MAXIMILIAN KARL

UNSUPERVISED CONTROL

MAXIMILIAN KARL

UNSUPERVISED CONTROL

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik

UNSUPERVISED CONTROL

Efficient Inference for Time Series Modelling and Intrinsic Motivation

Maximilian Karl

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr. Stephan Günnemann

Prüfer der Dissertation:

1. Prof. Dr. Patrick van der Smagt
2. Prof. Dr.-Ing. Sami Haddadin

Die Dissertation wurde am 30.04.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 09.12.2019 angenommen.

ABSTRACT

Intrinsic motivation without the need of defining an external reward has long been proposed for equipping artificial agents with behaviour previously only seen in living organisms. One of these techniques is called empowerment, which rewards the maximisation of information transfer between actions and the resulting sensor readings of an agent. While there exists intensive research on the possible behaviour of empowered agents in toy environments, only few results were obtained for complex robots and environments with continuous action and sensor spaces. Evaluating this reward requires firstly, a precise dynamics model of the agent and the environment, and secondly, computation of the expectation over all possible future results of its actions. In this work, an unsupervised algorithm for learning latent Markovian state space models from non-Markovian raw sensor sequences was developed for solving the first task. The second task was solved by introducing an efficient model-based method for computing empowerment. It is based on a lower bound on mutual information and Monte-Carlo sampling. Experiments on simulated robots show the expected behaviour presented in previous empowerment literature comprising biped balancing, flock behaviour of multi-agent systems, pendulum swing-up and grip stability. A final real-world experiment shows, that this methodology can also handle real-world data and fly a quadrocopter with collision avoiding behaviour.

ZUSAMMENFASSUNG

Künstliche Agenten durch intrinsiche Motivation und ohne externe Belohnung mit Verhalten auszustatten welche zuvor nur in lebenden Organismen zu beobachten war, wurde schon seit langem vorgeschlagen. Eine dieser Techniken nennt sich Empowerment, welche die Maximierung des Informationtransfers zwischen Aktionen und die daraus resultierenden Sensorwerte belohnt. Obwohl es schon viele theoretische Ergebnisse über das Verhalten von Empowerment maximierenden Agenten in simplen Umgebungen gibt, existieren nur wenige für komplexe Roboter und Umgebungen mit kontinuierlichen Aktions- und Sensorräumen. Um diesen Belohnungswert zu evaluieren benötigt man erstens ein genaues dynamisches Model des Agenten und seiner Umgebung und zweitens die Berechnung des Erwartungswertes aller durch die Aktionen induzierten möglichen zukünftigen Zustände. Als Lösung für die erste Aufgabe wird in dieser Arbeit ein unüberwachter Algorithmus vorgestellt, welcher latente Markovsche Zustandsräume aus nicht-Markovschen unverarbeiteten Sensorsequenzen lernen kann. Die zweite Aufgabe wurde durch die Entwicklung einer effizienten Modell-basierten Methode zur Berechnung von Empowerment gelöst. Diese basiert auf einer unteren Schranke für Transinformation kombiniert mit einer Monte-Carlo-Simulation. Experimente mit simulierten Robotern replizieren die in der bestehenden Literatur zu Empowerment vorgestellten Verhaltensmuster und beinhalten das balancieren eines Biped, Flockverhalten

in Multiagentsystemen, aufschwingen eines Pendels und Griffstabilität. Ein abschließendes Experiment zeigt, dass diese Methodik auch für Daten aus der realen Welt verwendet werden kann, und sogar einen Quadrocopter kollisionsvermeidend fliegen kann.

DANKSAGUNG

Meinem Doktorvater Patrick van der Smagt danke ich für sein Vertrauen in meine Ideen und die Möglichkeiten und Unterstützung diese umzusetzen. Außerdem danke ich meinem Mentor Justin Bayer der mir mit Rat und Tat immer zur Seite stand. Besonderer Dank gilt auch Djalel Benbouzid, Maximilian Sölch, Nutan Chen, Philip Becker-Ehmck und natürlich auch all meinen anderen Kollegen für deren Unterstützung, die diese Arbeit erst möglich gemacht hat. Meiner Familie danke ich für den Rückhalt, speziell meinem Vater der immer ein offenes Ohr für mich hatte.

Danke an denjenigen, der durch eine zerbrochene Tasse mein unermüdliches Interesse an Entropie geweckt hat.

CONTENTS

LIST OF ACRONYMS

nat	natural unit of information
SGVB	stochastic gradient variational Bayes
ELBO	evidence lower bound
VAE	variational auto-encoder
DVBF	deep variational bayes filters
DKF	deep Kalman filter
RL	reinforcement learning
ICA	independent component analysis
PCA	principal component analysis
PPCA	probabilistic principal component analysis
GRU	gated recurrent unit
LSTM	long short term memory
RNN	recurrent neural network
DOF	degree of freedom
iLQR	iterative linear quadratic regulator
FPGA	field programmable gate array
LIDAR	light detection and ranging sensors
I2C	inter-integrated circuit
IMU	inertia measurement unit
SPI	serial peripheral interface
PWM	pulse width modulation
HMC	Hamiltonian Monte Carlo
MH	Metropolis Hastings
MCMC	Markov chain Monte Carlo
PID	proportional–integral–derivative

LIST OF PUBLICATIONS

The following list is composed of publications of which parts, figures and results are incorporated into this thesis:

PUBLICATIONS

- Maximilian Karl, Maximilian Sölch, Justin Bayer and Patrick van der Smagt. „Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- Maximilian Karl, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt and Justin Bayer. „Unsupervised Real-Time Control through Variational Empowerment". In: (13th Oct. 2017). arXiv: 1710.05101 [stat].
- Maximilian Karl, Justin Bayer and Patrick van der Smagt. „Unsupervised Preprocessing for Tactile Data". In: (23rd June 2016). arXiv: 1606.07312 [cs, stat].
- Maximilian Karl, Justin Bayer and Patrick van der Smagt. „Efficient Empowerment". In: (28th Sept. 2015). arXiv: 1509.08455 [cs, stat].

List of publications that were created during the work on this thesis but have not been incorporated:

CO-AUTHORED

- N. Chen, M. Karl and P. van der Smagt. „Dynamic Movement Primitives in Latent Space of Time-Dependent Variational Autoencoders". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). Nov. 2016, pp. 629–636. DOI: 10.1109/HUMANOIDS.2016.7803340.
- Maximilian Karl, Artur Lohrer, Dhananjay Shah, Frederik Diehl, Max Fiedler, Saahil Ognawala, Justin Bayer and Patrick van der Smagt. „ML-Based Tactile Sensor Calibration: A Universal Approach". In: (21st June 2016). arXiv: 1606.06588 [cs].
- H. van Hoof, N. Chen, M. Karl, P. van der Smagt and J. Peters. „Stable Reinforcement Learning with Autoencoders for Tactile and Visual Data". In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Oct. 2016, pp. 3928–3934. DOI: 10.1109/IROS.2016.7759578.

- Christopher Wolf, Maximilian Karl and Patrick van der Smagt. „Variational Inference with Hamiltonian Monte Carlo“. In: (26th Sept. 2016). arXiv: 1609.08203 [stat].
- Justin Bayer, Maximilian Karl, Daniela Korhammer and Patrick van der Smagt. „Fast Adaptive Weight Noise“. In: (19th July 2015). arXiv: 1507.05331 [cs, stat].

Part I

MOTIVATION AND BACKGROUND

1 INTRODUCTION

Making robots interact with real world environments in a natural way seems to be inherently hard when trying to define their movement in a mathematical way. Living organisms easily learn complicated movements despite their huge variety of actuator configurations and sensor modalities. There has to be a common way of assessing the quality of interaction with the environment. Erwin Schrödinger made first steps in assigning a currency to self organising systems and called it negentropy, also referred to as free energy. Organisms exchange free energy with their environment to perform work. Other work linked free energy to information theoretic quantities, making it possible to implement and optimise these cost functions found in nature for artificial information processing systems. One of these formulations of intrinsic motivation is called empowerment and is defined as the channel capacity of the environment when sending information with actuators and receiving it back with the agents sensors. It will enable us to build robots with complex and arbitrary arrangements of sensors or actuators and control them in a general and information theoretic optimal way.

However, this method requires an accurate model of the agent and the environment, which should be trained on data in an unsupervised fashion. These models should furthermore express the dynamics in a compressed form as some latent state space form, as the observations might be high dimensional and redundant. The model can then be reused for exploiting the estimated intrinsic motivation cost function.

The next two chapters in this part cover prerequisites and background information for intrinsic motivation and its connection to entropy from physics and the inherent problems with modelling complicated distributions.

The three chapters in the second part of the thesis form the main contributions of this work. They show the evolution of individual methods needed for performing efficient unsupervised control. The first chapter presents the application of an unsupervised method for learning latent presentations from raw sensor data. An evaluation with high dimensional tactile sensor data shows, that the algorithm is able to compress the data into a compact low-dimensional representation with disentangled hidden states, linearly relatable to the true physical quantities. A possible use of such a compressed latent state for preprocessing of high-dimensional sensors for use in model-free control like reinforcement learning (RL) is shown. In the second chapter this sensor processing for static input is expanded with temporal information for processing time series data. Similar as in the static case, the algorithm is able to find a compressed representation for fully explaining the observed data. The algorithm or the resulting latent states can now also be used for supervised model-based control like optimal control. The final missing part for efficient unsupervised control is presented in the third chapter. Here a variational lower bound for mutual information makes the computation of empowerment tractable. At the same time the process for using the time series model for

control is described. Thus both parts make use of the learned system-dynamics. The chapter finishes with experiments on simulated agents and environments and a real world quadrocopter. The simulated experiments represent environments in previous intrinsic motivation literature. The real world experiment involves several achievements: (1) a state space model of distance sensors and their relation to actions is learned from real-world sensor data (2) empowerment steers the quadrocopter away from walls and controls hovering (3) the policy and the learned state space model is able to run locally on the quadrocopter with low computing power.

2 INTRINSIC MOTIVATION

Intrinsic motivation describes the satisfaction for activities that are done for their own sake and not as a reaction to some reward from the external world. It is thought as the source for exploration and curiosity but can also be used to describe self organisation. The latter appears when the change of the agents knowledge is excluded. The next chapters will focus on information theoretic definitions of intrinsic motivation and the choice will be backed up by connections between physics and information theory.

2.1 INTRINSIC MOTIVATION IN INFORMATION THEORY

2.1.1 *Interestingness*

Intrinsic motivation can also be defined out of an information theoretical standpoint as a theory of beauty and interestingness. In [80] the reward for a curious agent is defined as the interestingness of the received sensor data. Interestingness is defined as the time derivative of beauty, or compressibility, here defined as the Kolmogorov complexity of the observations. Following this reward an agent would try to move into states where it continuously is able to improve its internal representation of the external data. Once the observed data is perfectly compressed the current state looses its beauty, and the lack of improvement of the compression makes the state uninteresting.

2.1.2 *Empowerment*

Another intrinsic motivation formulation, as initially proposed in [79], is empowerment. It is defined as the channel capacity between the agents actions and the resulting state in the future. The channel capacity was initially developed for measuring the maximum amount of information that can be transferred over some transmission channel. Here the environment is defined as the transmission channel, with the actuators of the agent defined as transmitter and the sensors as receivers. Formally interactions with the environment are defined through a transition function $p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})$ with $\mathbf{z}$, $\mathbf{z}'$ and $\mathbf{u}$ as the current state, next state and executed action. The channel capacity for this transition is defined as

$$\mathcal{C}_{\mathbf{u}\to\mathbf{z}'}(\mathbf{z}) = \max_{\omega} \int \omega(\mathbf{u} \mid \mathbf{z}) \int p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}) \ln \frac{p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})}{\int \omega(\mathbf{u} \mid \mathbf{z})\, p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})\, d\mathbf{u}} \, d\mathbf{z}' d\mathbf{u} \tag{2.1}$$

$$= \max_{\omega} \int \omega(\mathbf{u} \mid \mathbf{z}) \int p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}) \ln \frac{p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})}{p(\mathbf{z}' \mid \mathbf{z})} \, d\mathbf{z}' d\mathbf{u}. \tag{2.2}$$

The channel capacity (eq. (2.1)) definition requires a source distribution $\omega(\mathbf{u} \mid \mathbf{z})$ used for sending information into the environment. The fraction in the argument of the logarithm then compares the result of the transition $p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})$ with

the marginal transition $p(\mathbf{z}' \mid \mathbf{z})$. This fraction of probabilities measures the difference between the change of an action and the combined influence of all actions from the source distribution. If for example, no action is able to change the next state, the marginal transition would have the same distribution and the logarithm of the fraction would be zero. When a different state can be reached for each action value then the logarithm would be maximal. Any outputs of the transition that are not influenced by the action will lead to a decrease of empowerment. This can also be seen as a form of counting all possible future states which was proven for the discrete state and action space in [78]. When used as a cost function, empowerment would grade each state and a controlled agent would then choose its action, such that the dynamics would lead him into the state with maximal empowerment. Following this cost function would drive an agent into states which maximise his amount of possibilities.

The behaviour resulting from optimising this intrinsic motivation definition has been extensively explored in various toy experiments. One of the most prominent is the inverted pendulum experiment, where empowerment alone is able to balance an inverted pendulum [75–77]. Another balancing result can be achieved in a bicycle environment [76]. When applied to a two dimensional grid world with obstacles and walls, agents try to avoid them and gather in regions from which more states are reachable [74, 78, 79]. When movable objects are added to that environment, empowerment increased in the proximity of those objects. In [73] and [78] the avoidance of other unpredictable and predator agents got examined, respectively. In environments with far more than two agents a sort of flocking and swarm behaviour can be observed [72]. In this case local empowerment computation and exploitation was able to generate global patterns and clusters. Instead of optimising actions, empowerment can also be used to change the appearance of an agent, especially its sensor configuration. In [75] the sensor evolution of an empowered agent is shown adapting to maximise its information input. This is possible as the transition distribution in the empowerment formulation heavily depends on the sensor structure.

Equation (2.2) describes the most used and initial form of empowerment but several modifications were added over the years. Some of them only need slight modifications or only add different interpretations for parts used in the equation. The literature also considers n-step empowerment where the space of actions gets expanded to sequences of actions and the transition performs n-steps through the environment. This form is used to measure influence over a larger horizon as some environments do not show their full non-linear dynamics when transitioning a single time-step.

2.2 PHYSICS OF ORGANISATION

Apart from the information theoretic approaches for defining the decision process there were much earlier attempts to define the behaviour of living organisms or the emergence of self organising structures. The objective in these cases is usually defined as the entropy or a change in entropy. In one of the earlier publications about these natural forces [71] explains that life and its actions can be defined as a local reverse of the second law of thermodynamics, which corresponds to a

constantly increasing entropy. In the following the irreversibility during physical processes is explained, also known as the source for increasing entropy, followed by the various definitions of the processes that makes organisms fight their constant decay.

2.2.1 *Irreversibility*

In physics, the second law of thermodynamics states that the entropy of a closed system always increases. This is also a property of systems that follow classical time reversible mechanics. Consider a box full of particles, they come in two types and particles of one of them are clustered in the centre. Now through movement and collisions the particles mix and with it the types are homogeneously distributed. A time-reverse state evolution would follow the exact same physical rules but the particles would seem to magically order themselves into the original clustering. Such a configuration, ordering all particles into the initial state, has an extremely low probability to appear randomly. The increase of this disorder or entropy is defined as the result of irreversible transitions. As soon as a particle collides with its surrounding it exchanges a small amount of energy. The large number of possible configurations of the surrounding makes it extremely unlikely to receive the same amount of energy back, making the transition irreversible. The original definition of the change of thermodynamic by Clausius states that

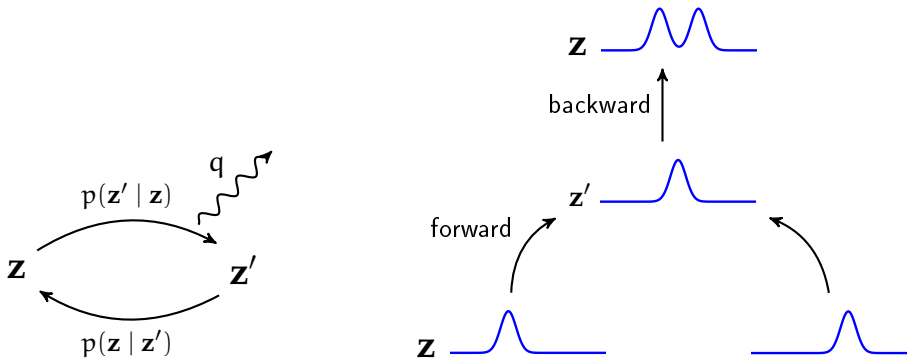
$$S_B - S_A \geqslant \int_A^B dq/T,$$

where the entropy change $S_B - S_A$ is related to the energy dissipated to the heat-bath divided by temperature T. The equality holds for reversible processes and the inequality for those that are irreversible.

Open systems however can retrieve negative entropy into their structure but inevitably increase the entropy of the surrounding. In physical systems this information exchange happens in the form of energy, but a unit conversion into information theoretic bits or natural unit of informations (nats) is possible. For acting autonomously in these environments, an agent has to estimate and optimize this flow of information. With the connection to information theory one can now use known tools like probabilistic modelling for estimating and optimising physical entropy.

The forward process can also be modelled probabilistically with $p(\mathbf{z}' \mid \mathbf{z})$, describing the evolution of the state $\mathbf{z}$ through time. When a transition exhibits irreversible changes to the state, then the reverse process $p(\mathbf{z} \mid \mathbf{z}')$ has a low probability when evaluated on the last state. Figure 2.1b illustrates this process on the collapse of two states, represented by the two Gaussian distributions in the bottom. The forward process maps both states into a single one which forces the backward process to model a multi-modal distribution when running backwards. The information about the previous state is lost and both initial states could have occurred with equal probability. In the physical world such an irreversible transition would generate and emit a small amount of energy q, just as depicted in fig. 2.1a.

There exists a conversion factor between these probabilities of the forward and backward process, the energy dissipation and the entropy production. This creates a link between physics and information theory. In the following recent formal



(a) Irreversible state transition emits a quantum of energy into the surrounding. The absorption makes the information irretrievable and the transition irreversible.

(b) Loss of information and irreversibility by collapsing modes in a probability function. In the final state distribution there is no information about how to get back to the distribution with two modes.

Figure 2.1: An irreversible process results in dissipated energy q. The erasure of information results in a distribution with higher variance.

definitions and experimental verifications of such connections will be presented and connected back to intrinsic motivation. Here they will be used to justify the use of empowerment as the choice for exploiting intrinsic motivation.

2.2.2 *Negentropy principle*

Schrödinger [70] coined the term *negentropy*, the negative entropy, and defined it as the fundamental objective for living organisms. It can be related to most physical processes and can be compared to free energy, but was named differently to avoid confusion. Negentropy measures the usable energy of a physical system. Usable energy can be stored in structure and order which holds true for energy stored in molecules, atoms mass or in an information system. Schrödinger came to the conclusion that according to this view of the world, everything in the universe can be considered living, but with different intensity, defined by the amount of order a system can achieve.

In a more recent publication [69] a similar idea was presented and called *causal entropic forces*, where the gradient of the path entropy of future states defines the force or action on an agent. This force points away from voids in the volume of future states and therefore maximises future entropy. It seems counter-intuitive here to maximise entropy whereas the negentropy principle tries to do the opposite. It is easy to assume that entropy should simply be reduced but instead an agent should seek states where negentropy can continuously flow. This will lead to states that are always on the border between chaos and order. Enough free energy is needed, in order to make significant changes to future states and future entropy. This means that causal entropic forces move the state into a region where high amount of free energy is available. This shows a similarity to empowerment, where

the number of possible future states is maximised and agents similarly collect and store potential energy in a number of environments. In [69] the causal entropic force theory is applied to a range of environments including a pendulum, a particle in confined space but also collaboration and tool use experiments.

2.2.3 *Free energy principle*

In [67, 68] a joint optimisation of generative model and action generators are discussed. The cost function is defined as a variational free energy and can be related to model evidence as well as to information gain. This is another example of the connection between learning a model of the environment and the connections to exploration or curiosity. Exploration is handled well by this cost by preferring states which reduce uncertainty of the observation model instead of only seeking unseen states.

2.2.4 *Crooks fluctuation theorem*

Most thermodynamic and entropic relations are applicable to systems in a so-called near equilibrium state, where the effects are linear. The expressions related to entropy production however can cover also systems far away from equilibrium. In [66] the following equality was derived, linking the forward and backward probability of a process to its entropy production:

$$\ln \frac{p(A \to B)}{p(B \to A)} = \omega = \frac{1}{k_B T}(W_{A\to B} - \Delta F).$$

The right hand part comes from the first law of thermodynamics where the total entropy production ω depends on the work done on the system $W_{A\to B}$ and the difference in free energy ΔF. The unit of this entropy change is measured in nats. T is defined as the temperature in Kelvin and k_B is the Boltzmann constant. Requirements for the systems are that they shall be stochastic but microscopically reversible. The term free energy is used for describing the amount of energy that can be used for performing work. Energy alone can not be used to quantify how much work is possible to be done. As already mentioned, Schroedinger explicitly stated in [70] that he renamed free energy to negentropy to avoid confusion. The relation between free energy and irreversibility described by the ratio between the probability distribution of the forward and the backward process forms a mathematical definition of Schrödingers negentropy.

2.3 INFORMATION THEORY AND PHYSICS

In the following chapter different theories about the connections between physical processes and information theory will be explored. One observation will be, that certain quantities like free energy or change in entropy can seamlessly be transferred from physics to an information theoretic change in information content with simple unit conversions. With this background empowerment can be connected to physical interactions and energy exchange. It also shows us the fundamental limitations of information processing through the Landauer principle.

First, the origin of physical entropy is introduced followed by experiments on physical processes which change their information content. Second, more general relations between energy, entropy and information are shown together with a mathematical link to Schrödingers negentropy from section 2.2.2. This then leads to the integration of transfer entropy, which can be considered as the internal measurement of the empowerment formulation, which will later be used as the cost function for controlling an agent.

2.3.1 *Szilard engine*

Other experiments tried to explain entropy production and later got supported by their connection to information theory. The Maxwell daemon is a hypothetical system which is able to circumvent the second law of thermodynamics. It has an overview about all particles confined in a box and controls a wall separating the box in two halves. As soon as a single particle is about to traverse the two halves in a certain direction the daemon opens the wall, whereas in the case of a particle travelling in the other direction, the wall will be closed. This procedure collects all the particles in one half, effectively reducing the entropy of the overall system and breaking the second law.

Szilard used a simplified system [65] with a single particle where the daemon is able to retrieve energy from the procedure, also called the *Szilard engine*. This time a piston takes the role of the wall and gets moved depending on the particle location such that it can move the piston for producing work. Szilard noticed that for moving the piston the daemon needs to store information about the position of the particle in its memory for selecting the right action, resetting the piston or not. This is a binary information since the particle can be either left or right of the final piston position. In the next round this bit of information needs to be erased, for making space for the current state of the particle, otherwise the whole past sequence of states has to be stored to conserve the second law. The second law of thermodynamics will only hold if this bit erasure cancels out with the energy gained by the piston. This theoretical experiment showed a first linkage between information processes and physics.

2.3.2 *Landauer's principle*

Landauer [64] argued that any computation and in particular any logical operator that is not reversible leads to a dissipation of a small amount of energy. Without this energy emission, the whole history of computation would need to be stored, otherwise the system would not be reversible at the lowest level. He also predicted that this amount of energy must be in the order of $k_B T$, with k_B the Boltzmann constant and T temperature.

The simplest operation for an irreversible process would be a function which forces a system with two possible states 0 and 1 into the same state 0. This process would be called bit erasure. In [64] a particle trapped in one of two wells is presented, defining the binary state of the system. One can now apply forces to push the particle in either direction and change the state of the system. If now a force is exerted in a certain direction then the system will always end up in the same final

binary state. This happens independently from the initial state and is effectively deleting one bit of information. Classical mechanics stops us from doing this irreversible transition and dictates that a specific amount of energy of $k_B T \ln 2$ has to be dissipated. The amount of energy is a result of the thermodynamic definition of entropy $S = k_B \ln W$ from Boltzmann. Since the number of states W was reduced from 2 to 1, the entropy was reduced by $-k_B \ln \frac{1}{2}$. This entropy decrease leads to the dissipation of energy with the amount of $k_B T \ln \frac{1}{2}$. In actual computers, this amount of energy is so tiny that it is negligible in comparison to the regular heat generation. But it paved the way for the connection between entropy in information theory and the entropy production in physical systems.

These theoretical results were later experimentally verified [63] by trapping a silica bead by an optical tweezer. The two wells were created by switching the tweezer and the bit erasure triggered by biasing the potentials, such that the particle always falls into one of the two states. They have shown that the average dissipated heat saturates at the Landauer limit.

This insight can be connected to the more general Crooks fluctuation theorem from section 2.2.4, where a bit erasure would affect the ratio of forward and backward probability. Similar as in fig. 2.1b a deletion of information has to result in a different reverse probability.

2.3.3 *Transfer entropy*

In a previous section it was shown that the Crooks fluctuation theorem can compute entropy production from the system transition probability distributions. The ratio of forward and backward transition probability describes the total entropy production as a combination of difference in free energy and work done on the system. The equations do not explain how to compute the individual parts of the total entropy production. For distinguishing between the internal entropy production and the external heat dissipation more information about the transition process is needed. Transfer entropy might be a way to describe and split both terms by using system transitions with additional conditions.

Schreiber [62] introduced *transfer entropy* as a replacement to the usual mutual information used for measuring. One problem of mutual information is that it does not measure directional or dynamical information. Transfer entropy is a difference between two entropies h_X and $h_{X,Y}$ defined as follows:

$$h_X = -\sum_{\mathbf{x}_{n+1}, \mathbf{x}_n^{(k)}} p\left(\mathbf{x}_{n+1}, \mathbf{x}_n^{(k)}\right) \log_2 p\left(\mathbf{x}_{n+1} \mid \mathbf{x}_n^{(k)}\right)$$

$$h_{X,Y} = -\sum_{\mathbf{x}_{n+1}, \mathbf{x}_n^{(k)}, \mathbf{y}_n^{(l)}} p\left(\mathbf{x}_{n+1}, \mathbf{x}_n^{(k)}, \mathbf{y}_n^{(l)}\right) \log_2 p\left(\mathbf{x}_{n+1} \mid \mathbf{x}_n^{(k)}, \mathbf{y}_n^{(l)}\right).$$

Which results in the following formulation for transfer entropy:

$$\begin{aligned} T_{Y\to X} &= h_X - h_{X,Y} \\ &= \sum_{\mathbf{x}_{n+1},\mathbf{x}_n^{(k)},\mathbf{y}_n^{(l)}} p\left(\mathbf{x}_{n+1},\mathbf{x}_n^{(k)},\mathbf{y}_n^{(l)}\right) \log_2 \frac{p\left(\mathbf{x}_{n+1} \mid \mathbf{x}_n^{(k)},\mathbf{y}_n^{(l)}\right)}{p\left(\mathbf{x}_{n+1} \mid \mathbf{x}_n^{(k)}\right)} \\ &= \mathbb{E}_{p(\mathbf{x}_{n+1},\mathbf{x}_n,\mathbf{y}_n)}\left[\log_2 \frac{p(\mathbf{x}_{n+1} \mid \mathbf{x}_n, y_n)}{p(\mathbf{x}_{n+1} \mid \mathbf{x}_n)}\right] \\ &= \mathbb{E}_{p(\mathbf{x}_{n+1},\mathbf{x}_n,\mathbf{y}_n)}[t_{Y\to X}] \end{aligned} \tag{2.3}$$

It measures the amount of information gained by including the information source Y, since it is the difference between the amount of information in X and the amount of information of the joint of X and Y. The regular transfer entropy is a weighted average (eq. (2.3)) of a local transfer entropy $t_{Y\to X}$, defined in [61]:

$$t_{Y\to X} = \log_2 \frac{p(\mathbf{x}_{n+1} \mid \mathbf{x}_n, y_n)}{p(\mathbf{x}_{n+1} \mid \mathbf{x}_n)} \tag{2.4}$$

In contrast to transfer entropy this local measure can also become negative in cases, when the additional information y misinforms about the resulting state.

In [59, 60] it is shown that there is a connection to thermodynamics by setting local transfer entropy equal to the energy dissipated by an irreversible process. The reasoning towards this equality also hints towards several interpretations on individual parts of transfer entropy. The uncertainty in $p(x_{n+1} \mid x_n, y_n)$ describes just the entropy production of the unobserved variables since it is conditioned on the context. This internal entropy increase can come from friction where the exact interaction between particles is not known and the process can thus be modelled only in a stochastic way. The other distribution $p(x_{n+1} \mid x_n)$ describes all the sources of entropy from above plus the entropy change through y_n. In this case, y_n got marginalised out, so that this distribution models the effect of all possible contexts on the system transition. Therefore the difference between the two entropies describes the external entropy production while the internal entropy production gets cancelled out:

$$t_{Y\to X} = \frac{\Delta S_{ext}}{k_B \log 2}$$

Compared to the Crooks theorem, where the sum of both external and internal entropy production is computed, here these two terms can be split as long as the actual interaction with the heat-bath is known, modelled by $p(x_{n+1} \mid x_n, y_n)$. The external entropy measured by the transfer entropy can not be set equal to the one appearing in the Crooks fluctuation theory equation because in the former case only the interactions with the y system are considered for computing the external entropy change. The additional information y takes the part of the actual interaction with the environment. It would be impossible to compute the actual difference between internal and external entropy production without this model of the interaction.

2.3.4 *Empowerment and transfer entropy*

This leaves us with a connection between empowerment and external entropy production. When the state $\mathbf{x}_n$ is used as the sensor reading and $\mathbf{y}_n$ is seen as an action sending information into the environment, then empowerment is the expected local transfer entropy (eq. (2.4)) under the source distribution and a single transition step. Empowerment therefore measures the maximum possible expected change in negentropy or free energy. A policy exploiting this reward will accumulate free energy, which allows the agent to release it later in the form of work on the system.

The introduction of transfer entropy and its relation to negentropy also opens up new possibilities for interpreting especially the one-shot transfer entropy as the reward. A similar reward is used in [58]. In contrast to empowerment, the mentioned publication has no split between policy and source distribution, meaning it uses the one-shot transfer entropy directly as the reward in a Bellman recursion for directed information. Additionally, they show a method for mixing regular extrinsic value with the information reward.

