that the principal failure mode occurs as a manifestation of the curse of dimensionality, due to the interdependency of latent variables arising when inferring the joint positions of articulated robots from vision.

Contents

1. Introduction

This study considers vision-based control of robots; to exert control with visual information as the primary input mode. In subsection 1.1 we provide a brief introduction to the field of robotics and the challenges that lie within. subsection 1.2 describes the key elements of vision-based robotics. We outline our contributions in subsection 1.3 and provide a brief overview of the structure of the rest of this document in subsection 1.4.

1.1. Robotics

Robotics was initially developed to create tools that would increase human productivity in difficult or monotonous activities. Since then, robotics has advanced to the point that it can now compete with and even outperform humans in a variety of tasks. Prior to the deep learning revolution, robotics relied upon fragile hand-crafted algorithms that were not adaptable and were tedious to design. The capabilities of modern robots have been increasing rapidly ever since.

Three essential areas that need to be improved [BK19] to move closer to the ideal of intelligent general-purpose robots are *perception*, *control*, and *planning*, these continue to be some of the most challenging issues in robotics and broadly depend on one another: to control, one must be able to perceive. To execute a plan, one must be able to control a robot. Perception can take many forms, one of which is processing visual information [SKK22] to infer or predict something about a robot and its environment.

1.2. Vision-based Robotics

In vision-based robotics, a robot's state is inferred from visual data and utilised in conjunction with a control strategy to carry out intended activities. After a specific action is taken, we would like to be able to anticipate what state a robot will be in. Consequently, there are three key elements of vision-based robot control:

1. State representation learning
2. Dynamical modelling
3. Control strategies

These three tasks are often decoupled and determined separately [HLF+18, WSBR15]. In addition to learning useful dynamical models, separating these elements requires us to learn action policies through reinforcement learning [DFR15], or use model predictive control [Ric05] strategies. The motivation underlying this approach is, since learned dynamical systems are task-independent, they can transfer towards policy search on a variety of tasks in the environment.

In [JBH21], the authors argue that decoupling these elements places unnecessary complexity and computational burden on control. Instead, they propose considering them jointly, by introducing additional inductive priors that simplify downstream control.

1.2.1. State Representation Learning

We consider monocular red-green-blue (RGB) visual observation data. Even at modest resolutions, a robot's perceived pixel space image I is extraordinarily high-dimensional. As such, the dynamical system describing the change in pixels between image frames is highly nonlinear and excruciatingly complex; learning dynamical models and control policies from this representation is a Herculean task. We require access to a suitable dynamical system in order to forecast the next state after a suggested action if we wish to perform any kind of control from pixels. To simplify this problem, we consider a low-dimensional state representation [KSBvdS17, WSBR15, ZVS+18] of the image I, as the encoding $\boldsymbol{x}(I)$.

1.2.2. Dynamical Models

Learning the dynamical model of a robotic system can be thought of as learning a world model, expressed in terms of a low-dimensional representation of the world a robot is in. In [HS18], the authors propose the unsupervised learning of world models in which an agent can then use this world model to learn action policies, without interacting with the real environment. The authors of [GMR16, DFR15] also follow this strategy.

Ideally, we would like to find the low-dimensional representation and dynamical model via unsupervised learning, in such a way that it facilitates direct control, instead of having to learn a policy in that space. This amounts to simply taking a step in the direction of the goal state, as opposed to running numerous trials for policy optimisation.

1.2.3. Statistical Machine Learning and Control

We would like to be able to guide the current state of a robotic system to some desired state over iterations. Learning action policies often involve using reinforcement learning, which is computationally expensive and time-consuming. It is preferable to utilise feedback control using tried-and-true proportional-integral-derivative (PID) controllers [SSZ10] since they are straightforward, dependable, and well-known. Standard proprioceptive joint-based torque control already uses PID control to guide individual joint angles towards desired values.

Machine learning and control theory are two fields that have recently collided. With the established field of control theory comes a wealth of tried-and-true techniques for the design and study of controllers and systems. State inference from pixels is a well-established application of statistical machine learning [WSBR15, HS18, KSBvdS17, BEPVDS19, JBH21].

It is pertinent that the modelling should be probabilistic, to facilitate principled quantification of uncertainty. This is paramount for safely and effectively reasoning under partial knowledge in unconstrained environments [BSB+04].

When learning low-dimensional representations and dynamical systems, control theory is usually not taken into account; as a result, model predictive control or reinforcement learning policy search is often necessitated. If one could infer every unique state of a robot from vision as the set of underlying joint angles, then, using only the error in state one could use control theory to guide a robotic system towards the goal state.

The main focus of this work is the joint state representation and dynamical modelling from pixels using a variational Bayesian approach. Moreover, this approach considers extensive inductive priors on the low-dimensional representation, to render it amenable to the direct application of control theory. This study builds on the work done in [JBH21], where the authors propose a variational recurrent neural network, the Newtonian variational autoencoder (NVAE), with suitable inductive priors such that its state inference can be used for simple feedback control.

1.3. Objectives and Contributions

With a significant emphasis on extending[1] the work of [JBH21], the main objective of this study is to investigate the field of vision-based robotics. First, we set out to reproduce the most important findings from the aforementioned research. We then build on their work by defining the conditions under which this paradigm excels and defining its boundaries.

The following contributions are made in this work:

- We successfully reproduce key results from [JBH21], starting with the original authors' research repository.
- We introduce the concept of latent gravity and show that the NVAE produces a *PI-controllable* latent space when trained on a robot subject to gravity, as opposed to forming a *P-controllable* latent space as before in the cases considered by the original paper.
- We describe and systematically investigate the challenges encountered when applying the NVAE to more complex scenarios than in the original paper.
- From these investigations, we conclude that the principal failure occurs as a manifestation of the curse of dimensionality due to interdependent latent variables inherent in inferring the joint positions of articulated robots from vision.

[1] This research project was originally titled "Enriching the NewtonianVAE with Latent Score-based Generative Modelling". After some examination, we determined that it would not make sense to integrate latent score-based generative modelling in the framework of variational recurrent neural networks after all. Thus we pivoted to analysing the limitations of the NVAE model, as to inform the path of future work.

1.4. Thesis Structure

A basic overview of feedback control systems and controllers is given in Chapter 2. We define feedback control and discuss its applications to robotic control problems, before providing a summary of PID control essentials.

Chapter 3 follows the development of the theory of the NVAE[2] model from [JBH21]. We begin with the variational autoencoder and extend it to sequence modelling to obtain the variational recurrent neural network. The NVAE is derived by incorporating suitable inductive priors into the variational recurrent neural network.

Chapter 4 describes the reproduction of key results from the original paper [JBH21].

In Chapter 5, we systematically investigate the limitations and failure modes of NVAE. Our investigations consider the influence of gravity, the number of joints and the influence of dynamic perspective.

Chapter 6 summarises the aims and findings of the research project and discusses possible future directions for this area of research.

[2] In this document NVAE refers to the Newtonian variational autoencoder from [JBH21]. It does not refer to the nouveau variational autoencoder from [VK20].

2. Control Systems

2.1. Feedback Control Systems

Control systems regulate the behaviour of systems using control loops. Typical closed-loop[3] feedback control systems [SSZ10] are used to control processes by comparing the process variable that is to be regulated to the setpoint (the desired value). Figure 1 illustrates the idea of closed-loop control.

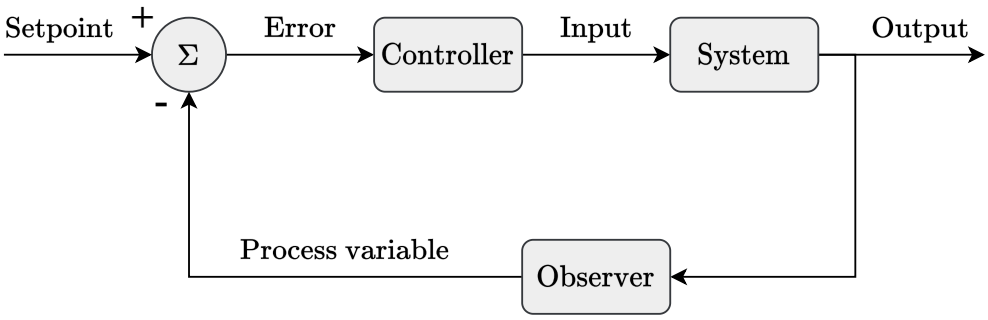


Figure 1: A block diagram of a simple closed-loop feedback control system. The continuous feedback loop aims to drive the error to zero, by guiding the process variable towards the setpoint.

In real robotic control scenarios where we have access to proprioceptive information, it is typical to use simple closed-loop control to direct our process variables $y(t)$ towards their setpoints $r(t)$ using the error $e(t)$. The process variable (also called the state) can be a scalar or vector quantity. When the state is a vector of $\mathbb{R}^D$, the error will also be a vector of $\mathbb{R}^D$; depending on the scenario this can be considered as D parallel control loops (there might be some correlation between state variables depending on the situation).

The theory underlying closed-loop feedback control largely considers linear time-invariant systems. Applying this to nonlinear time-varying systems typically requires linearising the dynamics around an operating point.

2.2. The State Observer

The state observer is possibly the most critical part of any control system. This element provides an estimate of the internal state of a system, using the available outputs produced by the system. Feedback control can only succeed when using an accurate state observer.

[3] Closed-loop as opposed to open-loop control, where the control signal is independent of the process variable, making this non-feedback control. An intuitive example of this is a regular toaster.

2.2.1. Observability and Controllability

In control theory, the notion of observability represents how well the internal state of a system can be inferred from its externally measurable outputs. In machine learning, our objective is often to learn models that perform this inference [LDRGF18]. It follows that in the union of control and machine learning, observability is of great consequence: the internal state information must be somehow encoded in the output signal for the system to be adequately observable.

Controllability is closely related to observability. The concept of controllability denotes the ability to predictably guide the robot through the entire configuration space of a system using permissible actions; this means any valid internal state should be reachable. For a system to be controllable, it first needs to be sufficiently observable.

