Learning Control Policies from Constrained Motion

Matthew Howard

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2009

Abstract

Many everyday human skills can be framed in terms of performing some task subject to constraints imposed by the task or the environment. Constraints are usually unobservable and frequently change between contexts.

In this thesis, we explore the problem of learning control policies from data containing variable, dynamic and non-linear constraints on motion. We show that an effective approach for doing this is to learn the unconstrained policy in a way that is consistent with the constraints.

We propose several novel algorithms for extracting these policies from movement data, where observations are recorded under different constraints. Furthermore, we show that, by doing so, we are able to learn representations of movement that generalise over constraints and can predict behaviour under new constraints.

In our experiments, we test the algorithms on systems of varying size and complexity, and show that the novel approaches give significant improvements in performance compared with standard policy learning approaches that are naive to the effect of constraints. Finally, we illustrate the utility of the approaches for learning from human motion capture data and transferring behaviour to several robotic platforms.

Acknowledgements

There are a number of people whom I wish to thank for their support, help and advice when conducting this research and writing this thesis.

First, I would like to thank my supervisor, **Sethu Vijayakumar**, for first inspiring me to delve into the world of machine learning for robotics and for all of the invaluable support, advice and guidance he has given me over the past three years, both academically and in regards to my development as a researcher.

Second, I would particularly like to thank **Stefan Klanke** for the close technical, academic and editorial support in conducting this research, developing the methods and algorithms, and desemminating it in the various publications. Many times Stefan has given up weekends, or worked late nights with me, to help meet paper deadlines. Without Stefan, this, 'the full paper', would certainly not have been possible.

Third, I would like to thank **Michael Gienger** for for his support particularly on the robotics side, and for all his patience with my naive questions during my several visits to HRI-EU. Michael has been a great supervisor and friend from whom I learnt a great deal about practical robotics, including the science and black arts of great prop-building for videos.

Fourth, I would also like to **Christian Goerick** for his work in helping make the collaboration with HRI-EU possible, and making it run as smoothly as it did. More than this, I found his advice invaluable, (if occasionally a little cryptic!) during my visits to HRI-EU.

Finally, I would also like to thank my girlfriend, **Maarit Laukkanen** for her kind support, in particular putting up with me when I'm away at the office all the time for the sake of work and deadlines, and for all of my friends and family for the same. I would also like to thank all the people in SLMC and at HRI-EU that have made working and studying at these places such a pleasurable experience.

Declaration

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

(*Matthew Howard*)

Table of Contents

List of Notation

Below is a list of symbols and abbreviations used throughout this thesis (unless an exception is noted in the text). Entries of the form $a(\cdot)$ denote an argument should be supplied to the function a, for example where there is a direct dependency on some quantity. In addition to the terms defined here, note that we use the convention of bold upper-case letters, $\mathbf{A}$, to denote matrices, bold lower-case letters, $\mathbf{a}$, to denote vectors and normal weighted font, a, to denote scalar terms.

Symbols

$\mathbf{x}$	State space coordinate.
$\mathbf{u}$	Observed action.
$\pi(\cdot)$	Policy mapping from states to actions.
t	Time.
T	Duration in time (e.g., of a trajectory).
$\psi(\cdot)$	Rheonomic or scleronomic constraint function.
$\mathbf{A}(\cdot)$	Pfaffian constraint matrix.
$\mathbf{N}(\cdot)$	Nullspace projection matrix.
$\mathbf{A}^T$	Transpose of $\mathbf{A}$.
$\mathbf{A}^\dagger$	Moore-Penrose pseudoinverse of $\mathbf{A}$, i.e., $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}$.
$\mathbf{I}$	Identity matrix.
$\mathbf{0}$	Vector of zeros.
$\dot{\mathbf{a}}$	Time derivative of $\mathbf{a}$.

$\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$	Position, velocity and acceleration in joint space.
$\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}$	Position, velocity and acceleration in task space.
τ	Torque in joint space.
$\mathbf{J}(\cdot)$	Jacobian matrix.
$\mathbf{M}(\cdot)$	Mass/inertia matrix.
$\mathbf{F_c}(\cdot)$	Coriolis and centrifugal forces.
$\mathbf{F_g}(\cdot)$	Gravitational force.
$\phi(\cdot)$	Scalar potential.
$\nabla_{\mathbf{x}} f$	Gradient of f with respect to **x**.
$\mathbf{\Phi}(\cdot)$	Vector potential.
$\nabla_{\mathbf{x}} \times \mathbf{f}$	Curl of **f** with respect to **x**.
$\breve{\phi}$	Potential, estimated from Euler integration.
$\tilde{f}(\cdot)$	Model estimate of $f(\cdot)$, e.g., $\tilde{\phi}(\mathbf{x})$ is the estimate of ϕ at point **x** taken from the global model of $\phi(\mathbf{x})$.
λ_i	*i*th eigenvalue of a matrix.
Λ	Diagonal matrix containing eigenvalues as the non-zero entries, i.e. $\Lambda = diag(\lambda_1, \cdots, \lambda_n)$.
$\mathbf{v}_i$	*i*th eigenvector of a matrix.
$\mathbf{V}$	Matrix containing eigenvectors as the columns, i.e. $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_n)$.
$\mathbf{H}$	Hessian matrix.
$\bar{\lambda}$	Regularisation parameter.
$\hat{\mathbf{a}}$	Normalised vector **a**.
$\hat{\mathbf{n}}$	Unit normal vector.
$\mathbf{P}$	Projection matrix.

$\mathbf{R}(\cdot)$	Rotation matrix. Argument denotes rotation angle.
$\bar{\mathbf{x}}$	Augmented state vector $\bar{\mathbf{x}} = (\mathbf{x}, 1)^T$.
$E[\cdot]$	Objective or error function. Arguments denote the quantity to be optimised.
$\mathcal{N}(\mu, \sigma)$	Gaussian distribution with mean μ and standard deviation σ.
$U[x_{min}, x_{max}]$	Uniform random distribution of x with $x_{min} \leq x \leq x_{max}$.
$\mathbf{A} \otimes \mathbf{B}$	The Kronecker product of the matrices $\mathbf{A}$ and $\mathbf{B}$.
$vec(\mathbf{A})$	The *vec* operation applied to the matrix $\mathbf{A}$. For example, for $\mathbf{A} \in \mathbb{R}^{2\times 2}$, $vec(\mathbf{A}) = (A_{11}, A_{21}, A_{12}, A_{22})^T$ where A_{ij} denotes the element of $\mathbf{A}$ on the ith row and jth column.

Abbreviations

AL	Apprenticeship Learning.
DMP	Dynamic Movement Primitives.
DPL	Direct Policy Learning.
HMM	Hidden Markov Models.
IRL	Inverse Reinforcement Learning.
MDP	Markov Decision Process.
NN	Nearest Neighbour.
PbD	Programming by Demonstration.
CLIK	Close Loop Inverse Kinematics.
LWPR	Locally Weighted Projection Regression.
nCPE	Normalised Constrained Policy Error.
nMSE	Normalised Mean Squared Error.
nUPE	Normalised Unconstrained Policy Error.

RBF	Radial Basis Function.
RMRC	Resolved Motion Rate Control.
WBM	Whole Body Motion Control.

Chapter 1

Introduction

A wide variety of everyday human skills can be framed in terms of performing some task subject to a set of constraints. Constraints may be imposed either by the environment (Ohta et al., 2004), the task (Calinon and Billard, 2007) or, more commonly, both. For example, when opening a door, the door acts as an environmental constraint that restricts the movement of ones hand along the opening arc of the door. When stirring soup in a saucepan, the sides of the pan prevent the spoon moving beyond their radius. Many tasks require self-imposed task constraints to be fulfilled in order to achieve adequate performance. For example, when pouring water from a bottle to a cup the orientation of the bottle must be constrained so that the stream of water falls within the mouth of the cup. When wiping a window, ones hand should be constrained to maintain contact with the wiping surface (Park and Khatib, 2006) and when climbing a ladder, constraints may be applied to the centre of mass or the tilt of the torso of the climber to prevent over-balancing. When manipulating or grasping solid objects the motion of ones fingers is constrained by the presence of the object (Sapio et al., 2006). In systems designed to be highly competent and adaptive, such as humanoid robots and robotic arms (Fig. 1.1), behaviour may be subject to a wide variety of constraints that are usually non-linear in actuator space and often discontinuous (Sentis and Khatib, 2006, 2005; Gienger et al., 2005; Sapio et al., 2005; Sentis and Khatib, 2004). Consider the task of running or walking on uneven terrain: the cyclic movement of the legs of the runner is constrained by the impact of the feet on the ground in a dynamic, discontinuous and unpredictable way.

A promising approach to providing robots with such skills as running and opening doors is to take examples of motion from existing demonstrators (e.g., from humans) and attempt to learn a control policy that somehow captures the desired behaviour (Argall et al. 2008; Billard et al. 2007; Schaal et al. 2003; see also Ch. 2). Such techniques offer (i) a simple, intuitive interface for programming robots, (ii) effective methods for motion recognition and classification (e.g., Inamura et al. 2004), and; (iii) accelerated optimisation of movements by using demonstration data to seed the solution (e.g., Schaal 1997).

However, while a wide variety of approaches for learning and representing movements have been proposed in recent years (Argall et al., 2008; Billard et al., 2007; Schaal et al., 2003), few have explicitly considered the problem of dealing with constraints on motion in learning. An important component of this is the ability to deal with the apparent variability in movements induced by varying constraints. For example, one wishes to learn a policy that allows one not only to open a specific door of a particular size (e.g. constraining the hand to a curve of a particular radius), but rather to open many doors of varying sizes (radii).

The focus of this thesis is on modelling control policies from movement data containing dynamic and uncertain constraints. The aim is to develop methods that allow the effect of constraints to be dealt with in an appropriate way during learning, with a view to improving existing methods that currently rely on traditional supervised learning techniques. In particular, we consider learning from movements that are subject to variable, dynamic, non-linear and even discontinuous constraints, and look for policies that can *generalise over constraints*.

The strategy we will use for this is to attempt to consolidate movement observations under different constraints in order to model the underlying *unconstrained policy* common to all. Learning the latter enables generalisation since we can apply new constraints to predict behaviour in novel scenarios. This is inspired by recent work in analytical dynamics (Udwadia, 2008) where an effective and intuitive strategy for analytically solving constrained motion problems has been to consider the effect constraints have in modifying the fundamental equations of motion of a system.