2.3.5 *Theories about connections to Life*

Independent to the published connections between empowerment and living organisms [74], there are several other attempts linking Crooks fluctuation theorem and entropy production to reproduction, metabolism and social behaviour [57, 69, 70]. In [57] the Crooks fluctuation theorem is used for explaining the reproduction of bacteria by assigning a higher probability to reproduction than for random desintegration. During their derivation another version of the Crooks fluctuation theorem appeared, which describes the inequality between entropy and reverse transition

$$\beta\langle\Delta Q\rangle + \ln p(\to \mathrm{I} \mid \mathrm{II}) + \Delta S_{int} \geqslant 0.$$

In this setting the entropies ΔS_{tot} and ΔS_{int} are defined as Shannon entropy and have the unit nat. β is defined as the inverse temperature $\frac{1}{k_B T}$. Together with the average entropy production

$$\langle\Delta S_{tot}\rangle = \Delta S_{int} + \beta\langle\Delta Q\rangle,$$

imposes some interesting results: By the fact that $p(\to \mathrm{I} \mid \mathrm{II}) \leqslant 1$, it holds that $\langle\Delta S_{tot}\rangle \geqslant 0$. Also ties to the previous relationship of transfer entropy and internal entropy production can be seen in these formulations. The term ΔS_{int} does not need to be averaged over multiple transitions compared to the other term $\langle\Delta Q\rangle$.

3 PROBABILISTIC MODELLING

Even though the presented intrinsic motivation definitions seem simple, all of them require the evaluation of a probabilistic transition model which represents not only the agents dynamics but also those of the environment. This hindered the previously missing experimental verification of above theories on real robotic systems. While modelling the dynamics of stiff robot arms is well solved, modelling soft robots or the environment usually represents a big challenge. Especially predicting the results of interactions with a dynamic environment is hard. The limited adaptiveness and learning for control limits us to use robots where sensors are calibrated and sensor quality is preferred over quantity. With learning and the autonomous behaviour of intrinsic motivation one can open a wider range of new robots for future applications.

The focus of this thesis lies therefore in learning accurate dynamical models from raw sensor data. Recent advances in approximate inference are not only helpful for learning generative models but also for estimating the expected value of information theoretic quantities as they appear in the intrinsic motivation formulations. The intractabilities found for computing intrinsic motivation are also related to the same intractabilities of learning generative models. Even optimising control policies can be cast into the framework of probabilistic inference as argued in [56]. In [55] the authors go even further and formulates several intrinsically motivated theories together with control into one inference framework.

The following section aims to give an overview over current unsupervised probabilistic modelling techniques and their limitations. The focus will be on learning generative models that are able to efficiently sample from $p(\mathbf{x})$, a distribution representing the underlying true data distribution. Another desired feature is the inclusion of action sequences and their influence on the predicted sensor values. In the following several methods for learning these probabilistic models will be presented.

3.1 CONTINUOUS LATENT VARIABLE MODELS

In the following it is assumed that the data points $\mathbf{x}$ of the desired model $p(\mathbf{x})$ have corresponding hidden representations with a probabilistic mapping $p(\mathbf{x} \mid \mathbf{z})$. The process of sampling then involves taking a sample from a prior distribution $p(\mathbf{z})$ and using this latent state as a condition for sampling from the generative model $p(\mathbf{x} \mid \mathbf{z})$. This process then creates samples from the data distribution $p(\mathbf{x})$. The distribution of the data points can therefore be expressed by taking the generative model and marginalising out the latent states:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}. \tag{3.1}$$

Several things of this distinction between observed and hidden variables are of interest. First the hidden state itself can be used as some feature representation

for preprocessing. But also the generative mapping can be used as a simulator for generating unseen data points with the same distribution as the recorded data. When the dataset is consisting of time series data and the generative process is conditioned on the action sequence, then whole trajectories can be generated. The observed influence of the actions on the trajectory can then be used to make decisions for optimising the outcome according to some control cost.

However, the problem is the intractability of inferring the hidden representation or especially finding the posterior distribution $p(\mathbf{z} \mid \mathbf{x})$. Most of the time, the generative model is even unknown and has to be learned as well. Having the inferred latent representation for each data-point eases the generative model learning part as it transforms the fitting of $p(\mathbf{x} \mid \mathbf{z})$ into a supervised learning task. Assumptions on the generative model can make the learning process tractable. The simplest is a linear assumption, referred to as probabilistic principal component analysis (PPCA) [54]:

$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I}).$$

The latent state $\mathbf{z}$ is mapped to data points $\mathbf{x}$ by the linear transformation $\mathbf{W}\mathbf{z} + \mu + \epsilon$ with additive noise $\epsilon \sim \mathcal{N}(\epsilon \mid \mathbf{0}, \sigma^2\mathbf{I})$. With a prior defined as $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{1})$ the marginal distribution is Gaussian as well:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} = \mathcal{N}(\mathbf{x} \mid \mu, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}).$$

This marginal is now easy enough to be solved either analytically or by iterative methods such as the EM-algorithm. PPCA can be seen as a probabilistic version of principal component analysis (PCA) when the observation noise standard deviation σ goes towards zero.

Another model with linear transformation but a non-Gaussian prior is independent component analysis (ICA) [54]. The prior is meant to factorise into independent components with usually heavy-tailed distributions such as:

$$p(z_i) = \frac{1}{\pi \cosh(z_j)},$$
$$p(\mathbf{z}) = \prod_i^N p(z_i).$$

The transformations to data points $\mathbf{x}$ is again done via a linear transformation $\mathbf{x} = \mathbf{W}\mathbf{z}$, this time with no additive noise or bias. With the help of the change of variable theorem the marginal density can be defined as:

$$p(\mathbf{x}) = \frac{1}{|\det \mathbf{W}|} p\left(\mathbf{W}^{-1}\mathbf{z}\right).$$

Most of the systems that are dealt with in the following chapters will show highly non-linear dynamics, making models with linear transformations like PCA or ICA unsuitable. This thesis will be focused on models that can be applied to arbitrarily complex data distributions while still being efficient.

3.2 INTRACTABILITY OF SAMPLING

While it is possible to evaluate the marginal likelihood analytically for simple linear models, computing the integral for non-linear models with high dimensional spaces is intractable. One attempt would be to solve the integral over all latent values by Monte-Carlo sampling:

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} = \mathbb{E}_{p(\mathbf{z})}[p(\mathbf{x} \mid \mathbf{z})] \approx \sum_{z \sim p(\mathbf{z})} p(\mathbf{x} \mid \mathbf{z}).$$

Here latent states from the prior $p(\mathbf{z})$ are sampled and then used to evaluate the generative distribution $p(\mathbf{x} \mid \mathbf{z})$. The density will show low probabilities for all pairs of $\mathbf{x}$ and $\mathbf{z}$ that do not fit together. The majority of the marginal likelihood will be made of high probabilities from data points with their corresponding latent state. This increases the amount of samples needed to properly evaluate the marginal likelihood. It would be much more efficient to use a proposal distribution for reducing the search space together with procedures for keeping the likelihood evaluation correct. This proposal method can then even be conditioned on the data-point $\mathbf{x}$.

A method for estimating expectations in such a way is *importance sampling*. The procedure starts by sampling from the proposal q, then weights $w_i = \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)}$ for the samples are computed followed by normalizing the weights. The function over which the expectation is being computed can then be evaluated at the sampled points $\mathbf{x}_i$ and weighted with the normalised ratios between target and proposal probability. This weighting can be derived by introducing the proposal q as a multiplication with 1 to the expectation. Especially for the case of marginal likelihoods the following equations hold:

$$\begin{aligned} p(\mathbf{x}) &= \int_{\mathbf{z}} p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} = \mathbb{E}_{p(\mathbf{z})}[p(\mathbf{x} \mid \mathbf{z})] \\ &= \int_{\mathbf{z}} \frac{p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})} q(\mathbf{z} \mid \mathbf{x})\, d\mathbf{z} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\frac{p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{x})}\right]. \end{aligned}$$

This importance sampling based estimation of the marginal likelihood is suited for evaluating the current likelihood of a model, but it is not suited for training model parameters. The distribution inside the expectation is defined over the whole dataset, which makes is impossible to handle the dataset in mini-batches. The usual factorisation of the distribution $p(\mathbf{x} \mid \mathbf{z})$ into mini-batches is not helpful, because a logarithm can not be applied for splitting these factors into additive parts.

Other methods for sampling from the posterior distribution include Markov chain Monte Carlo (MCMC) methods, such as Metropolis Hastings (MH) [53] or Hamiltonian Monte Carlo (HMC) [51, 52]. The benefit of these methods is, that they are unbiased given that they are run for a high number of iterations.

The MH algorithm [53] uses a Markov chain with a transition operator $q(\mathbf{z}_t \mid \mathbf{z}_{t-1})$. After each of these steps an acceptance probability A is computed as:

$$A = \min\left[\frac{p(\mathbf{x}, \mathbf{z}_t) q(\mathbf{z}_{t-1} \mid \mathbf{z}_t)}{p(\mathbf{x}, \mathbf{z}_{t-1}) q(\mathbf{z}_t \mid \mathbf{z}_{t-1})}, 1\right].$$

The next state in the chain is then kept with the probability A, otherwise the old state is used for the next sampling step. This acceptance step guides the random walk of the Markov chain, such that $p(\mathbf{x}, \mathbf{z}_t)$ is maximised, which represents the joint distribution $p(\mathbf{z}_t)p(\mathbf{x} \mid \mathbf{z}_t)$ of the graphical model. The chain will then converge to the true posterior distribution $p(\mathbf{z} \mid \mathbf{x})$.

HMC [51, 52] is another method that can be used for sampling from a complex distribution such as the posterior for a generative graphical model. It uses a simulation of particles with movements defined by a potential and a kinetic energy function. The state of these particles is composed of a momentum p and a position q and are associated to their respective kinetic energy $K(p)$ and potential energy $U(q)$. No friction is acting on the particles and the total energy, also called *Hamiltonian* $H(p, p) = U(q) + K(p)$, always stays the same while movement is generated by shifting energy between kinetic and potential energy. The following equations describe the evolution of position and momentum over time:

$$\frac{dq_i}{dt} = \frac{\delta H(q,p)}{\delta p_i} = \frac{\delta K(p)}{\delta p_i}$$
$$\frac{dp_i}{dt} = -\frac{\delta H(q,p)}{\delta q_i} = -\frac{\delta U(q)}{\delta q_i}$$

For solving this differential equation numerically usually the leapfrog method is used. Compared to the Euler method, the leapfrog method preserves the volume exactly. The leapfrog steps are defined as follows:

$$p_i^{(t+\epsilon/2)} = p_i^{(t)} - \frac{\epsilon}{2}\frac{\delta U(q)}{\delta q_i}(q^{(t)})$$
$$q_i^{(t+\epsilon)} = q_i^{(t)} - \epsilon\frac{\delta K(p)}{\delta p_i}(p^{(t+\epsilon/2)})$$
$$p_i^{(t+\epsilon)} = p_i^{(t+\epsilon/2)} - \frac{\epsilon}{2}\frac{\delta U(q)}{\delta q_i}(q^{(t+\epsilon)}).$$

Hamiltonian dynamics does not only have a physical interpretation, but can also be used for sampling from a probability distribution [52]. For this purpose the potential energy is defined as the logarithm of the probability density:

$$U(\mathbf{z}) = -\log p(\mathbf{x}, \mathbf{z}).$$

The kinetic energy is usually set as

$$K(p) = \frac{p^T M^{-1} p}{2}.$$

M is called the mass matrix. Hamiltonian Dynamics is then a substitute for the proposal distribution in the regular MH algorithm. The acceptance step is still being used but only corrects for numerical error of the leapfrog integration. With almost all transitions accepted, HMC is much more efficient than MH.

3.3 STOCHASTIC GRADIENT VARIATIONAL BAYES

Instead of inferring the hidden variable through the mentioned sampling procedures, an approximate posterior can be used. This posterior can be used to sample

from the latent space directly. This is possible since the usual complexity of the inference problem was shifted from the sampling domain to the approximate posterior domain.

One method for finding these approximate posteriors is stochastic gradient variational Bayes (SGVB) which has been proposed in [49, 50]. It is applicable to latent variable models by defining a variational lower bound on the log-likelihood $p(\mathbf{x})$, also called the evidence lower bound (ELBO). Starting out by defining $p(\mathbf{x})$ as the marginal likelihood by integrating out the latent states and introduce the approximate posterior as a fraction:

$$-\log p_\theta(\mathbf{x}) = -\log \int_{\mathbf{z}} p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) \mathrm{d}\mathbf{z} = -\log \int_{\mathbf{z}} \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})} p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) \mathrm{d}\mathbf{z}.$$

By using the Jensen inequality the denominator can be moved from the fraction into the logarithm leading to an expectation:

$$\begin{aligned} -\log p_\theta(\mathbf{x}) &\leqslant -\int_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \mathrm{d}\mathbf{z} \\ &= -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] + D_{\mathrm{KL}}\big(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z})\big) =: \mathcal{L}. \end{aligned} \tag{3.2}$$

3.4 REPRAMETRISATION TRICK

Applying methods from the deep learning domain, such as back-propagation [48] for maximising the lower bound w.r.t. the parameters ϕ and θ poses a problem when propagating errors through the sampling process of the expectation in eq. (3.2). The authors in [49] introduced the so called reparametrisation trick. It reformulates the expectations over distributions such that the parameters appear only in the value and not in the distribution. This only works when the sampling process of a distribution $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ can be expressed as another sampling step of an auxiliary variable $\epsilon \sim p(\epsilon)$ followed by a deterministic transformation $g_\phi(\epsilon, \mathbf{x})$. The expectation in eq. (3.2) can then be rewritten as:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] = \mathbb{E}_{p(\epsilon)}\big[\log p_\theta\big(\mathbf{x} \mid g_\phi(\epsilon, \mathbf{x})\big)\big].$$

3.5 VARIATIONAL AUTO-ENCODER

This loss function can also be interpreted as an auto-encoder with a sparsity regulariser and is called the variational auto-encoder (VAE). The reconstruction part $p(\mathbf{x} \mid \mathbf{z})$ of the loss is very similar to the regular mean squared error, especially in the case of Gaussian reconstruction distributions. The logarithm of this distribution leads to the same mean squared error term but with a scaling through by the variance of the Gaussian distribution. The KL-term $D_{\mathrm{KL}}(q(\mathbf{z}) \parallel p(\mathbf{z}))$ serves as a regulariser, since it measures how different the posterior and the prior are. Minimising this part would lead to the posterior getting more and more similar to the prior and transferring lesser information from the observation $\mathbf{x}$. Similar to the auto-encoder, both $p(\mathbf{x} \mid \mathbf{z})$ and $q(\mathbf{z} \mid \mathbf{x})$ are parametrised as neural networks. These networks would generate mean and variance of a Gaussian from the condition of the probability distribution. The parameters of the neural networks are then part

of θ and Φ of the loss function in eq. (3.2). The type of distribution parametrised by the neural networks is not limited to Gaussian distributions. Any distribution can be used where the sampling process is easily differentiable with respect to the distribution parameters. This is usually the case when the sampling process consists of a linear transformation of an uninformative distribution or if done by evaluating the cumulative distribution function with a uniform sample as input. This includes distributions like the Laplace, Exponential, Logistic or Cauchy. If more complicated distributions are required one can utilize normalizing flows from [47], where chains of transformations with easy to evaluate Jacobians are used.

The application of neuronal networks and stochastic gradient descend enables us to perform training of the model on current computing platforms like GPUs and make use of the many frameworks around machine and deep learning. Another benefit is that after the training the posterior distribution can be used as an efficient feed-forward model for performing inference.

Instead of plain neural networks also any differentiable function can be used. This enables the incorporation of structure to either recognition or generative models. This includes convolutional networks [46], sequential structures with variable length [44, 45] or auto-regressive and hierarchical structures [42, 43].

3.6 VARIATIONAL INFERENCE FOR HAMILTONIAN MONTE CARLO

In another publication [41] the SGVB framework got applied to MCMC. They found out that the acceptance ratio of the MH step of HMC can be added to the lower bound and be optimised together with the other parameters. Despite its importance for asymptotic convergence to the exact posterior distribution, the MH acceptance step is omitted in [41]. In [40] this acceptance step is added back into the algorithm. The structure of HMC does not leave a lot of room for optimisation except for predicting a good initial state, or optimising the mass matrix. Nothing prevents us from conditioning the creation of the matrix on the observation. A conditioning on the position is possible [39] but would make the algorithm unnecessarily complicated.

For training these free parameters of the initial state and the mass matrix, variational inference can be used, just as in [41]. The algorithm starts with a regular recognition model from VAE and uses it to predict the initial latent state of the Markov chain. Then several steps of the Hamiltonian dynamics transform this latent state into the final value to be evaluated by the generative model. When the intermediate states $\mathbf{y}$ of this Hamiltonian dynamics transformation are added to the regular lower bound $\mathcal{L}$, then following equations [41] hold:

$$\begin{aligned}
\log p(x) \geqslant \mathcal{L} &\geqslant \mathcal{L} - \mathbb{E}_{q(\mathbf{z}_T|\mathbf{x})}[D_{KL}(q(\mathbf{y} \mid \mathbf{z}_T, \mathbf{x}) \parallel r(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_T))] \\
&= \mathbb{E}_{q(\mathbf{y},\mathbf{z}_T|\mathbf{x})}[\log[p(\mathbf{x}, \mathbf{z}_T) r(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_T)] - \log q(\mathbf{y}, \mathbf{z}_T \mid \mathbf{x})] \\
&= \mathbb{E}_{q(\mathbf{y},\mathbf{z}_T|\mathbf{x})}[\log p(\mathbf{x}, \mathbf{z}_T) - \log q(\mathbf{z}_0 \mid \mathbf{x})] \\
&\quad + \sum_{t=1}^{T} \log[r_t(\mathbf{z}_{t-1} \mid \mathbf{x}, \mathbf{z}_t) / q_t(\mathbf{z}_t \mid \mathbf{x}, \mathbf{z}_{t-1})].
\end{aligned} \tag{3.3}$$

Here the intermediate state $\mathbf{z}_0, \mathbf{z}_1, \ldots \mathbf{z}_{t-1}$ are combined into $\mathbf{y}$. The purpose of the reverse distribution $r(\mathbf{y} \mid \mathbf{x}, \mathbf{z}_T)$ is to counteract the forward portability of

the Markov chain. Both the reverse and the forward probability can be factorised into local transition as shown in eq. (3.3). When the chain is balanced with $\frac{r_t(\mathbf{z}_{t-1}|\mathbf{x},\,\mathbf{z}_t)}{q_t(\mathbf{z}_t|\mathbf{x},\,\mathbf{z}_{t-1})} = 1$ the usual lower bound can be used without auxiliary variables. The authors in [47] recognised that this MCMC chain transition can be described as a flow and the acceptance ratio in eq. (3.3) relates to the distortion measured by the logarithm of the transformation determinant, that needs to be included in the lower bound.

3.7 APPROXIMATE INFERENCE FOR OPTIMAL CONTROL

It is well known [37, 38, 56], that optimal control and reinforcement learning can also be cast as an inference task. Instead of minimising expected cost, the success of a trajectory gets formulated as a Bayesian network:

$$p(r_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t) = \exp\left(-\beta \mathcal{C}_t(\mathbf{x}_t, \mathbf{u}_t)\right).$$

The scalar parameter β is used as an inverse temperature. Marginalising out the trajectory leaves us with:

$$\begin{aligned} p(r_t = 1) &= \int p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \prod_{t=1}^{T} \exp\left(-\beta \mathcal{C}_t(\mathbf{x}_t, \mathbf{u}_t)\right) \\ &= \int p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \prod_{t=1}^{T} p(r_t = 1 \mid \mathbf{x}_t, \mathbf{u}_t) \\ &= \int p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) p(r_t = 1 \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}). \end{aligned}$$

By introducing an approximate posterior for over trajectories, a lower bound can be defined on the success evidence [38]:

$$\begin{aligned} \log p(r_t = 1) =& \log \int p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})\, p(r_t = 1 \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \\ \geqslant& \int q_\pi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \log \frac{p(r_t = 1 \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})}{q_\pi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \\ =& \mathbb{E}_{q_\pi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\sum_{t=1}^{T} -\beta \mathcal{C}_t(\mathbf{x}_t, \mathbf{u}_t) \right] \\ & - D_{\mathrm{KL}}(q_\pi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \,\|\, p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})) \end{aligned}$$

When $q_\pi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ and $p_{\pi_0}(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ are factorised in the same way, the KL term can be simplified:

$$\log p(r_t = 1) \geqslant \mathbb{E}_{q_\pi(\mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\sum_{t=1}^{T} -\beta \mathcal{C}_t(\mathbf{x}_t, \mathbf{u}_t) \right] - D_{\mathrm{KL}}(\pi \,\|\, \pi_0).$$

This connection of optimal control and approximate inference enables us to use the tools variational inference, stochastic gradient descend and the reparametrisation trick for learning control policies.

Part II

MAIN WORK

4

EFFICIENT INFERENCE

As discussed in the previous chapter methods for creating probabilistic models of datasets, specifically for making predictions into the future are needed for estimating intrinsic motivation or performing control. In this chapter the focus is on the analysis and modelling of static data-points, meaning that time-series data is not handled yet. Unsupervised learning methods are used and applied to high dimensional datasets and later to analyse the resulting hidden representation. The following will focus on the class of methods that are able to train non-linear probabilistic models efficiently. Here new methods for joint training of generative models and efficient approximative inference models are explored.

Unsupervised methods are usually used as a preprocessing step for supervised tasks. This stems from their tendency to compress data into a more compact format while ensuring that all information is still contained in the hidden representation. The procedure of preprocessing involves the training of the unsupervised algorithm on raw data. Most often, PCA or ICA [36] is used. The recognition part of the algorithm, which transforms raw data into the hidden representation, gets used as the preprocessing step. The hidden representation is then used as the input to the next usually supervised task. The popular methods PCA and ICA are limited in their ability to represent complex datasets. PCA can only represent Gaussian distributed datasets well, while ICA can handle more complicated distributions but is still limited to orthogonal transformations of the data. Some data-sets however feature more complicated relationships that can not be handled by linear transformations. In the following the performance of the variational auto-encoder is tested on highly non-linear tactile data.

The following sections contain results, figures and parts of the publication [35].

4.1 VARIATIONAL AUTO-ENCODER FOR TACTILE DATA

In [35] unsupervised techniques got applied to tactile sensor data with the goal to improve robotic in-hand manipulation. The tactile sensors used here employ a sensor array together with a deformable exterior which makes the data highly non-linear and hard to calibrate to physical values. Calibrating each sensor independently to physical ground truth would be necessary for using it in a classical control setup, but would make the widespread application of these new sensors difficult. Usually a high amount of sensor and ground truth readings have to be recorded such that the learning algorithm can generalise and account for variations in the environment. An unsupervised preprocessing step makes it possible to bring the raw sensor data into a compressed format where only a small amount of labelled data is needed for creating a generalising mapping to ground truths. Unlabelled data is cheap when recorded autonomously by robots, without the need to restrain the sensor in a calibration setup. Apart from the calibration problem, these sensors suffer from drift and temperature effects and would need constant updates of their

calibration. A goal is therefore to have unsupervised methods which learn a model of the tactile sensors directly from raw data and are able to transform the data into a latent representation utilisable as a state for robot control as seen in [34]. A promising algorithm for preprocessing the raw tactile data is the VAE, as this algorithm is able to find non-linear relations between the hidden code and data-points and is able to model the dataset distribution probabilistically.

4.2 COMPARISON TO PREVIOUS WORK

Previous work on processing tactile data includes [32, 33] which concentrated on supervised training of the physical ground truths and also used those features for control. In [31] a preprocessing pipeline is presented which needed to be carefully designed and tuned. Apart from being very tedious, supervised training poses some more profound problems in the later application of those features. The manual selection of different external modalities might limit the information as originally received by the sensor. There might be unknown features which can not be easily defined by the usual ground truths like force, angles and surface texture. A piece of evidence can be seen in [30] where they recorded micro vibrations over different surface textures while varying the force level. This force variation changed the vibration spectrum so much that surface type and force can not easily be separated anymore. The usual methods like ICA and PCA already found their application for tactile data [29]. The preprocessed raw data were then mapped to ground truths. Also in this work they questioned if raw sensor data should rather be used for control tasks, but as stated in [34] it is necessary to transform raw data into a latent representation since the high dimensionality is problematic for current policy search methods.

4.3 EXPERIMENTAL SETUP

In the following experiments two tactile sensors with fundamentally different measurement principles were used. The BioTac [28] sensor uses 19 electrodes in an electrically conductive, but resistive liquid to measure deformation encased in a silicon skin. The electrodes are embedded in an internal stiff core. The second sensor from the iCub robot features a capacitive measurement of several electrodes against an outer ground plane [27]. The ground plane and the dielectric are the only soft and flexible components, making the iCub sensor much stiffer than the BioTac. Additionally to the 19 taxel the BioTac also features temperature and pressure sensors, but those were omitted in the experiments to make the comparison between the two sensors fairer.

The sensor data were recorded in a robotic setup, where ground truths like forces and angular positions can be accurately applied and measured. Also different curvatures and Shore hardnesses were applied to the tactile sensor. The different angles were set by using stepper motors controlled by an field programmable gate array (FPGA) PCI-Card (Mesa 5i25) on a regular PC running the Matlab Simulink Real-Time XPC operating system. The FPGA card enables the use of low level peripheral communication protocols like serial peripheral interface (SPI) and serial connections, which would otherwise not be available on regular desktop computers.

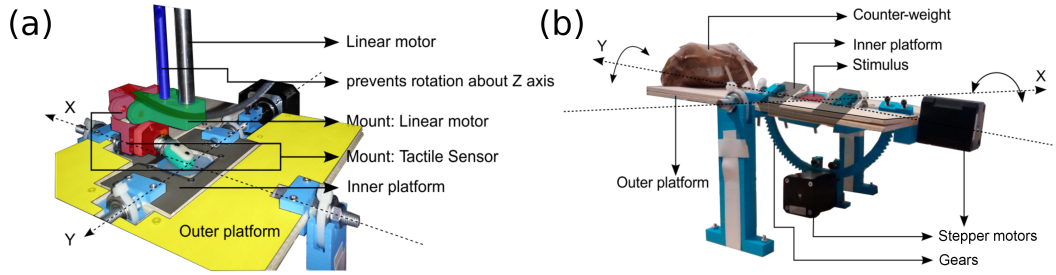


Figure 4.1: Structure of robot used for recording reproducible tactile sensor dataset. (a) Tactile sensor touching centre of 2 DOF gimbal platform. Material samples can be placed in the gimbal centre. (b) Detailed side view of gimbal platform.

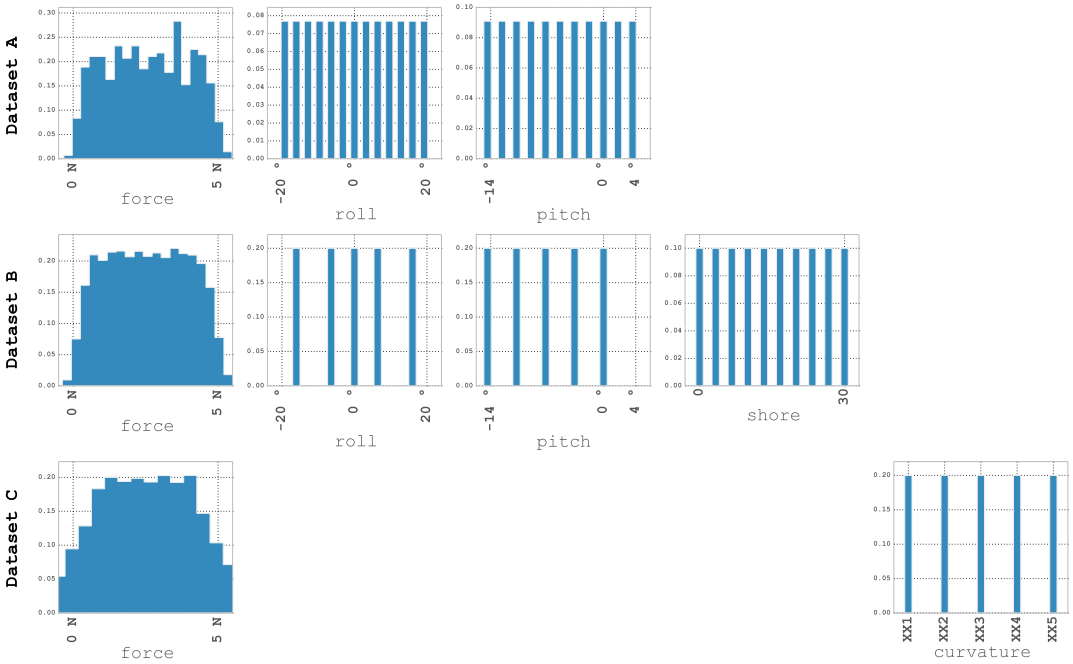


Figure 4.2: Distribution of ground truth recorded with robotic setup in fig. 4.1. Forces are controlled to be uniform in the range between 0 N and 5 N. The roll and pitch angles are set by the two stepper motors in the gimbal platform. (Dataset A) Set of recording with only variations in surface angles and force levels (Dataset B) Dataset with variation in surface angles, force levels and changing Shore hardness by manually switching samples on the gimbal platform. (Dataset C) Recording of curvature samples from fig. 4.3 with varying force levels but fixed surface angles.

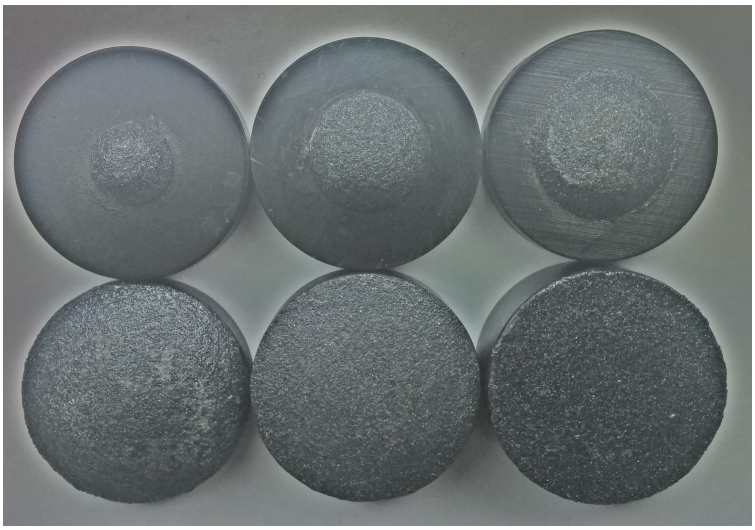


Figure 4.3: Curvature samples used for generation of dataset C (fig. 4.2). Starting from the top left corner the curvature values are 5 mm, 7.5 mm, 10 mm, 20 mm, 40 mm, flat.