2.3. Proportional Control

One of the simplest[4] forms of feedback control is proportional control (P-control), where the control signal $u(t)$ is a proportion K_p (the proportional gain) of the difference between the process variable $y(t)$ and the setpoint $r(t)$:

$$u(t) = K_\mathrm{p}\, e(t), \text{ where } e(t) = r(t) - y(t)\,. \tag{1}$$

If one can control a system by applying P-control only, then it is said the system is *proportionally controllable*.

2.4. Proportional-Integral-Derivative Control

The typical controller used in feedback control is the proportional-integral-derivative (PID) controller:

$$u(t) = K_\mathrm{p}\, e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}\, e(t)}{\mathrm{d}t}\,. \tag{2}$$

The gain terms $K_\mathrm{p}, K_\mathrm{i}$ and K_d contribute distinctively to the response of the controller: the proportional term guides the process variable directly towards its setpoint; the derivative term introduces inertia in the actuation, leading to smoother control trajectories; and the integral term is generally used to eliminate steady-state error.

The use of PID control is widespread and considered to be the gold standard for simple feedback control because it incorporates the proportional, derivative and integral modes. Additionally, it is simple to implement and tune [ZN42].

This chapter served as a brief introduction to control theory. We now redirect our focus to considering statistical machine learning to infer, from vision, the internal state of the robot we would like to control.

[4]The simplest form of feedback control is bang bang control. This strategy switches the control signal abruptly to its maximum or minimum. This typically is not useful in robot control settings.

3. Variational Vision-based State Inference

In this chapter, we develop a model that allows us to learn low-dimensional representations of images of robots that we can directly use for control. The core of variational methods for latent state representation learning is the variational autoencoder [KW13], which allows us to perform efficient approximate inference and learning in the presence of continuous latent variables with intractable posteriors. The variational autoencoder (VAE) extends to sequence modelling through the variational recurrent neural network [CKD+15], considering the temporal evolution of latent variables.

Additionally, we also aim to incorporate inductive priors to guide the formation of the low-dimensional latent representation, such that it can be used for simple feedback control [JBH21]. This poses two requirements: the latent space manifold must be smooth and disentangled.[5]

3.1. Variational Autoencoder

The variational autoencoder [KW13] is a popular framework for performing efficient approximate inference and learning in directed probabilistic models with continuous latent variables and intractable posterior distributions. This approach optimises a variational Bayesian approximation to the intractable posterior, without the need for analytic forms of the expectations. We consider the high-dimensional observed variable $\boldsymbol{x}$, as influenced by the low-dimensional latent variable $\boldsymbol{z}$ (which represents the underlying configuration of the robot).

3.1.1. Variational Bayes

We consider an independent and identically distributed (i.i.d.) data set of the continuous variable $\boldsymbol{x}$ assumed to be influenced by a hypothetical continuous latent variable $\boldsymbol{z}$ as in Figure 2; we will continue to use this notation throughout. The generative process is specified as follows: $\boldsymbol{z}$ is drawn from a prior distribution $p_\theta(\boldsymbol{z})$, then $\boldsymbol{x}$ is generated from a conditional distribution $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$. We assume that both the prior and likelihood distributions are from parametric families and that they are differentiable. It is not necessarily assumed that any marginals or posteriors are tractable. Typically these distributions are chosen to be isotropic Gaussians.

The posterior inference model $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ is called the probabilistic *encoder*, and $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ the probabilistic *decoder*; we will continue to use these terms in this document. The parameters ϕ of the variational approximation $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ to the intractable true posterior is learned jointly with the generative model parameters θ. This amortised variational

[5] Disentangled here refers to dimensions of the latent vector. Each element should fully represent only the state information of one angle or coordinate.

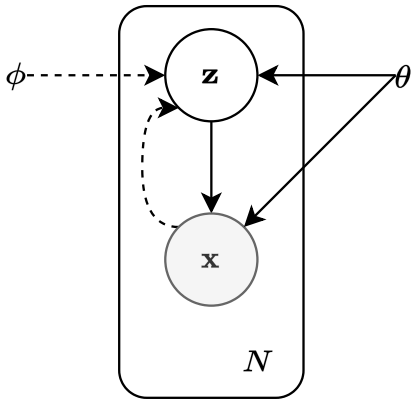


Figure 2: The graphical model underlying the variational autoencoder [KW13]. The solid lines denote the generative process, from the latent variable vector $\boldsymbol{z}$ to $\boldsymbol{x}$. The dotted lines represent the variational approximation $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ to the intractable posterior $p_\theta(\boldsymbol{z} \mid \boldsymbol{x})$.

Bayesian approach allows us to infer $\boldsymbol{z}$ from $\boldsymbol{x}$ cheaply, using the surrogate posterior distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$.

3.1.2. The Variational Lower Bound

The objective in this generative model is to maximise the aggregate marginal likelihood over the data set $\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}_i)$. To maximise this, we must first find the marginal:

$$\log p_\theta(\boldsymbol{x}) = \log \int p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p_\theta(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} \,. \tag{3}$$

We multiply with the approximate posterior distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ inside the integral without changing its value, and apply the definition of expectations:

$$\begin{aligned} \log p_\theta(\boldsymbol{x}) &= \log \int p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p_\theta(\boldsymbol{z}) \frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \, \mathrm{d}\boldsymbol{z} \\ &= \log \left(\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} p_\theta(\boldsymbol{z}) \frac{p_\theta(\boldsymbol{x} \mid \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \right) . \end{aligned} \tag{4}$$

We apply Jensen's inequality, the log identities and the definition of Kullback-Leiber (KL) divergence:

$$\begin{aligned} \log p_\theta(\boldsymbol{x}) &\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log p_\theta(\boldsymbol{z}) - \log q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \right] \\ \mathcal{L}(\theta, \phi; \boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \right] - \mathrm{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) || p_\theta(\boldsymbol{z})) \,. \end{aligned} \tag{5}$$

The model is trained by maximising the lower bound in Equation 5 w.r.t. the variational distribution parameters ϕ, and the generative model parameters θ. This cannot be done directly, so one must use Monte Carlo (MC) gradient estimation. However, the naïve estimator has high variance [PBJ12], making it unsuitable.

3.1.3. Reparameteristion Trick

The authors of [KW13] propose another MC gradient estimator where they reparameterise the latent random variable $\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ using a differentiable[6] transformation $g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x})$ of an auxiliary noise variable $\boldsymbol{\epsilon}$:

$$\boldsymbol{z} = g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}) \text{ with } \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \,. \tag{6}$$

This allows us to make differentiable MC estimates of expectations w.r.t. the approximate posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ directly, as in Equation 5. Generally, only one MC sample is used to approximate this expectation. The reparameterised expectation is then:

$$\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log p(\boldsymbol{x} \mid \boldsymbol{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\log p(\boldsymbol{x} \mid \boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon})\right], \, \boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\} \,. \tag{7}$$

Another way of looking at the reparameterisation trick is that it allows us to bypass the stochastic node in the computational graph since we cannot differentiate through it because sampling is not differentiable.

3.1.4. Autoencoding Variational Bayes

The vanilla VAE builds on this formulation with a few modelling assumptions. The prior over the latent variable $\boldsymbol{z}$ is taken to be an isotropic Gaussian $p_\theta(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$. We let the probabilistic decoder $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ be a multivariate Gaussian parameterised by neural network with parameters θ:

$$p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu_\theta}, \boldsymbol{\sigma_\theta}^2) \,. \tag{8}$$

Since the true posterior is intractable, we take the approximate posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ to be a multivariate Gaussian with diagonal covariance, parameterised by the encoder neural network with parameters ϕ,

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu_x}, \boldsymbol{\sigma_x}^2 \boldsymbol{x}) \,. \tag{9}$$

Once the approximate posterior distribution parameters are obtained, we draw $\boldsymbol{z}$ from it using the reparameterisation trick, after which it is passed to the decoder:

$$\boldsymbol{z} = g_\phi(\boldsymbol{x}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \,. \tag{10}$$

In this setting, the KL divergence term in (5) can be written in closed form [KW13],

[6] This step requires that the latent variables are continuous.

since both the prior and the approximate posterior are Gaussian:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \boldsymbol{x}) = \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left(\boldsymbol{\sigma}_j^2\right) - \boldsymbol{\mu}_j^2 - \boldsymbol{\sigma}_j^2\right) \\ + \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \,,\end{aligned} \tag{11}$$

where J is the dimensionality of the latent $\boldsymbol{z}$.

The variational lower bound objective function consists of two terms, the KL divergence and the probabilistic decoder cross-entropy. The KL divergence to the prior penalises the model for encodings that deviate from the (isotropic normal) prior. The decoder cross-entropy is also called the *reconstruction error* and is often calculated simply as mean squared error.[7]

The training of a VAE exhibits a trade-off between these two terms because the approximate posterior is conditioned on the observation. This causes a failure mode of the VAE, where the KL divergence dominates the objective, and the posterior becomes very close to the prior at the expense of the reconstruction.[8] This model is trained using Algorithm 1.

Algorithm 1 Autoencoding variational Bayes [KW13]

$\phi, \theta \leftarrow$ Initialise parameters.
repeat
  $\boldsymbol{x} \leftarrow$ Draw random sample from data set.
  $\boldsymbol{\epsilon} \leftarrow$ Draw random sample from noise distribution $p(\boldsymbol{\epsilon})$.
  $\boldsymbol{g} \leftarrow \nabla_{\theta,\phi} \mathcal{L}(\theta, \phi; \boldsymbol{x}, \boldsymbol{\epsilon})$ Calculate the gradients of the estimator.
  $\theta, \phi \leftarrow$ Update parameters using stochastic gradient descent (SGD).
until Convergence of parameters θ, ϕ.