In general, we will see that learning (unconstrained) policies from constrained motion data is a formidable task. This is due to several problems, such as (i) *unobservability* of constraints (ii) *non-convexity* of observations under different constraints, and; (iii) *degeneracy* in the set of possible policies that could have produced the observed movement under the constraint (Howard et al., 2009b, 2008a). We will discuss at length how these problems arise when learning in the constrained setting, and develop

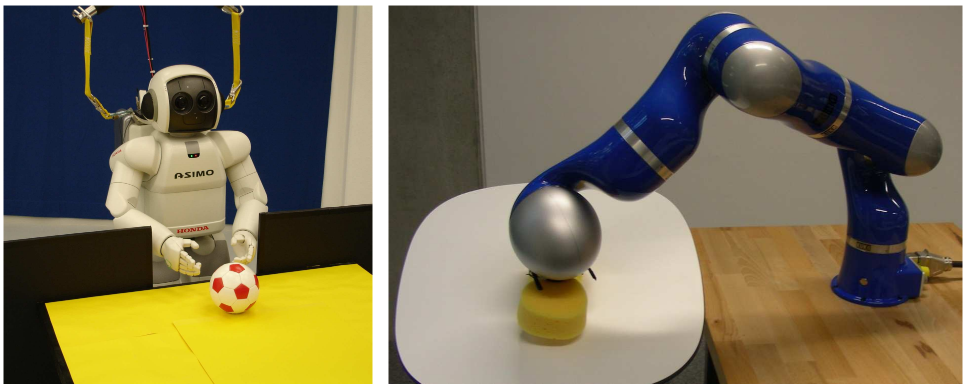


Figure 1.1: ASIMO humanoid robot (left) and anthropomorphic DLR light-weight arm (LWR-III) (right) used in our experiments.

several methods to overcome them, first for the special case of potential-based policies, and later for learning generic, arbitrary policies. We will show that despite these difficulties, given observations (i) under a sufficiently rich set of constraints it is possible to reconstruct the fully unconstrained policy; (ii) under an impoverished set of constraints we can learn a policy that generalises well to constraints of a similar class, and; (iii) under 'pathological' constraints we can learn a policy that, at worst, reproduces behaviour subject to those same constraints. Furthermore, achieving these is possible without the need for explicit knowledge of the constraints in force at the time of observation.

An extensive set of experiments are reported in order to validate the methods and to assess the performance of the various learning techniques developed. In these, learning is performed on data from several policies on complex, high-dimensional movement systems, subject to various realistic constraints. Furthermore, we illustrate the utility of the proposed approach for learning from human demonstrations and transferring behaviour to the ASIMO humanoid robot and the DLR robot arm (Fig. 1.1).

Thesis Outline

In the following, we give a short outline of the thesis highlighting the key content of each chapter. Below each description we also list references to articles in which the work has been published during the course of research, and highlight the original contributions made in the chapter.

In **Chapter 2**, we review the current state of the art in modelling movement for control and imitation and discuss related work specifically focused at dealing with constraints.

Original Contributions:

- *Review of imitation learning methods in terms of policy-based, trajectory-based, and indirect methods.*
- *Common assumption of invariance in constraints in existing constraint-focused works highlighted and analysed.*
- *Comparison of sources of variability in observations under invariant and variable constraints using a navigation task as an example.*

In **Chapter 3**, we discuss how constraints affect the kinematics and dynamics of movement in the light of recent theoretical work in analytical dynamics. We then go on to discuss how different classes of constraint within this model affects learning.

Original Contributions:

- *Constraints in imitation learning framed in terms of established principles of classical mechanics for the first time.*
- *Numerous examples of constrained systems provided, including examples from well-known kinematic and force control schemes.*
- *In-depth analysis of how constraints affect observations of movement from the viewpoint of learning, including degeneracy, non-convexity and problems with 'forced-action' constraints.*
- *Evidence for the feasibility of learning presented in terms of a geometric analysis of the problem.*

Publications:

- *Howard, M., Gienger, M., Goerick, C., and Vijayakumar, S. (2006). Learning utility surfaces for movement selection. In IEEE International Conference on Robotics and Biomimetics.*

- *Howard, M. and Vijayakumar, S. (2007). Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In Workshop on Robotics and Mathematics.*

In **Chapter 4**, we propose a method for learning policies from systems subject to variable constraints for the special case of kinematic, potential-based policies. We show that an effective method for representing movements under different constraints is to learn the unconstrained policy, and that this is possible without explicit knowledge of the constraints.

Original Contributions:

- *Basis for learning potential-based policies from constrained observations derived considering the relationship of observations to directional derivatives of the potential.*

- *Novel learning method developed based on local modelling of the potential and minimisation of global disagreement.*

- *Numerous experiments presented, highlighting enhanced performance and generalisation over constraints compared to standard policy learning techniques.*

Publications:

- *Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2008). Learning potential-based policies from constrained motion. In IEEE International Conference on Humanoid Robots.*

- *Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2008). Behaviour generation in humanoids by learning potential-based policies from constrained motion. Applied Bionics and Biomechanics, 5(4):195211.*

In **Chapter 5**, we propose a new method for learning generic policies from systems subject to variable constraints, removing the restriction to potential-based policies. We show that it is possible to learn arbitrary (e.g., rotational) policies, again without explicit knowledge of the constraints. Furthermore, we apply our approach to learning from human motion capture data.

Original Contributions:

- *Analysis of plausible risk functions and their implications for learning generic constrained policies.*
- *Novel risk function proposed based on optimising consistency with the constraints using an approximated projection.*
- *Numerous experiments presented, showing improved performance for arbitrary (including rotational) policies, and demonstrating application to real human data.*

Publications:

- *Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009). A novel method for learning policies from constrained motion. In IEEE International Conference on Robotics and Automation.*
- *Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009). A novel method for learning policies from variable constraint data. Autonomous Robots. (submitted).*

In **Chapter 6**, we present an extension of the general policy learning method aimed at improving robustness in learning. In particular, we deal with problems that arise when learning with the method proposed in Ch. 5 from data containing invariant or highly correlated constraints.

Original Contributions:

- *Analysis of model degeneracy problem for the novel method presented in Ch. 5.*
- *Novel extension of the constraint-consistent learning approach derived, based on dual optimisation of constraint-consistency and standard risk.*
- *Numerous experiments presented, demonstrating robust learning for constraints with differing levels of variability, highlighting how constraint-consistent learning can be combined with standard policy learning approaches.*

Publications:

- *Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009). Robust constraint-consistent learning. In IEEE International Conference on Intelligent Robots and Systems.*

Finally, in **Chapter 7**, we give conclusions and suggest directions for future work.

Chapter 2

Modelling Movement for Control and Imitation

2.1 Introduction

In this chapter, we review the current state of the art in modelling movement for control and imitation, to provide a background to the research done in this thesis, and to highlight the novel contributions made. To do this, we break the chapter into two parts.

In the first part we look at the general field of learning from movement data and discuss three major classes of methods as categorised by the types of model that they produce. Specifically, we consider methods that learn (i) *policy-based models* (ii) *temporal* or *trajectory-based models*, and (iii) *indirect models* of movement. Our aim in this part is to ground the work contained in this thesis in the context of the wider field of research.

In the second part we focus specifically on works that explicitly deal with constraints when modelling movement. We will see that much of the work in this area has focused on two main aspects of the modelling problem. First, there are studies that focus on *modelling constraints* from example movements. We will discuss several methods proposed for doing this, based on looking for at variance information in the observation data. Second, there are the studies that focus on *adaptation of movements to constraints*. We will review several works that attempt to model adaptation in terms of the optimal control framework, including work in the robotic and human motor control literature.

Finally, we will highlight a common assumption shared by these studies that limits their applicability for a number of problems of interest. Specifically, we will note that they all assume the constraints to be *invariant between observations* (or trials of adaptation). We therefore go on to discuss a third aspect of dealing with constraints in modelling movement that has received little attention in the literature. This is the problem of modelling movement where there are *dynamic, variable constraints* in the observations. We will show that current approaches do not consider this issue despite its frequent appearance in many real-world scenarios.

2.2 Statistical Modelling of Movement

A wide variety of approaches have been proposed for the statistical modelling of movement for purposes of control and imitation (Argall et al., 2008; Billard et al., 2007; Schaal et al., 2003). Broadly speaking most can be categorised into three major classes according to the nature of the models learnt and their domain of applicability. In this section we will give a brief overview of these different classes and the approaches proposed for learning them. We will look at their advantages and disadvantages in terms of their application domain. Our aim is to provide a background against which current work dealing with the role of constraints in movement can be compared.

2.2.1 Policy-based Modelling

A popular class of approaches to modelling movements can be termed that of *policy-based modelling* or *Direct Policy Learning* (DPL) methods[1] (Calinon and Billard, 2007; Alissandrakis et al., 2002; Grimes et al., 2007; Chalodhorn et al., 2006; Grimes et al., 2006; Schaal et al., 2007, 2003; Ijspeert et al., 2003, 2002b,a). In these approaches the idea is to represent demonstrated movements in functional form as some policy, i.e. as a mapping from states to actions

$$\mathbf{u} = \pi(\mathbf{x}) \,, \qquad \pi : \mathbb{R}^n \mapsto \mathbb{R}^d,$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$ are appropriately chosen state- and action-spaces, respectively. Assuming that the observed system can be adequately represented in this way,

[1]To clarify the terminology used, we refer to DPL as the supervised learning of policies from given motion data (e.g., from data recorded from a demonstrator). This is in contrast to the learning of policies from cost/reward feedback without the use of a value function, which is also sometimes referred to as DPL.

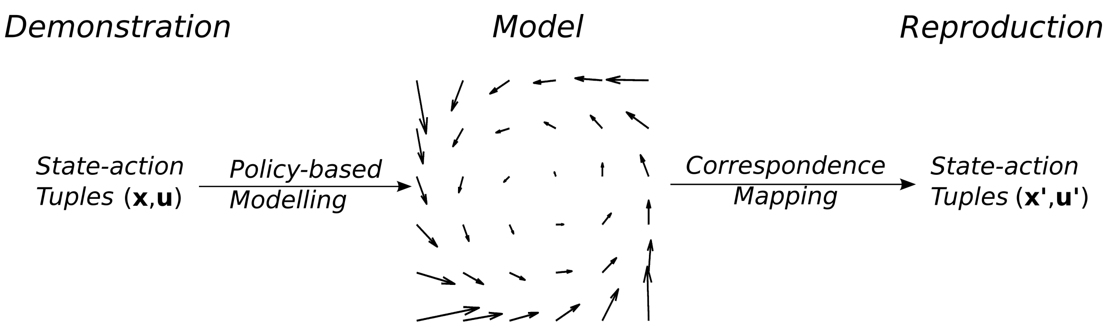


Figure 2.1: Policy based modelling. Demonstration data in the form of state-action tuples is used to form a policy model (vector field) that directly encodes the action for a given state. For the reproduction, a correspondence mapping must be defined from the demonstrator's state-action space $(\mathbf{x},\mathbf{u})$ to that of the imitator $(\mathbf{x}',\mathbf{u}')$.

the goal of DPL is to approximate the policy as closely as possible (Schaal et al., 2003). It is usually formulated as a supervised learning problem where it is assumed that observations of $\mathbf{u}$, $\mathbf{x}$ (often, though not necessarily, in the form of trajectories) are given and from these we wish to learn the mapping π.