The gimbal system for applying different angles is shown in fig. 4.1. The force magnitude is controlled in a closed loop with a force-torque sensor (ATI Nano 17) to keep the force distribution between the two sensors as similar as possible. The different values of each feature are kept as uniform as possible to reduce the amount of artefacts in the preprocessing due to unbalanced stimuli. Angles are open loop controlled to spread uniformly the range between $-19°$ to $19°$ in the roll direction and from $-3.6°$ to $18°$ in the pitch direction. For a uniform distribution of Shore hardness samples with a two component silicone (Smooth On Ecoflex) were fabricated which comes in a discrete set of Shore hardnesses while intermediate values were created by adding special softener chemicals. They were then calibrated with a Shore A measurement tool and selected to represent a sample set with equal distances between levels of hardness. The curvatures came in a predefined set of sizes as seen in fig. 4.3. The radii of the six curvatures range from 40 mm to 5 mm. The Shore hardness and curvature samples are placed in the centre of the gimbal platform and then applied with the force profile of a linearly increasing force from 0 N to 5 N with a follow-up decrease to 0 N at the same rate. Only the curvature recordings contain no change of platform angles. This setup was repeated for all gimbal angles which created a dataset where all combinations of the discrete states in fig. 4.2 are present in the dataset.

4.4 MODEL LEARNING

This dataset was then used to train a VAE as explained in section 3.3. A common network structure with the generative model was used:

$$\log p(\mathbf{x} \mid \mathbf{z}) = \log \mathcal{N}(\mathbf{x};\, \mu(\mathbf{z}), \sigma^2 \mathrm{I}). \tag{4.1}$$

It describes a Gaussian distribution where only the mean $\mu(\mathbf{z})$ is generated by the underlying neural network with two layers of 512 hidden states each. The

Table 4.1: Regression results on raw and unsupervised preprocessed data for BioTac sensor on surface dataset. Best result for each pair in bold.

	Linear Regression		Decision Tree Regression	
	raw	latent	raw	latent
Force [N]	0.32	**0.27**	**0.34**	0.36
Pitch [°]	4.02	**3.18**	**3.69**	3.70
Roll [°]	7.03	**4.79**	6.90	**5.96**

Table 4.2: Regression results on raw and unsupervised preprocessed data for iCub sensor on surface dataset. Best result for each pair in bold.

	Linear Regression		Decision Tree Regression	
	raw	latent	raw	latent
Force [N]	0.86	**0.75**	0.87	**0.81**
Pitch [°]	4.34	**2.92**	3.34	**2.80**
Roll [°]	5.23	**3.86**	**3.73**	4.84

standard deviation is shared across all data dimensions and is independent of the latent variable $\mathbf{z}$. In the recognition model, both mean and variance are dependent of the neural network output:

$$\log p(\mathbf{z} \mid \mathbf{x}) = \log \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma(\mathbf{x})^2 I).$$

This network has the same internal structure as the generative model and all networks use the identity function as the output transfer function. All the input data were normalised by subtracting the mean and dividing by the standard deviation.

Both BioTac and iCub data use the same network structure but different transfer functions with sigmoid transfer functions for the BioTac sensor and rectifier for the iCub sensor. The network for the iCub used adadelta[26] with step rate 0.1 for optimisation while for BioTac rmsprop[25] with step-rate 0.001 was used.

4.5 PREPROCESSING EVALUATION

For a first evaluation of the preprocessing quality the raw sensor data were compared to the latent representation after the preprocessing step in a classification and regression test. Linear regression and decision trees where trained to predict forces and angles.

The results of these prediction tasks are shown in the tables 4.1 to 4.4. In all cases linear regression on the latent representation outperforms the one trained on raw data. Decision trees on the other hand are, independent on the representation or tactile sensor, as good as the linear regression on latent representations. This is a foreseeable result, as decision trees can model much more complicated transformations compared to the linear regression algorithm. For the latent representation, the complicated transformations are already applied by the VAE preprocessing

Table 4.3: Regression results on raw and unsupervised preprocessed data for BioTac sensor on Shore dataset. Best result for each pair in bold.

	Linear Regression		Decision Tree Regression	
	raw	latent	raw	latent
Force [N]	**0.30**	0.34	**0.33**	0.37
Pitch [°]	3.45	**2.93**	**3.47**	3.62
Roll [°]	6.19	**4.48**	**6.19**	6.45
Shore [Shore A]	2.07	**1.94**	2.21	2.21

Table 4.4: Regression results on raw and unsupervised preprocessed data for iCub sensor on Shore dataset. Best result for each pair in bold.

	Linear Regression		Decision Tree Regression	
	raw	latent	raw	latent
Force [N]	0.56	**0.49**	0.68	**0.67**
Pitch [°]	2.45	**1.58**	2.50	**2.09**
Roll [°]	4.39	**2.47**	3.53	**3.47**
Shore [Shore A]	1.99	**1.37**	2.35	**1.76**

step. Figure 4.4 shows that both sensors already depict a highly non-linear mapping between forces and tactile sensor outputs. These plots show four applications of the 5N force profile at different angles. All together, it demonstrates that this preprocessing simplifies the data enough such that a linear transformation is able to map the tactile data to physical quantities. Also it is not deleting any important information in the data, as seen through the decision tree results, where no preference over raw or preprocessed data is visible.

The preprocessing was also evaluated on a classification task, where the six curvature classes are used as targets. Figure 4.5 and fig. 4.6 show the results for this dataset. The left side of each figure was generated by training linear classification on raw sensor data while the right side was trained on the latent representation. The results are represented by confusion matrices where a dark diagonal indicates correct classifications and off-diagonal squares show a grey tone colour when the wrong class gets predicted. The latent representation is much better suited for linear classification than raw sensor data.

4.6 DISCOVERY OF PHYSICAL QUANTITIES

The improved performance of linear regression on the latent representation suggests that the relevant information is already stored in a linear fashion in the hidden space. A close look on those latent dimensions whose distribution differs a lot from the prior distribution is presented in fig. 4.7. Dimensions that show a distribution similar to that of the prior distribution will contain less information about

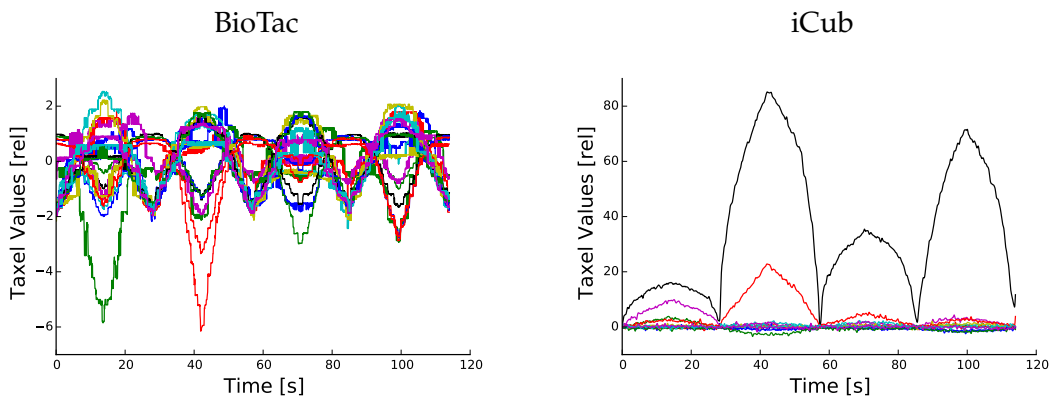


Figure 4.4: Time-series of BioTac (left) and iCub (right) recordings. For both sensors the same force profile 0 N to 5 N was applied. The linear slopes of the sawtooth force profile get transformend into sensor reading in a non-linear fashion. A lot of electrodes of the BioTac react to the same stimulus while for the iCub sensor only a few electrodes are active.

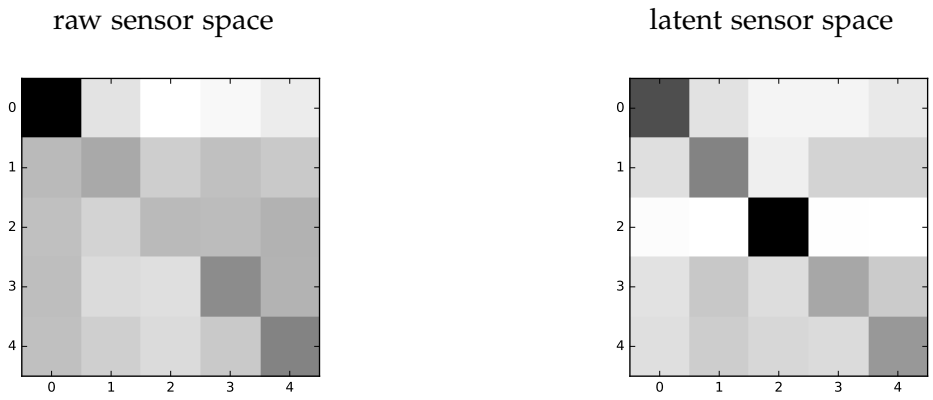


Figure 4.5: Results for linear classification applied to raw (left) and preprocessed (right) iCub sensor data. For showing the classification quality confusion matrices are used.

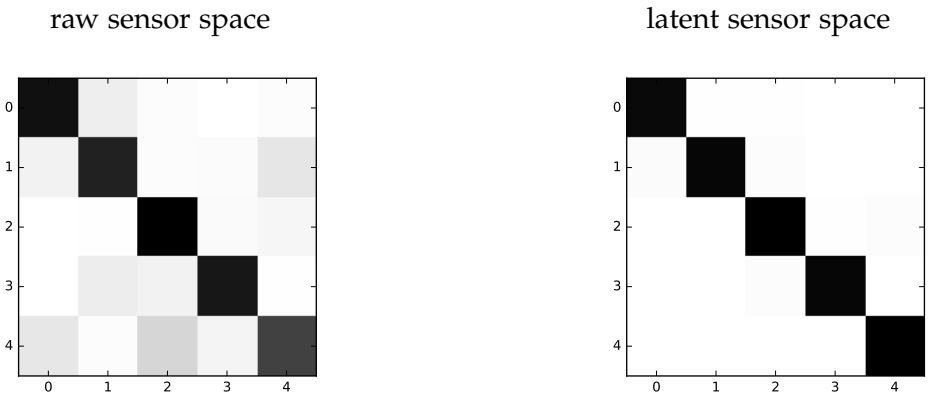


Figure 4.6: Results for linear classification applied to raw (left) and preprocessed (right) BioTac sensor data. For showing the classification quality confusion matrices are used.

the current sensor data input. Only three neurons remained after comparing the approximate posterior distribution to the standard normal distributed prior and keeping the dimensions with highest divergence. After this selection, the ground truths like force, roll angle and pitch angle were paired to each of the three latent dimensions. It turned out that each neuron encoded a specific physical quantity and they even related in an almost linear fashion to the ground truth. In fig. 4.7, the left column represents results on the BioTac sensor while the right shows the iCub sensor. Even for tactile sensors with completely different mechanical structure and measurement principle, the VAE found the same disentanglement into independent quantities similar to physical units. Only the ordering is arbitrary, but this poses no problems to subsequent learning algorithms or minimal calibration procedures in order to exploit this latent representation.

4.7 APPLICATION FOR CONTROL

The strength of the proposed methods does not solely lie in solving tactile tasks but help to create models of other complex systems, without the need for tedious labelling of data sets or handcrafting features suitable for control. Even without a learned representation of the transition through time these static representations can already be used for improving a robotic controller. For setting up a simple control environment, the gimbal platform from fig. 4.1 was equipped with a pole extending out of the rotation axis centre acting as an inverted pendulum. The motors got replaced by angular sensors such that the platform is moving without resistance. A simple neural network with 20 neurons in a single hidden layer with rectifier activations was used for predicting the next latent state. The input to the network is the current latent state together with a proposed action. During evaluation on the robot this model creates predictions for a fixed set of actions. The actions where the predicted latent state is closest to the desired position is send to the robot actuators. After each of the 30 roll-outs the model is retrained on

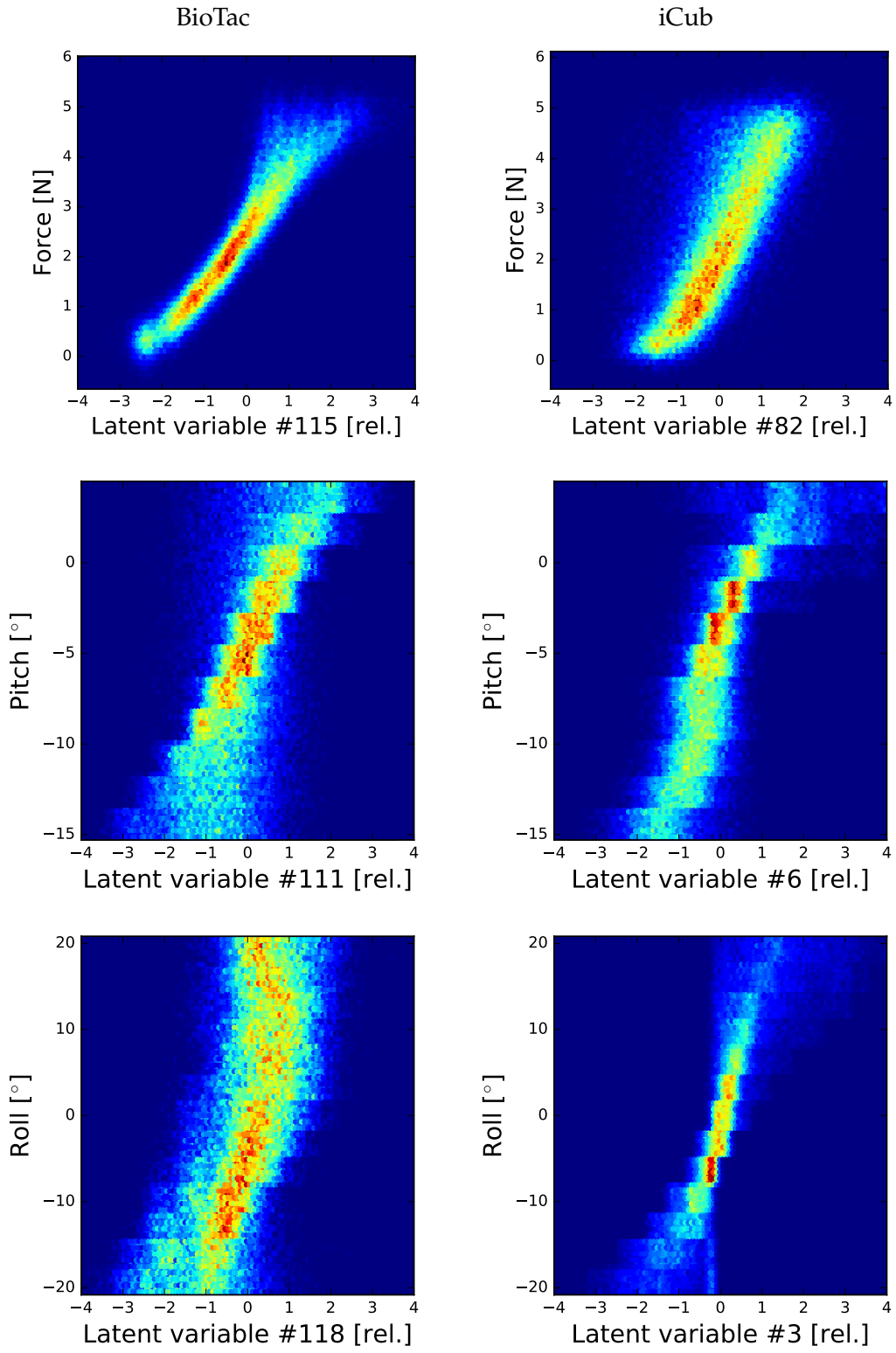


Figure 4.7: Relation between latent representation and ground truth. The VAE automatically identified a non-linear transformation into a latent space where each ground truth is stored in a separate dimension. The mapping even shows an almost linear relation between the latent value and the physical values. No ground truth was shown to the algorithm while training this mapping, here it was only used during the visualisations step.

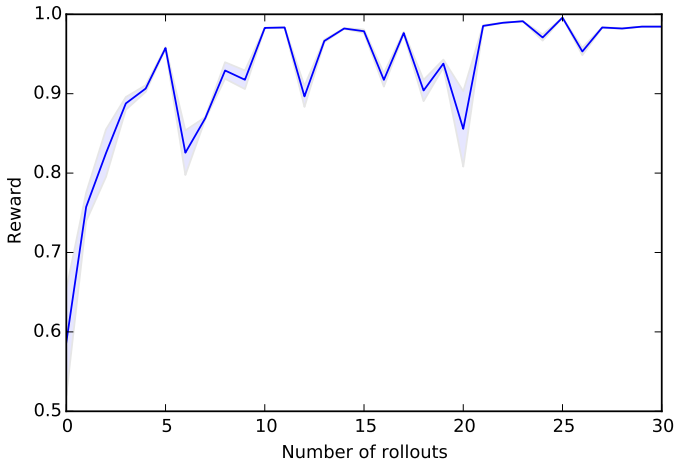


Figure 4.8: Reward during training of pole balancing task. A neural network was used to predict the next preprocessed tactile state given the current state and an action. Repeated execution with different actions of this network together with an error function on the predicted state are used to generate the next action. The rewards in the graph are averaged over 10 experiments. The last 10 time steps of each reward are used for calculating the reward. After each new roll-out the controller was retrained.

the accumulated data-point action pairs. The resulting increase in reward after 30 roll-outs is shown in fig. 4.8.

In [34] such a similar generated latent representation was used as the observation input to a policy search algorithm and provided much better performance as compared to raw sensor input. Additionally the paper showed that learning a dynamics model jointly with the latent space further improves the control learning. The resulting latent space representation used in the publication [34] suggests that the included action information shapes the latent space and makes it more regular, especially the dimensions which heavily rely on the dynamics, such as the velocity.

4.8 CONCLUSION ON PREPROCESSING

The modelling capabilities of the variational auto-encoder were demonstrated on highly non-linear data from two types of tactile sensors. Supervised methods like linear regression and decision trees were used for showing that all information about the relevant features are still encoded in the hidden representation. Not only did the experiments show that information is still present in the latent representations but also they improved the regression and classification capabilities of linear models. This solves the initial problem of accidentally limiting the number of features through supervised learning but also improves the discussed generalisation capabilities. This demonstrates a reduction of the labelling effort for supervised calibration of the sensors. Apart from that, the results showed that the compression capabilities of the latent space correctly recognised that for the tactile sensor data only three latent features are important. Those three features were collected into three latent dimensions and corresponded linearly to the true physical ground

truth: one neuron was linearly related to the normal force of the tactile sensor, while the other two neurons encoded the roll and pitch angle of the sensor relative to the touched surface. Even running the algorithm on two fundamentally different tactile sensors, the BioTac and the iCub sensor, produced the same result in terms of latent representation.

5 TIME SERIES MODELLING

Many tasks like optimising a control policy or estimating intrinsic motivation can benefit from a differential model of the environment. This reduces the need to excessively sample from the environment for estimating an error gradient. Again the environment has to be modelled probabilistically as the learned variance is required to act accordingly in uncertain state estimations. This leads us to a model $p(\mathbf{x}_{1:T})$ of the observed sequences $\mathbf{x}_{1:T}$. However a mere model of the, possibly non-Markovian, observational data is not enough. As a solution, a compressed latent space is learned from raw sensor data in which the state space transition is forced to be Markovian.

The following sections contain results, figures and parts of the publication [24].

5.1 REQUIREMENTS FOR STATE SPACE MODELS

Just as in the static data case the assumption here is, that the data set $\mathbf{x}_{1:T}$ is generated by a hidden process dependent on an unobserved variable $\mathbf{z}$. The data distribution can be represented as the marginal over all hidden variables:

$$p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) = \int p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})\, p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})\, d\mathbf{z}_{1:T}. \tag{5.1}$$

This marginal likelihood is very similar to the static case except for the additional condition on actions and the interpretation of each variable as a time sequence. However, state space models have further requirements which can be encoded into the graphical structure. There are two assumptions on the structure:

$$p(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t), \tag{5.2}$$

$$p(\mathbf{z}_{1:T} \mid \mathbf{v}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=0}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{v}_t). \tag{5.3}$$

The first assumption encoded in eq. (5.2) states that an observation $\mathbf{x}_t$ is only dependent on the local hidden state $\mathbf{z}_t$. It can be, that $\mathbf{x}_t$ only represents parts of the information of the actual state $\mathbf{z}_t$, which can make the transition from one $\mathbf{x}_t$ to another non-Markov. Equation (5.3) will force the transition to only require the last latent state and make the hidden dynamics Markovian. The variable $\mathbf{v}_t$ represents the source of uncertainty for the parameters and structure of the transitions and is part of the variable $\boldsymbol{\beta}_t = (\mathbf{w}_t, \mathbf{v}_t)$. The remaining part of $\boldsymbol{\beta}_t$ represents the process noise and describes the uncertainty of $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{v}_t)$ when a sample of $\mathbf{v}_t$ is given.

With these assumptions embedded into our marginal likelihood eq. (5.1) and factorizing the distributions over time series the marginal likelihood is defined as:

$$p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) = \iint p(\mathbf{v}_{1:T}) \prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t) \prod_{t=0}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{v}_t) \, d\mathbf{v}_{1:T} \, d\mathbf{z}_{1:T}. \tag{5.4}$$

5.2 PROBLEMS OF STATE-OF-THE-ART ALGORITHMS

Several applications of SGVB have been applied to time series data [19–23]. [19, 20] use non-linear transitions in a hidden layer of a recurrent neural network (RNN). The hidden deterministic transition receives samples from the observation model and the last latent distribution making the transition spread over several transformations. There is no clear distinction between a stochastic transition and the emission model.

Another similar algorithm is presented by [22]. Here the transition is directly defined as a neural network with a stochastic output. The only downside is, that this transition gets used only in the prior locally dependent on the last state, which only softly encodes the Markov properties. This results in a model that fails to extract state space features for time derivatives leading to an over powerful recognition model.

The authors in [23] present an alternative way of learning a time series model, but their inference model uses a very costly message passing and node potentials. The approach presented here infers the latent states directly instead of using an inference subroutine.

Closer to the work in this thesis, in [21] a VAE structure for single time-step prediction was used. They also applied the transition model to subsequent control tasks. Multiple observations need to be stacked to make the input Markovian. The number of time-steps needs to be individually tuned for every dataset. The inclusion of the second encoder time-step is not inspired by the ELBO but rather optimised by a tunable regularisation term.

In the model presented in this thesis the agglomeration of information of several observations is done automatically by the recognition model. The Markov assumptions in the transition forces the information necessary for single step prediction to accumulate in the latent state.

5.3 STOCHASTIC GRADIENT VARIATIONAL BAYES FOR TIME SERIES DISTRIBUTIONS

All tunable parameters lie within the neural networks representing the different distribution in the lower bound. Parameters of the transition can be shared between the prior distribution and the recognition model or viewed to be completely part of the generative model. All of these changes can be done without violating the lower bound and without significantly affecting the marginal likelihood. The choice of internal network structure can significantly affect the overall transition model quality resulting in long term predictions. By moving all stochasticity of the latent state

to the innovation noise variable $\boldsymbol{\beta}_{t-1}$, the marginal likelihood (eq. (5.4)) can be simplified to:

$$\begin{aligned} p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) &= \int p(\boldsymbol{\beta}_{1:T}) \prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t)\Big|_{\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}, \boldsymbol{\beta}_{t-1})} \mathrm{d}\boldsymbol{\beta}_{1:T} \\ &= \int p(\boldsymbol{\beta}_{1:T}) p_\theta(\mathbf{x}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \mathbf{u}_{1:T}) \, \mathrm{d}\boldsymbol{\beta}_{1:T}. \end{aligned} \tag{5.5}$$

The hidden state space variables $\mathbf{z}_t$ are now encoded inside the probability distributions in eq. (5.5). These internal deterministic relations are represented by the diamond nodes in fig. 5.1. This also holds for the state space assumptions which are encoded in the network structure inside $p_\theta(\mathbf{x}_{1:T} \mid \boldsymbol{\beta}_{1:T})$ by having no connections between variables which should not depend on each other.

The lower bound to the data likelihood can now be derived just as in the VAE case described in section 3.3:

$$\begin{aligned} \ln p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) &= \ln \int p(\boldsymbol{\beta}_{1:T}) p_\theta(\mathbf{x}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \mathbf{u}_{1:T}) \frac{q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})}{q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \mathrm{d}\boldsymbol{\beta}_{1:T} \\ &\geqslant \int q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \ln \left(\frac{p_\theta(\mathbf{x}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \mathbf{u}_{1:T}) p(\boldsymbol{\beta}_{1:T})}{q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \right) \mathrm{d}\boldsymbol{\beta}_{1:T} \\ &= \mathbb{E}_{q_\phi} \left[\ln p_\theta(\mathbf{x}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \mathbf{u}_{1:T}) - \ln q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) + \ln p(\boldsymbol{\beta}_{1:T}) \right] \qquad (5.6) \\ &= \mathbb{E}_{q_\phi} [\ln p_\theta(\mathbf{x}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \mathbf{u}_{1:T})] - D_{KL}\big(q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \parallel p(\boldsymbol{\beta}_{1:T})\big) \qquad (5.7) \\ &=: \mathcal{L}_{DVBF}(\mathbf{x}_{1:T}, \theta, \phi \mid \mathbf{u}_{1:T}). \end{aligned}$$

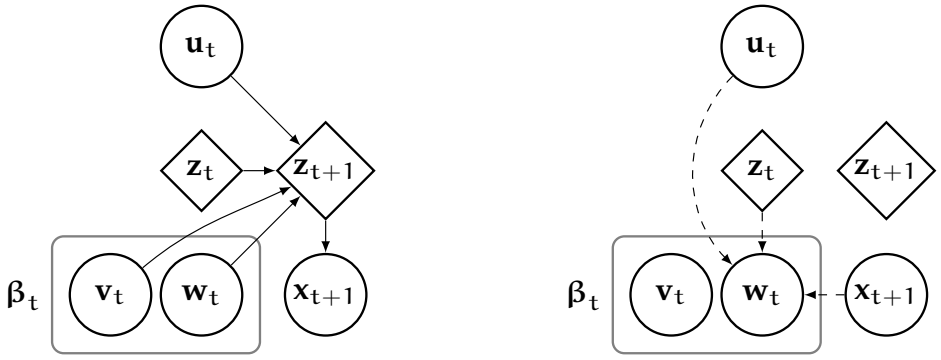
5.4 DEEP VARIATIONAL BAYES FILTERS

5.4.1 *Reparametrising the transition*

Previous approaches used the transition model as the prior for the latent space [22]. This resulted in gradients only flowing one time-step back, which lead to a recognition model directly encoding the observation into the latent state while ignoring the last latent state. It also meant that errors in the transition only led to local effects on the loss function, instead they should rather lead to changes in prediction error in future latent states.

For solving these problems, all transitions in deep variational bayes filters (DVBF) are kept in a long chain while the prior only acts locally through the innovation parameter β_t. In comparison to [22] this prevents the recognition model from directly sampling the latent states and forces it to actively correct the state predicted by the transition. This is very similar to the reparametrisation trick [49, 50] used for making a stochastic process differentiable w.r.t. its parameters. Here errors are back-propagated through a whole chain of transitions for optimising their parameters.

Two types of transition reparametrisations, that enable back-propagation through time are presented in the following. The first introduces an innovation noise variable which is a transformation of the previous observations during filtering. During generative sampling this variable is set to random noise.



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

Figure 5.1: a) Forward graphical model for predicting next latent state $\mathbf{z}_{t+1}$ and reconstruction of next observation $\mathbf{x}_{t+1}$. The dependence on only the last latent state $\mathbf{z}_t$ together with the reconstruction of the observation only from the current latent state enforce state space model assumptions. The input from $\boldsymbol{\beta}_t$ represents all stochasticity during the latent state space transition. b) Inference of the stochastic variables of the transition. All past latent states, observations and control inputs can be used for inferring the innovation noise.

The most general transition of this form

$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\beta}_t), \tag{5.8}$$

uses the last state $\mathbf{z}_t$, an action $\mathbf{u}_t$ and $\boldsymbol{\beta}_t$ as the only stochastic input to this deterministic function. It can be split into $\boldsymbol{\beta}_t = (\mathbf{w}_t, \mathbf{v}_t)$, where $\mathbf{w}_t$ is seen as process noise, while $\mathbf{v}_t$ is defined as stochasticity for the transition parameters. The posterior for representing both $\mathbf{w}_t$ and $\mathbf{v}_t$ can look like this:

$$q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}) = q_\phi(\mathbf{w}_{1:T} \mid \mathbf{x}_{1:T})\, q_\phi(\mathbf{v}_{1:T}). \tag{5.9}$$

When taking generative samples of the model, $\mathbf{w}_t$ needs to be sampled from the prior, while $\mathbf{v}_t$ should still be sampled from $q_\phi(\mathbf{v}_{1:T})$.

By introducing this innovation noise variable, the transition parameters can no longer be considered as part of the recognition model and the prior on $\boldsymbol{\beta}_t$ is now a standard normal distribution and independent on the transition parameters. The probability distribution of the transition is now implicitly defined by $\boldsymbol{\beta}_t$ and the structure of the transformation. When a tractable distribution density evaluation of the transition is required for later steps, this transformation needs to be invertible and distortions of the transformation need to be kept track of. Without this requirement the transition function can be chosen arbitrarily. The gradient computation through the transition which previously stopped at the KL-divergence cost now also receives gradient beyond that regulariser from errors later in the time series.

The graphical models in fig. 5.1 show the structure of the generative and inference structure. A functional structure of these models is shown in detail in fig. 5.2 and fig. 5.3a.