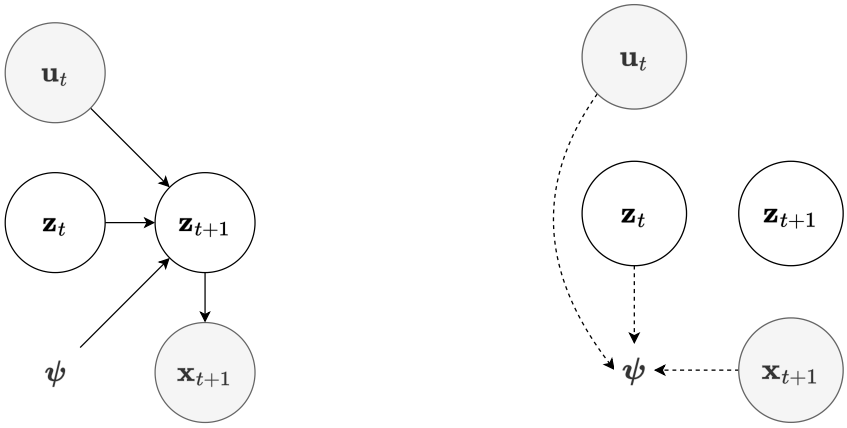
The VAE allows us to formulate a generative model for autoencoding, and specify a prior over the latent space of that autoencoder. This Bayesian formulation opens the door to incorporating further priors on the low-dimensional latent representations that we learn.

3.2. Variational Recurrent Neural Networks

When the data under consideration are sequences, the variational autoencoder is not directly applicable; it must be extended to sequence modelling. The variational recurrent neural network (VRNN) as proposed in [CKD+15], is a recurrent neural network with a variational autoencoder included at every timestep. A notable addition is that the relationship between latent variables across timesteps is also modelled as in Figure 3.

From this point on $\boldsymbol{u}$ denotes the actuation (in an actuated system). We have a sequence of T images $\boldsymbol{x}_{1:T}$, actuations $\boldsymbol{u}_{1:T}$, and the corresponding latent representations of the

[7] This corresponds to assuming a Gaussian generative model.
[8] This is known as posterior collapse because the posterior distribution collapses to the prior distribution.



(a) Forward graphical model. (b) Inference.

Figure 3: A generalised version of the graphical model in [KSBvdS17] for the VRNN state transition. The next latent state $\boldsymbol{z}_{t+1}$ depends on the previous latent state $\boldsymbol{z}_t$, the control input $\boldsymbol{u}_t$ and the transition model parameters $\boldsymbol{\psi}$.

images $\boldsymbol{z}_{1:T}$. The goal of training this model is to maximise the marginal data likelihood. The general form of the marginal data likelihood is given by:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) = \int p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) \, \mathrm{d}\boldsymbol{z}_{1:T} \,. \tag{12}$$

The two terms in the integral of Equation 12 are factorised as follows under a Markov assumption on latent states:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t \mid \boldsymbol{z}_t) \tag{13}$$

and

$$p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}) \,. \tag{14}$$

The approximate posterior of the sequence is given by:

$$q(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}) \,. \tag{15}$$

3.2.1. Variational Lower Bound

The model is trained by maximising the variational lower bound using the procedure described in Algorithm 1. The objective function is the timestep-wise variational lower

bound[9] given by:

$$\log p_\theta(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) \geq \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\boldsymbol{z}_t|\boldsymbol{x}_t,\boldsymbol{z}_{t-1},\boldsymbol{u}_{t-1})} \underbrace{\log p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_t)}_{\text{reconstruction}} - \underbrace{\mathrm{KL}(q_\phi(\boldsymbol{z}_t \mid \boldsymbol{x}_t)||p_\theta(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}))}_{\text{prior divergence penalty}} . \tag{16}$$

The VRNN model introduces the KL divergence of the approximate posterior to the transition prior $p(\boldsymbol{z}_{t+1} \mid \boldsymbol{z}_t, \boldsymbol{u}_t)$. This allows us to learn embeddings that conform to some specified latent dynamical system.

3.3. Latent Dynamical Systems

A dynamical system describes the state change $\boldsymbol{z}_t$ that occurs because of some action taken $\boldsymbol{u}_t$. Considering the dynamics of the latent state $\boldsymbol{z}_t$, as in [BEPVDS19], we impose a linear Gaussian dynamical system:

$$\boldsymbol{z}_{t+1} = \boldsymbol{A}(\boldsymbol{z}_t) \cdot \boldsymbol{z}_t + \boldsymbol{B}(\boldsymbol{z}_t) \cdot \boldsymbol{u}_t + \boldsymbol{C}(\boldsymbol{z}_t) + \varepsilon_t \text{ with } \varepsilon_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}^2 \boldsymbol{x}) \tag{17}$$

$$p(\boldsymbol{z}_{t+1} \mid \boldsymbol{z}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{z}_{t+1}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \boldsymbol{x}) . \tag{18}$$

To apply this to the context of stochastic gradient variational Bayes, this means the stochastic transition prior must be reparameterised as well in order to backpropagate through another stochastic node.

The authors of [WSBR15] enforce a locally linear latent dynamical system, as well as a variational autoencoder to map to and from the latent space. However, they do not use a sequence modelling approach and make temporal i.i.d. assumptions for this reason. This attempts to compensate for the loss of second-order information, due to modelling on the level of position (this is what NVAE addresses)

3.4. Newtonian Variational Autoencoder

The Newtonian variational autoencoder proposed by [JBH21] addresses the loss of second-order information by introducing linear second-order dynamics (modelling on the level of acceleration) in the latent space and constraining it such that the dimensions remain disentangled. This aids in interpretability and induces a smooth latent space in which we can successfully use proportional control.

This model explicitly treats position and velocity as two separate latent variables $\boldsymbol{z}$ and $\boldsymbol{v}$. For an actuated rigid body robot of D degrees-of-freedom and state vector $\boldsymbol{z} \in \mathbb{R}^D$, we

[9] The full derivation of this lower bound is available in Appendix 8 subsection 8.1.

model second-order latent dynamics:

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \boldsymbol{A}(\boldsymbol{z}, \boldsymbol{u}) \cdot \boldsymbol{z} + \boldsymbol{B}(\boldsymbol{z}, \boldsymbol{u}) \cdot \boldsymbol{v} + \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{u}) \cdot \boldsymbol{u}\,. \tag{19}$$

Only $\boldsymbol{z}$ is used as the stochastic state variable that is inferred by the approximate posterior $\boldsymbol{z}_t \sim q_\phi(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$. Then $\boldsymbol{v}$ is deterministic, naturally defined as:[10]

$$\boldsymbol{v}_t = (\boldsymbol{z}_t - \boldsymbol{z}_{t-1})/\Delta t\,, \tag{20}$$

where $\boldsymbol{z}_t \sim q_\phi(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ and $\boldsymbol{z}_{t-1} \sim q_\phi(\boldsymbol{z}_{t-1} \mid \boldsymbol{x}_{t-1})$.

Returning to the marginal data likelihood of the variational recurrent neural network in Equation 12, the generative model is once again factorised under a Markov assumption, with auxiliary latent velocity:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t \mid \boldsymbol{z}_t) \tag{21}$$

and

$$p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}; \boldsymbol{v}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_t)\,. \tag{22}$$

The transition prior is chosen to be a Gaussian distributed prediction from the linear dynamical system:

$$p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_t) = \mathcal{N}(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1} + \Delta t \cdot \boldsymbol{v}_t, \sigma^2) \tag{23}$$

$$\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \Delta t \cdot (\boldsymbol{A} \cdot \boldsymbol{z}_{t-1} + \boldsymbol{B} \cdot \boldsymbol{v}_{t-1} + \boldsymbol{C} \cdot \boldsymbol{u}_{t-1})\,, \tag{24}$$

with $[\boldsymbol{A}, \log(-\boldsymbol{B}), \log(\boldsymbol{C})] = \mathrm{diag}(f_\psi(\boldsymbol{z}_t, \boldsymbol{u}_t))$, and f_ψ is a neural network with parameters ψ. The following additional restrictions are applied:

- **Diagonal transition matrices** $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$ encourages correct coordinate relations because linear combinations of dimensions are eliminated.
- **B is strictly negative** to induce an inertial effect as per the intuition of velocity.
- **C is strictly positive** to ensure correct directional relations between the action $\boldsymbol{u}$ and the position $\boldsymbol{z}$.

The distributions $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_t)$ and $q_\phi(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$ are diagonal Gaussians parameterised by neural networks.

[10] This Δt is known and constant for every image frame.

3.4.1. Variational Lower Bound

This model is also trained by maximising the marginal likelihood of the data. The variational lower bound[11] of this model is given by:

$$\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) \geq \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{t-1})q(\boldsymbol{z}_{t-2}|\boldsymbol{x}_{t-2})} \Bigg(\underbrace{\mathbb{E}_{p(\hat{\boldsymbol{z}}_t|\boldsymbol{z}_{t-1},\boldsymbol{u}_{t-1})} \log(\boldsymbol{x}_t \mid \hat{\boldsymbol{z}}_t)}_{\text{predicted present reconstruction}} - \underbrace{\mathrm{KL}(q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)||p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1})}_{\text{transition prior divergence}} \Bigg). \tag{25}$$

It is noteworthy that this variational lower bound specifies reconstruction based on the predicted present state $\hat{\boldsymbol{z}}_t$ as per the dynamical model. This is known to place a great emphasis on the predictive capability of the dynamical model through the transition prior [HLF+18, KSBvdS17].

In this chapter, we followed the development of the theory underlying the NVAE model. In the following chapter, we turn our attention to reproducing key results from [JBH21].

[11] The variational lower bound as given in [JBH21] is incorrect. The reconstruction term should have a log, and the KL divergence term should be negative. The corrected derivation is available in Appendix 8 subsection 8.2.

4. Reproduction of Key Results

In the paper [JBH21], the authors show the development of controllable latent spaces using 2 degrees-of-freedom (DOF) robots in the two-dimensional plane, not subject to gravity. These environments, as well as how we set them up, are described in Appendix 7. We outline the reproduction of the key results obtained in the paper below.