A schematic of policy-based learning is shown in Fig. 2.1, illustrating the data flow from demonstration data, through learning, to movement reproduction. As can be seen, this approach requires data in the form of tuples of states and actions. For the reproduction, policy-based models produce atomic state-dependent actions; however, if the policy is applied in closed loop (i.e. state feedback is given after applying action $\mathbf{u}$) then trajectories are produced. It should also be noted that, as with all policy-based approaches, the choice of state- and action-space is problem specific (Schaal et al., 2003) and, when used for imitation learning, depends on the *correspondence* between the state-action space of the demonstrator $(\mathbf{x},\mathbf{u})$ and that of the imitator $(\mathbf{x}',\mathbf{u}')$. For example, if we wish to learn the policy a human demonstrator uses to wash a window, and transfer that behaviour to an imitator robot, an appropriate choice of $\mathbf{x}$ may be the Cartesian coordinates of the hand, which would correspond to the end-effector coordinates of the robot. Transfer of behaviour across non-isomorphic state- and action-spaces (for example, if the demonstrator and imitator have different embodiments) is also possible by defining an appropriate state-action metric (Alissandrakis et al., 2007).

The policy-based approach to learning from observed behaviour has appeared in some form or another in many early works on Programming by Demonstration (PbD) (reviews can be found in Argall et al. 2008; Billard et al. 2007; Schaal et al. 2003). In recent years several authors have taken the policy-based approach using sophisticated, non-parametric supervised learning techniques to model observation data. Recently popular examples of such techniques include *local learning methods* (Peters and Schaal, 2008a; Vijayakumar et al., 2005, 2002; Schaal and Atkeson, 1998; Atkeson and Schaal, 1997; Atkeson et al., 1997) and *Bayesian probabilistic methods* (Calinon and Billard, 2007; Grimes et al., 2007; Wang et al., 2006; Urtasun et al., 2006; Chalodhorn et al., 2006; Grimes et al., 2006; Hsu et al., 2005; Grochow et al., 2004). However, in recent times a particularly appealing approach has been to combine models learnt from data with sets of dynamical systems for use as control policies. In the robotics community, this is commonly known as the *Dynamic Movement Primitives* (DMP) approach (Schaal, 2006; Degallier et al., 2006; Nakanishi et al., 2004; Ijspeert et al., 2003, 2002a,b, 2001).

Example: Dynamic Movement Primitives

Dynamic Movement Primitive learning (Schaal, 2006; Degallier et al., 2006; Nakanishi et al., 2004; Ijspeert et al., 2003, 2002a,b, 2001) is the generic term used for approaches that approximate movement data with models either entirely consisting of a set of dynamical systems (Ijspeert et al., 2001), or a combination of dynamical systems and non-parametric regression models (e.g. Ijspeert et al. 2003, 2002b,a). The great strength of these approaches is that they combine several beneficial properties of dynamical systems with the convenience of a learnt model. In other words, the dynamical systems can be chosen to ensure that the reproduced movement has certain desirable properties, such as guarantees on stability, reachability or controllability (Sontag, 1998). On the other hand, the learning of non-parametric models can help to provide a simple interface to modulate these systems in a data-driven way, without having to engineer the dynamical system from scratch.

An early proponent of this approach was Ijspeert et al. (2001) who proposed a method for fitting mixtures of second-order dynamical systems to human trajectory data. This was tested in a trajectory tracking task on a simulated humanoid robot and resulted in trajectories that were stable against perturbations during task execution (Ijspeert et al., 2001). Since then, several extensions have been proposed, such as learning discrete movements with stable attractor landscapes (Ijspeert et al., 2003, 2002b) and rhythmic movements with periodic systems (Ijspeert et al., 2002a). Nakanishi et al.

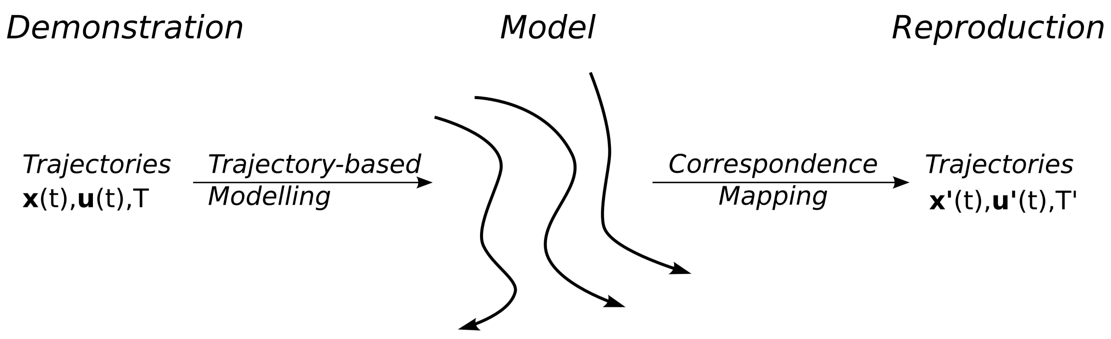


Figure 2.2: Trajectory based modelling. Demonstration data in the form of trajectories is modelled in a way that preserves the order and timing of states and actions. For the reproduction a correspondence mapping is used to produce trajectories in the imitator's state-action space.

(2004) proposed the use of DMPs in the form of neural oscillators for learning bipedal walking. Degallier et al. (2006) proposed a method for switching between discrete and rhythmic dynamical systems for learning a humanoid drumming task. Finally, Park et al. (2008) recently proposed a method to combine DMPs with dynamic potential functions to incorporate simple obstacle avoidance behaviour.

DMPs and other policy-based methods are a relatively simple, effective approach to modelling movement for control and imitation. By framing the problem as a supervised learning of the mapping between states and actions, they can draw on many sophisticated supervised learning techniques. This means they can be learnt very quickly and efficiently, and are suitable for fast, real-time prediction of actions for closed-loop control.

2.2.2 Temporal & Trajectory-based Modelling

A second important class of approaches for modelling movement are those based on temporal and trajectory-based modelling. These use time series modelling techniques in an attempt to capture and exploit time dependencies and the sequential nature of the observed movement data (Dietterich, 2002).

A schematic of trajectory-based modelling is shown in Fig. 2.2, illustrating the data flow. In this class of approaches data arrives in the form of trajectories; that is, strings of states and actions with timing information, such as the duration T. Usually it is assumed that some pre-processing of the data is performed to segment the trajectories (i.e. to determine the start and end of the movement) and to deal with synchronisation (i.e. to ensure time correspondence between different example movements). The trajectories are combined into models capturing the temporal structure of the movement, which can then be used to predict the instantaneous action at some instantaneous state along a trajectory $\mathbf{x}(t)$, or a complete open-loop trajectory through states and actions. It should be noted that for movement reproduction, the correspondence problem must again be solved, similar to the policy-based approaches (ref. Sec. 2.2.1).

Several approaches have been proposed for trajectory-based modelling, such as spline fitting of salient via points (Aleotti and Caselli, 2006; Asfour et al., 2006; Calinon et al., 2005; Ude, 1993) and models based on autoregressive techniques such as recurrent neural networks (Ijspeert and Cabelguen, 2006; Tani and Yamamoto, 2002; Ijspeert, 2001; Morita and Murakami, 1997; Morita, 1996). In the recent literature a particularly popular approach has been the use of Hidden Markov Models (HMMs) (Lee and Nakamura, 2007; Takano et al., 2006; Lee and Nakamura, 2006; Inamura et al., 2005, 2004; Inamura and Nakamura, 2003), to model movement data. In the robotics community the major approach using HMMs is commonly termed the *mimesis model* of imitation learning.

Example: Mimesis Model

The *mimesis model* (Lee and Nakamura, 2007; Takano et al., 2006; Inamura et al., 2005, 2004; Inamura and Nakamura, 2003; Inamura et al., 2002) was one of the first approaches to take advantage of HMMs to unify behaviour modelling, recognition, and generation, as well as handling issues of correspondence all under the same probabilistic framework.

In the framework proposed by Inamura et al. (2004), a database of demonstrated trajectories is maintained and used for learning models of behaviour using discrete and continuous left-right HMMs. The parameters from the continuous models are used to define salient reference points in the phase space of the learner robot, which are then used as states in the discrete HMM models. Due to the greater computational efficiency of discrete HMMs, these are used for the recognition and generation of movements. Finally, to ensure that the learnt movements are viable for the robot to perform (i.e., to deal with correspondence issues) the database of movements is augmented with trajectories generated from the learnt HMMs in the phase space of the robot, causing the reference points to be updated for recognising and generating movement.

Several extensions and applications of the mimesis model have also been reported. For example, Lee and Nakamura (2007) propose a method for mimicking human movements from marker data using on-board monocular vision. Takano et al. (2006) propose a hierarchical version of the mimesis model where two lower level HMMs are used to model primitive behaviours of two agents, and an upper level HMM is used to model sequences of interactions between the lower level behaviours. They apply this to a kick-boxing match where the interaction between two human combatants is reproduced by a robot interacting with a human. Inamura et al. (2005) use verbal commands captured from voice-recognition software to highlight points of attention in demonstrations and facilitate learning.

The mimesis model and other trajectory-based methods are particularly suited to modelling behaviour where order and sequence is important, for instance strings of actions aimed at some outcome. An example would be that of making tea, where the sequence of actions (i.e., boiling the water, adding the teabag, pouring from the teapot etc.) is important. They are also useful for modelling behaviours where the timing of actions is important, such as when to initiate a movement or the duration for which a movement should be performed. There are still some issues to resolve in trajectory-based learning, such as the segmentation and time synchronisation of observations, but methods to cope with these problems are currently active areas of research.

2.2.3 Indirect & Inverse Optimal Modelling

The third major class of movement modelling approaches that we consider here can be broadly termed *indirect or inverse optimal modelling* methods. In these, certain assumptions are made about the *movement generation process*, and based on these a model is learnt that reproduces the movement when operated upon by that same, or a similar, process.