The optimisation of this model often got trapped in local minima. In these minima the network learned to encode individual observations directly into the latent space and reconstructed observations ignoring the transition output. It therefore

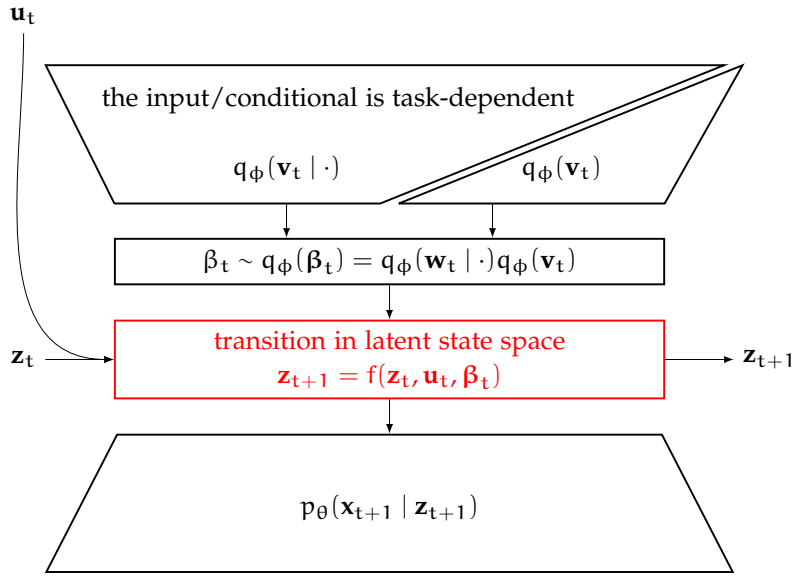
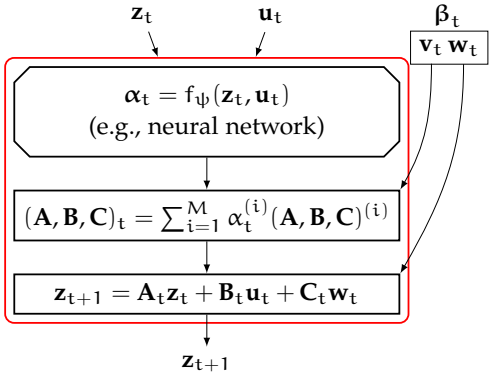
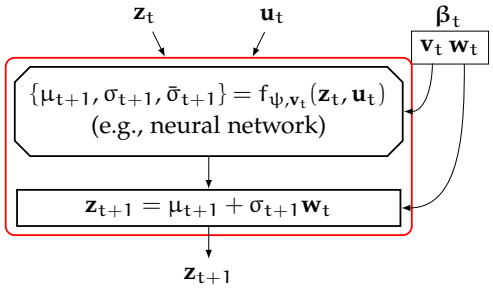


Figure 5.2: Structure of DVBF for a single time step. The box in red represents the generic type of transition and can be replaced with either a non-linear neural network based transition or with a locally linear transition. The encoder part in the top predicts β_t from the current observation. The decoder in the bottom then reconstructs the observation using the filtered latent state $\mathbf{z}_{t+1}$. The probability distribution of the transition is dependent on β_t and its processing inside the deterministic transition, see section 5.4.1 for more information.



(a) Locally linear transition.



(b) Non-linear transition with Gaussian distribution.

Figure 5.3: (a) Locally linear transition from section 5.4.3. Matrices in the transition are generated by linearly combining a set of base matrices. The weights α_t are dependent on the last latent state and action in a non-linear fashion. The local linear transition will follow a multivariate Gaussian distribution dependent on the choice of $\mathbf{C}_t$ and $\mathbf{w}_t$. (b) Non-linear transition from section 5.4.4. Mean and variance are generated by a neural network and transition uses the reparametrisation trick by transforming the sample $\mathbf{w}_t$.

learned no reasonable system dynamics inside the transition. In order to solve this instability an annealed version of (5.6) was employed:

$$\begin{aligned}\mathcal{L}_{\text{DVBF}} = \mathbb{E}_{q_\phi}\big[c_i \ln p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}) - \ln q_\phi(\boldsymbol{\beta}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \\ + c_i \ln p(\mathbf{w}_{1:T}) + \ln p(\mathbf{v}_{1:T})\big].\end{aligned} \tag{5.10}$$

The inverse temperature parameter $c_i = \max(1, 0.01 + i/T_A)$ emphasizes the maximisation of entropy of the approximate posterior. As the number of training steps i reaches T_A, c_i reaches 1 and the ELBO is back at its normal form. [17, 18, 47] show that this method smooths out the error landscape.

5.4.2 *Gated recognition model*

The second method for structuring the recognition and transition model for DVBF makes use of Gaussian multiplication similar as in the original Kalman filter formulation. In this setup both, the recognition model and the transition, predict the current latent state as a Gaussian distribution. The two probability densities then get multiplied and renormalized:

$$q(\mathbf{z}_{t+1} \mid \ldots) \propto q_{\text{meas}}(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t+1}, \mathbf{u}_{1:t}) \times p_{\text{trans}}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t). \tag{5.11}$$

Here the conditioning of the inverse emission model on the whole sequence is shown as the most general representation, but it can certainly be relaxed and conditioned on the last state and current observation alone. In the case of whole sequences an RNN can be used for processing the data into sufficient statistics for the distribution. For the multiplication of two Gaussian distributions the resulting probability density is again Gaussian. The closed form solution of this operation results in the following equations[16]:

$$\begin{aligned}\mu_q &= \frac{\mu_{\text{trans}}\sigma^2_{\text{meas}} + \mu_{\text{meas}}\bar{\sigma}^2_{\text{trans}}}{\sigma^2_{\text{meas}} + \bar{\sigma}^2_{\text{trans}}}, \\ \sigma^2_q &= \frac{\sigma^2_{\text{meas}}\bar{\sigma}^2_{\text{trans}}}{\sigma^2_{\text{meas}} + \bar{\sigma}^2_{\text{trans}}}.\end{aligned} \tag{5.12}$$

These operations give the transition and recognition model the possibility to assign a belief to their predictions. The resulting latent state is then properly weighted regarding these beliefs.

μ_{trans} and $\bar{\sigma}^2_{\text{trans}}$ come from the transition model p_{trans}, while μ_{meas} and σ^2_{meas} come from the inverse measurement model q_{meas}:

$$\begin{aligned}p_{\text{trans}}(\mathbf{z}_{t+1} \mid \ldots) &= \mathcal{N}\big(\mu_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t), \bar{\sigma}^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)\big), \\ q_{\text{meas}}(\mathbf{z}_{t+1} \mid \ldots) &= \mathcal{N}\big(\mu_{\text{meas}}(\mathbf{z}_{1:t}, \mathbf{x}_{1:t+1}, \mathbf{u}_{1:t}), \sigma^2_{\text{meas}}(\mathbf{z}_{1:t}, \mathbf{x}_{1:t+1}, \mathbf{u}_{1:t})\big).\end{aligned} \tag{5.13}$$

With the equations from eq. (5.12), the resulting probability density for the recognition model is then defined as:

$$q(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t+1}, \mathbf{u}_{1:t}) = \mathcal{N}(\mu_q, \sigma^2_q). \tag{5.14}$$

The KL-divergence can be directly used between this approximate posterior $q(\mathbf{z}_{t+1} \mid \mathbf{z}_{1:t}, \mathbf{x}_{1:t+1}, \mathbf{u}_{1:t})$ and some prior transition $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ just as used in algorithms like deep Kalman filter (DKF). This prior is the generative transition distribution, of which only the mean is shared between the fusion step and the prior transition. Would the standard deviation $\bar{\sigma}^2_{\text{trans}}$ also be shared between the fusion step and prior, then the above KL divergence minimisation would try to pull $\bar{\sigma}^2_{\text{trans}}$ and σ^2_q to the same value. This leads to the following relationship in the case of a minimal KL-divergence:

$$\begin{aligned}
\sigma^2_q = \frac{\sigma^2_{\text{meas}} \bar{\sigma}^2_{\text{trans}}}{\sigma^2_{\text{meas}} + \bar{\sigma}^2_{\text{trans}}} &\stackrel{!}{=} \bar{\sigma}^2_{\text{trans}} \\
0 &= \bar{\sigma}^2_{\text{trans}} - \bar{\sigma}^2_{\text{trans}} \frac{\sigma^2_{\text{meas}}}{\sigma^2_{\text{meas}} + \bar{\sigma}^2_{\text{trans}}} \\
0 &= \bar{\sigma}^2_{\text{trans}} \frac{\bar{\sigma}^2_{\text{trans}}}{\sigma^2_{\text{meas}} + \bar{\sigma}^2_{\text{trans}}} \\
0 &= \frac{(\bar{\sigma}^2_{\text{trans}})^2}{\sigma^2_{\text{meas}} + \bar{\sigma}^2_{\text{trans}}}
\end{aligned}$$

This equation shows that sharing the variance and optimising the KL-divergence cost will force σ^2_{meas} towards infinity and $\bar{\sigma}^2_{\text{trans}}$ towards zero. Even though this attractor is counteracted by the reconstruction cost, it showed during experiments that this produced local minima which were hard to escape by the optimiser. As a solution an extra variance σ^2_{trans} is introduced for the prior distribution while the mean of the transition is still shared between the posterior and the prior model:

$$p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) = \mathcal{N}\big(\mu_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t), \sigma^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)\big). \tag{5.15}$$

The parameters μ_{trans}, $\bar{\sigma}^2_{\text{trans}}$ and σ^2_{trans} are predicted by one neural network with last state $\mathbf{z}_t$ and action $\mathbf{u}_t$ as inputs and μ_{meas} and σ^2_{meas} are predicted by another neural network conditioned on the observations and actions. The updated graphical structure with the Gaussian multiplication step is shown in fig. 5.4.

For fitting this divergence into the previous framework from section 5.4.1 with innovation noise variables and a standard normal Gaussian prior, the transformations can be moved from the prior to the posterior side of the KL-divergence. This is done by subtracting the transition mean and dividing by the transition standard deviation such, that the prior is transformed into a standard normal distribution. The value for the KL-divergence does not change when these transformations are applied, as long as they are homeomorphic. The parameters of $\mathbf{w}_t \sim \mathcal{N}(\mu_{\mathbf{w}_t}; \sigma^2_{\mathbf{w}_t})$:

$$\mu_{\mathbf{w}_t} = \frac{\mu_q - \mu_{\text{trans}}}{\sigma_{\text{trans}}}, \qquad \sigma^2_{\mathbf{w}_t} = \frac{\sigma^2_q}{\sigma^2_{\text{trans}}},$$

can then be used in the same way as the previous distribution of $\mathbf{w}$ inside the KL computation with a standard normal distribution as the prior just as shown in eq. (5.7) where $\mathbf{w}$ is part of $\boldsymbol{\beta}$.

With this change it was possible to omit the annealing step and still get good generative models. However, it should be noted that this multiplication does not come from the same mathematical derivation as in the Kalman filter. Here it is only

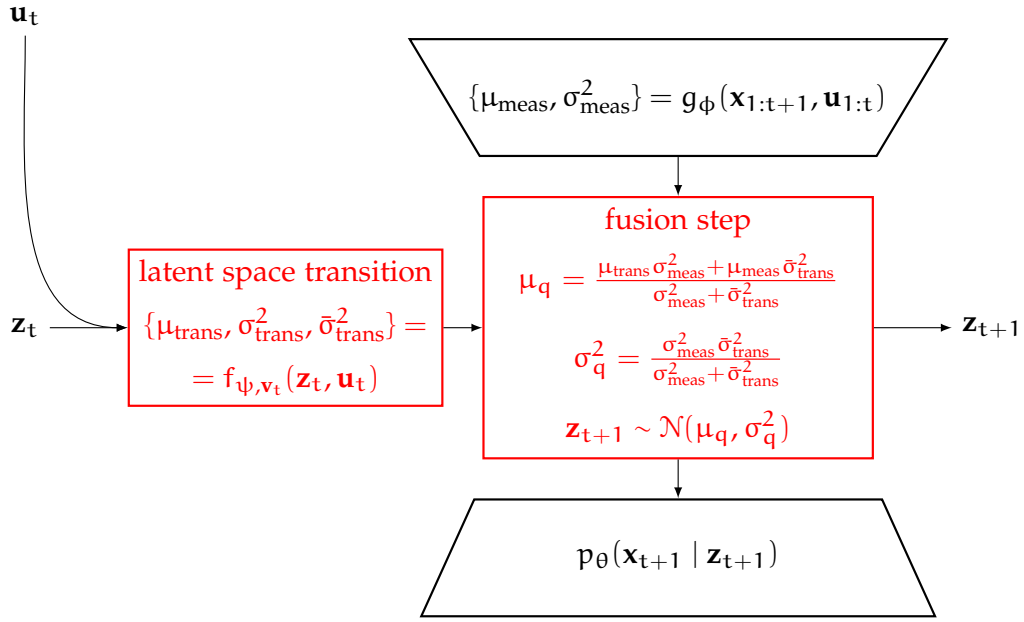


Figure 5.4: Architecture of the gated recognition model with Gauss multiplication fusion step. Graphical model for one transition under state space model assumptions for the fusion step. The additional variance output σ^2_{trans} of the transition remains unused. The fusion step is only used for computing the ELBO or during generative sampling the model.

used as a method to weight the two outputs of the transition and the recognition model according to their confidence. In fact it is misleading to denote the parameter $\bar{\sigma}^2_{trans}$ in a similar fashion as the variance of a Gaussian. The fractions of variances in eq. (5.12) can instead be combined into a gate value $\alpha = \frac{\sigma^2_{meas}}{\sigma^2_{meas} + \bar{\sigma}^2_{trans}}$ and its annihilating $1 - \alpha$:

$$\mu_q = \frac{\mu_{trans}\sigma^2_{meas} + \mu_{meas}\bar{\sigma}^2_{trans}}{\sigma^2_{meas} + \bar{\sigma}^2_{trans}} = \alpha\mu_{trans} + (1 - \alpha)\mu_{meas}.$$

The variance fusion changes to

$$\sigma^2_q = \alpha\bar{\sigma}^2_{trans} + (1 - \alpha)\sigma^2_{meas}.$$

This leads to another interpretation of this prediction and measurement fusion: When the two variances are combined to a single parameter, a similarity to a long short term memory (LSTM) or gated recurrent unit (GRU) layer can be made. Such gating mechanisms give neural networks the capability to control a multiplicative element in the network and enable better gradient flow [15].

[43] applied a similar technique for combining the information from bottom-up recognition and top-down prior connections. However they ignore, that optimising the KL divergence with a shared variance between approximate posterior and prior results in instable training.

5.4.3 *Locally linear transitions*

The transition can be implemented in different ways depending on the desired application. This can either be a non-linear function like a neural network or something with more restrictions on the structure. A popular choice would be a linear transition where the current state gets multiplied with a matrix to generate to the next state. By setting the matrix as a function of $\mathbf{z}_t$ the model will describe a locally linear system. The authors from [21] chose such a system which makes the learned dynamics usable with iterative linear quadratic regulator (iLQR) for locally optimal control. The transition is then defined like this:

$$\mathbf{z}_{t+1} = \mathbf{A}_t \mathbf{z}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{C}_t \mathbf{w}_t, \qquad t = 1, \dots, T, \tag{5.16}$$

where $\mathbf{A}_t, \mathbf{B}_t$, and $\mathbf{C}_t$ are the transition matrices defined as functions of $\mathbf{z}_t$ and $\mathbf{u}_t$. The variable $\mathbf{w}_t$ can be seen as process noise and is also used in this sense when performing generative sampling. However, during filtering it should be seen as a form of innovation noise coming from the recognition model which corrects the current trajectory with input from observations.

The matrix generating functions are parametrised as a linear combination of base matrices. Only the weights $\boldsymbol{\alpha}_t$ are now state and action dependent, and generated by a non-linear function:

$$\boldsymbol{\alpha}_t = f_\psi(\mathbf{z}_t, \mathbf{u}_t) \in \mathbb{R}^M, \tag{5.17}$$

$$\mathbf{A}_t = \sum_{i=1}^{M} \alpha_t^{(i)} \mathbf{A}^{(i)}, \qquad \mathbf{B}_t = \sum_{i=1}^{M} \alpha_t^{(i)} \mathbf{B}^{(i)}, \qquad \mathbf{C}_t = \sum_{i=1}^{M} \alpha_t^{(i)} \mathbf{C}^{(i)}.$$

When a Bayesian interpretation of the transition parameters [14] is used, then these base matrices are drawn from the posterior $q_\phi(v_t)$:

$$\mathbf{v}_t \sim q_\phi(\mathbf{v}_t)$$
$$\mathbf{v}_t = \left\{ \mathbf{A}^{(i)}, \mathbf{B}^{(i)}, \mathbf{C}^{(i)} \mid i = 1, \dots, M \right\}.$$

A softmax transfer function is used in the last layer of f_ψ from eq. (5.17) forcing a convex combination of base matrices. Together with the Bayesian treatment this forms a powerful regularization. The inference of the innovation noise $\mathbf{w}_t$ can either be done by direct modelling through the posterior distribution as well as through the gating mechanism described in the last section. When the gating mechanism is used, a second $\mathbf{C}$ matrix has to be added for the additional variance in the fusion step.

5.4.4 *Non-linear transitions*

The deterministic transition function can also directly be described through a non-linear mapping. There are several choices for including the noise variables de-

pending on the application. The simplest one is achieved by directly using all parameters as input to a non-linear function:

$$\mathbf{z}_{t+1} = f_\psi(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\beta}_t). \tag{5.18}$$

With this parametrisation only the first version of the fusion step can be used where β_t variables are directly predicted by the recognition model. To benefit from the gating mechanism (section 5.4.2) a Gaussian parametrisation of the transition is needed. The transition is then composed of two steps, first a generation of mean and variance and then a sampling process where the innovation variable is included:

$$\begin{aligned} \{\mu_{t+1}, \sigma_{t+1}, \bar{\sigma}_{t+1}\} &= f_\psi(\mathbf{z}_t, \mathbf{u}_t), \\ \mathbf{z}_{t+1} &= \mu_{t+1} + \sigma_{t+1}\mathbf{w}_t. \end{aligned} \tag{5.19}$$

The second variance $\bar{\sigma}_{t+1}$ is used for the Gaussian fusion in the recognition model gating mechanism. The graphical structure of this transition can be seen in fig. 5.3b. This type of transition can be used, when no specific transition structure is required.

5.5 EVALUATION

The proposed method got evaluated on several toy datasets. They were chosen such that their state space is easy to plot but still depicting complex dynamics. It was tested on a pendulum system and a ball trapped in a box where the state is encoded as a high dimensional image. The latent representations are then evaluated on their ability to generate accurate long term predictions, but also on their ability to represent hidden states for transforming the non-Markovian observations into a Markovian state.

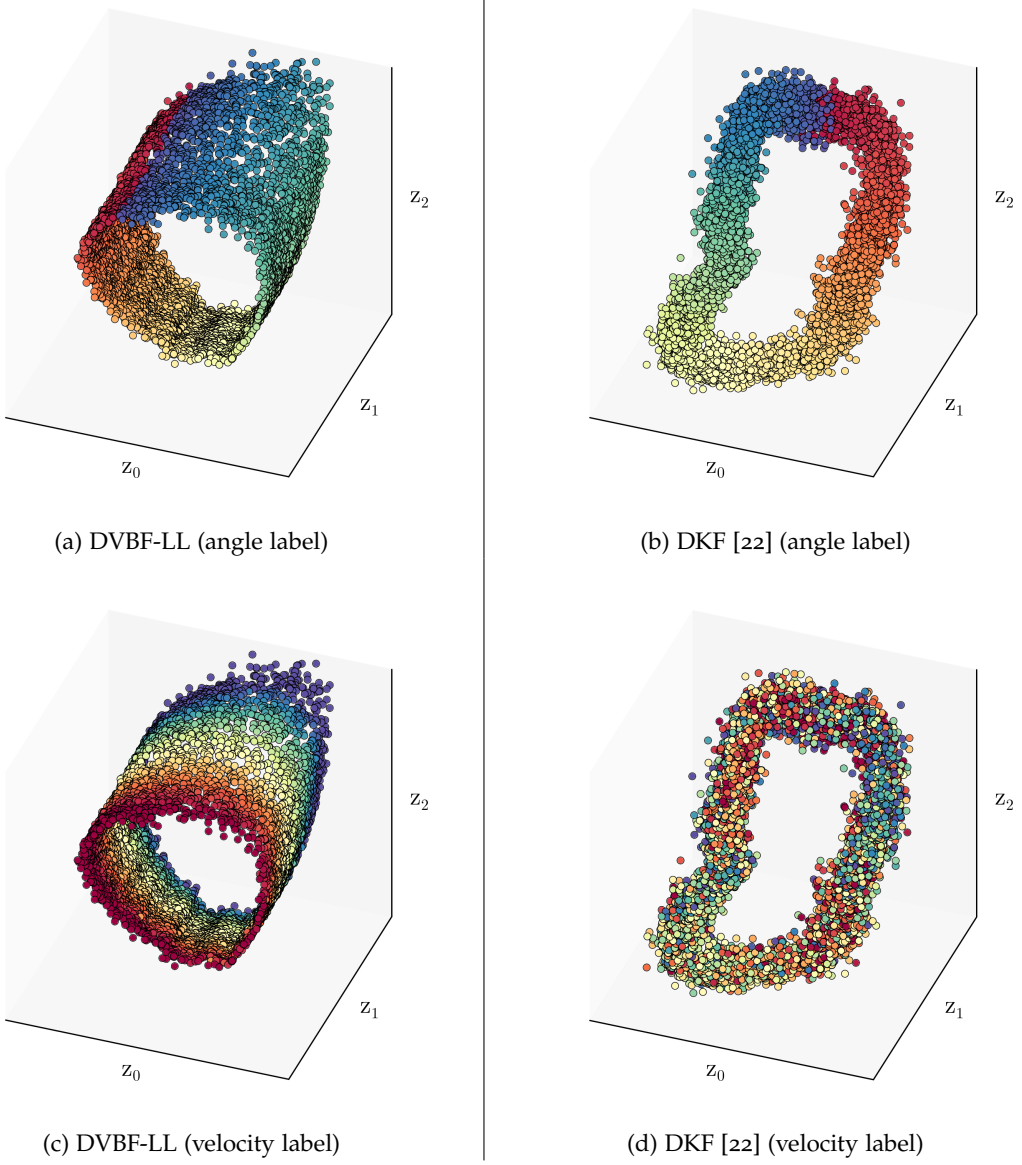
5.5.1 *Dynamic pendulum*

For the pendulum environment data set was generated by simulating the differential equation

$$ml^2\ddot{\varphi}(t) = -\mu\dot{\varphi}(t) + mgl\sin\varphi(t) + u(t),$$

with a force control $u(t)$ set to random noise and the other parameters set to $m = l = 1, \mu = 0.5, g = 9.81$. The angle $\varphi(t)$ of the pendulum is then rendered into a 16x16 image. This transformation into images deletes information about the angular velocity $\dot{\varphi}(t)$ and makes the environment partially observable. The velocity information can however be recovered when combining several observations. Goal of this experiment is to verify that the DVBF model is capable of transforming these non-Markovian image sequences into a latent Markovian state space.

The model used for learning this environment uses the locally linear transition from section 5.4.3, together with the innovation noise update and the annealed ELBO. Detailed information about model parameters for DVBF can be found in appendix A.3. Parameters of the pendulum experiment for the DKF model are shown in appendix A.5.



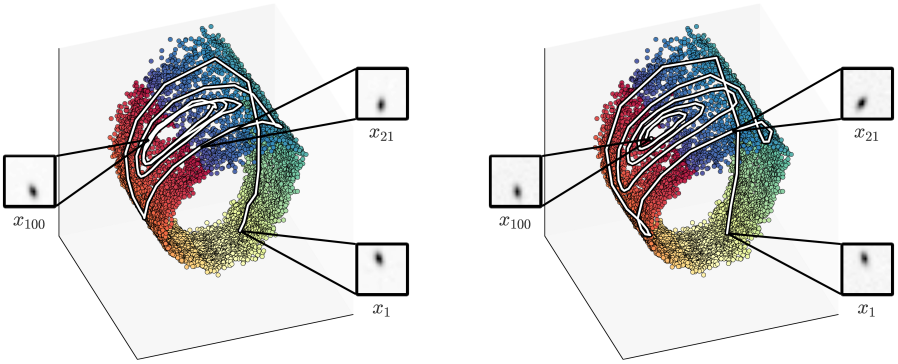
(a) DVBF-LL (angle label)

(b) DKF [22] (angle label)

(c) DVBF-LL (velocity label)

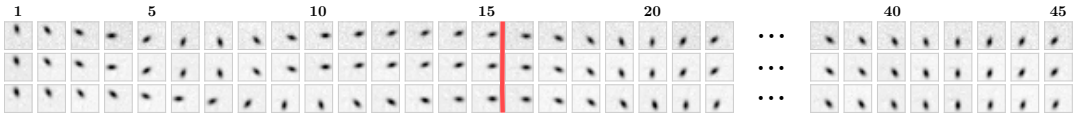
(d) DKF [22] (velocity label)

Figure 5.5: Comparison of latent representations of DVBF and DKF (from [22]). Both models were trained on the same pendulum dataset. (a, c) The latent space of DVBF coloured with the ground truth angle (a) and velocity (c). (b, d) The latent space of DKF coloured with the ground truth angle (b) and velocity (d). In comparison to the DVBF results DKF is able to encode angular information of the pendulum in the latent space, but fails to encode several observations into velocity information.



(a) Latent walk sampled from generative model without observation intput.

(b) Latent walk with filtered states.



(c) Sequence of observations from pendulum environment (top), reconstructions (middle), generative samples after starting from the same initial latent state (bottom).

Figure 5.6: (a) Generative latent walk through latent state space of pendulum environment. Walk starts from a single filtered latent state. Three observation reconstructions from selected points of the walk are shown and connected to the corresponding state. (b) Reconstructed latent walk through latent state space of pendulum environment. Each point of the trajectory corresponds to an actual observation. Three of these observations are shown connected to the corresponding latent state. (c) Exemplary observation trajectory with samples beyond training data length (red line) and its reconstructions and generative samples. Only the reconstruction model has access to the full observation trajectory, while the generative model only received an initial latent state. The generative model shows valid reconstructions even far beyond the training sequence length. The skipped part of the sequence can be found in fig. A.1.

In fig. 5.5 the results of this experiment are shown. In fig. 5.5a and fig. 5.5c the latent variables are shown. Both plots embed the same points but with different ground truths used as colouring. The angle got encoded into two dimensions, onto a circular shape, similar to an encoding of the form $(\sin(\alpha), \cos(\alpha))$. The circle also got axis aligned without explicitly stating this property to the algorithm. The velocity is encoded perpendicular to this angle-circle effectively creating a barrel or cylindrical shape. However the angular velocity was never explicitly present in the dataset, but is necessary to predict the next latent state which in turn minimises the KL divergence. With the reconstruction approximately the same, this creates a bigger ELBO than just encoding the states necessary for observation reconstruction.

The DKF algorithm also encoded the angle in its latent state, but fails to compress it into a small representation. Instead the path of a full rotation is spread across all three dimensions as seen in fig. 5.5a. The latent space in fig. 5.5d additionally shows no correlation with the velocity ground-truth, indicating that the recognition model failed to compress several images into a Markovian state space.

Table 5.1: ELBO values for DVBF and DKF model on pendulum test set.

	Lower Bound	=	Negative Reconstruction Error	−	KL divergence
DVBF-LL	798.56		802.06		3.50
DKF	784.70		788.58		3.88

An important feature necessary for subsequent tasks is a good generative model which runs for several time steps without corrections from the observation model. Essentially this generative model can be regarded as a learned simulator and will also be used as such in the subsequent chapters.

This capability was tested on the pendulum model and sampled 100 time step trajectories. These trajectories in latent space are shown in fig. 5.6a, in observation space in fig. 5.6c and with the recognition model enabled in fig. 5.6b and the middle row of fig. 5.6c. The red line in fig. 5.6c represents the lengths of sequences during training, showing that the model generalises well beyond that limit. The latent walk in fig. 5.6a if unrolled along the cylinder length closely resembles the known phase portraits of the pendulum with the attractor at its centre. Figure 5.6c compares the dataset with the observation walk in the dataset (top) to the filtered walk (middle) and the fully generative walk (bottom) where only the starting condition is dependent on the data. The generative samples follow the true dataset closely with only small but physically plausible deviations.

The values of the ELBO in table 5.1 show a slightly better bound for the method presented here compared to DKF. This is an indication that the ELBO does only slightly reflect the actual improvement on generative quality.

5.5.2 *Bouncing ball*

In the second experiment a ball is trapped in a two dimensional box. The environment is set to elastic collisions such that the ball bounces of the walls in case of collisions. This creates highly non-linear dynamics, as the point of collision involves a sharp direction and velocity change. The latent state dimensions were set to the true system state space size of four dimensions: two for the position of the ball and two for the velocities. Additional model parameters are described in appendix A.4.

The results for this experiment are shown in fig. 5.7. Even in this vastly different environment from the pendulum environment similar results were achieved in terms of latent space shape. Again physical values are aligned with the axes, even latents encoding positions and velocities are separated into individual dimensions. Especially the Cartesian coordinates of the ball are mapped to the latent space without almost any distortions. This is an interesting feature as there are no explicit Cartesian coordinates present in the dataset, since the position is only encoded as an image. The distortion can be analysed by the checker-board pattern shown in fig. 5.7a. The pattern is based on the true positions of the ball and then mapped onto the corresponding latent points. If there is no distortion visible and all checker-board pads are square and of equal size, then the model even learned

a distance preserving Cartesian coordinate system from a raw stream of image observations.