We were given access to the research repository of the authors of [JBH21]. From this, we regenerated the datasets using the relevant environments and wrote training code for the NVAE model. We experienced some initial difficulty in reproducing any results; we consulted with the first author of [JBH21], who advised us about the practicalities and implementation details of this model. In particular, the model is sensitive to random seed and highly susceptible to posterior collapse. He advised that the results in the original paper were obtained by incrementing the timestep of the latent state prediction by one; this is discussed below.

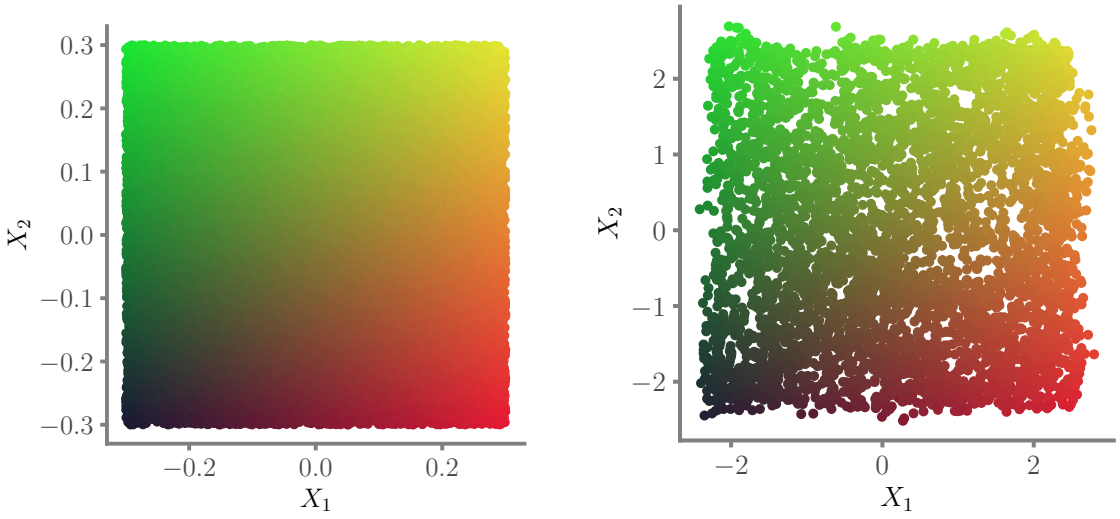
4.1. Pointmass

The pointmass environment (see Appendix 7) consists of a robot that can move around in a two-dimensional space, not subject to gravity. The actions recorded in the training set are normalised to be centred around zero in the range $[-1, 1]$. Figure 4 shows the latent space as well as a P-control trajectory for this environment.

It is worth elaborating on the latent space visualisation in Figure 4 as similar plots will occur throughout this thesis. Consider the true space (the manifold of valid configuration states of the robot) as a plane with an arbitrary smooth colour gradient across, such that each point has a unique colour. Images representing points in this true space are encoded to latent space using the encoder of the NVAE, while the colour of the particular point is preserved. This shows us the smoothness of the latent manifold and its relative correspondence to the true space. [12]

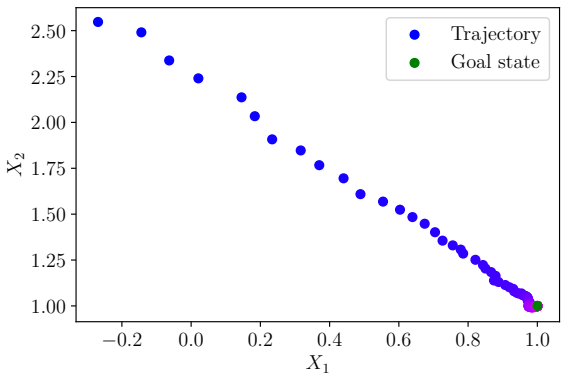
This environment was particularly susceptible to posterior collapse. We turned to the standard way to work around this: annealing the KL divergence term over iterations, such that it starts small and gradually increases in weight. This gives the encoder network a chance to learn to reconstruct well before the latent space is smoothed by the KL term. The authors of [ZSB+20] suggest implementing batch normalisation in the encoder of a VAE may help to prevent posterior collapse. We did not find that this is true in this case.

[12] For a more quantitative metric of evaluating the suitability, consider the KL divergence to the transition prior as in 25.



(a) True space.

(b) Latent space.



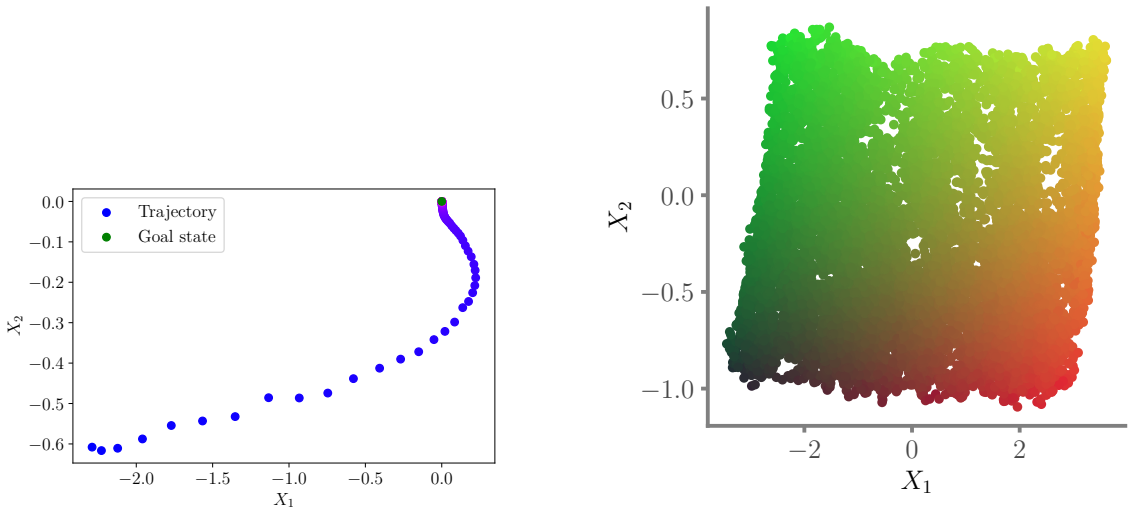
(c) P-control trajectory.

Figure 4: The true space, latent space and a P-control trajectory for the pointmass2D environment. The latent space was obtained by encoding test points that were not included in the training data. This procedure is described further below. The P-control trajectory converges on the goal state, indicating the desired *P-controllability*.

4.2. Reacher

The reacher (see Appendix 7) environment consists of a robot that can move around in a two-dimensional space, not subject to gravity. The actions recorded in the training set are normalised to be centred around zero in the range $[-1, 1]$.

The variational lower bound for the NVAE (Equation 25) considers prediction and reconstruction of the present latent state as estimated by the dynamical model. In the Appendix of [JBH21], the authors specify that they increase this by one timestep.[13] Figure 5 shows the latent space as well as a P-control trajectory for this environment.



(a) P-control trajectory.

(b) Latent space.

Figure 5: The latent space and P-control trajectory for the reacher2D environment. The latent state manifold was obtained in the same way as in Figure 4. The P-control trajectory converges on the goal state, indicating the desired *P-controllability*.

This system has more complex dynamics than the pointmass, because the mapping from the action to state space is highly nonlinear since we are modelling the state as the angle configuration of the joints.

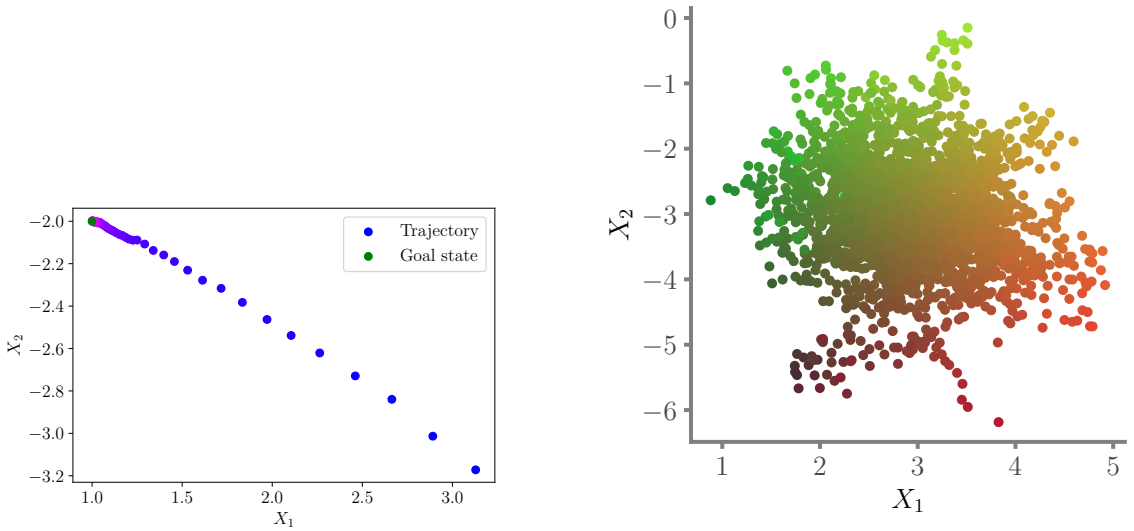
Training with the variational lower bound as in Equation 25 results in a latent space of separate locally smooth areas. The remedy for this is to increase the state prediction timestep by one in the objective function. Our intuition is that this forces the latent space to extend the range of any local smoothness, such that one hopefully ultimately gets global smoothness. This strategy proves to be very effective in this case. This environment was not found to be susceptible to posterior collapse, no KL annealing was necessary.

[13] In practice they reconstruct one timestep in the future. This renders the variational lower bound mathematically incorrect, but still provides a valid objective for gradient-based optimisation. This choice is also helpful insofar as forcing the model to learn an effective latent dynamical model.