A schematic of indirect modelling is illustrated in Fig. 2.3. Depending on the method, the observation data required for indirect modelling may take either the form of trajectories or tuples of states and actions. In addition to this, information on the generation process is needed, such as parameters determining the generation (e.g., discounting factors γ or the state dynamics function $f(\mathbf{x},\mathbf{u})$) or rules determining how to transform the model to recover the movement (e.g., through optimisation of movement with respect to the model). Similarly, appropriate information on the movement gener-

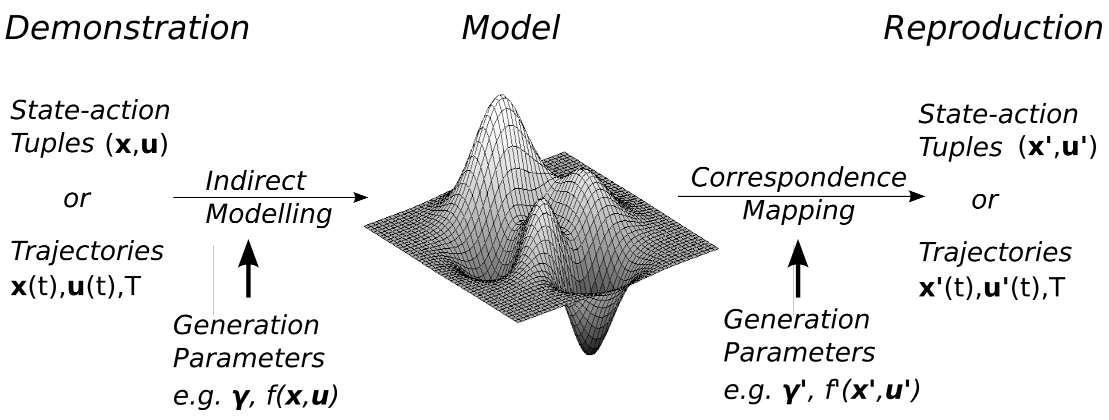

Demonstration
Model
Reproduction
State-action Tuples (**x**,**u**)
or
Trajectories **x**(t),**u**(t),T
Indirect Modelling
Generation Parameters e.g. γ, f(**x**,**u**)
Correspondence Mapping
Generation Parameters e.g. γ', f'(**x'**,**u'**)
State-action Tuples (**x'**,**u'**)
or
Trajectories **x'**(t),**u'**(t),T

Figure 2.3: Indirect modelling. Demonstration data in the form of state-action tuples or trajectories is combined with some assumed movement generation process and parameters. For decoding the movement from the model the same or similar movement generation process is used to find a movement in the imitator's state-action space. An example movement generation process could be reinforcement learning, with given state dynamics for the demonstrator $\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u})$ and imitator $\dot{\mathbf{x}}' = f'(\mathbf{x}',\mathbf{u}')$.

ation process must be defined for the reproduction in the imitator's state-action space. It should be noted that the latter need not exactly match the former; for example, the state-dynamics might change or a different optimisation process may be used.

A simple example of a class of functions that can be used to indirectly model movement is that of scalar potentials (Park et al., 2008; Brillinger, 2007; Khatib et al., 2004; Ohnishi and Imiya, 2007; Conner et al., 2003; Rimon and Koditschek, 1992) which can be used to model a certain class of policies (see Ch. 4). In this case the movement generation process is simply that of taking the gradient of the potential function[2]. However, a family of indirect modelling approaches that has been growing in popularity recently is that of *inverse optimal methods,* such as *inverse reinforcement learning* and *apprenticeship learning*.

Example: Inverse Optimal Modelling

In recent years, approaches based on inverse optimal control such as *Apprenticeship Learning* (AL) (Kolter et al., 2008; Neu and Szepesvári, 2007; Ng, 2006; Ratliff et al., 2006a; Maire and Bulitko, 2005; Abbeel and Ng, 2005, 2004) and *Inverse Reinforce-*

[2]Note that this can be thought of as a special case of inverse optimal modelling under infinite discounting, i.e. using a greedy one-step look-ahead for optimisation.

ment Learning (IRL) (Ziebart et al., 2008; Neu and Szepesvári, 2007; Freire da Silva et al., 2006; Ramachandran and Amir, 2006; Abbeel and Ng, 2004; Ng and Russell, 2000; Russell, 1998; Atkeson and Schaal, 1997) become increasingly popular as methods for indirect modelling of movement. These approaches attempt to model observed movements through their underlying reward/cost function, assuming that they come from some optimally controlled system. In other words, the movement generation process for these models is that of forward optimisation of the control with respect to the model (e.g. through reinforcement learning, Sutton and Barto 1998).

Early proponents[3] of this approach were Ng and Russell (2000) who first derived the theoretical basis for IRL from the Bellman equations and proposed heuristic algorithms for its solution. Their derivation was presented in terms of a discrete state-action space where the reward function was simply a vector of values containing the reward at each state. They showed that, given a (finite-state) Markov Decision Process (MDP), with a knowledge of (or samples from) the optimal policy π^*, discount factor γ and state-transition probabilities $\mathbf{P}_a$, linear programming with heuristics can be used to find the reward vector most consistent with the optimal policy (Ng and Russell, 2000).

Since then the method has been extended in different ways. For example, Abbeel and Ng (2004) describe a method for efficient learning when the reward is composed of a weighted linear combination of known features, and apply the approach to learning different styles of driving in a simple simulated driving game. Ratliff et al. (2006a) present a similar approach but are able to incorporate data from multiple MDPs. They demonstrate their method on a number of navigation tasks based on 2-D satellite images (Ratliff et al., 2007, 2006a,b). Alternative approaches to the basic IRL problem have also been suggested. For example, Ramachandran and Amir (2006) proposed a formulation that defined a distribution over possible reward functions. This enabled them to apply Bayesian inference to find the most likely reward. In contrast, Neu and Szepesvári (2007) use natural gradients to optimise the fit between the observed optimal actions and the reward.

One of the appealing features of indirect modelling approaches such as IRL and AL is the promise of enhanced generalisation beyond that of standard spatio-temporal generalisation (e.g. predicting behaviour in unseen parts of the state space). In approaches

[3]While here we review recent literature related to the statistical learning from demonstration data, it should be noted that in fact the inverse optimal control problem was first posed by Kalman (1964), and some solutions can be found as far back as Casti (1980).

such as these, the models learnt may be decoupled from the dynamics of the observed systems in some way. For example, in IRL, the learnt reward function may be used for optimisation under a new set of state dynamics to produce qualitatively similar behaviour (Neu and Szepesvári, 2007). With this in mind, in Ch. 4 we will pursue an indirect approach for modelling movement based on learning potential functions in order to generalise over different dynamics characterised by different constraints.

2.3 Incorporating Motion Constraints

Despite the wide range of approaches proposed for modelling movement for control and imitation, relatively few studies deal explicitly with the effect of constraints on movement. In the majority of the examples presented above, only unconstrained movements, for example, squatting and kicking (Inamura and Nakamura, 2003) or tracing figures (Ijspeert et al., 2003), are considered. Alternatively, in some cases constraints may exist implicitly in the movement; for example, when beating a drum (Degallier et al., 2006) there is a constraint that prevents the drumstick penetrating the skin. However, usually in these cases the constraints are kept invariant between different demonstrations and also for the reproduction; for example, by keeping the position and orientation of the drum fixed. Note that, provided this invariance in the constraints holds, these approaches are effective. The reason for this is that, in effect, the constraint can be implicitly absorbed into the model itself (for more details on learning from consistently constrained observations, please refer to Sec. 3.3). However, as we will see, these methods face difficulties if this implicit assumption is violated.

Commonly, when constraints are explicitly considered, studies focus on two particular issues of the modelling problem. Broadly speaking, these can be categorised as (i) *inferring constraints* based on variance in the observations, and (ii) *adaptation of movements* in the presence of a constraint. In this section we review works aimed at dealing with these issues and discuss their benefits and shortcomings with respect to modelling movement from constrained data. In particular we will note that, in common with the standard learning approaches described in the preceding sections, these also make the key assumption of *invariant constraints*.

Finally, we will go on to discuss a third, complementary aspect of modelling movement in the presence of constraints that has that has received relatively little attention in the literature. This is the issue of dealing with *variability in the constraints* con-

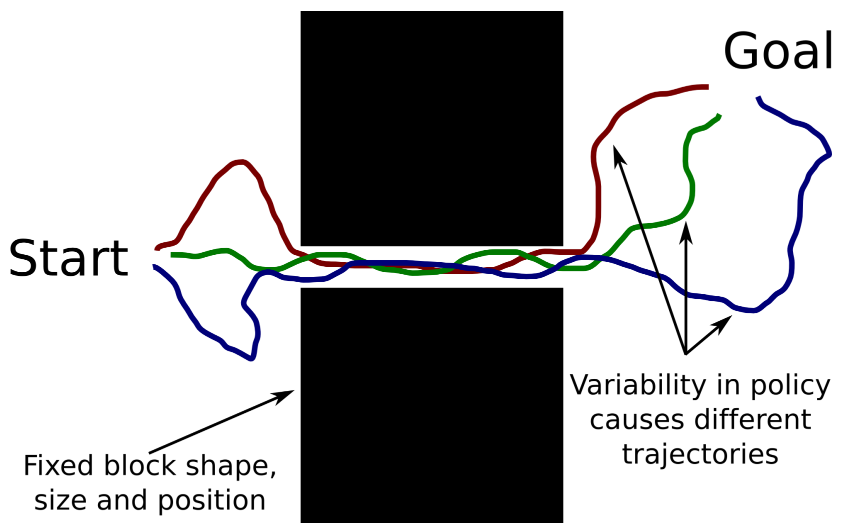

Goal
Start
Variability in policy causes different trajectories
Fixed block shape, size and position

Figure 2.4: Inferring constraints from observed movements using variance information. In the presence of high noise causing perturbations, demonstrated movements (coloured lines) exhibit high variability in unconstrained regions (here, in the regions left and right of the two obstacles (black boxes)). In contrast, in constrained regions (central area between the boxes) the variability is reduced due to the constraints. This variability can be used to infer the presence and location of constraints.

tained in the observations. We discuss how existing approaches do not address this issue despite its appearance in many real-world scenarios.

2.3.1 Inferring Constraints from Variable Observations

In a number of studies, several authors have proposed methods to use demonstrated movement data to infer information about constraints, commonly by looking at the variance in a set of observations (Delson and West, 1993, 1994b,a, 1996; Ogawara et al., 2002; Calinon and Billard, 2007; Guenter et al., 2007; Calinon and Billard, 2008; Hersch et al., 2008) These approaches all share two common assumptions, namely that (i) unconstrained movements contain high variability between trials (e.g. due to noise or environmental perturbations present during demonstrations); and (ii) consistency in observations indicates a constraint on the motion.