5.6 MODEL-BASED CONTROL

Before applying this model to maximise an intrinsic motivation cost, one can already demonstrate the proposed learning method in combination with supervised control costs. The goal is to find a policy $\pi(\mathbf{u}_t \mid \mathbf{z}_t)$ that maps from the learned latent representation $\mathbf{z}_t$ to control actions $\mathbf{u}_t$ such that a cost function $\mathcal{C}(\mathbf{z}_t)$ is minimal at all points in time. The expected future cost is defined as:

$$\mathcal{C}_\pi := \mathbb{E}_\pi\left[\sum_{t=1}^{T} \mathcal{C}(\mathbf{z}_t)\right] \approx \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T} \mathcal{C}\left(\mathbf{z}_t^{(m)}\right). \tag{5.20}$$

The expectation is computed over all trajectories $\mathbf{z}_{1:T}$, which are dependent on the samples from a policy π and generated by a recursive call of the learned transition function. At each call the transition receives the latest action from the policy and the last latent state. The following factorised distribution illustrates the sampling process:

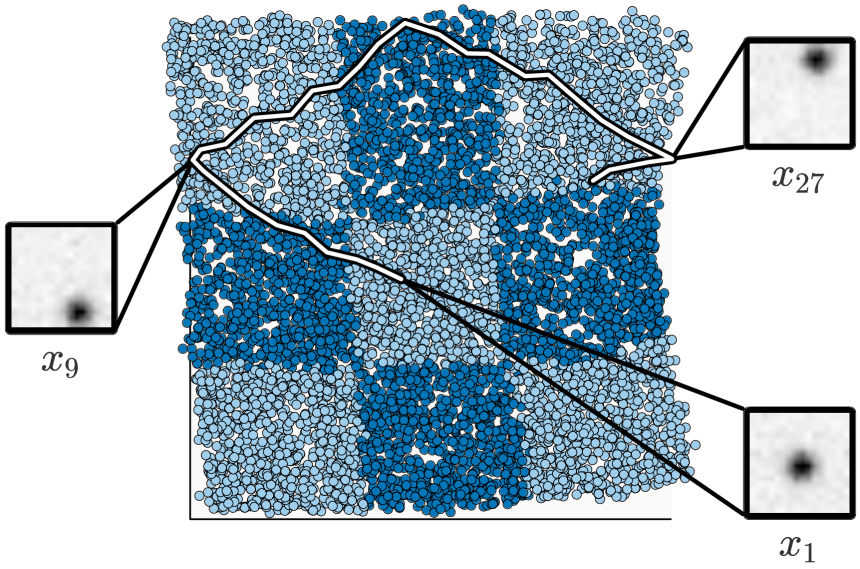
$$\mathbf{z}_{1:T},\, \mathbf{u}_{1:T-1} \sim p(\mathbf{z}_1) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{u}_t,\, \mathbf{z}_t)\, \pi(\mathbf{u}_t \mid \mathbf{z}_t). \tag{5.21}$$

Similar as in [13], the expected cost is estimated by Monte-Carlo sampling indicated in eq. (5.20) by summing the costs of M sampled trajectories. The reparametrisation trick for propagating through the sampling process from section 3.3 can also be applied here for propagating the control cost gradient through the chain of sampled transitions back into the policy parameters. Not only can this policy search be formulated as an inference problem (section 3.7), even efficient inference methods can be reused here. The loss function used is then the Monte-Carlo sampled cost eq. (5.20) with an additive regulariser in the form of a KL-divergence between the policy and a baseline policy. Such a modified value function was also used in [12], and a brief derivation can be found in section 3.7. The cost function is then changed to:

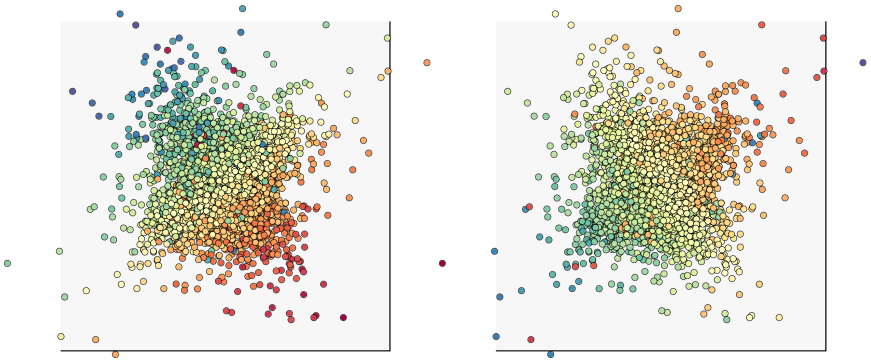
$$\begin{aligned}\mathcal{C}_\pi &:= \mathbb{E}_\pi\left[\sum_{t=1}^{T} \beta\mathcal{C}(\mathbf{z}_t) + D_{KL}(\pi(\mathbf{u}_t \mid \mathbf{z}_t) \parallel \pi_0)\right] \\ &\approx \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T}\left[\beta\mathcal{C}\left(\mathbf{z}_t^{(m)}\right) + D_{KL}\left(\pi\left(\mathbf{u}_t^{(m)} \mid \mathbf{z}_t^{(m)}\right) \parallel \pi_0\right)\right].\end{aligned} \tag{5.22}$$

The parameter β is used for tuning the influence of the policy regulariser. The value function, measuring the future expected cost conditioned on a starting state $\mathbf{z}_1$, is defined as:

$$\mathcal{V}_\pi(\mathbf{z}_1) := \min_\pi \mathbb{E}_\pi\left[\sum_{t=2}^{T} \mathcal{C}(\mathbf{z}_t)\middle|\mathbf{z}_1\right]. \tag{5.23}$$



(a) Latent walk of bouncing ball colliding with box boundaries.



(b) Latent space coloured with velocity data from ground truth.

Figure 5.7: Visualisations of the latent space of the bouncing ball environment. DVBF automatically identified latent states that correspond almost linearly to the physical ground truth state. (a) The two latent dimensions that correspond to position information. The checker board pattern is generated using ground truth position information. The regular shape and squareness of the checker board shows that the latent space preserves the distance. (b) Velocities are automatically stored in the remaining latent dimensions and orthogonality between the two velocity components is preserved in latent space.

This process is comparable to model-predictive control with the difference, that predictions are only generated while training the policy and not during online execution. After policy optimisation the prediction and action selection process is amortized into the policy.

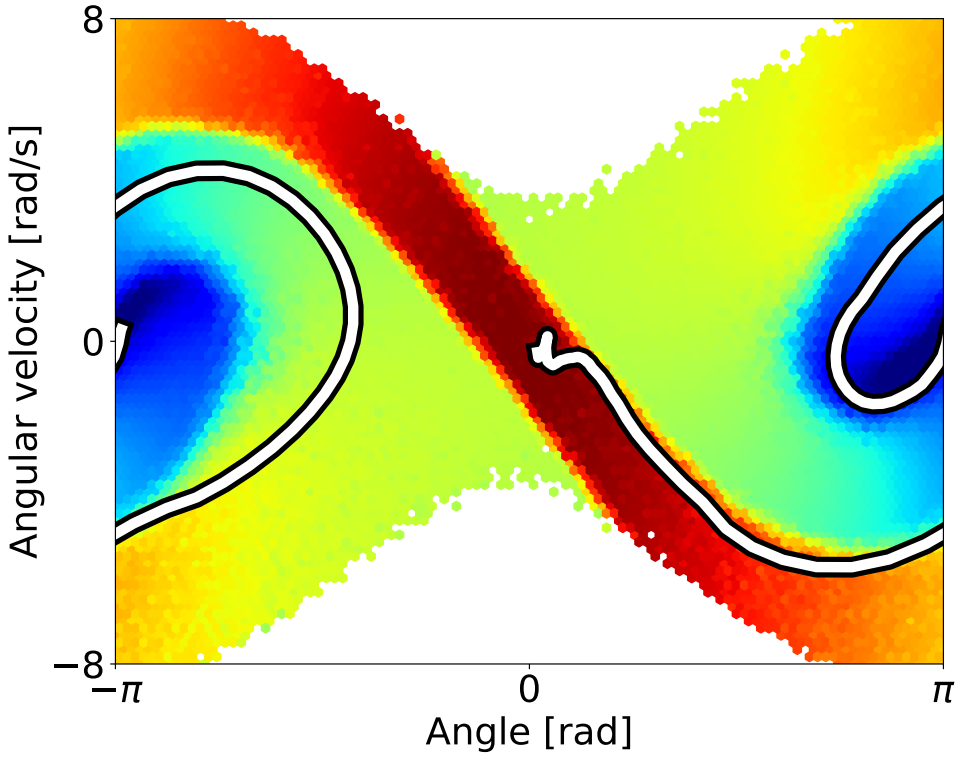
The cost function can be defined in several ways depending on the available information about the system. When the desired target position in sensor space is known, an option is to define costs as differentiable functions dependent on the observation and the last action. During model rollouts this cost function is then evaluated on emission model samples. In most reinforcement learning environments the cost function is not accessible and only samples of rewards are received during rollouts of the environment. In this case rewards can be added to the dataset during model training and interpreted as another observation, which forces the emission model to learn a probabilistic model of the reward. The input to the model gets simply redefined as the concatenation of observation and reward samples:

$$\mathbf{x} = \{\mathbf{x}_{obs}, x_{reward}\}. \tag{5.24}$$

The cost is then defined through a sampling process from the emission model of DVBF:

$$\begin{aligned} x_{reward} &\sim p(x_{reward} \mid \mathbf{z}_t) \\ c(\mathbf{z}_t) &= -\mathbf{x}_{reward}. \end{aligned}$$

A short experiment demonstrates this way of learning the reward function. For generating a dataset, the pendulum environment `Pendulum-v0` from the OpenAI Gym library[11] was used. This environment features samples of the reward paired with each observation as output. With these observation reward pairs and action inputs a DVBF model was trained, which was then used to generatively roll out predictions for optimising the control cost from eq. (5.22). The resulting value landscape when using the policy search method from above can be seen in fig. 5.8. Monte-Carlo sampling was used to evaluate the value function (eq. (5.23)) at each point in state space.



(a) Value function for inverted pendulum swingup, Monte-Carlo sampled using dynamic model learned with DVBF. Cost function learned with same model by concatenating reward to observation as shown in eq. (5.24).

(b) Exemplary supervised control cost based swingup sequence.

Figure 5.8: Results for the pendulum experiment with supervised control cost.

6 UNSUPERVISED CONTROL

This chapter will provide methods for efficiently computing empowerment, followed by evaluations on simulated environments with both given dynamics and dynamics trained with the unsupervised learning algorithms from previous chapters.

In this part empowerment will be used as the intrinsic motivation. In order to analyse the performance of the algorithm developed in this work, the results are compared with literature about theoretical and toy experiment results.

The following sections contain results, figures and parts of the publication [10].

6.1 LIMITATIONS OF PREVIOUS METHODS

Let us revisit the structure of the empowerment formulation from section 2.1.2:

$$\mathcal{E}(\mathbf{z}) = \max_{\omega} \int \omega(\mathbf{u} \mid \mathbf{z}) \int p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}) \ln \frac{p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})}{\int \omega(\mathbf{u} \mid \mathbf{z})\, p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})\, d\mathbf{u}} d\mathbf{z}' d\mathbf{u}. \tag{6.1}$$

The main obstacle of using empowerment is its computational intractability. In the continuous case in eq. (6.1) three integrals have to be computed and maximised with respect to the source distribution $\omega(\mathbf{u} \mid \mathbf{z})$. This problem has been circumvented by restricting the environments to small discrete state and action spaces [9, 75] where the influence on the environment can be evaluated by enumerating all possible states using the Blahut-Arimoto algorithm [8].

A few also tried to estimate empowerment in continuous space. [76] developed a method for learning a model of the system transition with Gaussian processes and a Monte-Carlo integration of the empowerment value. The action space is still discrete which limits the applicability to real world robotics tasks.

Another limitation can come from the way empowerment is exploited. The easiest way is to evaluate the empowerment value of the next states for every action and greedily choosing the one with highest value. Valleys in the empowerment landscape whose transition might lead to higher rewards in a later state are therefore ignored. Usually this is circumvented by using n-step empowerment, looking farther into the future for evaluating the state.

One solution for enabling both continuous state and action spaces is to linearise a given system and then use the analytical solution for the channel capacity of linear systems. This strategy was evaluated on the pendulum environment in [77]. In the next section the idea of using a linearised transition model will be reiterated, but instead of linearising a possibly highly non-linear model observations will be transformed into a latent space where the transitions are locally linear. In later sections this chapter will also apply efficient empowerment estimation to non-linear transitions.

6.2 EMPOWERMENT FOR LINEAR TRANSITIONS

For a channel where the action influence is a linear relationship, empowerment can be solved analytically. A typical approach for computing channel capacity is to linearise the system and use formulations for linear channels as in [7]. Then the solution of computing the channel capacity from [7] can be used. After linearisation the transitions is described as linear transitions like in [77]:

$$\mathbf{S} = \mathbf{TA} + \mathbf{Z}$$

where $\mathbf{A}$ is the applied action, $\mathbf{T}$ is a matrix describing its influence on the next state $\mathbf{S}$ and $\mathbf{Z}$ has a Normal distribution $\mathbf{Z} \sim \mathcal{N}(0, K_s)$. The matrices $\mathbf{T}$ and $\mathbf{Z}$ can be extracted by linearisation of a given non-linear transition function or by fitting them to samples of actions and resulting next states. This fitting procedure can be achieved by a pseudo-inverse on a tensor with action state pairs, as described in [77]. The actions $\mathbf{A}$ are designed to be Normal distributed with a specific limit on the variance, which is comparable to a power constraint. This power constraint is needed as the linear nature would allow changes by the action of any magnitude and would therefore leave the channel capacity of a linear channel unlimited. In the physical reality every control input and its influence is limited which either allows no linear transitions or bounds the range of action inputs.

Computing empowerment for these locally linear channels can be done very efficiently in the order of a singular value decomposition. Most of the computational complexity comes from transforming the linear system transition into independent channels. The channel capacity C can then be computed independently for each dimension by:

$$C = \max_{P_i} \sum_i \frac{1}{2} \log(1 + \sigma_i P_i), \tag{6.2}$$

where σ_i represent the singular values of the linear transformation matrix of the whitened dynamics system. The power constraints P_i have to sum up to a constant maximal power P and their distribution is optimized to maximise the channel capacity by the water-filling algorithm [6]. Water-filling is done by increasing the power level P_i of the channel with highest σ_i until an increase of the next channel power would raise the total channel capacity more. The overall procedure is described in algorithm 1 and can be simplified dependent on the structure of the noise covariance. Detailed derivation of the procedure can be found in [73] and in [77].

6.2.1 *Empowerment for locally linear transitions*

The channel capacity computation of linear channels gives us a chance of estimating empowerment efficiently under the assumption that a local linear approximation is accurate enough. For highly non-linear systems, linearisation can discard important dynamics, leading to wrong empowerment values. When the transition matrices directly learned from data, noise also poses a problem. This problem is discussed for a pendulum environment in [77] where randomly drawing from the

Algorithm 1 Empowerment calculation for linear channels

Require: Linearised dynamics $\mathbf{S} = \mathbf{T}'\mathbf{A} + \mathbf{Z}$ with $Z \sim \mathcal{N}(0, K_s)$

$K_s = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$	▷ Split additive noise covariance matrix using SVD.
$\mathbf{T} = \sqrt{\mathbf{\Sigma}}^{-1}\mathbf{U}^T\mathbf{T}'$	▷ Transform transition matrix according to state dependent noise.
$\mathbf{T} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$	▷ Split transition matrix using SVD.
$C = \max_{P_i} \sum_i^N \frac{1}{2} \log(1 + \sigma_i P_i)$	▷ Compute empowerment individually for each channel with singular values $\sigma_i = \Sigma_{ii}$ and power constraints.

action space instead of taking values from a regular grid renders the empowerment landscape unusable.

This problem can be solved by transforming the non-linear observations into a hidden transformation and train a locally linear model in this latent state space. For this purpose DVBF is used for learning locally linear state space models, as presented in the previous chapter. The transition needs to have a specific structure as in eq. (5.16) with linear transition matrices conditioned on the last latent state. This forces the latent space during training to warp the observation space such that the linearisation does not limit the quality of observation reconstruction. One of these resulting smooth latent spaces can be observed in the pendulum experiment in section 5.5.1. The resulting latent space in fig. 5.5 already shows a structure beneficial for learning an accurate locally linear transition. The jump between the pendulum angle 0 and 2π would pose a problem for linearisation of the state space, not so in the learned latent space where the angle is automatically mapped into a two-dimensional ring with a periodic and continuous angle space.

With this method the pseudo-inverse method for finding the linear transition matrix can be replaced by extracting the transition matrix from the learned locally linear transition. While the locally linear transition matrix is now defined as a neural network conditioned on the last latent state, also the water-filling algorithm described in section 6.2 can be replaced with a neural network. This parametric function would take the current state as input and predict the resulting power distribution along the channels. The output is constraint to sum up to one by using a softmax transfer function for the output. Its parameters can be learned jointly with the policy parameters by stochastic gradient ascent on the empowerment value.

6.2.2 *Evaluation*

First experiments on robotic environments using the linear channel approach from section 6.2 already show some of the envisioned results. In one of them the dataset recorded in section 4.3 was used for training the individual linear channels by using the pseudo-inverse. The actions were defined as the derivative of the force applied and the observation was the raw tactile data of the BioTac sensor.

The background of this experiment was to use the resulting empowerment value as a cost function to estimate grip stability in robotic hands. The resulting empowerment values for each force can be seen in fig. 6.1. An increase in force increases the

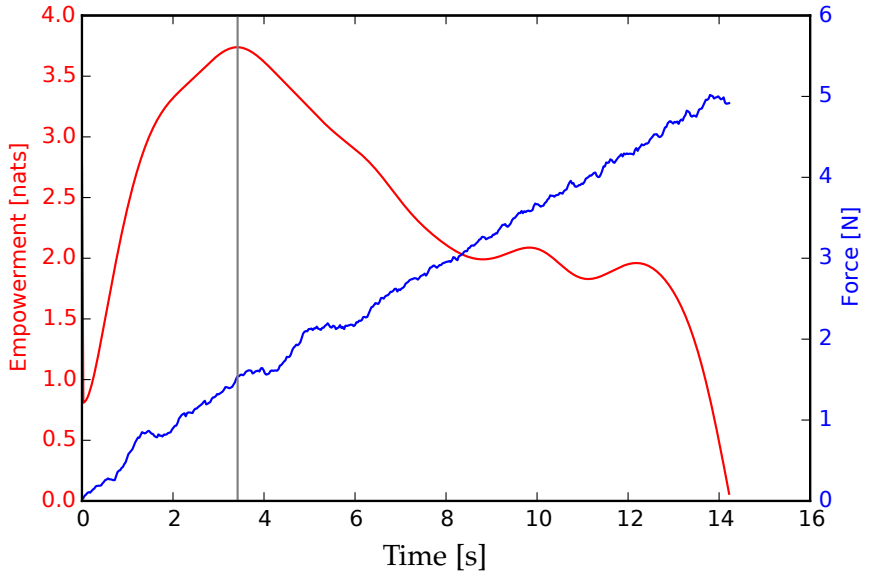


Figure 6.1: Empowerment computed with local channel approach with force change as actions and tactile sensor as observations. The amount of information flowing through the sensor (red) reaches a maximum when the normal force (blue) increases and reaches a maximum of information transfer between a force of 1 and 2 N. Saturation and too little force limit the amount of information transmitted and in turn limit empowerment.

empowerment value up to a maximum point with a force between 1 and 2 N. Only a small amount of liquid inside the BioTac skin is displaced at low forces. With an increased force more taxel receive a reasonable change as more liquid and skin gets deformed. At some force point the taxel activation saturates and the information flowing through the sensor is reduced. Even in this simple setup without a complicated interaction with objects slipping out of the robots fingers, some sense of an initial grip force policy can be explained by sensor behaviour alone.

6.3 EMPOWERMENT FOR NON-LINEAR TRANSITIONS

These previous approximations on empowerment are limiting the evaluation of the transition. They either linearise the system dynamics and thus destroy a lot of valuable system information [76], artificially limit their environment to only be composed of discrete states [9, 75] or learn directly from samples of the environment [78]. Even the method of leaning a locally linear state space is not applicable to all scenarios, as large latent spaces are needed for complex environments and highly non-linear systems pose a limit for learning accurate locally linear transitions.

6.3.1 *A bound on mutual information*

Evaluating empowerment on any non-linear function would be desirable and would open its application on more transition models learned with deep neural networks. If the transition is also differentiable, variational methods can be used, similar to

the ones used during model training, and directly use back-propagation through the dynamics. As the empowerment formulation

$$\begin{aligned}\mathcal{E}(\mathbf{z}) = \mathcal{C}_{\mathbf{u}\to\mathbf{z}'}(\mathbf{z}) &= \max_{\omega} \int \omega(\mathbf{u} \mid \mathbf{z}) \int p\big(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}\big) \ln \frac{p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})}{p(\mathbf{z}' \mid \mathbf{z})} \, d\mathbf{z}' d\mathbf{u} \\ &= \max_{\omega} \mathcal{I}\big(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\big) \end{aligned} \tag{6.3}$$

can be viewed as a channel capacity and is based on mutual information $\mathcal{I}$, the lower bound from [5] can be used to compute the inner part of the max operator. The difference between empowerment and regular channel capacity lies in the renaming of the transmitter and receiver into source and next state. Here the maximum information flowing from actions $\mathbf{u}$ through the environment back into the sensors $\mathbf{z}$ is measured. The source distribution from the channel capacity literature is denoted here as $\omega(\mathbf{u} \mid \mathbf{z})$ and used for maximising the channel capacity. It should not be confused with the policy used later on, especially with the close similarity between empowerment and the Bellman equation. This similarity however enables the use of stochastic optimal control techniques for optimising the source distribution later on.

For showing how the lower bound from [5] is applied to empowerment one must first expand the mutual information term from eq. (6.3):

$$\begin{aligned}\mathcal{I}\big(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\big) &:= D_{\mathrm{KL}}\big(p\big(\mathbf{z}', \mathbf{u} \mid \mathbf{z}\big) \parallel p\big(\mathbf{z}' \mid \mathbf{z}\big)\, \omega(\mathbf{u} \mid \mathbf{z})\big) \\ &= \iint p\big(\mathbf{z}', \mathbf{u} \mid \mathbf{z}\big) \ln \frac{p(\mathbf{z}', \mathbf{u} \mid \mathbf{z})}{p(\mathbf{z}' \mid \mathbf{z})\, \omega(\mathbf{u} \mid \mathbf{z})} \, d\mathbf{z}' d\mathbf{u} \\ &= \iint \omega(\mathbf{u} \mid \mathbf{z}) p\big(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}\big) \ln \frac{p(\mathbf{z}', \mathbf{u} \mid \mathbf{z})}{p(\mathbf{z}' \mid \mathbf{z})\, \omega(\mathbf{u} \mid \mathbf{z})} \, d\mathbf{z}' d\mathbf{u},\end{aligned}$$

where the transition $p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})$ is taken from the learned system dynamics model and the distribution $\omega(\mathbf{u} \mid \mathbf{z})$ is parametrised arbitrarily and then trained by maximising the mutual information. The problem is the marginal transition

$$p\big(\mathbf{z}' \mid \mathbf{z}\big) = \int \omega(\mathbf{u} \mid \mathbf{z})\, p\big(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}\big) \, d\mathbf{u},$$

which is dependent on both $p(\mathbf{z}' \mid \mathbf{u}, \mathbf{z})$ and $\omega(\mathbf{u} \mid \mathbf{z})$ and can not be parametrised by an additional function independent of the parameters of the transition. This is the main reason which makes eq. (6.3) intractable. The problem is similar to the intractability of the marginal likelihood for the latent variable model from section 3.1. Here this problem is circumvented by making sure that the marginal transition $p(\mathbf{z}' \mid \mathbf{z})$ does not appear in the cost since it is not necessary to optimise it directly.

The joint transition $p(\mathbf{z}', \mathbf{u} \mid \mathbf{z})$ is then factorised into $p(\mathbf{z}' \mid \mathbf{z})$ and $p(\mathbf{u} \mid \mathbf{z}', \mathbf{z})$. This introduction of the *planning distribution* $p(\mathbf{u} \mid \mathbf{z}', \mathbf{z})$ helps us to cancel out the marginal transition:

$$\mathcal{I}\big(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\big) = \iint \omega(\mathbf{u} \mid \mathbf{z}) p\big(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}\big) \ln \frac{p(\mathbf{u} \mid \mathbf{z}', \mathbf{z})}{\omega(\mathbf{u} \mid \mathbf{z})} \, d\mathbf{z}' d\mathbf{u}.$$

Here the lower bound from [5] comes into play and allows us to replace the planning distribution with its variational approximation $q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})$.

$$\begin{aligned}\mathcal{I}\big(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\big) &\geqslant \iint \omega(\mathbf{u} \mid \mathbf{z}) p\big(\mathbf{z}' \mid \mathbf{u}, \mathbf{z}\big) \ln \frac{q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})}{\omega(\mathbf{u} \mid \mathbf{z})} \, d\mathbf{z}' d\mathbf{u} \\ &=: \hat{\mathcal{I}}\big(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\big).\end{aligned} \tag{6.4}$$

The planning distribution helps us to interpret how this formulation measures the influence on the environment in a different way. It is still an expectation over the next states under the source and transition distribution, but this time the planning distribution probability gets compared to the source distribution. The planning distribution receives the current and next state pair and tries to predict the action that caused the transition, pairs where this prediction is hard results in action distributions with high variance and in low empowerment. The source distribution acts as a prior and normalises the result across different contexts $\mathbf{z}$.

The inequality holds for arbitrary approximate planning distributions, but the gap can be large dependent on the closeness between approximate and true planning distribution:

$$\mathcal{I} - \hat{\mathcal{I}} = \mathbb{E}_{\mathbf{z}' \sim p(\mathbf{z}'|\mathbf{z})} \left[D_{KL}\left(p\left(\mathbf{u} \mid \mathbf{z}', \mathbf{z}\right) \parallel q\left(\mathbf{u} \mid \mathbf{z}', \mathbf{z}\right) \right) \right]. \tag{6.5}$$

By maximizing the lower bound with respect to the parameters of $q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})$ the gap can be made smaller. The bound on mutual information then generates the following bound on empowerment:

$$\begin{aligned} \mathcal{E}(\mathbf{z}) = \max_{\omega} \mathcal{I}\left(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\right) &\geqslant \max_{\omega} \hat{\mathcal{I}}\left(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\right) \\ &= \max_{\omega} \mathbb{E}_{p(\mathbf{z}', \mathbf{u}|\mathbf{z})} \left[\ln \frac{q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})}{\omega(\mathbf{u} \mid \mathbf{z})} \right]. \end{aligned} \tag{6.6}$$

6.3.2 *Efficient empowerment optimisation*

In the following the efficient maximisation of empowerment will be explained. This maximisation makes use of the above defined bound in eq. (6.4) to approximate the measured mutual information between the action and sensor inputs. In this step the maximisation of mutual information and minimisation of the inequality gap is joined, defined in eq. (6.5), by jointly optimizing eq. (6.6) with respect to the parameters from both source distribution $\omega(\mathbf{u} \mid \mathbf{z})$ and planning distribution $q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})$. When the optimal source and planning distribution are used, the mutual information in eq. (6.4) term can be directly used for computing the empowerment value. In the following section the source maximisation will be omitted in the equations, as this is implicitly happening during the stochastic back-propagation step as described in section 3.4 for optimising all parameters.

The derived lower bound of empowerment still contains integrals over the continuous values of the next state and action. This intractability is solved by using a Monte-Carlo sampling approach just as in [49, 50]:

$$\begin{aligned} \hat{\mathcal{I}}\left(\mathbf{z}'; \mathbf{u} \mid \mathbf{z}\right) &= \mathbb{E}_{p(\mathbf{z}', \mathbf{u}|\mathbf{z})} \left[\ln \frac{q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})}{\omega(\mathbf{u} \mid \mathbf{z})} \right] \\ &\approx \frac{1}{N} \sum_{n=1}^{N} \left[\ln q\left(\mathbf{u}^{(n)} \mid \mathbf{z}, \mathbf{z}'^{(n)}\right) - \ln \omega\left(\mathbf{u}^{(n)} \mid \mathbf{z}\right) \right], \end{aligned} \tag{6.7}$$

where $\mathbf{u}^{(n)} \sim \omega(\mathbf{u} \mid \mathbf{z})$ and $\mathbf{z}'^{(n)} \sim p\left(\mathbf{z}' \mid \mathbf{z}, \mathbf{u}^{(n)}\right)$. As shown in [49, 50] these two sampling steps from the source and the transition distribution need to be differentiable for properly applying the reparametrisation trick (cf. section 3.3). Another assumption that was adopted from [49, 50] is, that if during optimisation a large

batchsize is used, single samples with $N = 1$ in the Monte-Carlo step are acceptable. All functions representing the distributions in this equation should have a smooth gradient without discontinuities. For the source and the planning distribution neural networks are a natural choice. Depending on the given model of the agent the transition can have restrictions which might hinder a proper gradient propagation. In most of the cases the transition is learned in a smooth latent space with DVBF which showed to be a good parametrisation for the use in approximating empowerment.

The back-propagation of gradients through the transition is crucial here. Maximising eq. (6.7) for computing the channel capacity seems to increase the entropy of the source distribution without evident bounds. The first term in eq. (6.7) counteracts this entropy increase, as the next latent state $\mathbf{z}'$ is affected by samples from the source distribution. These samples of the source distribution are passed through the transition and their effect on the environment is represented in the resulting latent state $\mathbf{z}'$. The effect of the environment includes certain limitations like saturation of the motor current, maximum velocities or friction. Those limitations change the information transmitted through the environment and reduce the action reconstruction capability of the planning distribution. Increasing the entropy of source distribution would then cover too much area in the actions space where the expected one-shot transfer entropy is too low and the overall mutual information would be less than the maximum capacity of the channel. A transition or environment without any of such restrictions can definitely result in an unbounded entropy of the source distribution, but those transitions are not realistic nor interesting for the use with intrinsic motivation, c.f. "Tragedy of the Greek Gods" from [4]. Even though [78] uses the same lower bound on empowerment, their lack of an explicit model of the environment forces them to artificially limit the entropy of the source distribution. They use a variational solution for the source distribution which couples the planning distribution to the source resulting in a squared error loss for optimising the source.

6.3.3 *Empowerment exploitation*

It is now possible to compute empowerment in an efficient way for continuous valued states and actions. The next step is to exploit this cost for doing intrinsically motivated control. For that the already learned dynamics model can be reused together with optimal-control for finding the control policy. With the methods proposed in section 5.6 make it possible to use the learned non-linear system transition to optimise a control policy by minimising a differentiable cost function. The policy can then be trained by stochastic gradient descend and back-propagate errors through state space paths sampled from policy and transition distributions. The trajectories sampled with

$$\mathbf{z}_{1:T}, \mathbf{u}_{1:T-1} \sim p(\mathbf{z}_1) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{u}_t, \mathbf{z}_t) \pi(\mathbf{u}_t \mid \mathbf{z}_t), \tag{6.8}$$

Algorithm 2 Joint training of policy π, source ω and approx. planning distribution $q(\mathbf{u} \mid \mathbf{z}', \mathbf{z})$

Require: Dynamics $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$, cumulation horizon T, initialisations for π, ω, and q
repeat
▷ Estimate cumulated empowerment $\mathcal{C}$, eq. (6.9)
for $m = 1 : M$ **do**
▷ π-step: Sample dynamics with policy π:
$\mathbf{z}_{1:T}^{(m)}, \mathbf{u}_{1:T-1}^{(m)} \sim p(\mathbf{z}_1) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{u}_t, \mathbf{z}_t) \pi(\mathbf{u}_t \mid \mathbf{z}_t)$
for $t = 1 : T$ **do**
▷ ω-step: Sample one step with policy ω:
$\mathbf{u}_t^{\omega} \sim \omega\left(\mathbf{u}_t \mid \mathbf{z}_t^{(m)}\right),\ \mathbf{z}_{t+1}^{\omega} \sim p\left(\mathbf{z}_{t+1} \mid \mathbf{u}_t^{\omega}, \mathbf{z}_t^{(m)}\right)$
▷ One-shot MI, eq. (6.7)
$\hat{\mathcal{I}}_t^m \leftarrow \ln q_\theta\left(\mathbf{u}_t^{\omega} \mid \mathbf{z}_{t+1}^{\omega}, \mathbf{z}_t^{(m)}\right) - \ln \omega\left(\mathbf{u}_t^{\omega} \mid \mathbf{z}_t^{(m)}\right)$
end for
end for
▷ Cost and KL towards policy prior π_0
$\mathcal{C} \leftarrow \sum_m \sum_t \left[\beta \hat{\mathcal{I}}_t^m + D_{KL}\left(\pi\left(\mathbf{u}_t^{(m)} \mid \mathbf{z}_t^{(m)}\right) \| \pi_0\right)\right]$
▷ Gradient ascent in ω and q with some learning rate λ
$\theta \leftarrow \theta + \lambda_\theta \nabla_\theta \mathcal{C}$
▷ Gradient ascent in π with some learning rate λ
$\chi \leftarrow \chi + \lambda_\chi \nabla_\chi \mathcal{C}$
until convergence

are then used to estimate the cost:

$$\begin{aligned} \mathcal{C}_\pi &:= -\mathbb{E}_\pi\left[\sum_{t=1}^{T} \hat{\mathcal{E}}(\mathbf{z}_t) + D_{KL}(\pi(\mathbf{u}_t \mid \mathbf{z}_t) \| \pi_0)\right] \\ &\approx -\frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} \left[\beta \hat{\mathcal{E}}\left(\mathbf{z}_t^{(m)}\right) + D_{KL}\left(\pi\left(\mathbf{u}_t^{(m)} \mid \mathbf{z}_t^{(m)}\right) \| \pi_0\right)\right]. \end{aligned} \tag{6.9}$$

This is the cost definition from section 5.6, where $\mathcal{C}(\mathbf{z}_t)$ got replaced with the lower bound on empowerment $\hat{\mathcal{E}}(\mathbf{z}_t)$. By minimizing this cost function with respect to the policy parameters the agent is trained to exploit empowerment.