4.3. Fetch

The fetch robot (see Appendix 7) environment consists of a 4-DOF[14] robot that can move around in a three-dimensional space, not subject to gravity. The actions recorded in the training set are normalised to be centred around zero in the range $[-1, 1]$. It is important to note that the state of this robot is modelled as the coordinates of the endpoint effector, not the joint angles. This means the nonlinearity in this situation is already resolved via the inverse kinematics in the simulator, which maps the endpoint effector actuation to joint rotations. Figure 6 shows the latent space as well as a P-control trajectory for this environment.

[14]The first three DOF refer to the x, y and z coordinates of the endpoint effector. The last DOF is the state of the gripper tongs.



(a) P-control trajectory.

(b) Latent space of first two latent coordinates.

Figure 6: The first two coordinates of the latent state manifold are shown. The plot was obtained in the same way as in Figure 4. The P-control trajectory converges on the goal state, indicating the desired *P-controllability*.

In this chapter, we reproduced key results from [JBH21]. We found that the embeddings learned by the NVAE can successfully be used to exert convergent P-control. In the next chapter, we identify and challenge the particular set of boundaries in which these original experiments were performed.

5. Analysis of Limitations

In the following chapter, we systematically investigate the limitations and failure modes of NVAE. We consider the influence of gravity, the number of joints, the joint dependencies of the robot, and the influence of dynamic perspective. These are all considerations in real robotic systems. Additional information and visualisations of the environments used in this chapter are available in Appendix 7.

5.1. Towards Real Robots

In the previous chapter, we reproduced the key results from [JBH21]. These illustrated that the NVAE model worked effectively under a set of particular conditions. The commonalities of these scenarios were:

1. There is no gravitational force acting on the robot.
2. The key parts of the robot are fully observed and no strong perspective image effects occur.
3. When considering the state as joint angles, the robot had relatively few joints (two), resulting in a low-dimensional latent space.

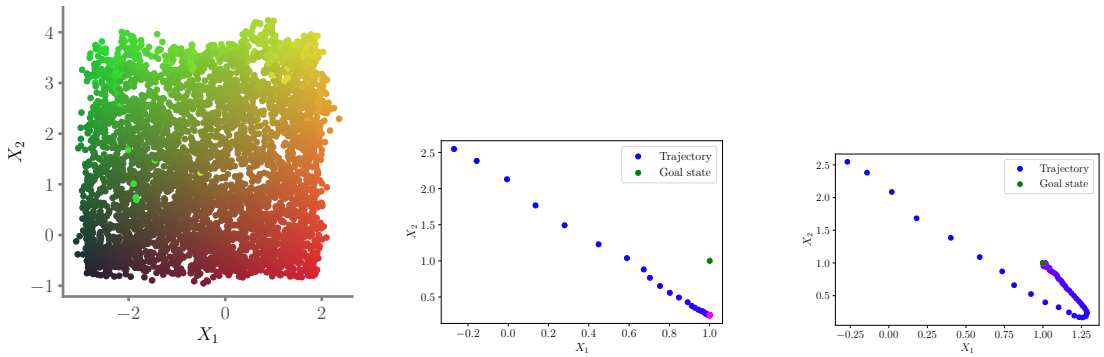
We would like to use the NVAE to control more complex and realistic robots from vision. Robots used in research and industry typically have many joints, move in three dimensions and are subject to the force of gravity. It is therefore pertinent to investigate the robustness and efficacy of the NVAE under these conditions, to identify possible limitations and failure modes of this model.

5.2. The Influence of Gravity

The original experiments conducted in [JBH21] made use of robotic environments not subject to gravity. It is essential to understand how the NVAE responds when gravity acts upon the robot during training. The effect of gravity in the latent space ("latent gravity") should theoretically be handled by the $\boldsymbol{C}$ term of the latent dynamical system in Equation 17. To verify this, we again consider the pointmass environment from Chapter 4 and add a downward gravity vector in MuJoCo.[15]

Figure 7 indicates that the latent space forms appropriately despite gravity. This suggests that the latent gravity vector is well compensated for. We can see that only using proportional control results in a steady-state error. Fortunately, this is easy to correct by introducing the integral term in the controller. We can say that the latent space formed by the NVAE is *PI-controllable* under the influence of gravity.

[15]This was not entirely straightforward, as MuJoCo is still poorly documented. The best resource available is the xml reference, which tends to be suitably vague.



(a) Latent space. (b) P-control trajectory. (c) PI-control trajectory.

Figure 7: The latent space and control trajectories for the pointmass environment, subject to gravity of $g = -0.1\ \mathrm{m} \cdot s^{-2}$ in the y direction. Subfigures 7b and 7c show the latent state trajectories that result after using model for control. The P-control trajectory converges on the goal state, indicating the desired *P-controllability*.

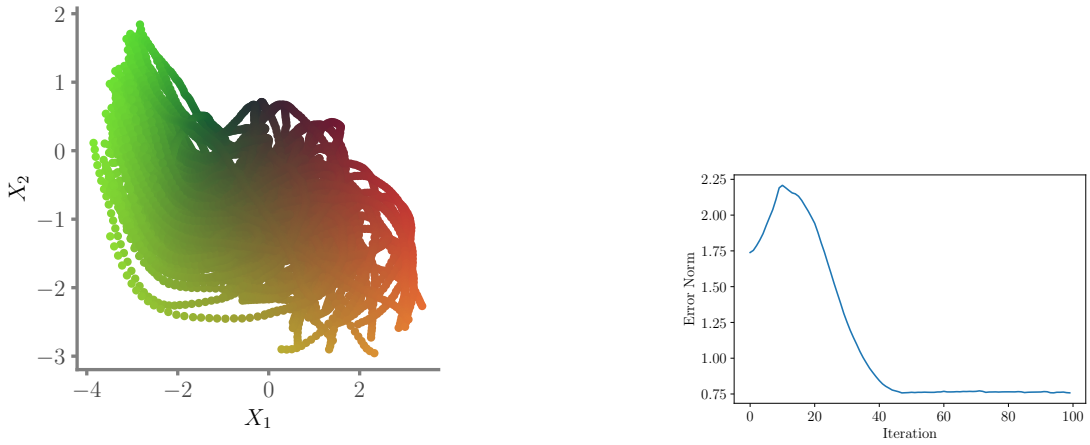
5.3. Three-dimensional Multi-jointed Robots

Real robots move in three spatial dimensions and typically have more than two joints. This subsection begins by considering the Franka Emika Panda[16] [MuJ22], an articulated robot with 7-DOF (excluding the gripper). Further information about this robot and the environment setup is provided in Appendix 7. We view the Panda as a topical robot that a vision-based state inference and control model should be able to work with. We propose to investigate this by starting with the 7-DOF Panda robot. Unfortunately, the naïve application of NVAE to the Panda robot does not produce a proportionally controllable or smooth latent space. Therefore, we begin our investigation by simplifying the problem and constraining the Panda to be functionally equivalent to the reacher.

5.3.1. The Orthogonal Panda

The Panda is constrained to be effectively the same as the reacher: all joints are fixed, except for the shoulder and elbow. The camera is positioned orthogonally to the robot, such that the arm is fully in view. We do this to start from conditions that are known to produce controllable latent spaces and build from there. Figure 8 shows the results of this experiment. The latent space forms smoothly and corresponds well with the true space; proportional control converges in this scenario, as expected.

[16] We tried to use the Panda model from [MuJ22] a few days after it was first published. Unfortunately, there were bugs in the model which caused strange behaviour. These bugs were fixed on the repository a few weeks later.



(a) Latent space.

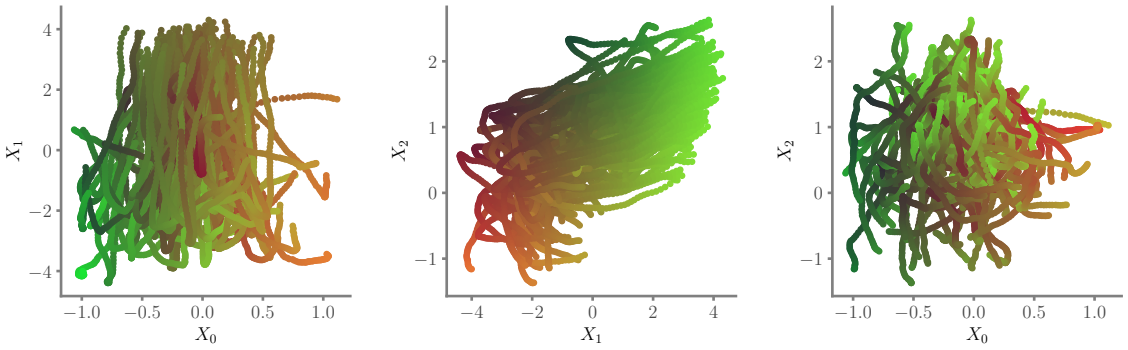
(b) P-control trajectory.

Figure 8: This Figure shows the latent space of the constrained two-jointed Panda, where it is equivalent to the reacher environment from the previous chapter, as well as a P-control trajectory showing that this latent space is sufficiently smooth as to be *P-controllable*.

5.3.2. Introducing the Base Joint

Introducing the base joint of the Panda allows it an additional degree of freedom with which to rotate itself towards and away from the camera, changing the perspective. Figure 9 presents the pairwise latent spaces (since the latent space is three-dimensional now, we visualise it pairwise) formed in this situation.

We can see that in Figure 9 when the base joint is at zero degrees and the robot is orthogonal as before, an area of local smoothness forms. This suggests that there may be some difficulty in learning globally smooth latent spaces from perspectives other than orthogonal, or under changing perspectives.



(a) Latent space, X_0, X_1.

(b) Latent space X_1, X_2.