To illustrate the concept behind these approaches, consider the navigation problem shown in Fig. 2.4. There, the task is to learn a control policy to get from the start state to the goal state, given a set of expert demonstrations. In this environment, the task is made harder by the presence of two large obstacles (black boxes), that constrain the possible movements that can be taken (i.e., movement is restricted by the obstacles when in the centre of the space, between the two boxes). Now consider that we are given a set of demonstrations of an expert performing the task, where there is a lot of noise or perturbations between the different demonstrations. This noise will affect the trajectories seen in different ways, depending on the constraints. For example in the unconstrained regions (e.g. near the start and goal states in Fig. 2.4) there may be a large variance in the paths taken by the demonstrator as he or she gets knocked off course by the noise. In the constrained regions (e.g. in the central region between the blocks), however, the effect of the noise will be reduced, since the constraints effectively prevent perturbations in the vertical direction, since the obstacles cannot be penetrated. Under the assumption that the constraint is fixed between the different demonstrations (i.e. the shape, position, orientation and size of the boxes is fixed), and that the noise (or the chance of a perturbation) is consistent across the space, we can look for these regions where there is low variance in the demonstrations to find out where constraints are present.

One of the first studies to exploit ideas such as these in the context of PbD, was that of Delson and West (1994b). They describe a method for programming a robot to find the shortest path between a set of obstacles to a target by recording human demonstrations with a force gripper, similar to the set up described in Fig. 2.4. The data was used to identify admissible (i.e. obstacle-free) regions by looking at the range of positions visited by the recorded trajectories. The task was such that the demonstrator would always successfully avoid the obstacles and would pass obstacles on the same side. This meant that the regions visited could be assumed to be safe from collisions and to lie on the path to the goal. Once these admissible regions had been identified, a path planning algorithm was applied to find the shortest path to the target within the safe region. Note that, the constraint that trajectories found by the path planner must lie within this region is an approximation of the true environmental constraints (i.e. the those physically induced by the obstacles). Taking this approach, Delson and West (1994b) were able to program the robot to navigate this environment (i) with the guarantee of obstacle avoidance, and (ii) in a manner more efficient than the original demonstrations (since the robot always took the same shortest path, without the variability of the human).

Since then several authors have also attacked this problem with similar approaches.

For example, Ogawara et al. (2002) proposed an approach to search for 'essential interactions' between objects in a manipulation task. Sequences of candidate interactions were analysed to determine which were common to all in a set of demonstrations, using dynamic programming sequence matching. Calinon and Billard (2008, 2007) considerably extended the approach by applying more sophisticated techniques to encode the statistics of the demonstrations. Instead of using simple bounds on admissible trajectories based on the range of the features (Delson and West, 1994b,a), Gaussian mixture models were used to form a probabilistic model of the demonstrations, and Gaussian mixture regression to reproduce the maximum likelihood movement. They report better generalisation, and smoother trajectories when learning from smaller data sets compared to the original range-based approach. Furthermore, their approach has been applied to several new tasks such as laying a table (Calinon and Billard, 2008) and grasping and moving a chess piece (Calinon and Billard, 2007). Finally, Hersch et al. (2008) suggested an extension to the approach to modulate dynamical systems for robust trajectory tracking, similar to DMPs (ref. Sec. 2.2.1).

These approaches based on looking at inter-trial variance to infer constraints have been successful for several scenarios of interest. Specifically, these involve situations where *constraints are consistent across trials* so that observed variability can be put down to noise and perturbations in the control. However, there are also other sources of variability in constrained movement which this model cannot explain. This includes the effect of *adaptation of the controller* to the constraints. The latter has also received some attention in the constrained movement modelling literature. We turn to this in the next section.

2.3.2 Adaptation to Constraints

A second area in which constraints have been considered explicitly when modelling motion concerns the issue of *adaptation to constraints*. Here, investigation is concerned with ways in which systems can change their behaviour when experiencing a new constraint, commonly with the rationale that the constraint prevents the existing control strategy from performing adequately. Frequently, work in this area makes the assumption that adaptation is performed starting with some seed policy (e.g., learnt under one set of constraints) which is then modified to tackle a new set of constraints (Ohta et al., 2004; Svinin et al., 2005; Guenter et al., 2007). Note that, again, a key

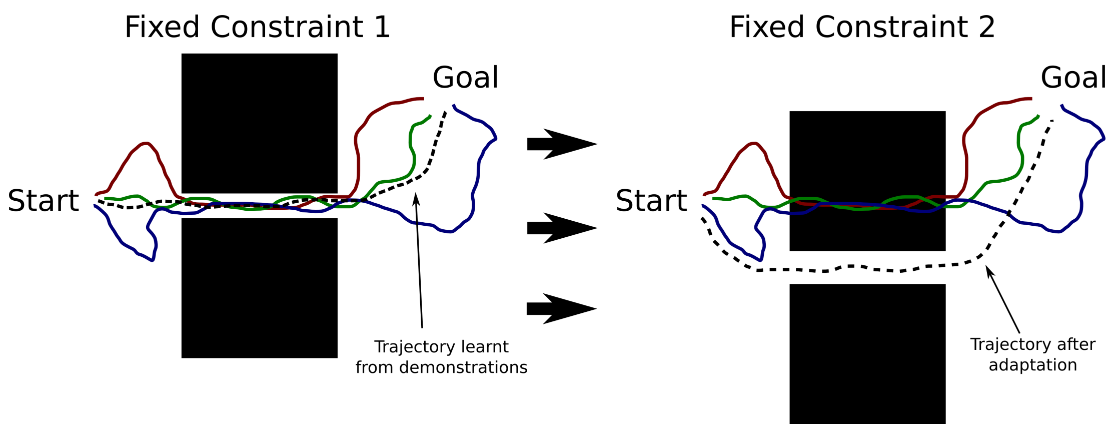

Fixed Constraint 1
Goal
Start
Trajectory learnt from demonstrations
Fixed Constraint 2
Goal
Start
Trajectory after adaptation

Figure 2.5: Adaptation to unseen constraints. In existing approaches, when a new constraint (Constraint 2) is presented to an existing controller (Control 1), that controller is adapted to generate a new controller (Control 2), tailored to the new constraint. Note that the new constraint is usually held constant during adaptation.

assumption of these approaches is that the new constraints are kept *invariant* during the adaptation, so that any variations come from a change in the controller (policy).

To illustrate the concept, let us return to the navigation example described in the preceding section. Again consider that we have to move from some start position to the same goal in the presence of two obstacles. However, this time consider that (i) we start with some initial control policy, for example, learnt in the context of the original set of constraints (e.g. learnt from demonstrations or through reinforcement learning), and; (ii) we are then presented with a new scenario in which we must perform the same task under a new (fixed) constraint, for example with the two obstacles shifted to a new position, as shown in Fig. 2.5. Under the new constraint, adaptation proceeds to change the original policy to improve performance, here, by altering the path of the nominal trajectory.

Most existing studies on adaptation looking at the role of constraints use this model to explain changes in behaviour; namely that an existing policy (possibly adapted to one set of constraints), is adapted (usually through a process of optimisation) to a new fixed constraint. Examples of this can be found in the human and robotics literature.

For example, in the human motor control literature Ohta et al. (2004) and Svinin et al. (2005) recently performed experimental studies of human adaptation to externally applied (i.e. environmental) movement constraints. The goal of these studies was to look at ways in which humans interact with physical objects in the environment that constrain movement; for example, when turning the handle of a coffee grinder (Svinin et al., 2005). They approached the problem in terms of the optimal control framework (Bertsekas, 2007) by analysing how humans adapted their movements under an unfamiliar constraint, and comparing this with optimisation of several established cost functions from the human motor control literature. In their experiments, subjects were asked to make point-to-point movements along a closed-path (i.e., a loop, starting and ending at the same point), subject to constraints on the shape of the path. The constraints were enforced by use of a modified manipulandum to which several metal sheets could be attached. Each of the sheets had different closed-path groves cut into them (e.g. circles, ellipses, clover-leaves) and, when attached, meant that the handle of the manipulandum could only travel along the groove. Essentially this reduced the mobility of the subjects' hands to one degree of freedom, corresponding to the angular position along the path. Angular velocity and normal and tangential (to the path) force profiles were collected and compared to simulated minimum jerk and torque/force-change (in hand and joint space), and minimum muscle effort control. They found that after several trials of training under the new constraint subjects adapted their movement in a way that was consistent with optimisation of a combined hand-force- and joint-torque-change cost function.

A similar model has also been proposed in the robotics domain. For example, Guenter et al. (2007) report a method for handling previously unseen constraints when learning from demonstration data. The aim of this work was to find a way to adapt policies to new constraints that are not contained in the demonstrations, while avoiding having to learn a new policy from scratch. Following earlier work by Calinon and Billard (2007), policies in the form of Gaussian mixture models were initially learnt from a set of teacher demonstrations. These were then adapted to new constraint conditions through a reinforcement learning approach. For this, parameters of the policy model were directly optimised using episodic natural actor-critic reinforcement learning (Peters and Schaal, 2008b). It should be noted that the parameters of the constraints (e.g. obstacle positions and shapes) were not explicitly given to the algorithm, but were held constant between trials during optimisation. The approach was tested for two tasks. In the first the task was to place an object in a box when a (previously unseen) obstacle was placed between the robot's hand and the box so that the robot had to adapt the movement and reach over the obstacle. In the second a policy for grasping a chess piece was learnt, then adapted by changing the constraint on the direction of approach of the hand to the piece.

Studies such as these confirm our intuition that under certain circumstances adaptation is required to cope with new constraints. This can be due to the existing control strategies becoming ineffective under the new constraint, or simply a drive to improve performance when some specific constraint is experienced a number of times and thus becomes familiar. Similar to the methods described in Sec. 2.3.1, these adaptive studies also assume that *constraints are invariant* between trials and all changes to the movement are due to adaptation of the control policy.

2.3.3 Variability in Constraints

In the preceding sections, we outlined a number of studies that explicitly consider constraints when modelling movement. We noted that a key assumption made in these studies is that there is invariance in constraints. In particular, they assume consistency in the constraints between observations and try to explain differences between observations in terms of *variability in the control*; either due to noise and perturbations (Sec. 2.3.1) or adaptation of the controller to the constraint (Sec. 2.3.2). However, while these are important aspects of dealing with variability in observations of constrained movement, there is a third aspect to the problem that has not been considered in the studies presented so far. This is the issue of *variability in the constraints*.