The above steps are summarised as pseudo code in algorithm 2. Compared to greedy optimisation of empowerment as used in [4], here it is not necessary to evaluate empowerment during control for finding the right action, as the long term reward maximisation is already amortised into the learned policy. The sampling and inference steps are all done off-line as all of the necessary information is stored in the learned model of the environment. The policy has then amortised all of the computational cost into a single function which can easily be evaluated in real-time on a robot. Moreover all empowerment maximising distributions are trained to be functions dependent on the current state, and can even be used for evaluating empowerment on the fly after training.

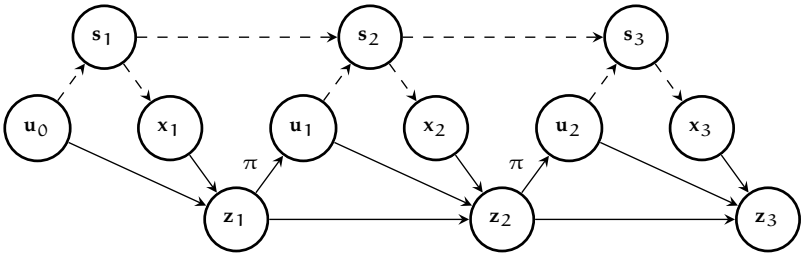


Figure 6.2: Graph representing the action perception loop together with the learned model. Dashed lines show interactions and dependencies that happen in the real environment where no gradient or model is available. The continuous lines represent all functions and transformations which are fully known. Actions $\mathbf{u}_t$ are affecting both the real environment and the latent trajectory and update steps from observations $\mathbf{x}$ correcting the latent state.

6.3.4 *Comparison to action perception loops in empowerment literature*

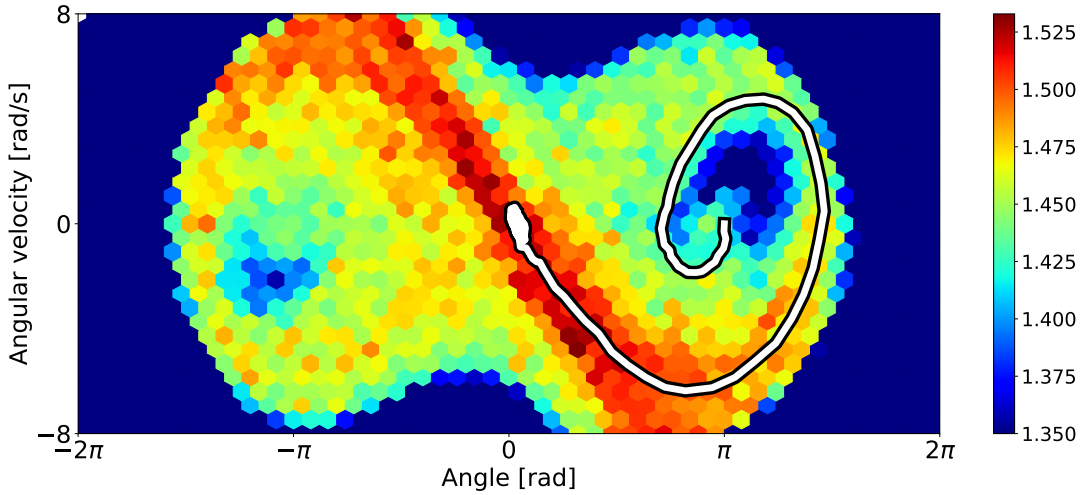
In the empowerment literature different channels are considered for computing the mutual information. Either the information is transmitted from action to the full environment state, next observation or next memory state. Each one might have its own benefits in certain situations, but this thesis concentrates on the full environment state setting. However as there is no access to the true system state, a learned model has to be used. A representation of this perception action loop is shown in fig. 6.2. It shares similarity with the memory based empowerment computation from [72] but the latent dynamics is forced to mimic the true state dynamics in the DVBF model learning step. Therefore the latent state can not be considered as memory according to the definitions in the current empowerment literature. However there is a second option for the choice of the path for computing the mutual information. Instead of the next latent state the next observation can be used directly in the mutual information estimation. The mutual information in eq. (6.3) then gets replaced with

$$\mathcal{I}\left(\mathbf{x}'; \mathbf{u} \mid \mathbf{z}\right)$$

where $\mathbf{x}'$ is the observation that was generated from $\mathbf{z}'$ by the observation model.

6.3.5 *Evaluation in simulation*

Before applying the presented method on complex environments, they must first be simulated as simplified environments similar to those described in previous empowerment publications, such as the pendulum environment [75–77], moving through space obstructed by walls [74, 78, 79] or swarm behaviour [72]. The experiments in the following sections include bouncing balls, where a ball is trapped in a confined space and the pendulum environment, which serves as an example for empowerment exploitation, where the model of the environment is known in advance. In the end the algorithm will be evaluated on more complicated environments such as a bipedal walker and a multi agent environment with several balls trapped in a box.



(a) Pendulum state space coloured with empowerment values and with swing-up trajectory in white. The swing up procedure is controlled by a policy trained by maximising empowerment as reward. The colour represents the accumulated empowerment value computed with eq. (5.23) and the empowerment lower bound from eq. (6.7). An horizon of 30 time-steps was used for the value function.

(b) Rendering of the pendulum swing-up sequence from fig. 6.3a.

Figure 6.3: Results for the pendulum experiment.

Pendulum swing-up

In the pendulum experiment the state transition was provided as the numerical solution of its differential equation. There the state is two dimensional with angle and angular velocity while the action is one dimensional and represents the torque at the pendulum pivot. It is implemented such, that error can be back-propagated through the dynamics. Parameters and details about the differential equations can be found in appendix B.2. Compared to previous experiments with the pendulum this is the first time where empowerment is optimised for an agent with non-linear dynamics where both actions and observations have a continuous value. [76] used a continuous state space but discretised actions. Earlier work in [77] still had to linearise the dynamics for computing empowerment for a fully continuous pendulum system.

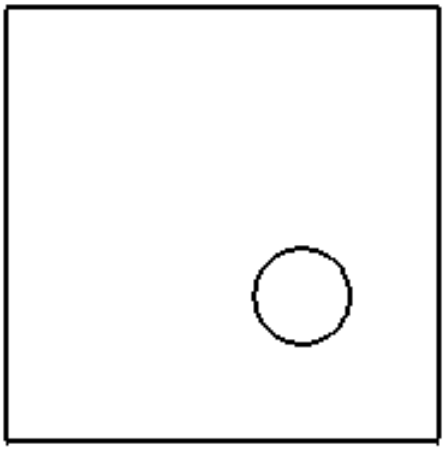
The resulting expected future empowerment landscape is shown in fig. 6.3a. This plot was generated by evaluating the empowerment value $\mathcal{V}_\pi$ by Monte-Carlo sampling as in eq. (5.23) for each starting point, binning close starting points and averaging the values. A stable upright pendulum state is represented by the centre of the graph where the angle and the velocity is zero. This region also displays the highest empowerment values with gradual increase from both ends of a diagonal through the centre. This diagonal represents states where the current velocity already counteracts gravity and only a slight push from the action is needed for coming closer to the upright position. A spiral pattern on both sides of the diagonal creates a swing-up motion when followed greedily in each time-step. The values presented in fig. 6.3a are a combination of the empowerment reward with

an evaluation of the value function from eq. (5.23), where both contributed to the smoothness of the value function. This comes from the similarity of the Bellman equation and the empowerment formulation. Both are using an auto-regressive sampling process from the transition and some kind of policy. This holds especially true for n-step empowerment, where a whole chain of transition samples is taken, and a similar landscape can be generated with the empowerment reward alone. Here the angles fed as observations are unbound and can represent higher values than π. This creates an unbalance in the states beyond $[-\pi, \pi]$, without changing the overall smoothness of the empowerment landscape. However, it makes the policy favour a certain direction for the swing-up sequence. The white trajectory represents the swing-up procedure from fig. 6.3b which follows the empowerment gradient to the top. Previous work like [76, 77] come to the same results regarding empowerment landscapes and successful pendulum swing-up.

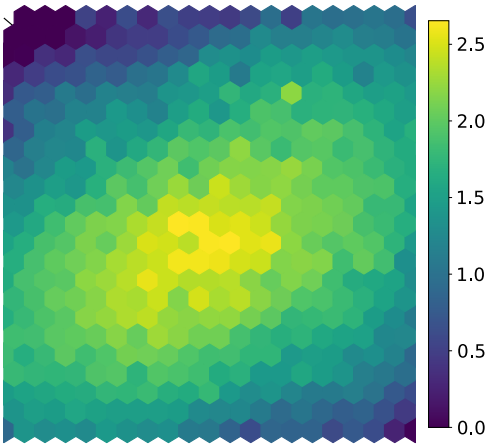
Bouncing ball experiment

In the bouncing ball environment a movable ball got trapped in a box, with their respective sizes represented in fig. 6.4a, and provided it with an action representing the velocity of the ball. The position is then defined as the discrete integration of that velocity action. Such particles in limited space also got evaluated with other definitions of intrinsic motivation, not based on empowerment, such as [69]. Collisions are non-elastic, meaning that a ball that hits the walls seems to stick to walls and only actions away from the wall can move the ball towards the centre. This time the transition and interactions with the environment are learned by DVBF with the parameters noted in appendix B.3.

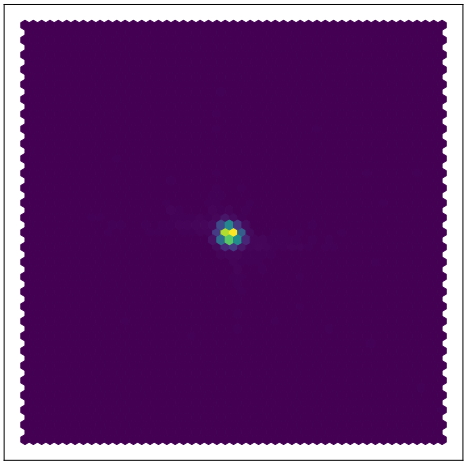
Again similar results as reported in previous literature [72, 74, 78, 79] are achieved where an empowered agent can chose between being limited in its actions by a nearby wall or staying in an open space with lots of possible states to reach. Figure 6.4c shows the distribution of an empowered agent which tries to remain in the centre between the confining walls. In fig. 6.4b the empowerment values for each state in the box are shown with a clear maximum in the centre. The actions that lead to these states are shown in fig. 6.4d, all pointing towards the centre of the box. In comparison to the empowered policy, uniformly distributed actions would result in a higher number of states near walls and corners due to the non-elasticity of contacts. A second experiment performed on the same environment is shown in fig. 6.5. There, just as in the other experiments in simulated environments, the full sequence of latent states was sampled from the generative model and used as the starting states for maximising the empowerment lower bound. The generatively sampled latent states are then dependent on the policy which will affect the learning dynamics. As soon as the policy optimisation gets stuck in a certain state space area, more values from that area will be passed to the empowerment optimisation. These dynamics make it much harder to tune hyper-parameters. Using the full sequence of dream states as starting states for optimising empowerment will lead to increased variance in the final policy and goal position. When using only filtered states as starting states a much sharper result can be achieved as seen in fig. 6.4c.



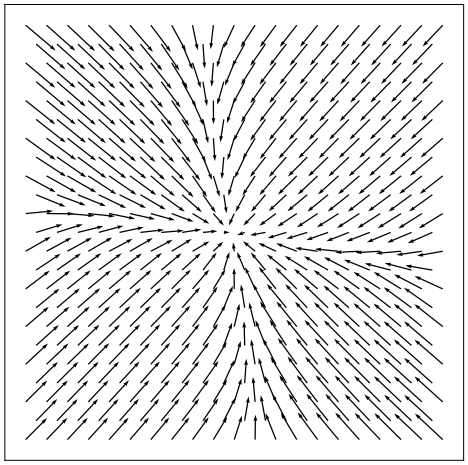
(a) Rendering of the ball in a box environment.



(b) Ball position space with corresponding empowerment values estimated wiht eq. (6.7).

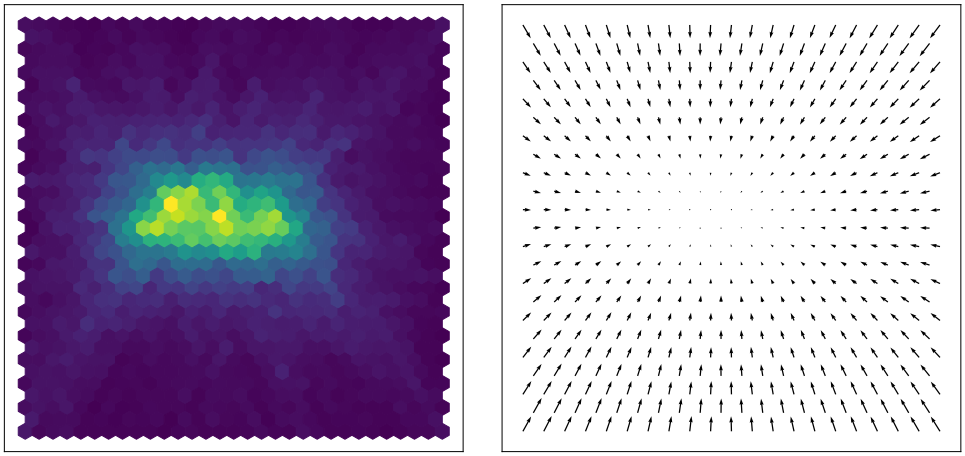


(c) Distribution of the ball position after running empowerment exploiting policy in simulated environment.



(d) Ball position space with force vectors sampled from policy trained by maximising eq. (6.9) using empowerment as reward.

Figure 6.4: Empowerment exploitation results for the single ball in a box experiment. The ball moves into the centre where it has the maximum possible number of future options. In contrast to the other experiments empowerment was only optimised on filtered latent spaces.



(a) Distribution of the ball position after running empowerment exploiting policy in simulated environment.

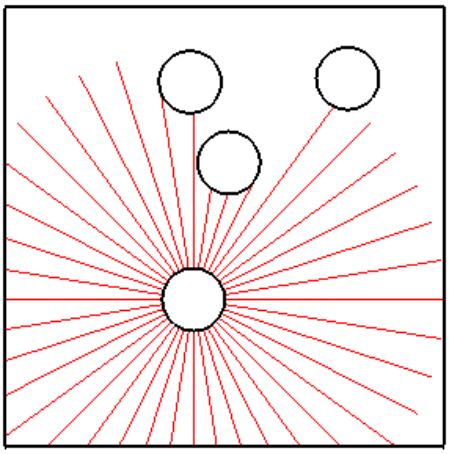
(b) Ball position space with force vectors sampled from policy trained by maximising eq. (6.9) using empowerment as reward.

Figure 6.5: Results for the single ball in a box experiment with empowerment maximisation optimised on the full trajectory of dream states. Using dream states can lead to higher variance in final ball position as compared to using only filtered latent states seen in fig. 6.4.

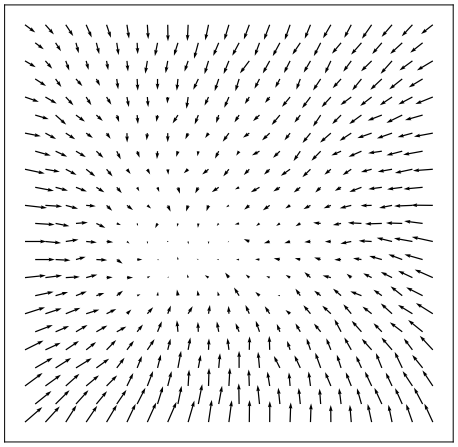
Multiple balls in a box

For making the ball in a box environment more interesting the number of balls is increased and actions now represent forces instead of velocities. Every agent acts independently during evaluation, where each agent received the learned model and policy, trained on data-points from all agents. They have no means of communication with each other except through physical interaction in the environment. For perceiving the presence of other agents 40 distance sensors were introduced equally distributed around the agents outline. Together with its own position and velocity, these sensors form the observation of the agent. These distance sensor readings can be compared to the LIDAR frequently used in robotics to map the robots environment. The maximum distance and distribution of these virtual LIDAR sensors can be seen in fig. 6.6a in the form of red lines. Detailed parameters are shown in appendix B.4.

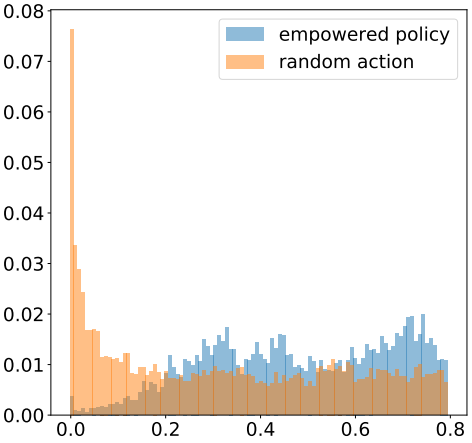
The results of an empowered agent in this environment is shown in fig. 6.6b, fig. 6.6c and fig. 6.6d. Averaged over all different agent constellations the policy (fig. 6.6b) exhibits similar behaviour as in the single ball case. The agents try to stay away from walls and remain in the centre of the box. The distribution of distances between each pair of balls as shown in fig. 6.6d indicates the balls tendency to stay away from each other at a constant distance of approximately 0.3. In the distribution of wall to ball distances in fig. 6.6c no such clear optimal distance is visible, the wall is repelling with a plateau towards the centre of the box. This clear difference comes from the fact that other agents react and avoid the agent in control, creating an additional form of influence to the environment. Acting through other agents by pushing them around increases its empowerment. The total behaviour



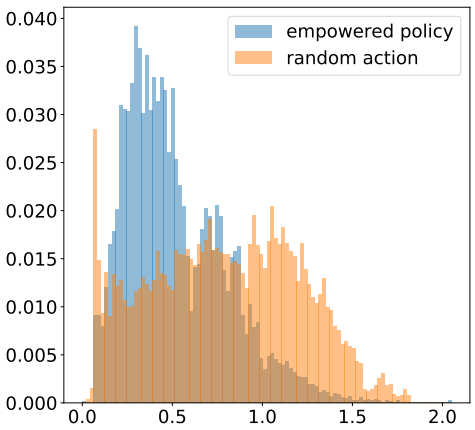
(a) Rendering of multi agent environment. Each agent can sense the local environment through LIDAR. The sensor readout of one agent is shown in red.



(b) Ball position space with force vectors sampled from policy trained with empowerment as reward. This force field is averaged over all agents and their configuration in space.



(c) Distribution of ball to wall distances. The wall in the first spacial dimension was used for the distance computation. Distances around 0.8 represent the box centre.



(d) Distribution of ball to ball distances. Empowered agents tend to keep a certain distance forming a flock in the centre of the box.

Figure 6.6: Results for the multi agent environment. Each agent operates independently without access to other agent states except the information through distance sensors. Here an empowered agent avoids wall while forming a swarm in the centre of the box.

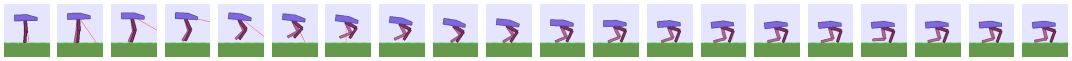


Figure 6.7: Bipedal walker balancing without external cost. Empowerment makes the agent find a stable position on two legs.

of the agent is therefore to move towards the centre, but keeping the distance to other agents constant, essentially creating a cluster of agents.

Bipedal balancing

Applied to a bipedal robot, empowerment should also keep the agent in a balancing position just as in the pendulum example. This time the state and action space has a higher number of dimensions and the balancing pivot is not actuated like in the inverted pendulum. Instead, the knees and hips of the walker are equipped with actuators. The environment used here comes from the OpenAI Gym library[11]. Here the dynamics are modelled with DVBF and use as input a sequence of all observations that are provided by the environment: angular position and speed of the head, both knee and hip joints, global velocity, contact sensors of the feet and ten distance sensors pointed towards the ground. Parameters for training the model are shown in appendix B.5. Exploited empowerment on this robot makes the robot balance and reach a stable position. One of such stabilising position is shown in fig. 6.7. As soon as the feet lose contact to the ground or the robot gets tilted so much that there is no escape from falling over and empowerment drops to a minimum.

6.3.6 *Multi-step empowerment*

The experiment on the pendulum environment showed that the combination of the empowerment reward and the value function is necessary for finding a good swing-up trajectory. For greedily choosing the action for the next step as in [4], a 1-step empowerment cost would not be sufficient for swinging the pendulum up. As a solution the authors in [4] decided to use the n-step empowerment formulation where the mutual information between the sequence of n actions and the final state after n-steps is used in the empowerment formulation. This is however computationally more expensive than 1-step empowerment, but can be compared to the complexity of computing the value function. The prediction and accumulation of several transition steps is simply moved from the value function into the mutual information for measuring the amount of options in n-steps.

Empowerment with n-steps can also be computed with the new method explained in section 6.3.2 with only slight modifications. The number of action values predicted by the source and the planning distribution get increased by a factor of n and the transition gets sampled multiple times with a different section of the longer action vector fed into each transition. The intermediate states are then discarded and only the first and last state are used by the planning distribution. When the intermediate states are not discarded the variant of the method will be called path-empowerment. Alternatively to the changed sampling process DVBF can also be retrained with only every nth data-point of the time series. The result for a pen-

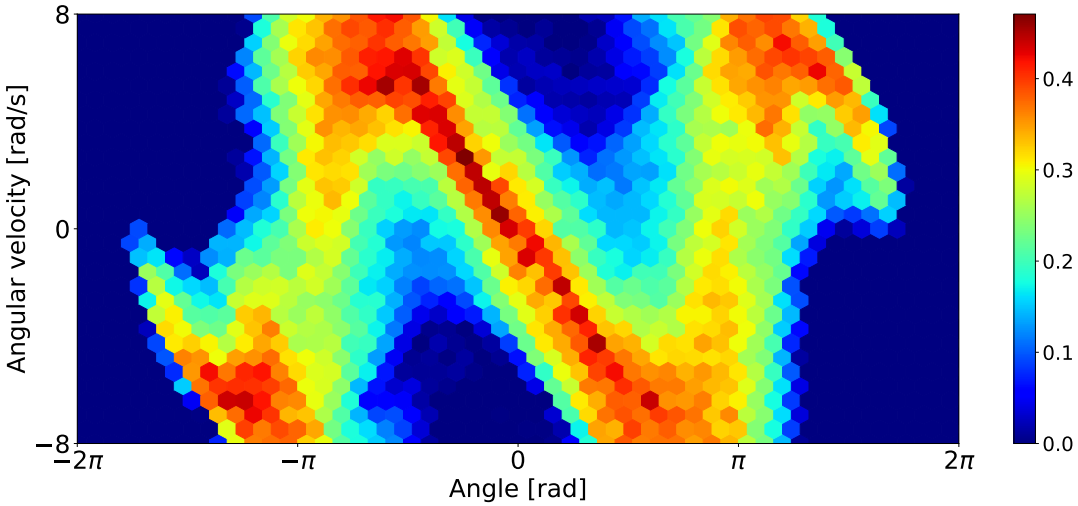


Figure 6.8: Empowerment landscape for 5-step empowerment on pendulum environment. In contrast to fig. 6.3a the values here come directly from the Monte-Carlo evaluation of the empowerment function (eq. (6.7)) without accumulation.

dulum with 5-step empowerment can be seen in fig. 6.8. This time instead of the accumulated value the plain empowerment function (eq. (6.7)) is shown. Similar to fig. 6.3a, diagonal patterns with high reward are visible which could be followed greedily for a swing-up trajectory.

7

EVALUATION ON FLYING ROBOTS

This chapter will show how to apply the above described methods on actual hardware. A quadrocopter was chosen as an appropriate candidate for the first application. For benefiting of the proposed machine learning methods the flying robot is equipped with additional sensors. These sensors do not need to be calibrated or placed at specific positions since the whole dynamic will be learned from raw data streams. The usual setup for quadrocopter experiments includes a marker based tracking system. A camera array records the illuminated markers and triangulates their position to compute exact Cartesian coordinates and orientations of the robot. Cartesian coordinates make it easier to apply the known model of the quadrocopter for employing control. Here a different approach is taken and only local sensing will be used to control the robot. This makes it possible to perform all computation on-board, as no position data needs to be transferred. This methodology can work with arbitrary sensors in unknown position configurations and it can enable usage of robots that would otherwise be too hard to model by classical methods. The sensors do not need to be calibrated and their activation can be highly correlated with each other. The only constraint is that they should be able to convey enough information about the current state for controlling the robot appropriately. In the situations where it is unclear if the current sensor configuration conveys to little information to the controller, the same methods as for unsupervised control can be used to assess the current information throughput through the robot hardware. Channel capacity from actuators to the sensors can be used to measure the size of the bottleneck that a sensor or actuator choice or configuration might imply. This approach to perform all sensing, sensor processing and movement generation on board of the robot also serves as a showcase for the capabilities of the presented time-series modelling and use for policy optimisation. It can be used even beyond the use of intrinsic motivation as the reward function.

7.1 CONSTRUCTION AND EVOLUTION OF QUADROCOPTER HARDWARE

During the learning phase the quadrocopter would bump several times with walls or other objects, and therefore needs to be protected against crashes. The main problem are the fragile and fast spinning propellers which are damaged after the slightest touch with other objects. The solution was to build a frame around the quadrocopter, also serving as a structure for mounting the different distance sensors. After a few iterations with wooden structures the material of choice was changed to laser cut polyamide sheets. The final cutting pattern for the laser cutter can be seen in fig. 7.1. It is meant to be cut out of 2 mm polyamide sheets and then snapped together with the interlocking notches all around the edges of each part. These notches are then secured by a glue made out of polyamide dissolved in formic acid, which forms a frame out of solid plastic after the acid has been evaporated. It is especially lightweight due to cut-outs at regions that are

unimportant for structural strength and increased stiffness through vertical beams throughout the frame. The curved edges on the outer wall of the frame and the glued clip mechanisms provide extra strength. The frame also features rectangular holes for mounting the laser distance sensors, shown in fig. C.1a, all around the quadrocopter as well as a mounting position in the centre for the main computer. Inside that protective frame, an x-shaped structure of carbon fibre beams for connecting motors and flight-controller is mounted. The motors are connected through 3D printed adapters (see fig. C.2a) to this carbon fibre cross. For connecting the flight controller firmly to the carbon fibre frame the model shown in fig. C.2c is used. It also serves as a holder for the battery which is secured using a Velcro strap. The rigid connection through the plastic mount and the carbon frame to the motors is important for a stable flight, as any flexing and high amplitude vibration between the motor and the inertia measurement unit (IMU) would destabilise the high frequency feedback loop for levelling the quadrocopter inside the flight controller. This was a problem in early versions where oscillations of the quadrocopter frame were transferred to the proportional–integral–derivative (PID) output and caused overheated motors and instabilities during flight. The stability during flight should be tested with a manual controller every time the mechanical structure is changed. When the robot is not controllable with manual command and feedback through the human pilot chances are high that no machine learning algorithm is able to fly the machine.

Such information loss can also occur during the transmission of commands to the flight controller. Flight controller offer several options for the transmission of the control signals. One can typically choose between analogue signals such as pulse width modulation (PWM) for each channel or a single digital serial connection. The PWM signal does not use analogue voltages, but can still be considered analogue, as the time from one flange to another is a continuous value and is influenced by the impedance of the wire and the environment. Initial tests showed that for the transmission from the main computer to the flight controller a digital connection should be used, as it is less affected by external disturbances and can even employ error correction.

Another source of loss of controllability can hide inside the flight controller. The first version of the quadrocopter used the Illuminati 32 Flight as flight controller with an MPU6050 chip as the IMU. After every collision with walls this flight controller and IMU combination changed its internal horizon a tiny bit. This change caused an offset between actions and the flight velocity. During random exploration this problem is not apparent as these tiny offsets vanish in the overall noise. Only during final execution of the learned policy this deviation is visible through extreme velocity changes directly after collisions. The second version of the quadrocopter features the newer flight controller CL Racing F7 which uses the ICM20602 IMU. Both versions were compared in a motion tracked and PID position controlled experiment with manually induced collisions. The second version showed no horizon offset after collisions and stayed stable in the same positions with actions at neutral position.

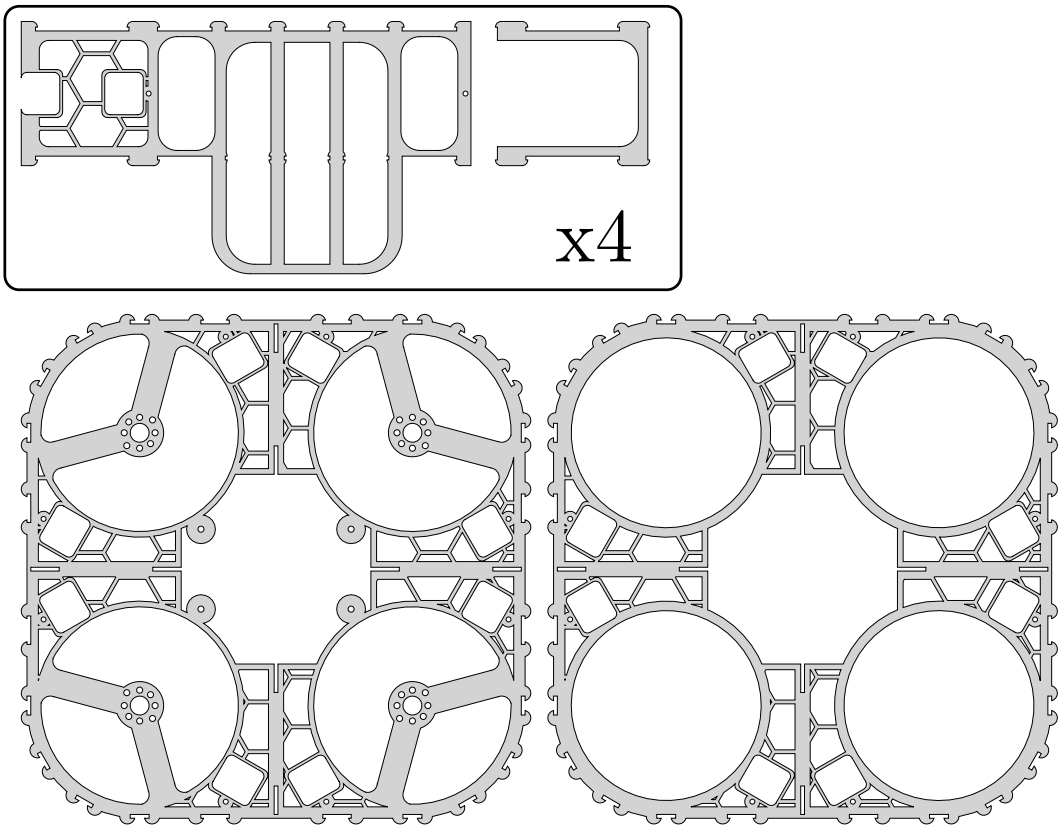


Figure 7.1: Structure of quadrocopter protective frame.