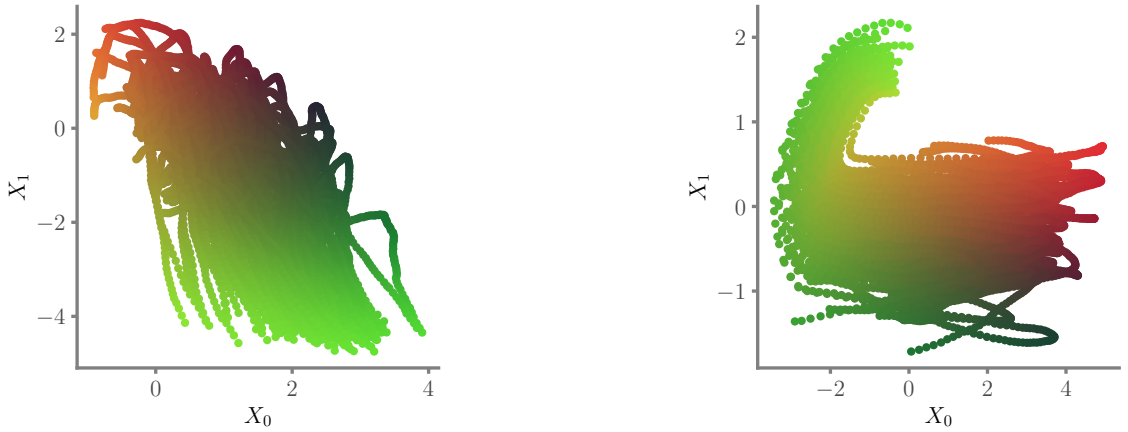
(c) Latent space X_0, X_2.

Figure 9: The three subfigures depict the three unique combinations of pairwise latent space plots. While some local smoothness is observable in Subfigure 9b, the rest of the latent space is not smooth.

5.3.3. Perspective Image Effects

To investigate how the model responds to perspective effects alone, the base joint is fixed once again, but at 45 degrees. In this scenario, the Panda has 2-DOF but is pointed away from the camera, causing a perspective effect. Figure 10a shows the latent space once again forms to be globally smooth and therefore proportionally controllable.

This result suggests that it is possible to learn globally smooth latent spaces despite static perspective effects. Another possible root cause of the result in Figure 9 could be that the perspective, in that case, is dynamic.



(a) Panda fixed 45-degree perspective latent space.

(b) Panda dynamic perspective latent space.

Figure 10: Subfigure 10a shows the latent space of a two-jointed Panda with the base joint fixed at 45 degrees. Subfigure 10b shows the latent space when the Panda has two joints, but one of those joints causes dynamic perspective.

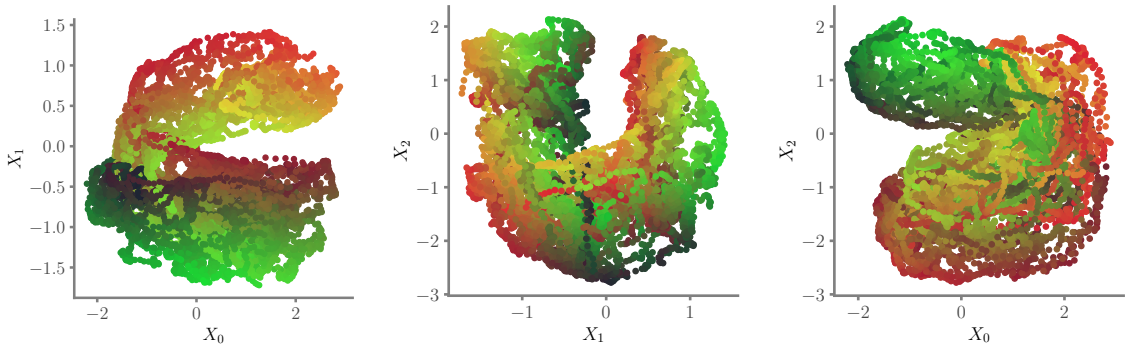
To rule out dynamic perspective as the root cause of failure, we deactivate the shoulder joint and consider the Panda only with base and elbow joints. This yields a 2-DOF system with a dynamic perspective. Figure 10b indicates that even when faced with learning an embedding robust to dynamic perspectives, the model still produces a smooth and disentangled latent space. We, therefore, turn to investigate the intrinsic dimensionality of the latent space.

5.3.4. Articulated Robotic Structures

Articulated robots are structured such that the visual appearance of a joint strongly depends on the configurations of the other joints connecting themselves to the base.[17] This is in contrast to segmented robots, where the visual appearance and configuration of each joint are independent.

[17] This is not the case when observed information is proprioceptive, such as receiving the joint angles directly from angle encoders. In this case, the measurements of the joints are independent.

To investigate the possible failure mode due to the articulated robotic structure, we again consider the reacher robot. This time, we extend the reacher environment by adding one more identical limb[18] to the system, endowing it with 3-DOF. The resultant latent space is shown in Figure 11. We observe limited localised smoothness and no disentanglement. This points towards the increased complexity of latent variable dependency, which can be viewed as a manifestation of the curse of dimensionality.



(a) Latent space X_0, X_1. (b) Latent space X_1, X_2. (c) Latent space X_0, X_2.

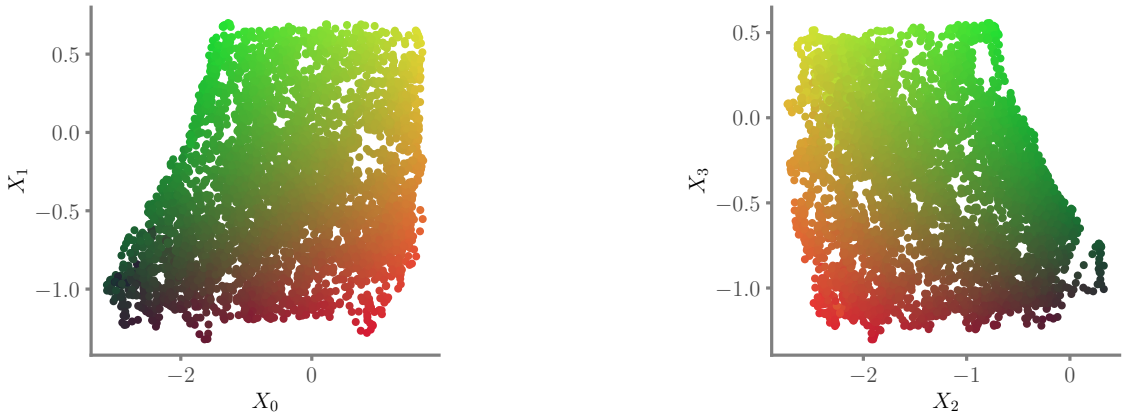
Figure 11: The three subfigures depict the three unique combinations of pairwise latent space plots. There is limited localised smoothness in the latent space and no disentanglement.

5.3.5. High-dimensional Latent Spaces

To verify whether NVAE is capable of handling inherently higher-dimensional latent spaces, we return to the 2-DOF reacher robot. We duplicate this robot and place it pivoting around the same point as the first.[19] These two robots are constrained to separate halves of the image frame, such that they cannot overlap.

Figure 12 shows that the model can successfully separate the dynamics of the two independent systems, and form globally smooth latent spaces in this scenario. This shows that the failure in Figure 11 is not inherent to purely an increase in latent dimensionality, but rather likely comes as the result of the increased complexity of latent variable dependencies.

[18] See Appendix 7.
[19] See Appendix 7.



(a) First reacher latent space.

(b) Second reacher latent space.

Figure 12: The latent spaces of the NVAE trained on two independent reacher robots occupying the same frame. The covered areas of the latent spaces are smooth and disentangled, indicating that it is controllable.

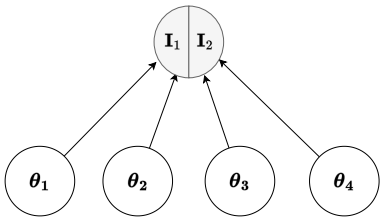
5.4. A Graphical Models Point of View

Figure 13 shows an illustration of the scenario in vision-based robotics when inferring the state as joint angles. Notice that the angle variables influencing the image of the robot are independent before the image is observed. After the image is observed, the variables become entangled, because the visual appearance of some joints affects the visual appearance of other joints. This presents a significantly more complex interdependent relationship than independence.

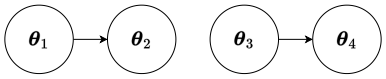
Recall from subsection 3.1.4 that one of the fundamental assumptions of the VAE is that the latent factors are independent (the posterior is chosen as a diagonal Gaussian). This assumption is in direct conflict with the model in Figure 13. Increasing the DOF of an articulated robot therefore exponentially increases the complexity of the latent variable dependencies, which the typical modelling assumptions of the VAE are not inherently equipped to handle. This need not be the case, as one can easily introduce a full-covariance matrix in the VAE.

The experiments detailed in subsection 5.3 support this hypothesis. We saw that for the 7-DOF articulated Panda to produce a smooth and controllable latent space, we needed to constrain it to 2-DOF. This result is also reflected in the 3-DOF reacher experiment. We showed that dynamic perspective is likely not responsible for this failure mode. The dual and 3-DOF reacher experiments ultimately suggest that the failure mode is not because of the number of latent variables, but rather due to the nature of the relationship amongst the latent variables in the setting of vision-based robotics when considering the joint angles as the underlying state.

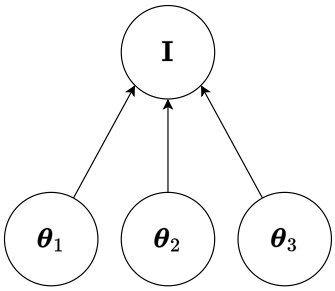
As illustrated in Figure 13, the same problem does not apply when we model the



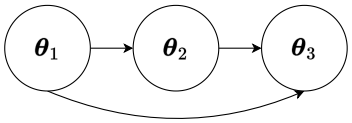
(a) $p(\theta_1, \theta_2, \theta_3, \theta_4, \boldsymbol{I})$



(b) $p(\theta_1, \theta_2, \theta_3, \theta_4 \mid \boldsymbol{I})$



(c) $p(\theta_1, \theta_2, \theta_3, \boldsymbol{I})$



(d) $p(\theta_1, \theta_2, \theta_3 \mid \boldsymbol{I})$

Figure 13: The probabilistic graphical models describing what occurs when modelling the state of a robot from vision. Subfigures 13a and 13b reflect the double reacher scenario (Figure 12). Subfigures 13c and 13d refer to the three-jointed reacher scenario (Figure 11). This figure indicates that the relationship among the variables changes once the image is observed because the visual appearance of the joint angles becomes dependent on one another.

coordinates of the endpoint effector (or any independent factors) as the state, because the latent variables are weakly entangled after observing the image. In this case, the exponentially increasing interdependency of variables manifests itself in the curse of dimensionality. In subsection 6.2 we propose possible remedies for this situation.