Variability in constraints can occur in many everyday behaviours. For example, consider the problem of opening doors in an everyday environment such as an office. This is a very simple behaviour in which a successful strategy would involve grasping the door handle and pulling it open. Here, we can identify an environmental constraint imposed by the door; namely, that the hand is forced to travel along the opening arc of the door. Note also that this constraint is specific to each particular door, that is different doors may have different widths or open in different directions (e.g. depending on which side of the door the hinges are attached) and this affects the *observed outcome* of the control (i.e. the shape and size of the opening arc). However, note also that despite the different constraints, the same control strategy (i.e. ‘pulling’) would be effective to open many different doors without the need for a specialised controller for each.

As another example, consider also the task of stirring soup in a saucepan. There, we can also identify an environmental constraint; in this case the sides of the pan constrain the movement (i.e., the spoon cannot leave the radius of the pan). Again, this

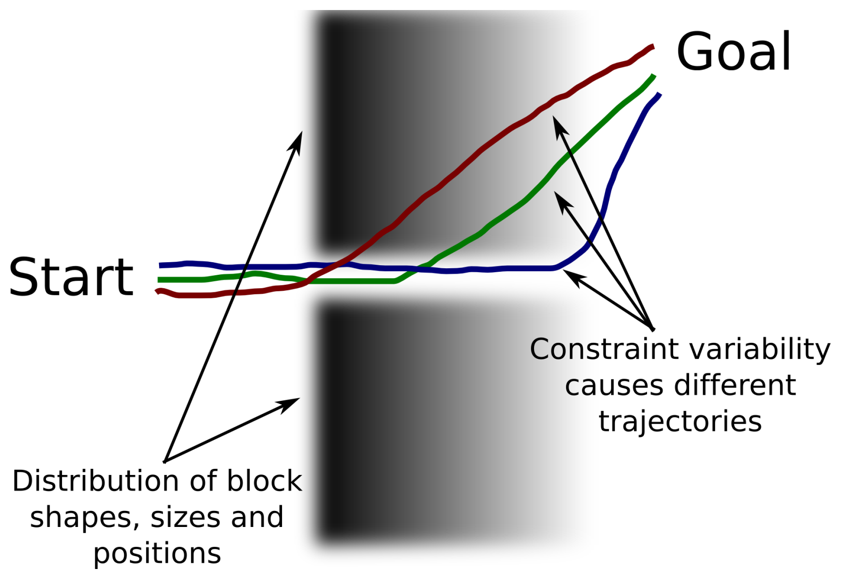

Goal
Start
Constraint variability causes different trajectories
Distribution of block shapes, sizes and positions

Figure 2.6: Inferring the control from variable constraints. Assuming consistency in the control and high variability in the constraints it is predicted that consistency in the observations can be used to model the control. This is the complement of the problem of inferring the constraints (cf. Fig. 2.4).

constraint is dependent on the particular saucepan (e.g. different saucepans may have different sizes and shapes). However, note also that despite these differences, the same periodic control strategy is effective for stirring in many different saucepans.

The point of these examples is to highlight a situation that is largely neglected in studies such as those described in Sec. 2.3.1 & Sec. 2.3.2. There, the focus was on *variability in the control* under a set of *consistent constraints*. However, in these examples, a more appropriate approach seems to be the complement of this; namely, to consider *consistency in the control under variable constraints*. In many cases such as these, approaching the problem in this way is more intuitive, and can offer a simpler explanation of variations in movement that may be observed.

To compare how the concept of this approach differs from that of the approaches described in the preceding two sections, we can again look at our navigation example, but this time from the viewpoint of constraint variability (ref. Fig. 2.6). Consider, again, that we wish to learn a control policy that takes us from some start state to the goal from a set of demonstrations. However, this time, consider that the constraints are different in each of our demonstrations; for example, the width of the obstacles varies for different demonstrations. This is illustrated in Fig. 2.6, where we represent the distribution of different obstacles (i.e. constraints) as shaded regions (so, for example, darker regions indicate a higher likelihood of the obstacle occupying that space). Note that, since the same basic task is performed in each demonstration, we can assume a similar policy would be used. However, note also that under this set up, even assuming perfect consistency in the policy, one would see variability in the movement *due to variability in the constraints*. For example, in Fig. 2.6, we plot three example trajectories that would result from applying a simple point attractor policy (with the attractor point at the goal) for three different obstacle widths (constraints). In this case, differences in the paths can be attributed purely to variability in the constraints.

One of the main novel contributions of the research contained in this thesis is to study the problem of modelling movement in this scenario; in particular, how to make consistent models of movement when our observation data contains high variability in the constraints. This differs significantly from the approaches described in the preceding two sections, both in terms of the assumptions about the constraints, and the goals of learning (for example, we no longer focus on finding the constraints, but look instead for consistency in the controls). There exist a number of motivations for approaching the problem in this way.

For example, one major benefit of handling such cases is that, using such a model, we can hope to predict movement in such a way that *generalises over constraints*. That is, if we can model the controller that is consistent with a set of demonstrations observed under a given set of constraints, we can hope to re-apply that policy to accurately predict movement *under a new set of constraints*. Intuitively, if we observe that our demonstrator uses the same policy successfully under a wide variety of constraints, then presumably the demonstrator finds that policy sufficiently *robust* against the effects of those different constraints for successful performance. Following this, we can then reasonably assume that, by learning this policy and applying it under similar constraint scenarios, this robustness will also be transferred to the learner. For instance in the door-opening example, by learning the control strategy used by a demonstrator to open several familiar doors (i.e. under a set of ‘training constraints’), we may learn a generalised ‘pulling’ control strategy. Assuming this to be successful for the constraints (doors) seen in the training data, we could then apply the controller to new settings with unfamiliar constraints, for example, to open a new door (e.g. one that opens upwards instead of to the side).

Related to this, another possible motivation is to use our model to get a new per-

Policy	Constraint	Modelling	Adaptation
Fixed	Fixed	✓	–
Fixed	Variable	✗	–
Variable	Fixed	✓	✓
Variable	Variable	✗	✗

Table 2.1: Array of possible scenarios in which constraints may affect actions observed. Ticks and crosses indicate whether the given combination can be handled with existing approaches, from the viewpoint of (i) statistical modelling of behaviour data, and (ii) adaptation of behaviour.

spective on the classification of different observations in the movement recognition setting. That is, if we are given a new set of observations (possibly under different constraints), we can check these against our current model and assess their similarity. For example, we could look at the distance in parameter space for the generalised door-opening controller and a similar controller learnt for another task, such as drawer-pulling. This would enable us to group qualitatively similar tasks in the same category, ignoring the specific differences due to the constraints. In other words models learnt by this approach offer alternative, meaningful, abstractions of the observed movement, i.e. we can say two movements are qualitatively or quantitatively similar, *up to a difference in constraints*.

Finally, this alternative approach to the problem of modelling constrained movement can also be used to complement, and improve understanding of the existing studies on control-variability and adaptation under constraints. For example, we might look for ways to detect the source of observed variability (i.e. due to the control, the constraints or both) and improve our models of behaviour. This could be used, for example, to determine when adaptation occurs or is necessary in different constrained scenarios. A further possibility in this direction would be to decompose the variability in observations into its constituent parts, such as adaptation, noise, changes to the constraint, and other perturbations, further improving the quality of our models of behaviour.

2.3.4 Filling the Gap

Based on the analysis in the preceding three sections, we are now in a position to chart how different aspects of modelling movement in the presence of constraints have been studied or solved in the existing literature. In the rows of Table. 2.1 we consider four cases in which constraints could affect the actions observed from some policy, categorised in terms the different possible sources of variability. In the columns we indicate whether the case has been (or could trivially be) handled with existing methods from the viewpoint of (i) statistical modelling of behaviour data, and (ii) adaptation of behaviour. Note that here, the variability of a quantity may be due to noise, external perturbations or adaptation, and is assumed to occur across data sets when modelling, or across learning trials during adaptation. Note also that we only consider adaptation in terms of changes to the policy (hence under the assumption of a fixed policy it does not make sense to deal with the adaptation problem).

As can be seen from the table, the simplest case, namely a learning a fixed policy under fixed constraints, is straightforward using standard approaches such as those described in Sec. 2.2 (this will also be discussed in greater detail in Sec. 3.3.1 & 3.3.2). For the case of a fixed constraint, but variability in the policy (e.g. due to noise in the actions), methods such as those described in Sec. 2.3.1 are well-suited for modelling the movement from a given set of observations, and can even be used to infer information about the constraints. Furthermore, under a fixed constraint, it has been shown that one can optimise an existing policy to improve performance under that constraint, for example using the methods described in Sec. 2.3.2.

However, looking at the cases where variability in the constraints occurs (rows 2 and 4 of Table 2.1) there is a clear gap in current research. Clearly, methods such as those described in Sec. 2.3.1 and Sec. 2.3.2 that rely on the assumption of a fixed constraint are not suitable for such cases. Furthermore, as discussed in detail in the next chapter, most standard regression approaches also fail in the presence of variable constraints (essentially due to a mismatch between the effect of constraints and the noise model commonly assumed in such approaches; see Sec. 3.3.2). In this thesis, our goal is to take the first steps toward filling this gap by proposing methods specifically tailored to such conditions.

Specifically, our goal is tackle the first of the gaps highlighted in Table 2.1; namely the problem of modelling data where, under the assumption of fairly small variation in the control, the main source of variability in the observations can be attributed to variability in the constraints. As discussed in Sec. 2.3.3, there are many real-world scenarios where this is the case, allowing our methods to be directly applied to learning in such scenarios. Furthermore, we hope that by studying this problem we may highlight principles that may then be used in filling in the other gaps, such as adaptation in the presence of variable constraints, and modelling in the presence of the combined variation of policies and constraints. By doing this we hope to extend our knowledge about modelling movement within the domain of dealing with constraints.

2.4 Conclusion

In this chapter, we have reviewed a variety of methods for modelling movement data from observations. We discussed several paradigms for learning models of movement, in particular we highlighted methods that learn (i) policy-based, (ii) trajectory-based and (iii) indirect models. In doing this we aimed to provide a snap-shot of the current state of the art in the movement modelling literature, against which the work in this thesis can be compared.

In the context of this broader field of work, we then went on to focus specifically on studies that explicitly address the role of constraints in motion. We noted that within this more restricted field, work has primarily been directed at two goals (i) inferring the constraints in force during demonstrations, and (ii) adaptation to (fixed) constraints. We noted that these two approaches rely on a common assumption: That the constraints across demonstrations (or adaptation trials) are invariant, meaning that any variation in movements is attributed purely to variation in the control.