7.2 QUADROCOPTER WITH DISTANCE SENSING

A goal during the empowerment experiment with quadrocopters was to sense and compute everything locally. This excludes external positioning systems as well as heavy communication of sensor or action data. The quadrocopter therefore needs to be equipped with a 360 degree of view sensing capability. For this purpose single time-of-flight laser sensors where spread across the quadrocopter hull. The position of the sensors in the hull can be seen in fig. 7.2. They are soldered on breakout boards as shown in fig. C.1b and then mounted in 3D printed holders as depicted in fig. C.1a for installing them in the quadrocopter hull. As the distance sensor in the first version of the quadrocopter the VL53L0X distance sensor was used. This sensor is able to measure distances up to 2 m but are clipped to 1.5 m for model learning since readings in between that range become increasingly noisy. A second version of the quadrocopter uses the newer VL53L1X sensor with an increased distance of up to 4 m. These time-of-flight sensors measure distances by sending an infrared laser beam pulse and measuring the time it needs to come back to them. The communication with the on-board Raspberry Pi is done over the inter-integrated circuit (I2C) bus. Each chip comes with the same address used for identifying the sensors on the bus and have to be set to individual addresses during start-up. For this purpose two I2C multiplexer are used to individually communicate and set individual addesses for each sensor. The overall circuit used

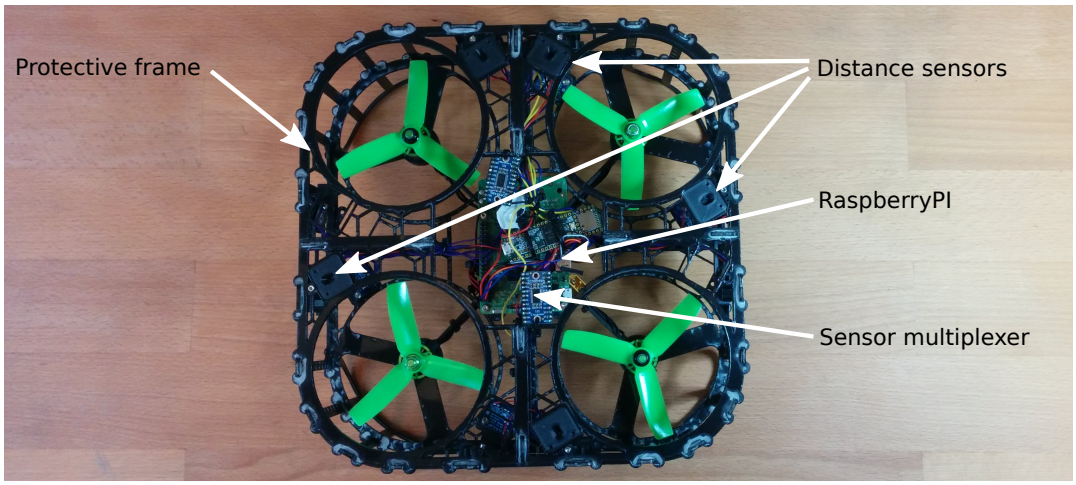


Figure 7.2: Structure of quadrocopter hardware.

for the distance sensor is shown in fig. C.3. After this initialisation step all ports of the multiplexer are opened such that all sensor are connected to the same bus.

7.3 LEARNED MODEL OF QUADROCOPTER DYNAMICS

For computing empowerment and optimising the policy an accurate model of the quadrocopter dynamics is needed. For that, a DVBF model is learned from section 5.4 on the raw sensor data sequence. The input observation vector consists of the 16 LIDAR, 3D acceleration, 3D angular velocity and battery voltage data. The action inputs are the concatenated roll, pitch, throttle and yaw inputs from 16 time-steps. Just as in the simulated experiments from section 6.3.5 first the DVBF model is trained on this data, then the parameters of the policy and the lower bound on empowerment get optimised jointly with the algorithm described in algorithm 2. The initial dataset for learning the model was recorded by manually flying the quadrocopter evenly in all positions in the cage and by making sure enough collisions with walls are made. In the second version of the quadrocopter this process was assisted by a PID controller on the position from the motion tracking system. This controller pushed the quadrocopter back towards the centre when no manual controller commands are executed. During tuning the model parameters this data set was augmented with roll-outs from the intermediate policies. Before training these data have to be preprocessed by clipping at the sensor limits and scaling them into the range between -1 and 1. This preconditions the inputs for proper use in neural networks. Also the noisy output of the policy has to be post processed to support the flight controller. The raw output of the policy is a Gaussian distribution with mean and variance predicted through a feed-forward neural network. The flight controller cannot handle noisy input, therefore the inputs for throttle are accumulated with a leaky integrator. Roll and pitch outputs are processed by a discretisation step mapping into three states: maximum inclination in both directions and a zero action. In the second version of the quadrocopter this integrator was changed to an moving average filter over the sequence of actions. Each run of the policy produces 16 steps times 4 channels which are averaged such that the channel

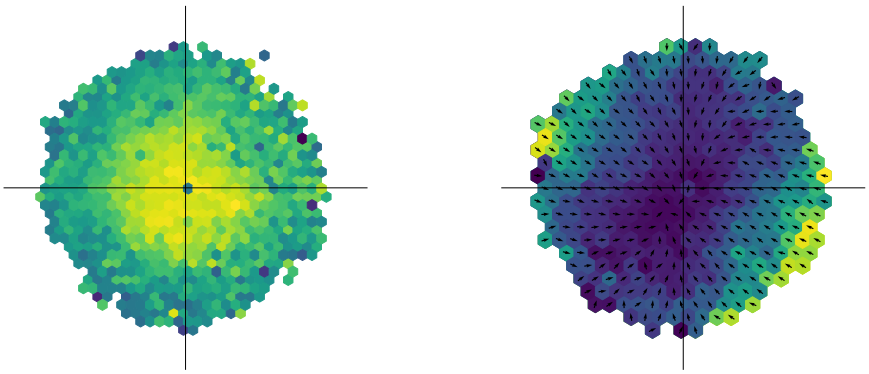
output before filtering is kept constant for 16 steps. In both versions the output is clipped to stay in the range between 592 and 1392. The throttle range is different with limits between 350 and 1300 and the yaw channel is kept completely at neutral position. These limits are not the same as those used for scaling the inputs to the learned DVBF model. The model limits might have a wider range for capturing all influences of actions on the systems state change. Limits of the policy output prevent the quadrocopter from flipping or ascending too fast. The DVBF model in both versions used a modified version of the locally linear transition from eq. (5.17). Instead of directly predicting the weight vector α with a feed-forward neural network it is created by adding a stochastic layer in f_ψ from eq. (5.17). The stochastic layer is defined by a conditional Gaussian just as the encoder in DVBF which is again regularized with an additional KL-divergence term pulling the stochastic layer towards a standard normal prior. This structure is similar to that from [3] when omitting the transition and inference model for the stochastic layer. The detailed structure of the individual neural networks used for the model, policy, planning and source distribution can be found in appendix C.1.

7.4 RESULTING POLICIES

Before running the trained filter and policy on the quadrocopter, the resulting policy can be visualised in a two-dimensional space using the LIDAR sensor data. Even though there is no absolute positioning system and the distance sensors are bound by a maximum distance smaller than the robot cage, their values can be used to compute an offset from the cage centre position. This offset is computed as the sum of all normalised sensor directions in the horizontal plane of the quadrocopter and multiplied with their measured distance. When all sensors are maxed out, the scaled direction vectors have the same length and cancel out. As soon as an object appears in the sensors proximity some of the sensors will read smaller distances and the sum of scaled vectors will point in the opposing direction. In this space a position close to the origin should be a desired position for an empowerment exploiting policy. The centre in this sensor projection represents all positions in space that are equally distant to all surrounding objects. This two-dimensional offset was used for creating the figures in fig. 7.3 and fig. 7.4. In fig. 7.3a points are coloured by the empowerment value of the corresponding state with a maximum at the origin of the graph. Figure 7.3b and fig. 7.4 show the policy optimised for exploiting empowerment. The arrows represent the direction in which the policy tries to move the offset while the colour represents the length of the vectors. The policies for both version 1 and 2 look very similar in terms of their actions pointing towards the centre of the cage. None of the drawbacks of the initial flight controller are visible at this stage yet. Both versions seem to move the robot to the centre of the cage.

7.5 LATENT SPACE OF QUADROCOPTER

The latent space for the model used in fig. 7.3 can be seen in fig. 7.5. The positions of the individual hexagons are represented in the same way as the policy representation from fig. 7.3, with the centre of the plot representing a position where all



(a) Empowerment landscape of quadrocopter position in cage.

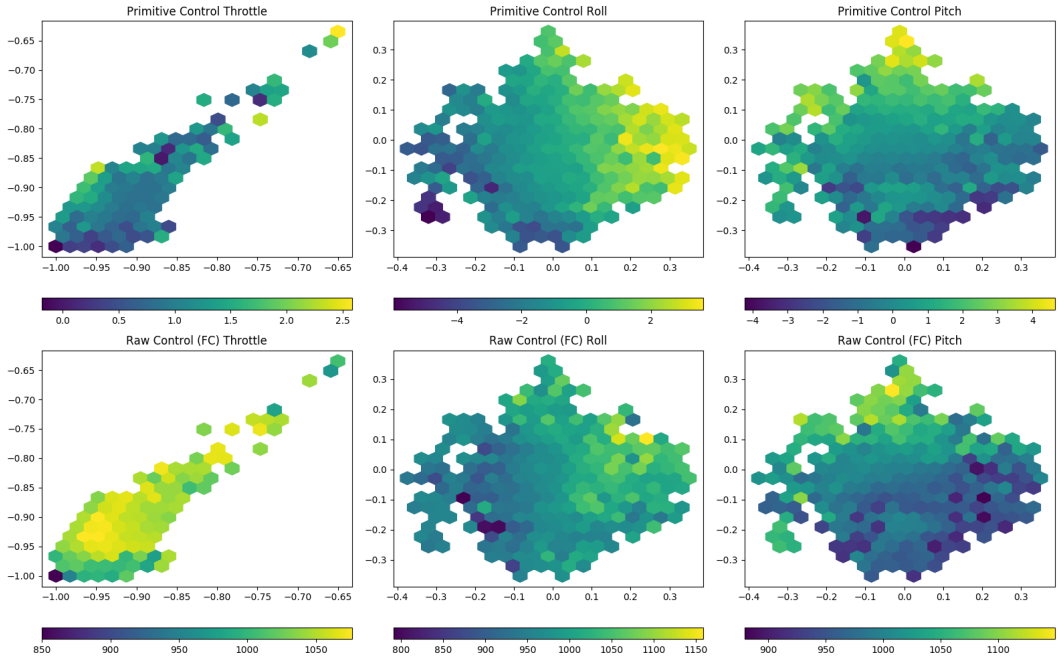
(b) Resulting policy exploiting empowerment.

Figure 7.3: Results for the quadrocopter experiment. Several quadrocopter of version 1 were used to create this dataset. The results in this plot are computed on training data.

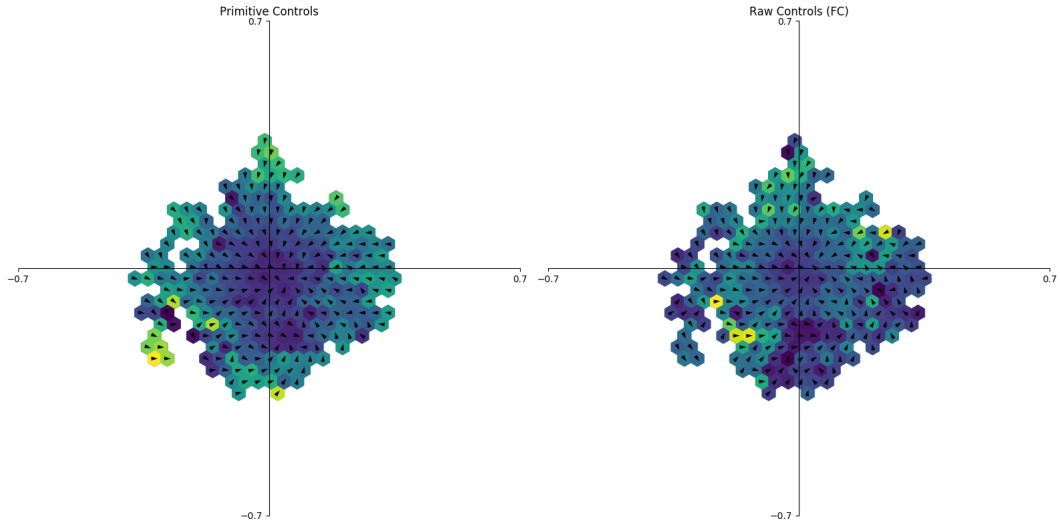
sensors show an equal distance in all directions. The colouring represents the value of the respective latent dimension. Several latent dimensions can be made out that represent the positions away from any wall with a clear maximum in the centre. The majority of the latent dimensions represent a specific area in the perimeter around the centre. The remaining dimensions are distributed randomly and show no clear position encoding.

7.6 EXECUTION OF LEARNED MODEL ON HARDWARE

For evaluating the learned policy, the trained parameters and graph information are uploaded to the Raspberry Pi located on the quadrocopter. The network is saved as a tensorflow [2] graph file by the python program used for training the networks. The tensorflow 1.3 c library is then used for loading that graph and executing it on the robot hardware. This main program is responsible for reading in all sensors, evaluating the tensorflow graph followed by the transmission of the resulting actions to the flight controller. Timing of the sensor readings and action execution is handled by individual threads and the real-time clock. Synchronisation between threats is handled by a custom communication library, an early version of the framework presented in [1]. An overview over the structure of that control loop is shown in fig. 7.6. Feedback is achieved through the filtering step in the upper right corner of fig. 7.6, where the current state is updated with the current measurement. The policy, the DVBF filter step and the delay for actions are all handled by the tensorflow graph, making it easy to change parameters and structure in the graph generation without recompilation of the main program. The computational power of the Raspberry Pi is more than enough as only a single step of the DVBF model and the policy has to be executed. Also no gradient update



(a) Individual dimensions of policy samples. Top row represents sampled outputs of the policy distribution and bottom row shows control outputs after filtering. The roll and pitch channels show high driving signal as soon as the state of the quadrocopter is not in a centred position.



(b) Roll and pitch dimensions of policy samples combined as quiver plot with (left) sampled outputs of the policy distribution (right) control outputs after filtering. A clear trend towards the center of the cage is visible.

Figure 7.4: Samples from policy trained on data recorded with version 2 of the quadrocopter. Computed on the validation dataset.

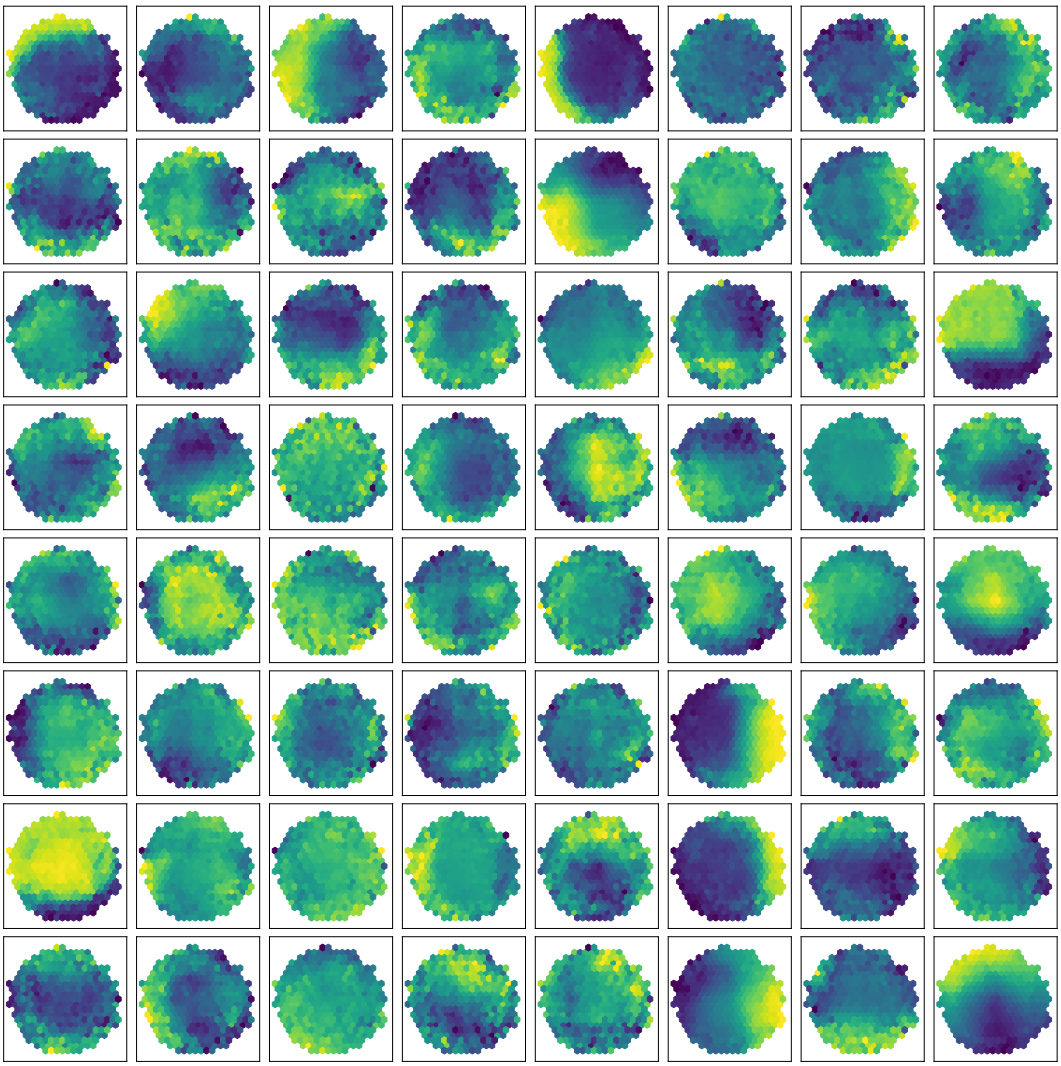


Figure 7.5: Latent state space of the DVBF model trained on quadrocopter data. Each of the 64 subplots represents a single latent state activation depending on the quadrocopter position. The unsupervised algorithm was able to split the actual Cartesian space into regions and assigns them to individual states. The results in this plot are computed on training data.

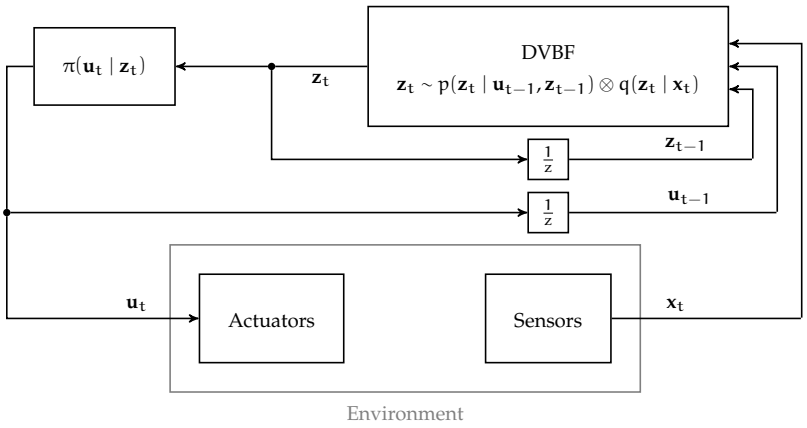


Figure 7.6: Structure of high level quadrocopter controller with DVBF. The envrionment is composed of the systems outside of the main control computer, including the flight controller, the motor driver and the sensor hardware dynamics.

or back-propagation needs to be performed during execution. The empowerment computation is not executed, as all behaviour is already amortised into the policy.

The policies shown in section 7.4 also show valid behaviour when executed on the real hardware. One of these trajectories in sensor space together with frames from a video recording are shown in fig. 7.7. Again the two-dimensional mapping of LIDAR sensor readings from fig. 7.3 is used here to map the trajectory of the flying quadrocopter. Certain positions in this sensor state are mapped to still frames of a video recording of the flight. The trajectory shows how the quadrocopter is moving from the upper left corner of the cage towards the centre of the cage and then gets perturbed and moves towards the lower left wall. The region in which the perturbation happens shows a low policy strength and no clear direction as presented by the dark blue colouring and arrow direction. Only when it approaches closer to the wall, the policy produces higher actions with a clear direction away from the wall and makes the quadrocopter brake very close to the wall in frame 41 and returns to the middle of the cage in frame 51. The fluctuation around the centre position in the cage can be attributed to the high variance of the low range LIDAR sensors of version 1 of the quadrocopter and to the noisy relationship between actions and states due to changing horizons after collisions. The walls and the upper area of the plexiglass cage create reflections and the large distances increase the noise in the sensors reading even more. The sensors pointing towards the floor however gather data from less reflective surfaces and showed a less noisy measurement. This has a huge impact on the quality of the policy in this direction. The resulting trajectory after running the policy on hardware shows a constantly hovering quadrocopter without being told so, except through the intrinsic motivation cost function. The floor is another wall that should be avoided to increase ones future options.

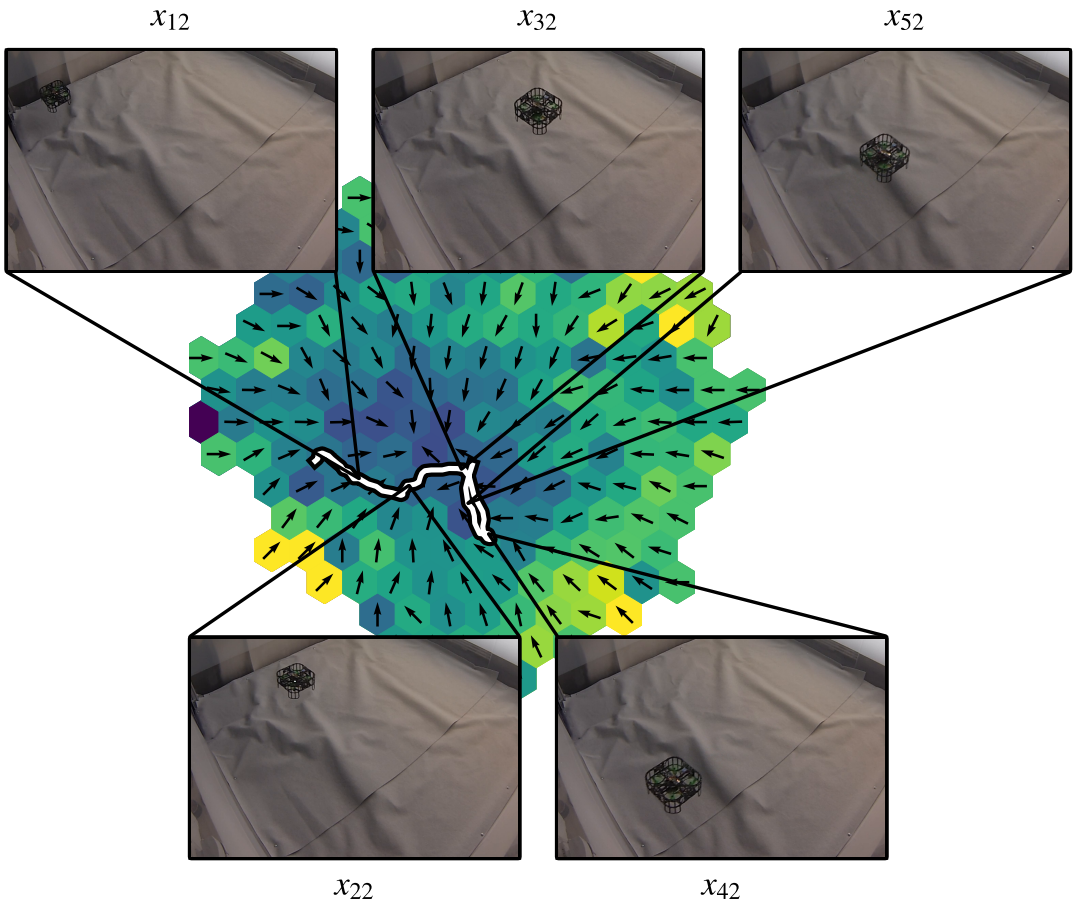


Figure 7.7: Example flight after applying the learned policy to the quadrocopter. The white line represents the trajectory of the quadrocopter in x-y-positions computed from the LIDAR readings. The quadrocopter first flies out of the corner and then arrives at the cage centre in frame 32, then gets disturbed and drift towards the wall in the lower left. Just before it is about to hit the wall after frame 42, the quadrocopter breaks and returns to the centre. The background in this plot was computed on the training dataset while the trajectory represents an unseen trajectory.

8

CONCLUSION AND FUTURE WORK

This thesis presents an efficient method for computing and exploiting an intrinsic motivation called empowerment. A novel method for learning latent state space models from raw data was developed. It is capable of predicting far into the future with a learned transition conditioned on control inputs. With this learned interaction between actions and observations control policies can be learned by using the model as a simulator. A variational lower bound makes the computation of empowerment tractable and was then combined with the policy learning algorithm. This empowerment exploiting method was applied to various simulated environments such as the pendulum, a moving ball, multi-agent systems. A real world experiment showed that the presented method is also capable of learning the model of a quadrocopter and controlling it with a minimum of computational resources.

The results from this thesis can also be used beyond the optimisation of intrinsic control. This includes control tasks with supervised cost functions or preprocessing raw sensor data into a compressed latent state. In future work, the here presented methodology could be unified as proposed in [55] for creating a model where one single unsupervised cost function is used to learn models of the environments and control policies maximising intrinsic motivation at the same time.

Part III

APPENDIX

A

TIME SERIES MODELLING

This chapter contains parts and parameters from [24] and lists the parameters used in chapter 5.

A.1 ANNEALED KL-DIVERGENCE

The KL-divergence in the annealed version eq. (5.10) of the ELBO from eq. (5.6) can be computed analytically for certain distributions. For the annealed KL-divergence between two Gaussian distributions the following equation holds:

$$\mathbb{E}_q[-\ln q(\mathbf{z}) + c_i \ln p(\mathbf{z})] = c_i \frac{1}{2}\ln(2\pi\sigma_p^2) - \frac{1}{2}\ln(2\pi\sigma_q^2) + c_i \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2}$$

A.2 ADDITIONAL EXPERIMENT PLOTS

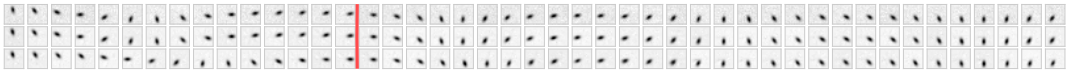


Figure A.1: Ground truth and samples from recognition and generative model. Complete version of fig. 5.6 with all missing samples present.

A.3 IMPLEMENTATION DETAILS FOR DVBF IN PENDULUM EXPERIMENT

- Input: 500 episodes of 15 timesteps with 16^2 observation dimensions and motor babbled 1 action dimension
- Latent Space: 3 dimensions
- Observation Network $p(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t; \mu(\mathbf{z}_t), \sigma)$: 128 ReLU + 16^2 identity output
- Recognition Model: 128 ReLU + 6 identity output

$$q(\mathbf{w}_t|\mathbf{z}_t, \mathbf{x}_{t+1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{w}_t; \mu, \sigma),$$
$$(\mu, \sigma) = f(\mathbf{z}_t, \mathbf{x}_{t+1}, \mathbf{u}_t)$$

- Transition Network $\alpha_t(\mathbf{z}_t)$: 16 softmax output
- Initial Network $\mathbf{w}_1 \sim p(x_{1:T})$: Fast Dropout BiRNN with: 128 ReLU + 3 identity output

- Initial Transition $\mathbf{z}_1(\mathbf{w}_1)$: 128 ReLU + 3 identity output
- Optimizer: adadelta, 0.1 step rate
- Inverse Temperature: $c_0 = 0.01$, updated every 250th gradient update, $T_A = 10^5$ iterations
- Batch-size: 500

A.4 IMPLEMENTATION DETAILS FOR DVBF IN BOUNCING BALL EXPERIMENT

- Input: 500 episodes of 15 timesteps with 16^2 observation dimensions and motor babbled 2 action dimension
- Latent Space: 4 dimensions
- Observation Network $p(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t; \mu(\mathbf{z}_t), \sigma)$: 128 ReLU + 16^2 identity output
- Recognition Model: 128 ReLU + 8 identity output

$$q(\mathbf{w}_t|\mathbf{z}_t, \mathbf{x}_{t+1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{w}_t; \mu, \sigma),$$
$$(\mu, \sigma) = f(\mathbf{z}_t, \mathbf{x}_{t+1}, \mathbf{u}_t)$$

- Transition Network $\boldsymbol{\alpha}_t(\mathbf{z}_t)$: 16 softmax output
- Initial Network $\mathbf{w}_1 \sim p(x_{1:T})$: Fast Dropout BiRNN with: 128 ReLU + 4 identity output
- Initial Transition $\mathbf{z}_1(\mathbf{w}_1)$: 128 ReLU + 4 identity output
- Optimizer: adadelta, 0.1 step rate
- Inverse Temperature: $c_0 = 0.01$, updated every 250th gradient update, $T_A = 10^5$ iterations
- Batch-size: 500

A.5 IMPLEMENTATION DETAILS FOR DKF IN PENDULUM EXPERIMENT

- Input: 500 episodes of 15 timesteps with 16^2 observation dimensions and motor babbled 1 action dimension
- Latent Space: 3 dimensions
- Observation Network $p(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t; \mu(\mathbf{z}_t), \sigma(\mathbf{z}_t))$: 128 Sigmoid + 128 Sigmoid + 2 16^2 identity output
- Recognition Model: Fast Dropout BiRNN 128 Sigmoid + 128 Sigmoid + 3 identity output

- Transition Network $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_{t-1})$: 128 Sigmoid + 128 Sigmoid + 6 output
- Optimizer: adam, 0.001 step rate
- Inverse Temperature: $c_0 = 0.01$, updated every 25th gradient update, $T_A = 2000$ iterations
- Batch-size: 500

B

UNSUPERVISED CONTROL PARAMETERS

This chapter contains parts and parameters from [10] and lists the parameters used in chapter 6.