6. Conclusions

In this work, we considered joint state representation and dynamical modelling from pixels using a variational Bayesian approach. In particular, we considered the work of [JBH21], where they proposed the NVAE, a variational recurrent neural network for learning proportionally controllable latent spaces via suitable inductive priors. We began by outlining the development of the theory for this approach and reproducing key results from the original paper. We proceeded to investigate the limitations of this model when trying to apply it to more complex and realistic robotic scenarios.

6.1. Findings

- We successfully reproduce key results from [JBH21], starting from the original authors' research repository.
- We introduce the concept of latent gravity and show that the NVAE produces a *PI-controllable* latent space when trained on a robot subject to gravity, as opposed to forming a *P-controllable* latent space.
- We find that the model is capable of handling dynamic perspective effects and latent spaces with dimensionality larger than two.
- Our investigation suggests that the model fails due to a fundamental modelling conflict: the modelling assumption of a VAE is the latent variables are independent of each other. When we model the state as joint angles, the latent variables become entangled as in Figure 13. This situation is difficult to model using a vanilla VAE, and becomes exponentially more complex as the number of joints increases.

6.2. Future work

The following paragraphs detail promising areas of study identified for future work.

6.2.1. Enriching the Latent Variable Relationships

The most notable limitation of the NVAE is the failure mode we have focused on in subsection 5.4, which occurs due to the interdependency of latent variables when modelling the state as joint angles from vision. Future work should aim to address this. We hypothesise that including normalising flows [RM15] on the encoder network will allow the latent variables to entangle nonlinearly and hence obtain a better fit of the posterior.

Future work should also look at new approaches to incorporating arbitrary dependency structures of latent variables in VAEs, as in [MK22]. We hypothesise that modelling the structure in Figure 5.4 directly might assist in resolving this failure mode.

6.2.2. Object Manipulation

The next frontier for this research is to try and manipulate objects in the scene, using the NVAE's embeddings for control. This should consider the robustness of the state embeddings when other objects are present or moving around in the scene. Training with objects initialised in random locations might facilitate some degree of invariance to object location. We should account for these objects by designing a representation learning system that facilitates this. One would likely also incorporate reinforcement learning policies to perform these complicated actions.

6.2.3. Observability and Controllablility

In subsection 2.2.1 we discussed two important aspects of control systems: observability and controllability. In control theory [SSZ10] we typically consider linear time-invariant systems:

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{26}$$

where $\boldsymbol{x}$ is the internal system state, $\boldsymbol{y}$ is the system output and $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are the output, input and state matrices. This continuous linear system is observable if and only if

$$\operatorname{rank} \begin{bmatrix} \boldsymbol{C}\boldsymbol{A}^0 \\ \boldsymbol{C}\boldsymbol{A}^1 \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{n-1} \end{bmatrix} = n, \tag{27}$$

where n is the number of state variables. However, in our case $\boldsymbol{y}$ is a nonlinear function of the state through the decoder $h(\boldsymbol{z})$. We can compute the Jacobian of this with respect to the state using reverse-mode automatic differentiation:

$$\boldsymbol{C}' = \mathbf{J}_{\boldsymbol{x}} h \tag{28}$$

This poses the question: based on this, could we observe degradation in the rank of the observability matrix in situations where the robot is not observable or controllable? If one were to create a heatmap of this for all valid states, what might emerge?

More generally, within this integration of control theory and statistical machine learning, several interesting new questions emerge:

- Can we derive criteria to determine which states are controllable and observable?
- Can we then use this to choose the best camera angle to control a given robotic system?
- Can we take this into account in planning, so that we only follow trajectories where the state space is smooth and observable?

Environment	Space Dimensionality	DOF	Δt (s)	State Space
Pointmass	2D	2	0.5	Coordinates
Two-jointed reacher	2D	2	0.1	Angles
Fetch	3D	4	0.1	Coordinates
Three-jointed reacher	2D	3	0.1	Angles
Double reacher	2D	4	0.1	Angles
Panda	3D	7	0.01	Angles

Table 1: This table details the particulars of each robotic simulation environment used.

7. Experimental Setup

This appendix details the robotic experimental setup we used in chapters 4 and 5. Following the authors of [JBH21], we used the (now open-sourced) multi-joint dynamics with contact (MuJoCo) framework [TET12] as the physics engine supporting the robotic simulations. Likewise, we made use of the DeepMind control library [TMD+20] to interface MuJoCo with Python.[20]

The DeepMind control library provides a simple API for accessing MuJoCo robots and environments that come in the form of `xml` files. This sets up and initialises the physics engine, and provides a Python wrapper that provides access to pixel observations of the specified size and allows for action inputs.

7.1. Robotic Simulation Environments

The initial pointmass and reacher environments are from [TMD+20], while the fetch robot is from [BCP+16]. The Panda environment is from [MuJ22]. The robotic environments as they were used are shown in Figure 14. Each environment operates at a sampling period Δt as specified in Table 1.

7.1.1. Generation of Training Data

The training data was generated by initialising the robots with zero velocity on the joints, and a random uniformly sampled state vector. Uniformly sampled random actuations are then applied for a number of timesteps, recording the true state and pixel observations for one episode. This is repeated for a number of episodes. We used 1000 episodes and 50 to 100 timesteps per episode. All images were downsampled to be RGB images of dimensions $64 \times 64 \times 3$. The reacher environments were limited to angle ranges which prevent full rotation and collision with itself, or in the case of the double reacher the other robot. The Panda was also constrained to prevent full joint rotations.

[20] The only exception to this is the fetch robot. It is from OpenAI Gym [BCP+16].

7.2. Function Approximation and Optimisation

7.2.1. Neural Network Architectures

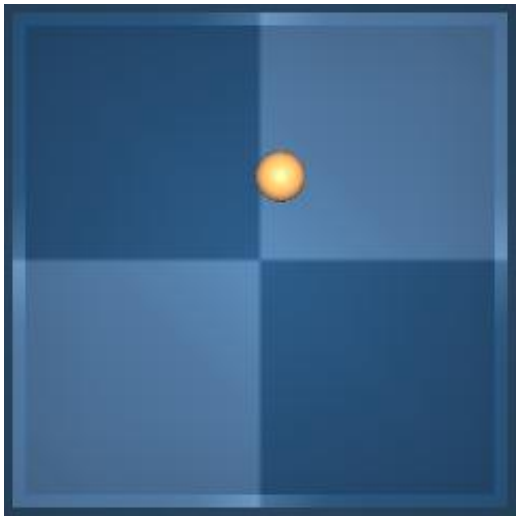
In the encoder, we used a convolutional neural network with four convolutional layers, with numbers of filters $[512, 64, 32, 16]$ terminating in a fully connected layer of 512 neurons that then parameterise the Gaussian stochastic position.

The latent dynamics model's matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ from Equation 19 were trained as the output of a multilayer perceptron of 3 layers with 32 neurons each. The capacity of this physics prediction network was varied to ensure it is not under-parameterised.

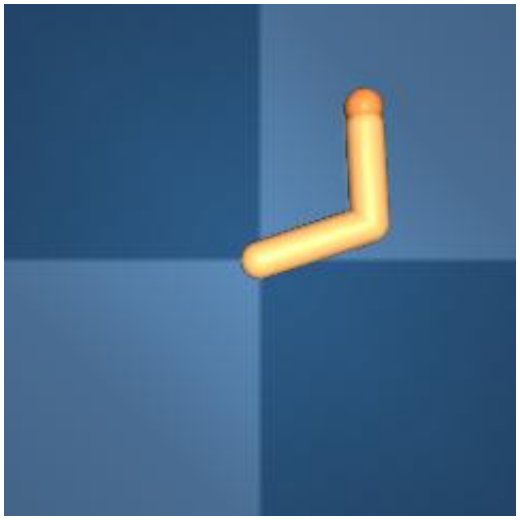
7.2.2. Optimisation

We used the Adam optimiser [KB14] throughout, with an initial learning rate of 1e^{-4} to 3e^{-4}. We found that training does not converge when using batch sizes other than 1, even if batch normalisation [IS15] is introduced to the dynamics model.

An important practical matter to take note of when working with this model is that it is extremely sensitive to the initial parameters (random seed). This makes reproducing results more difficult. We recommend running experiments more than once to verify result consistentency.



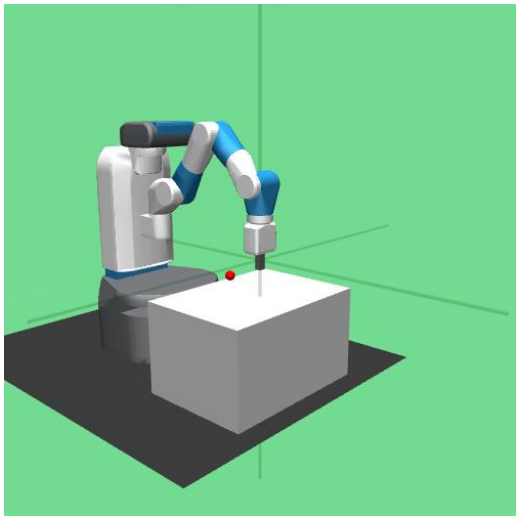
(a) Pointmass.