We then went on to discuss a third source of variation in the observations, that of *variation in the constraints* and discussed examples of where this is a more intuitive and fitting description of several real-world examples. We also discussed the motivation for studying the effect of variable constraints, both in terms of possible applications in, for example, imitation learning and movement recognition. Furthermore, we discussed how studying this problem fills a large hole in existing research on methods to model movement, and how filling this gap may enhance these existing models.

In the next chapter, we take a detailed look at the effect constraints have on movement observations. In particular, we outline a formal model for constraints based on recent work in analytical dynamics (Udwadia and Kalaba, 1996). We will then look at how different classes of constraint affect our observations in different ways and discuss the implications these have for learning.

Chapter 3

Effect of Constraints on Dynamics and Learning

3.1 Introduction

In this chapter we take a closer look at the the effect of constraints on the dynamics of systems and the implications this has for learning. The analysis is broken into two parts.

First, we will introduce a formal model that can be used to describe the dynamics of constrained motion. This is based on principles of analytical dynamics (Udwadia and Kalaba, 1996) and can be used to describe both natural and controlled dynamics in terms of a generic set of constraints. In the robotics literature this formalism is extensively used in the kinematic and force control of rigid body systems such as manipulators and humanoid robots (Udwadia, 2008; Peters et al., 2008; Sapio et al., 2006; Gienger et al., 2005; Sentis and Khatib, 2004; Buss, 2004; Bruyninckx and Khatib, 2000; Nakamura, 1991; Khatib, 1987; Liégeois, 1977). However, the formalism is generic and can be applied to a wide class of systems (Udwadia and Kalaba, 1996). We will discuss how such constraints affect motion and provide illustrative examples of how the formalism is used in control of redundant manipulators. In general, we will see that an intuitive way to understand the constrained behaviour of a system is is to consider how the unconstrained behaviour is modified by the constraints.

Second, we will discuss how constraints affect the observations and the problems these induce when attempting to learn a policy to capture that behaviour (Howard et al., 2008a, 2009a, 2008b; Howard and Vijayakumar, 2007; Howard et al., 2006). We will see that different problems arise depending on the constraint setting. In particular, we will identify three such settings and discuss examples of where these might be observed in everyday life.

Finally, we will discuss our strategy for overcoming these problems in order to learn the best possible model from the given data. These principles will be used in the learning approaches developed in later chapters.

3.2 Constraint Model

The constraint model assumed throughout this thesis is based on principles of analytical dynamics. Here, we will briefly outline the formalism and its relation to everyday control tasks. For a more thorough treatment of these principles we refer the reader to any standard text on analytical dynamics, such as Udwadia and Kalaba (1996).

Following the policy-based approach (ref. Ch. 2), in this thesis we consider policies that can be described as autonomous systems of the form

$$\mathbf{u}(t) = \pi(\mathbf{x}(t)) \ , \qquad \pi : \mathbb{R}^n \mapsto \mathbb{R}^d, \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$ are appropriately chosen state- and action-spaces, respectively, and π is some stationary mapping between the two. For example, in kinematic control, the state vector could be the joint angles, $\mathbf{x} \equiv \mathbf{q}$, and the action could be the velocities $\mathbf{u} \equiv \dot{\mathbf{q}}$. In forced-based control a suitable state might be $\mathbf{x} \equiv \mathbf{q}, \dot{\mathbf{q}}$, with actions corresponding to applied torques $\mathbf{u} \equiv \tau$.

We assume the policies to be subject to a set of hard, k-dimensional, Pfaffian constraints (where $k \leq d$)

$$\mathbf{A}(\mathbf{x},t)\pi(\mathbf{x}) = \mathbf{b}(\mathbf{x},t) \tag{3.2}$$

where $\mathbf{A}(\mathbf{x},t) \in \mathbb{R}^{k \times d}$ is some rank-k matrix and $\mathbf{b}(\mathbf{x},t) \in \mathbb{R}^k$ is some vector. Together these two terms describe the constraints on the policy. This constraint equation can be thought of as the result of a set of k constraints of the form

$$\psi_i(\mathbf{x},t) = 0; \quad i = 1, 2, \cdots, k \tag{3.3}$$

where $\psi_i(\mathbf{x},t)$ are smooth functions which, when differentiated, give the constraint relation (3.2) (Udwadia and Kalaba, 1996). The effect of the constraints is to modify the policy actions so that they are projected into the nullspace of the constraints. This means that the actions we observe under the constraints have the form

$$\mathbf{u}(\mathbf{x},t) = \mathbf{A}(\mathbf{x},t)^{\dagger}\mathbf{b}(\mathbf{x},t) + \mathbf{N}(\mathbf{x},t)\pi(\mathbf{x}) \tag{3.4}$$

where $\mathbf{N}(\mathbf{x},t) \equiv (\mathbf{I} - \mathbf{A}^{\dagger}\mathbf{A}) \in \mathbb{R}^{d\times d}$ is a projection matrix that, in general, has a non-linear dependence on time and state, and $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix[1].

The formalism outlined is generic and can be used to handle constraints in a wide variety of systems (Udwadia, 2008). For example, constraints of this form (3.2) commonly appear in scenarios where manipulators interact with solid objects, for example, when grasping a tool or turning a crank or a pedal; also referred to as contact constraint scenarios (Park and Khatib, 2006; Murray et al., 1994; Mattikalli and Khosla, 1992). Such constraints are also common in the control of redundant degrees of freedom in high-dimensional manipulators (Liégeois, 1977; Khatib, 1987; Peters et al., 2008), where policies such as (3.4) are used, for example, to aid joint stabilisation (Peters et al., 2008), or to avoid joint limits (Chaumette and Marchand, 2001), kinematic singularities (Yoshikawa, 1985) or obstacles (Choi and Kim, 2000; Khatib, 1985) under task constraints. As an example: Setting $\mathbf{A}$ to the Jacobian that maps from joint-space to end-effector position increments, and setting $\mathbf{b} = \mathbf{0}$, would allow any motion in the joint space provided that the end-effector remained stationary.

The formalism can also readily be applied to learning policies based on dynamic quantities such as torques or (angular and linear) momentum subject to constraints (e.g., see Peters et al. 2008 and Kajita et al. 2003, respectively). In such cases, it is assumed that the systems are subject a set of ‘ideal constraints’ in the sense that they satisfy d’Alembert’s principle. D’Alembert’s principle is a classical result in analytical mechanics that characterises ideal constraints as those that can be described by a set of *forces of constraint* which, for movements that satisfy the constraints, do no work. The constraint formalism (3.1)-(3.4) used here is consistent with this principle for constraints on system dynamics involving forces (Udwadia and Kalaba, 1996).

Finally, it should be noted that such constraints are also not limited to manipulator kinematics and dynamics; for example, Antonelli et al. (2005) apply it to team coordination in mobile robots, and Itiki et al. (1996) use the formalism to model the dynamics of jumping.

In general, the effect of the constraints (3.2)-(3.4) is to disallow policy actions in some sub-space of the system (specifically, the space orthogonal to the image of

[1]Note that a list of symbols is provided at the start of this thesis as a quick reference of the notation used.

$\mathbf{N}(\mathbf{x},t))$. Additionally, for constraints with non-zero $\mathbf{b}(\mathbf{x},t)$, certain actions may be 'enforced' by the constraint that are not derived from the policy. For example, if the policy controls a movement of an arm through the joint-space velocities, then grabbing the arm and holding it stationary would constrain the movement (a $\mathbf{b}(\mathbf{x},t) = \mathbf{0}$ constraint). Additionally, holding the arm and moving it against the direction of the policy would correspond to a non-zero $\mathbf{b}(\mathbf{x},t)$ constraint. To better illustrate the effect of such constraints on system dynamics, in the following we outline some examples from the kinematic and force-based control of redundant manipulators.

3.2.1 Example: Resolved Motion Rate Control

A simple example of a constraint-based control scheme that directly corresponds to (3.2)-(3.4) is Resolved Motion Rate Control (RMRC) (Whitney, 1969; Liégeois, 1977). RMRC is a popular scheme for velocity-based control of rigid-body manipulators. It assumes a linearised forward model

$$\dot{\mathbf{r}} = \mathbf{J}(\mathbf{q},t)\dot{\mathbf{q}} \tag{3.5}$$

where $\mathbf{r} \in \mathbb{R}^k$ and $\mathbf{q} \in \mathbb{R}^n$ are the task- and joint-space coordinates respectively, $\dot{\mathbf{r}}$ and $\dot{\mathbf{q}}$ denote the task- and joint-space velocities and $\mathbf{J}(\mathbf{q},t)$ is the Jacobian relating the two. Note that, in general, the Jacobian is time-dependent reflecting the fact that the task-space may change (for example, a humanoid using this control scheme may switch from using both hands to manipulate an object to using just one; Gienger et al. 2005). In such a system, a typical task is to realise some trajectory $\mathbf{r}^*(t)$ in the chosen task-space. This places a constraint on the joint space velocity of the system, i.e. the joint space velocity must be such that $\mathbf{r}^*(t) - \mathbf{r}(t) = \mathbf{0}$ (this is also commonly known as Closed-Loop Inverse Kinematics (CLIK), e.g. see Chiacchio et al. 1991). The solution is given by the Liégeois inverse kinematic model (Liégeois, 1977)

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q},t)^\dagger\dot{\mathbf{r}} + (\mathbf{I} - \mathbf{J}(\mathbf{q},t)^\dagger\mathbf{J}(\mathbf{q},t))\mathbf{a} \tag{3.6}$$

where $\mathbf{a} \in \mathbb{R}^n$ is an arbitrary vector. The Liégeois model (3.6) can be used to represent most velocity-control methods (English and Maciejewski, 2000). Note that in this scheme, we can directly identify the state and action vectors $\mathbf{x} \equiv \mathbf{q}$, $\mathbf{u} \equiv \dot{\mathbf{q}}$ and the constraint relation (3.2) $\mathbf{A}(\mathbf{x},t) \equiv \mathbf{J}(\mathbf{x},t)$, $\mathbf{b}(\mathbf{x},t) \equiv \dot{\mathbf{r}}$. In (3.6) we are free to choose the vector $\mathbf{a}$ as we wish and usually this is done by defining some policy. In tracking tasks, policies may be chosen to complement the task constraints; for example, to promote stability in the joint-space (Peters and Schaal, 2008a) or to optimise the movement in some way (Nakamura, 1991). A popular choice is to define a policy that optimises a potential function (English and Maciejewski, 2000; Nakamura, 1991)

$$\mathbf{a} \equiv \pi(\mathbf{q}) = -\nabla\phi(\mathbf{q}). \tag{3.7}$$

Using a such a policy, it is guaranteed that $\phi(\mathbf{q})$ will be minimised by the joint space velocity vector $\dot{\mathbf{q}}$ at every time step (for $\dot{\mathbf{r}} = \mathbf{0}$, the decrease is monotonic (Nakamura, 1991)). Potentials can then be chosen whose minima correspond to some 'desirable' joint configuration; for example, those that avoid joint limits (Chaumette and Marchand, 2001), kinematic singularities (Yoshikawa, 1985) or obstacles (Choi and Kim, 2000; Khatib, 1985).