B.1 GENERAL EXPERIMENT SETTINGS

Before actions are fed into the transition or the environment they got post-processed with the tanh function and then scaled to fit the action range of the environment.

B.2 PENDULUM

DYNAMICS

The pendulum dynamics in OpenAI Gym [11] were reimplemented in tensorflow in order to backpropagate errors through the transition. Friction was added to be comparable to prior work on empowerment for the pendulum [76].

$$
\begin{aligned}
\Delta t &= 0.05 \\
g &= 10 \\
m &= 1 \\
l &= 1 \\
u &\leftarrow \min(\max(u, u_{\min}), u_{\max}) \\
\dot{\theta} &\leftarrow \dot{\theta} + (-\frac{3g}{2l}\sin(\theta+\pi) + \frac{3}{ml^2}(u - 0.05\,\dot{\theta}))\Delta t \\
\theta &\leftarrow \theta + \dot{\theta}\Delta t \\
\dot{\theta} &\leftarrow \min(\max(\dot{\theta}, -8), 8)
\end{aligned}
$$

PARAMETERS

Network parameters: Number of action dimensions $n_u = 1$

- Policy $\pi(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + 128 tanh + 128 tanh + 128 tanh + {n_u identity, n_u exp}
- Source $\omega(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + 128 tanh + 128 tanh + 128 tanh + {n_u identity, n_u exp}
- Planning $\pi(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{z}_{t+1})$: 128 tanh + 128 tanh + 128 tanh + 128 tanh + {n_u identity, n_u exp}
- β changing from 5 to 2000 over 700 epochs
- Horizon for policy optimisation $T = 30$

B.3 SINGLE BALL IN A BOX

DYNAMICS

Single ball in a box simulated with Box2D based on OpenAI Gym environments.
Inelastic collisions, density 5.0, friction 1.1, radius 0.67, box square length 5.96

PARAMETERS

Number of action dimensions $n_u = 2$
Number of latent dimensions $n_z = 32$

- Policy $\pi(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + {$n_u$ identity, n_u exp}
- Source $\omega(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + {$n_u$ identity, n_u exp}
- Planning $\pi(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{z}_{t+1})$: 128 tanh + {$n_u$ identity, n_u exp}
- Recognition model $q_{\text{meas}}(\mathbf{z}_t \mid \mathbf{x}_t)$: 128 relu + {$n_z$ identity, n_z square}
- Transition model $\mu_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$, $\sigma^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$, $\bar{\sigma}^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$: 128 sigmoid + {$n_z$ identity, n_z square, n_z square}
- Generative model $p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{u}_t)$: {128 relu + n_x identity, 1 square}
- $\beta = 50$
- Horizon for policy optimisation $T = 20$

B.4 MULTIPLE BALLS IN A BOX

DYNAMICS

Multiple balls in a box simulated with Box2D based on OpenAI Gym environments.
Inelastic collisions, density 5.0, friction 1.1, radius 0.66, box square length 9.29

PARAMETERS

Number of action dimensions $n_u = 2$
Number of latent dimensions $n_z = 16$

- Policy $\pi(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + {$n_u$ identity, n_u exp}
- Source $\omega(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + {$n_u$ identity, n_u exp}
- Planning $\pi(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{z}_{t+1})$: 128 tanh + {$n_u$ identity, n_u exp}
- Recognition model $q_{\text{meas}}(\mathbf{z}_t \mid \mathbf{x}_t)$: 128 relu + {$n_z$ identity, n_z square}
- Transition model $\mu_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$, $\sigma^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$, $\bar{\sigma}^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$: 128 sigmoid + {$n_z$ identity, n_z square, n_z square}

- Generative model $p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{u}_t)$: {128 relu + n_x identity, 1 square}
- $\beta = 20$
- Horizon for policy optimisation $T = 20$

B.5 BIPEDAL ROBOT

DYNAMICS

BipedalWalker-v2 environment from OpenAI Gym.

PARAMETERS

Number of action dimensions $n_u = 4$
Number of latent dimensions $n_z = 64$

- Policy $\pi(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + {$n_u$ identity, n_u exp}
- Source $\omega(\mathbf{u}_t \mid \mathbf{z}_t)$: 128 tanh + {$n_u$ identity, n_u exp}
- Planning $\pi(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{z}_{t+1})$: 128 tanh + {$n_u$ identity, n_u exp}
- Recognition model $q_{\text{meas}}(\mathbf{z}_t \mid \mathbf{x}_t)$: 128 relu + {$n_z$ identity, n_z square}
- Transition model $\mu_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$, $\sigma^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$, $\bar{\sigma}^2_{\text{trans}}(\mathbf{z}_t, \mathbf{u}_t)$: 128 sigmoid + {$n_z$ identity, n_z square, n_z square}
- Generative model $p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{u}_t)$: {128 relu + n_x identity, 1 square}
- $\beta = 60$
- Horizon for policy optimisation $T = 50$

C

QUADROCOPTER

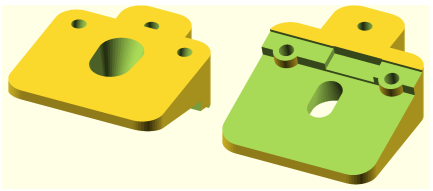
The following chapter show parameters, electronic circuits and 3D models used in chapter 7.

C.1 NETWORK PARAMETERS FOR QUADROCOPTER VERSION 2

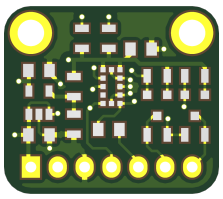
Dataset is cut into windows of 40 time steps in intervals of 20 steps between windows.
Δt between time steps: 144 ms
Number of action dimensions $n_u = 64$
Number of latent dimensions $n_z = 64$
Batch size for model and policy training: 64

- Policy $\pi(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{u}_{t-1}^{channel})$: 256 tanh + {$n_u$ identity, n_u softplus}
- Source $\omega(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{u}_{t-1}^{channel})$: 256 tanh + {$n_u$ identity, n_u softplus}
- Planning $\pi(\mathbf{u}_t \mid \mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{u}_{t-1}^{channel})$: 256 tanh + {$n_u$ identity, n_u softplus}
- Recognition model $q_{meas}(\mathbf{z}_t \mid \mathbf{x}_t)$: 256 relu + {$n_z$ identity, n_z square}
- Transition model: locally linear, $n_\alpha = 32$, $n_{slds} = 32$, slds-network: 128 relu + 128 relu + {n_{slds} identity, slds softplus }, affine transformation from slds sample to alpha,
- Generative model $p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{u}_t)$: {256 relu + n_x identity, 1 exp}
- Empowerment steps: 2
- Path-empowerment steps: 2
- $\beta = 3000$
- Horizon for policy optimisation $T = 20$

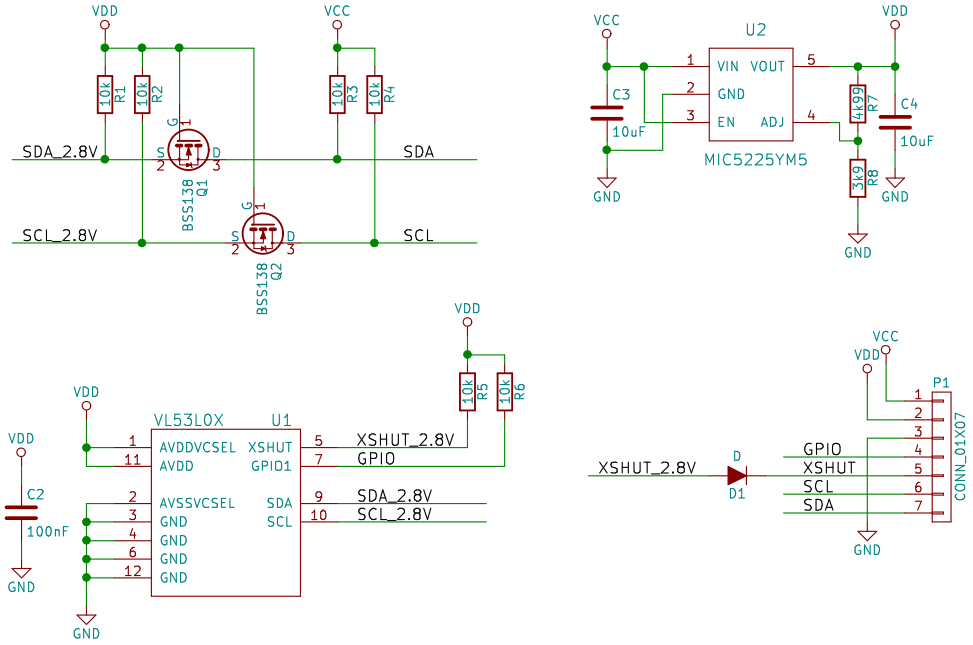
C.2 HARDWARE



(a) Laser distance sensor mount. Keeps the sensor a certain angle relative to the rectangular quadrocopter frame walls and protects the sensor from collisions and incoming light from directions other than the sensing direction.

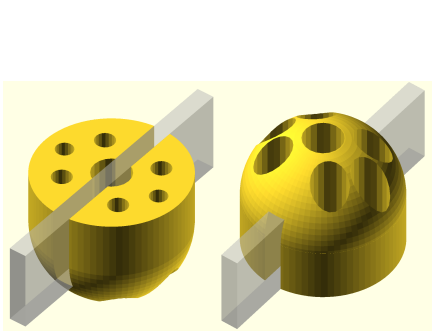


(b) Printed circuit board for VL053XL time of flight sensor chip, voltage regulator and logic shifter. Replication of Adafruit breakoutboard modified for adjustable voltage regulator.

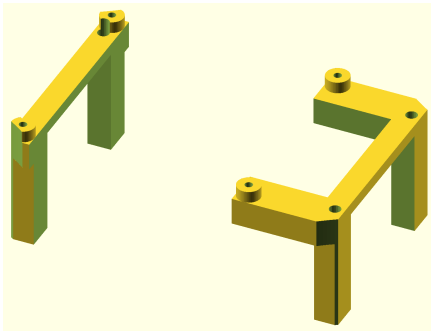


(c) Schematics for VL053XL time of flight sensor chip, voltage regulator and logic shifter. Replication of Adafruit breakoutboard modified for adjustable voltage regulator.

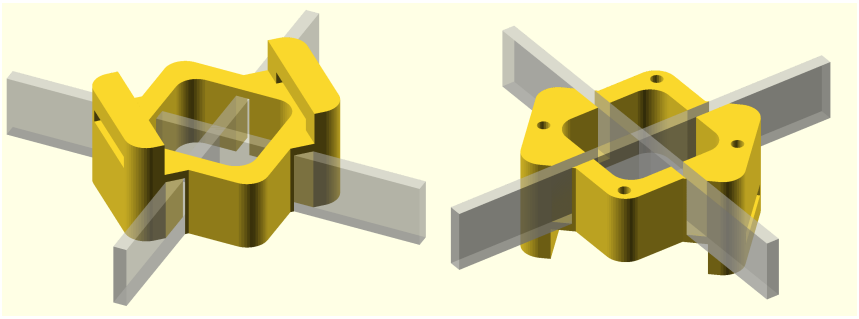
Figure C.1: Laser sensor printed circuit board, schematic and plastic mount.



(a) One of the four 3D printed motor mounting parts. They are glued to the main carbonfiber frame, here in parts shown in transparent grey.



(b) Adapter for mounting a Raspberry Pi in the centre of the quadrocopter. The flightcontroller is seated underneath the Raspberry Pi between the posts.



(c) Part for mounting flight controller and battery. It is placed in the centre of the carbon fiber cross. The flight controller is mounted on the flat side with four screws while the battery is secured with a velcro strap looping through the two rectangular holes.

Figure C.2: Models for 3D printed parts for mounting motors and printed circuit boards inside the quadrocopter frame.

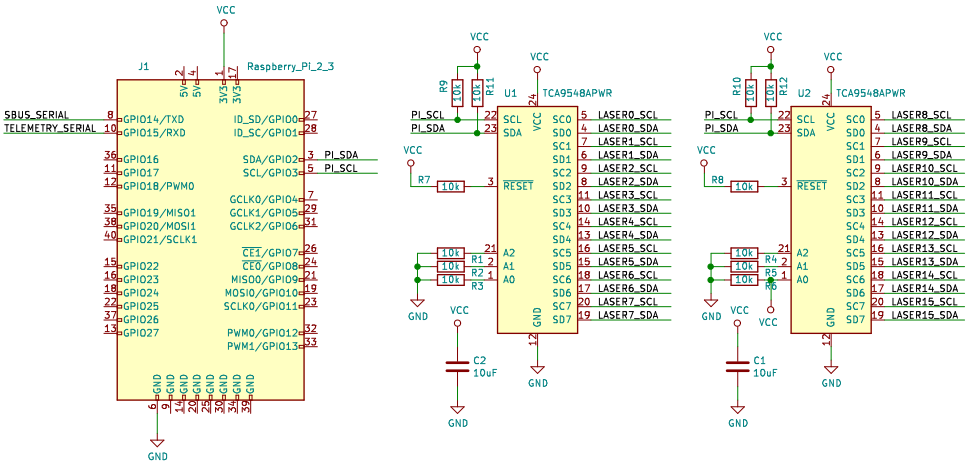


Figure C.3: Multiplexing of laser sensors signals and connection to Raspberry Pi. Two TCA9548A breakout boards from Adafruit are used to multiplex the 16 laser distance sensors.

BIBLIOGRAPHY

[1] Felix Frank, Alexandros Paraschos and Patrick van der Smagt. „ORC - A Lightweight, Lightning-Fast Middleware“. In: *3rd IEEE International Conference on Robotic Computing, IRC 2019, Naples, Italy, February 25-27, 2019*. IEEE, 2019, pp. 337–343. ISBN: 978-1-5386-9245-5. DOI: 10.1109/IRC.2019.00061.

[2] Martín Abadi et al. „TensorFlow: A System for Large-Scale Machine Learning“. In: (27th May 2016). arXiv: 1605.08695 [cs].

[3] Philip Becker-Ehmck, Jan Peters and Patrick van der Smagt. „Switching Linear Dynamics for Variational Bayes Filtering“. In: *https://openreview.net/forum?id=B1MbDj0ctQ* (27th Sept. 2018).

[4] Christoph Salge, Cornelius Glackin and Daniel Polani. „Empowerment – an Introduction“. In: (7th Oct. 2013). arXiv: 1310.1863 [nlin].

[5] David Barber Felix Agakov. „The IM Algorithm: A Variational Approach to Information Maximization“. In: *Advances in Neural Information Processing Systems* 16 (2004), p. 201.

[6] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 11th July 2006. ISBN: 978-0-471-74881-6.

[7] Telatar Emre. „Capacity of Multi-Antenna Gaussian Channels“. In: *European Transactions on Telecommunications* 10.6 (12th Sept. 2008), pp. 585–595. ISSN: 1124-318X. DOI: 10.1002/ett.4460100604.

[8] Raymond W. Yeung. „The Blahut-Arimoto Algorithms“. In: *A First Course in Information Theory*. Information Technology: Transmission, Processing and Storage. Springer, Boston, MA, 2002, pp. 215–231. ISBN: 978-1-4613-4645-6. DOI: 10.1007/978-1-4419-8608-5_10.

[9] Alexander S. Klyubin, D. Polani and C. L. Nehaniv. „Empowerment: A Universal Agent-Centric Measure of Control“. In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 1. Sept. 2005, 128–135 Vol.1. DOI: 10.1109/CEC.2005.1554676.

[10] Maximilian Karl, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt and Justin Bayer. „Unsupervised Real-Time Control through Variational Empowerment“. In: (13th Oct. 2017). arXiv: 1710.05101 [stat].

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang and Wojciech Zaremba. „OpenAI Gym“. In: (5th June 2016). arXiv: 1606.01540 [cs].

[12] Mohammad Gheshlaghi Azar, Vicenç Gómez and Hilbert J. Kappen. „Dynamic Policy Programming“. In: *Journal of Machine Learning Research* 13 (Nov 2012), pp. 3207–3245. ISSN: ISSN 1533-7928.

[13] H. J. Kappen. „Path Integrals and Symmetry Breaking for Optimal Control Theory“. In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.11 (2005), P11011. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2005/11/P11011.

[14] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu and Daan Wierstra. „Weight Uncertainty in Neural Network“. In: *International Conference on Machine Learning*. International Conference on Machine Learning. 1st June 2015, pp. 1613–1622.

[15] Sepp Hochreiter and Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[16] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press, 24th Aug. 2012. 1096 pp. ISBN: 978-0-262-01802-9.

[17] Z. Ghahramani and G. E. Hinton. „Variational Learning for Switching State-Space Models“. In: *Neural Computation* 12.4 (Apr. 2000), pp. 831–864. ISSN: 0899-7667. pmid: 10770834.

[18] Stephan Mandt, James McInerney, Farhan Abrol, Rajesh Ranganath and David Blei. „Variational Tempering“. In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. 2nd May 2016, pp. 704–712.

[19] Justin Bayer and Christian Osendorfer. „Learning Stochastic Recurrent Networks“. In: (27th Nov. 2014). arXiv: 1411.7610 [cs, stat].

[20] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville and Yoshua Bengio. „A Recurrent Latent Variable Model for Sequential Data“. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett. Curran Associates, Inc., 2015, pp. 2980–2988.

[21] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker and Martin Riedmiller. „Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images“. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2746–2754.

[22] Rahul G. Krishnan, Uri Shalit and David Sontag. „Deep Kalman Filters“. In: (16th Nov. 2015). arXiv: 1511.05121 [cs, stat].

[23] Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams and Sandeep R Datta. „Composing Graphical Models with Neural Networks for Structured Representations and Fast Inference“. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett. Curran Associates, Inc., 2016, pp. 2946–2954.

[24] Maximilian Karl, Maximilian Sölch, Justin Bayer and Patrick van der Smagt. „Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data“. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[25] Tijmen Tieleman and Geoffrey Hinton. *Lecture 6.5—RmsProp: Divide the Gradient by a Running Average of Its Recent Magnitude*. 2012.

[26] Matthew D. Zeiler. „ADADELTA: An Adaptive Learning Rate Method“. In: (22nd Dec. 2012). arXiv: 1212.5701 [cs].

[27] A. Schmitz, M. Maggiali, L. Natale, B. Bonino and G. Metta. „A Tactile Sensor for the Fingertips of the Humanoid Robot iCub“. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Oct. 2010, pp. 2212–2217. DOI: 10.1109/IROS.2010.5648838.

[28] J. Fishel. „Design and Use of a Biomimetic Tactile Microvibration Sensor with Human-like Sensitivity and Its Application in Texture Discrimination Using Bayesian Exploration“. Ph.D. dissertation. University of Southern California, 2012.

[29] N. Wettels and G. E. Loeb. „Haptic Feature Extraction from a Biomimetic Tactile Sensor: Force, Contact Location and Curvature“. In: *2011 IEEE International Conference on Robotics and Biomimetics*. 2011 IEEE International Conference on Robotics and Biomimetics. Dec. 2011, pp. 2471–2478. DOI: 10.1109/ROBIO.2011.6181676.

[30] N. Wettels, J. A. Fishel, Z. Su, C. H. Lin and G. E. Loeb. „Multi-Modal Synergistic Tactile Sensing“. In: *Tactile Sensing in Humanoids—Tactile Sensors and beyond Workshop, 9th IEEE-RAS International Conference on Humanoid Robots*. 2009.

[31] V. Ciobanu, A. Petrescu, N. Hendrich and Jianwei Zhang. „Tactile Sensor Value Preprocessing Pipeline“. In: *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*. System Theory, Control and Computing (ICSTCC), 2013 17th International Conference. Oct. 2013, pp. 674–680. DOI: 10.1109/ICSTCC.2013.6689038.

[32] Zhe Su, Jeremy A. Fishel, Tomonori Yamamoto and Gerald E. Loeb. „Use of Tactile Feedback to Control Exploratory Movements to Characterize Object Compliance“. In: *Frontiers in Neurorobotics* 6 (2012), p. 7. ISSN: 1662-5218. DOI: 10.3389/fnbot.2012.00007. pmid: 22855676.

[33] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme and S. Schaal. „Force Estimation and Slip Detection/Classification for Grip Control Using a Biomimetic Tactile Sensor“. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. Nov. 2015, pp. 297–303. DOI: 10.1109/HUMANOIDS.2015.7363558.

[34] H. van Hoof, N. Chen, M. Karl, P. van der Smagt and J. Peters. „Stable Reinforcement Learning with Autoencoders for Tactile and Visual Data“. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Oct. 2016, pp. 3928–3934. DOI: 10.1109/IROS.2016.7759578.

[35] Maximilian Karl, Justin Bayer and Patrick van der Smagt. „Unsupervised Preprocessing for Tactile Data“. In: (23rd June 2016). arXiv: 1606.07312 [cs, stat].

[36] A. Hyvärinen and E. Oja. „Independent Component Analysis: Algorithms and Applications“. In: *Neural Netw.* 13.4-5 (May 2000), pp. 411–430. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(00)00026-5.

[37] Gerhard Neumann. „Variational Inference for Policy Search in Changing Situations“. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. Omnipress, 28th June 2011, pp. 817–824. ISBN: 978-1-4503-0619-5.

[38] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess and Martin Riedmiller. „Maximum a Posteriori Policy Optimisation“. In: *International Conference on Learning Representations* (2018).

[39] Mark Girolami and Ben Calderhead. „Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods“. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (1st Mar. 2011), pp. 123–214. ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2010.00765.x.

[40] Christopher Wolf, Maximilian Karl and Patrick van der Smagt. „Variational Inference with Hamiltonian Monte Carlo“. In: (26th Sept. 2016). arXiv: 1609.08203 [stat].

[41] Tim Salimans, Diederik Kingma and Max Welling. „Markov Chain Monte Carlo and Variational Inference: Bridging the Gap“. In: *International Conference on Machine Learning*. International Conference on Machine Learning. 1st June 2015, pp. 1218–1226.

[42] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever and Max Welling. „Improved Variational Inference with Inverse Autoregressive Flow“. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett. Curran Associates, Inc., 2016, pp. 4743–4751.

[43] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby and Ole Winther. „Ladder Variational Autoencoders“. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett. Curran Associates, Inc., 2016, pp. 3738–3746.

[44] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu and Geoffrey E. Hinton. „Attend, Infer, Repeat: Fast Scene Understanding with Generative Models“. In: (28th Mar. 2016). arXiv: 1603.08575 [cs].

[45] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende and Daan Wierstra. „DRAW: A Recurrent Neural Network For Image Generation“. In: *International Conference on Machine Learning*. International Conference on Machine Learning. 1st June 2015, pp. 1462–1471.

[46] Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli and Josh Tenenbaum. „Deep Convolutional Inverse Graphics Network“. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett. Curran Associates, Inc., 2015, pp. 2539–2547.

[47] Danilo Rezende and Shakir Mohamed. „Variational Inference with Normalizing Flows“. In: *International Conference on Machine Learning*. International Conference on Machine Learning. 1st June 2015, pp. 1530–1538.

[48] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. *Learning Internal Representations by Error Propagation*. ICS-8506. California Univ. San Diego La Jolla Inst. for Cognitive Science, Sept. 1985.

[49] Diederik P. Kingma and Max Welling. „Auto-Encoding Variational Bayes“. In: *Proceedings of the International Conference on Learning Representations* (2014).

[50] Danilo Jimenez Rezende, Shakir Mohamed and Daan Wierstra. „Stochastic Backpropagation and Approximate Inference in Deep Generative Models“. In: *PMLR*. International Conference on Machine Learning. 27th Jan. 2014, pp. 1278–1286.

[51] Simon Duane, A. D. Kennedy, Brian J. Pendleton and Duncan Roweth. „Hybrid Monte Carlo“. In: *Physics Letters B* 195.2 (3rd Sept. 1987), pp. 216–222. ISSN: 0370-2693. DOI: 10.1016/0370-2693(87)91197-X.

[52] Radford M. Neal. „MCMC Using Hamiltonian Dynamics“. In: (8th June 2012). arXiv: 1206.1901 [physics, stat].

[53] W. K. Hastings. „Monte Carlo Sampling Methods Using Markov Chains and Their Applications“. In: *Biometrika* 57.1 (1st Apr. 1970), pp. 97–109. ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97.

[54] Christopher Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 1st Jan. 2006. ISBN: 978-0-387-31073-2.

[55] Martin Biehl, Christian Guckelsberger, Christoph Salge, Simón C. Smith and Daniel Polani. „Expanding the Active Inference Landscape: More Intrinsic Motivations in the Perception-Action Loop“. In: (21st June 2018). arXiv: 1806.08083 [cs].

[56] Konrad Rawlik, Marc Toussaint and Sethu Vijayakumar. „Approximate Inference and Stochastic Optimal Control“. In: (20th Sept. 2010). arXiv: 1009.3958 [cs, stat].

[57] Jeremy L. England. „Statistical Physics of Self-Replication“. In: *The Journal of Chemical Physics* 139.12, 121923 (21st Aug. 2013). ISSN: 0021-9606. DOI: 10.1063/1.4818538.

[58] Stas Tiomkin and Naftali Tishby. „A Unified Bellman Equation for Causal Information and Value in Markov Decision Processes“. In: (5th Mar. 2017). arXiv: 1703.01585 [cs].

[59] Mikhail Prokopenko and Joseph T. Lizier. „Transfer Entropy and Transient Limits of Computation“. In: *Scientific Reports* 4 (23rd June 2014). DOI: 10.1038/srep05394.

[60] Mikhail Prokopenko, Joseph T. Lizier and Don C. Price. „On Thermodynamic Interpretation of Transfer Entropy“. In: *Entropy* 15.2 (1st Feb. 2013), pp. 524–543. DOI: 10.3390/e15020524.

[61] Joseph T. Lizier, Mikhail Prokopenko and Albert Y. Zomaya. „Local Information Transfer as a Spatiotemporal Filter for Complex Systems“. In: *Physical Review E* 77.2, 026110 (15th Feb. 2008). DOI: 10.1103/PhysRevE.77.026110.

[62] Thomas Schreiber. „Measuring Information Transfer“. In: *Physical Review Letters* 85.2 (10th July 2000), pp. 461–464. ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.85.461. pmid: 10991308.

[63] Antoine Bérut, Artak Arakelyan, Artyom Petrosyan, Sergio Ciliberto, Raoul Dillenschneider and Eric Lutz. „Experimental Verification of Landauer's Principle Linking Information and Thermodynamics". In: *Nature* 483.7388 (8th Mar. 2012), pp. 187–189. ISSN: 0028-0836. DOI: 10.1038/nature10872.

[64] R. Landauer. „Irreversibility and Heat Generation in the Computing Process". In: *IBM J. Res. Dev.* 5.3 (July 1961), pp. 183–191. ISSN: 0018-8646. DOI: 10.1147/rd.53.0183.

[65] Leo Szilard. „On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings". In: *Behavioral Science* 9.4 (1964), pp. 301–310. ISSN: 1099-1743. DOI: 10.1002/bs.3830090402.

[66] Gavin E. Crooks. „Entropy Production Fluctuation Theorem and the Nonequilibrium Work Relation for Free Energy Differences". In: *Physical Review E* 60.3 (1st Sept. 1999), pp. 2721–2726. DOI: 10.1103/PhysRevE.60.2721.

[67] Karl Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas Fitzgerald and Giovanni Pezzulo. „Active Inference and Epistemic Value". In: *Cognitive Neuroscience* 6.4 (2nd Oct. 2015), pp. 187–214. ISSN: 1758-8928. DOI: 10.1080/17588928.2015.1020053. pmid: 25689102.

[68] Karl Friston. „The Free-Energy Principle: A Unified Brain Theory?" In: *Nature Reviews Neuroscience* 11.2 (Feb. 2010), pp. 127–138. ISSN: 1471-0048. DOI: 10.1038/nrn2787.

[69] A. D. Wissner-Gross and C. E. Freer. „Causal Entropic Forces". In: *Physical Review Letters* 110.16 (19th Apr. 2013), p. 168702. DOI: 10.1103/PhysRevLett.110.168702.

[70] Erwin Schrödinger. *What Is Life*. Cambridge: University Press, 1948.

[71] William James Sidis. *The Animate and the Inanimate*. The Gorham Press, 1925.

[72] P. Capdepuy. „Informational Principles of Perception-Action Loops and Collective Behaviours". Thesis. University of Hertfordshire, 20th Jan. 2011.

[73] Christoph Salge, Cornelius Glackin and Daniel Polani. „Empowerment and State-Dependent Noise - An Intrinsic Motivation for Avoiding Unpredictable Agents". In: MIT Press, 2nd Sept. 2013, pp. 118–125. ISBN: 978-0-262-31709-2. DOI: 10.7551/978-0-262-31709-2-ch018.

[74] Daniel Polani. „Information: Currency of Life?" In: *HFSP Journal* 3.5 (Oct. 2009), pp. 307–316. ISSN: 1955-2068. DOI: 10.2976/1.3171566. pmid: 20357888.

[75] Alexander S. Klyubin, Daniel Polani and Chrystopher L. Nehaniv. „Keep Your Options Open: An Information-Based Driving Principle for Sensorimotor Systems". In: *PloS One* 3.12 (2008), e4018. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0004018. pmid: 19107219.

[76] Tobias Jung, Daniel Polani and Peter Stone. „Empowerment for Continuous Agent-Environment Systems". In: *Adaptive Behavior* 19.1 (1st Feb. 2011), pp. 16–39. ISSN: 1059-7123. DOI: 10.1177/1059712310392389.

[77] Christoph Salge, Cornelius Glackin and Daniel Polani. „Approximation of Empowerment in the Continuous Domain". In: *Advances in Complex Systems* 16 (02n03 May 2013), p. 1250079. ISSN: 0219-5259, 1793-6802. DOI: 10.1142/S0219525912500798.

[78] Shakir Mohamed and Danilo Jimenez Rezende. „Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning“. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett. Curran Associates, Inc., 2015, pp. 2125–2133.

[79] Alexander S. Klyubin, Daniel Polani and Chrystopher L. Nehaniv. „All Else Being Equal Be Empowered“. In: *Advances in Artificial Life*. Ed. by Mathieu S. Capcarrère, Alex A. Freitas, Peter J. Bentley, Colin G. Johnson and Jon Timmis. Lecture Notes in Computer Science 3630. Springer Berlin Heidelberg, 2005, pp. 744–753. ISBN: 978-3-540-28848-0.

[80] Jürgen Schmidhuber. „Developmental Robotics, Optimal Artificial Curiosity, Creativity, Music, and the Fine Arts.“ In: *Connect. Sci.* 18.2 (2006), pp. 173–187. DOI: 10.1080/09540090600768658.