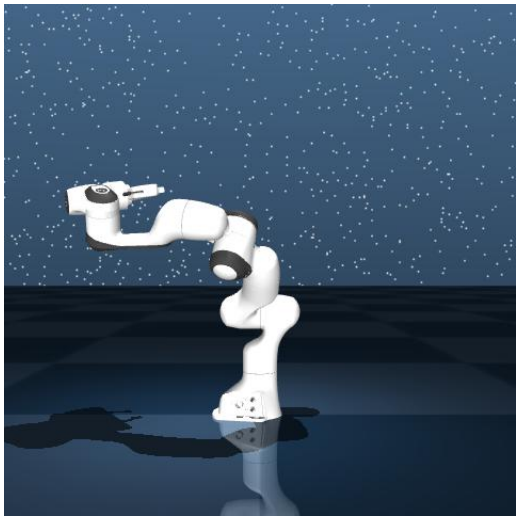
(b) Two-jointed reacher.

(c) Three-jointed reacher.

(d) Double two-jointed reacher.



(e) Fetch.



(f) Panda.

Figure 14: The six subfigures depict the robotic simulation environments that were used in this work.

8. Derivations of Variational Lower Bounds

8.1. Variational Recurrent Neural Network

The marginal data likelihood we would like to maximise is:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) = \int p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) \, \mathrm{d}\boldsymbol{z}_{1:T} \,. \tag{29}$$

The two factors in the integral of Equation 12 are factorised as follows under an observation independence and Markov assumption as in Figure 3:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) = \prod_t p(\boldsymbol{x}_t \mid \boldsymbol{z}_t) \tag{30}$$

and

$$p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) = \prod_t p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}) \,. \tag{31}$$

The approximate posterior of the sequence is given by:

$$q(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}_{1:T}) = \prod_t q(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \,. \tag{32}$$

Taking the log and introducing the approximate posterior,

$$\begin{aligned} \log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) &= \log \left(\prod_t \int p(\boldsymbol{x}_t \mid \boldsymbol{z}_t, \boldsymbol{u}_{t-1}) p(\boldsymbol{z}_t \mid \boldsymbol{u}_{t-1}, \boldsymbol{z}_{t-1}) \frac{q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)}{q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)} \, \mathrm{d}\boldsymbol{z}_t \right) \\ &= \sum_t \log \left(\mathbb{E}_{q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})} \left[\frac{p(\boldsymbol{x}_t \mid \boldsymbol{z}_t) p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1})}{q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)} \right] \right) \,. \end{aligned} \tag{33}$$

Now applying Jensen's inequality, we obtain:

$$\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) \geq \sum_t \mathbb{E}_{q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)} \left(\log \left[\frac{p(\boldsymbol{x}_t \mid \boldsymbol{z}_t) p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{u}_{t-1})}{q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})} \right] \right) \,. \tag{34}$$

Per the definition of KL divergence, this yields:

$$\begin{aligned} \log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) \geq \sum_t \mathbb{E}_{q_\phi(\boldsymbol{z}_t \mid \boldsymbol{x}_t, \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1})} & \underbrace{\log p(\boldsymbol{x}_t \mid \boldsymbol{z}_t)}_{\text{reconstruction}} \\ & - \underbrace{\mathrm{KL}(q(\boldsymbol{z}_t \mid \boldsymbol{x}_t) || p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}))}_{\text{prior divergence penalty}} \,. \end{aligned} \tag{35}$$

8.2. Newtonian Variational Autoencoder

This is the derivation as presented by [JBH21], with minor corrections as mentioned in subsection 3.4. Once again we consider the marginal data likelihood:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) = \int p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) \, \mathrm{d}\boldsymbol{z}_{1:T} \,. \tag{36}$$

The variable $\hat{\boldsymbol{z}}$ refers to the estimate of the positional latent variable $\boldsymbol{z}$ at time t as per the learned transition dynamics at that stage. The two factors in the integral of Equation 12 are further factorised as follows:

$$\begin{aligned} p(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) &= \prod_t p(\boldsymbol{x}_t \mid \boldsymbol{z}_t) \\ &= \prod_t \int p(\boldsymbol{x}_t \mid \hat{\boldsymbol{z}}_t) p(\hat{\boldsymbol{z}}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_{t-1}) \, \mathrm{d}\boldsymbol{z}_t \end{aligned} \tag{37}$$

and

$$p(\boldsymbol{z}_{1:T} \mid \boldsymbol{u}_{1:T}) = \prod_t p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_{t-1}) \,. \tag{38}$$

where velocity is calculated as $\boldsymbol{v}_t = (\boldsymbol{z}_t - \boldsymbol{z}_{t-1})/\Delta t$. The approximate posterior of the sequence is given by:

$$q(\boldsymbol{z}_{1:T} \mid \boldsymbol{x}_{1:T}) = \prod_t q(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \,. \tag{39}$$

We take the log, introduce the approximate posterior and apply Jensen's inequality:

$$
\begin{aligned}
&\log p(\boldsymbol{x}_{1:T} \mid \boldsymbol{u}_{1:T}) = \\
&\log \Bigg(\int \prod_t \frac{q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)}{q(\boldsymbol{z}_t \mid \boldsymbol{x}_t)} \int p(\boldsymbol{x}_t \mid \hat{\boldsymbol{z}}_t) p(\hat{\boldsymbol{z}}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_{t-1}) \, \mathrm{d}\hat{\boldsymbol{z}}_t \\
&\qquad \prod_t p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_{t-1}) \, \mathrm{d}\boldsymbol{z}_{1:T} \Bigg) \\
&\geq \int \prod_t q(\boldsymbol{z}_t \mid \boldsymbol{x}_t) \Bigg(\sum_t \log \Bigg[\int p(\boldsymbol{x}_t \mid \hat{\boldsymbol{z}}_t) p(\hat{\boldsymbol{z}}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_{t-1}) \, \mathrm{d}\hat{\boldsymbol{z}}_t \\
&\qquad + \sum_t \log \frac{p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1})}{(\boldsymbol{z}_t \mid \boldsymbol{x}_t)} \Bigg] \Bigg) \, \mathrm{d}\boldsymbol{z}_{1:T} \\
&= \sum_t \int q(\boldsymbol{z}_{t-1} \mid \boldsymbol{x}_{t-1}) q(\boldsymbol{z}_{t-2} \mid \boldsymbol{x}_{t-2}) \Bigg(\log \Bigg[\int p(\boldsymbol{x}_t \mid \hat{\boldsymbol{z}}_t) p(\hat{\boldsymbol{z}}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_{t-1}) \, \mathrm{d}\hat{\boldsymbol{z}}_t \Bigg] \\
&\qquad - \mathrm{KL}(q(\boldsymbol{z}_t \mid \boldsymbol{x}_t) || p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1}) \Bigg) \, \mathrm{d}\boldsymbol{z}_t \\
&\geq \sum_t \mathbb{E}_{q(\boldsymbol{z}_{t-1} \mid \boldsymbol{x}_{t-1}) q(\boldsymbol{z}_{t-2} \mid \boldsymbol{x}_{t-2})} \Bigg(\underbrace{\mathbb{E}_{p(\hat{\boldsymbol{z}}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1})} \log(\boldsymbol{x}_t \mid \hat{\boldsymbol{z}}_t)}_{\text{predicted present reconstruction}} \\
&\qquad - \underbrace{\mathrm{KL}(q(\boldsymbol{z}_t \mid \boldsymbol{x}_t) || p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{u}_{t-1})}_{\text{transition prior divergence}} \Bigg) .
\end{aligned}
\tag{40}
$$

References

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

[BEPVDS19] Philip Becker-Ehmck, Jan Peters, and Patrick Van Der Smagt. Switching Linear Dynamics for Variational Bayes Filtering. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 553–562. PMLR, 09–15 Jun 2019.

[BK19] Aude Billard and Danica Kragic. Trends and Challenges in Robot Manipulation. *Science*, 364, 06 2019.

[BSB+04] David Bellot, Roland Siegwart, Pierre Bessiere, Adriana Tapus, Christophe Coue, and Julien Diard. *Bayesian Modeling and Reasoning for Real World Robotics: Basics and Examples*, pages 186–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[CKD+15] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[DFR15] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, feb 2015.

[GMR16] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, International Conference on Machine Learning*, 2016.

[HLF+18] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels, 2018.

[HS18] David Ha and Jürgen Schmidhuber. World Models. 2018. cite arxiv:1803.10122.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015.

[JBH21] Miguel Jaques, Michael Burke, and Timothy Hospedales. NewtonianVAE: Proportional Control and Goal Identification from Pixels via Physical Latent Spaces. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4452–4461, 2021.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[KSBvdS17] Maximilian Karl, Maximilian Solch, Justin Bayer, and Patrick van der Smagt. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. *ArXiv*, abs/1605.06432, 2017.

[KW13] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2013.

[LDRGF18] Timothee Lesort, Natalia Diaz-Rodriguez, Jean-Franocis Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, dec 2018.

[MK22] Jacobie Mouton and Steve Kroon. SIReN-VAE: Leveraging Flows and Amortized Inference for Bayesian Networks, 2022.

[MuJ22] MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022.

[PBJ12] John Paisley, David Blei, and Michael Jordan. Variational Bayesian Inference with Stochastic Search, 2012.

[Ric05] Arthur G. Richards. Robust constrained model predictive control. 2005.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows, 2015.

[SKK22] Abhilasha Singh, V. Kalaichelvi, and R. Karthikeyan. A Survey on Vision Guided Robotic Systems with Intelligent Control Strategies for Autonomous Tasks. *Cogent Engineering*, 9(1):2050020, 2022.

[SSZ10] Sy Najib Sy Salim and Maslan Zainon. *Control Systems Engineering*. 01 2010.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[TMD+20] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and Tasks for Continuous Control. *Software Impacts*, 6:100022, 2020.

[VK20] Arash Vahdat and Jan Kautz. NVAE: A Deep Hierarchical Variational Autoencoder, 2020.

[WSBR15] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[ZN42] J. G. Ziegler and Nathaniel B. Nichols. Optimum Settings for Automatic Controllers. *Journal of Dynamic Systems Measurement and Control-transactions of The Asme*, 115:220–222, 1942.

[ZSB+20] Qile Zhu, Jianlin Su, Wei Bi, Xiaojiang Liu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. A Batch Normalized Inference Network Keeps the KL Vanishing Away, 2020.

[ZVS+18] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning, 2018.