An example of how the choice of nullspace policy affects the behaviour of a simple three-link manipulator is given in Fig. 3.1. There, trajectories are shown for two policies subject to the constraint that the end-effector follows a given trajectory in Cartesian end-effector space.

3.2.2 Example: Force/Torque Control

For force or torque control of rigid body systems a similar constraint-based control scheme can be formulated (Udwadia, 2008; Peters et al., 2008, 2005; Udwadia, 2003; Bruyninckx and Khatib, 2000). This scheme assumes a robot model based on the Lagrangian equations of motion

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{F_c}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F_g}(\mathbf{q}) \tag{3.8}$$

where $\tau \in \mathbb{R}^n$ is the applied torque/force, $\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}} \in \mathbb{R}^n$ are joint-space positions, velocities and accelerations $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is an inertia/mass matrix, $\mathbf{F_c}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^n$ represents centrifugal and Coriolis forces and $\mathbf{F_g}(\mathbf{q}) \in \mathbb{R}^n$ is the gravity. Here, the constraint relation

$$\mathbf{A}(\mathbf{q},\dot{\mathbf{q}},t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q},\dot{\mathbf{q}},t) \tag{3.9}$$

is used to determine desired trajectories or forces according to the task (Peters and Schaal, 2008a; Udwadia, 2008; Peters et al., 2005; Udwadia, 2003; Bruyninckx and Khatib, 2000). Substituting (3.9) into (3.8) gives (Peters et al., 2005; Udwadia, 2003; Bruyninckx and Khatib, 2000)

$$\tau = \mathbf{T}^\dagger(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) + (\mathbf{I} - \mathbf{T}^\dagger\mathbf{T})\mathbf{a} \tag{3.10}$$

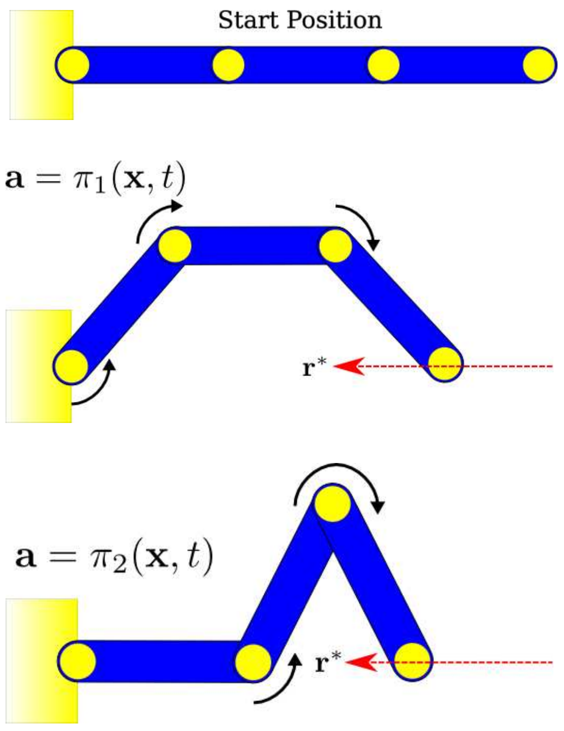

Start Position
$\mathbf{a} = \pi_1(\mathbf{x}, t)$
$\mathbf{r}^*$
$\mathbf{a} = \pi_2(\mathbf{x}, t)$
$\mathbf{r}^*$

Figure 3.1: Two ways to utilise the null-space under the constraint that the end-effector tracks a desired trajectory $\mathbf{r}^*(t)$ (dashed line) with a three link arm. The choice $\mathbf{a} = \pi_1(\mathbf{x}, t)$ uses the three joints equally, whereas the choice $\mathbf{a} = \pi_2(\mathbf{x}, t)$ uses the second and third joints more.

where for compactness we define $\mathbf{F} \equiv -\mathbf{F_c} - \mathbf{F_g}$, $\mathbf{T} \equiv \mathbf{A}\mathbf{M}^{-1}$ and $\mathbf{a}$ is an arbitrary vector. Here, a weighted pseudoinverse is often used, in which case different choices of weighting matrix $\mathbf{W}$ determine the control paradigm (Peters and Schaal, 2008a; Peters et al., 2005). For example, resolved acceleration kinematic control ($\mathbf{W} = \mathbf{M}^{-2}$) or the Operational Space Formulation (Khatib, 1987) ($\mathbf{W} = \mathbf{M}^{-1}$) both fit into this framework. As before, the vector $\mathbf{a}$ can be freely chosen, but is commonly used to implement some stabilising policy (Peters and Schaal, 2008a). An example of the effect of two different choices of policy is given in Fig. 3.2.

3.3 Learning from Constrained Policies

In the preceding section, we outlined a generic formulation of constraints applied to motion and illustrated how the dynamics of systems are affected by the constraint. In

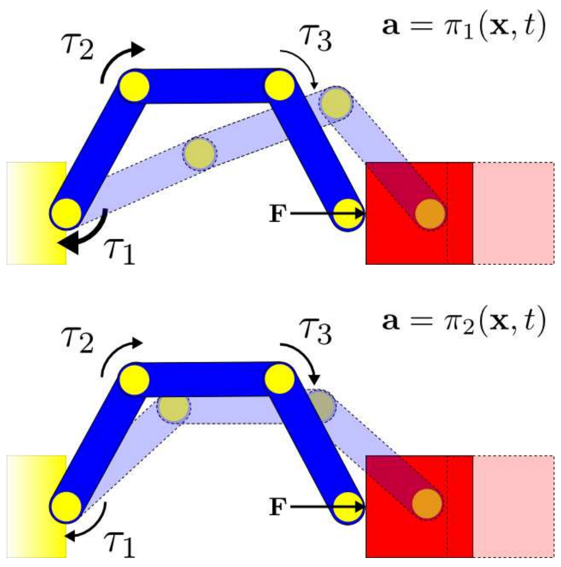

$\mathbf{a} = \pi_1(\mathbf{x}, t)$
τ_2
τ_3
τ_1
F
$\mathbf{a} = \pi_2(\mathbf{x}, t)$
τ_2
τ_3
τ_1
F

Figure 3.2: Two ways to utilise the null-space in dynamics control when applying a force F to mass (box) with a fixed-base three link arm. The upper scheme applies a large torque to the base joint, a medium torque to the second joint and a small torque to the third joint. The lower scheme uses equal torques for each joint. The choice of $\mathbf{a}$ in (3.10) determines the scheme used.

general, we saw that policies are projected into the nullspace of the constraints (i.e. the nullspace of $\mathbf{A}$). This results in changes in the observed behaviour, that in turn will affect how we can learn from observations.

Here, and throughout the thesis, we make certain assumptions on what is contained in the given observation data. In particular, we will assume that the data consists of observations of states and actions $(\mathbf{x}_n, \mathbf{u}_n)$, usually in the form of trajectories. Furthermore, we assume that the constraints are not explicitly observable (i.e. we do not have direct access to $\mathbf{A}(\mathbf{x},t)$, $\mathbf{b}(\mathbf{x},t)$ or $\mathbf{N}(\mathbf{x},t)$), and data sets are not labelled with respect to the constraints (i.e. observations may not all come from the same set of constraints). Our goal throughout is to find the underlying policy $\pi(\mathbf{x})$. In the following we characterise the implications of different classes of constraint for learning that policy. We first consider the simplest case (i.e. unconstrained observations) then go on to look at more complex problems, such as stationary and forced-action constraints.

3.3.1 Unconstrained Systems

The simplest class of systems that we may encounter in learning is that of directly observed *unconstrained* policies. In terms of the constraint formalism outlined in the preceding section, this corresponds to having $\mathbf{A} = \mathbf{0}$ (a matrix of zeros) and $\mathbf{b} = \mathbf{0}$ in (3.2)-(3.4) so that the projection operator is simply the identity matrix $\mathbf{N} \equiv \mathbf{I}$. Under such conditions the policy is fully observable, i.e. the observations are simply

$$\mathbf{u} = \pi(\mathbf{x}).$$

From the viewpoint of learning, the case of unconstrained policies is straightforward. Provided the policy is autonomous, the mapping $\pi : \mathbf{x} \to \mathbf{u}$ is static and so lends itself well to supervised learning techniques, such as those described in Ch. 2.

As an example, consider a simple unconstrained policy to extend a jointed finger as shown in Fig. 3.3(a). There, the policy simply moves the joints towards the zero (outstretched) position. The vector field representation of the behaviour is shown in red in Fig. 3.3(c). Learning the unconstrained policy in this case is simply a matter of finding a good fit to this vector field; for example, by fitting DMPs (Ijspeert et al., 2003, 2002b) or non-parametric modelling (Peters and Schaal, 2008a) (ref. Sec. 2.2.1).

3.3.2 Stationary Constraint Systems

The second class of systems that we may encounter are those subject to *stationary constraints*. These are constraints which act as hard restrictions on the actions available to the policy, but do not enforce actions as in the system (3.4). In other words, these are systems where $\mathbf{b} = \mathbf{0}$, so satisfy the constraint relation

$$\mathbf{A}(\mathbf{x},t)\mathbf{u} = \mathbf{0}. \tag{3.11}$$

This means that the observations consist of a projection of the policy into the nullspace of $\mathbf{A}(\mathbf{x},t)$:

$$\mathbf{u}(\mathbf{x},t) = \mathbf{N}(\mathbf{x},t)\pi(\mathbf{x}(t)). \tag{3.12}$$

The effect of the constraints (3.11)-(3.12) is to disallow motion in some sub-space of the system, specifically the space orthogonal to the image of $\mathbf{N}(\mathbf{x},t)$.

Constraints of the form (3.11) commonly appear in scenarios where manipulators interact with solid objects; for example, when grasping a tool or turning a crank or a pedal, i.e., contact constraint scenarios (Park and Khatib, 2006; Murray et al., 1994;

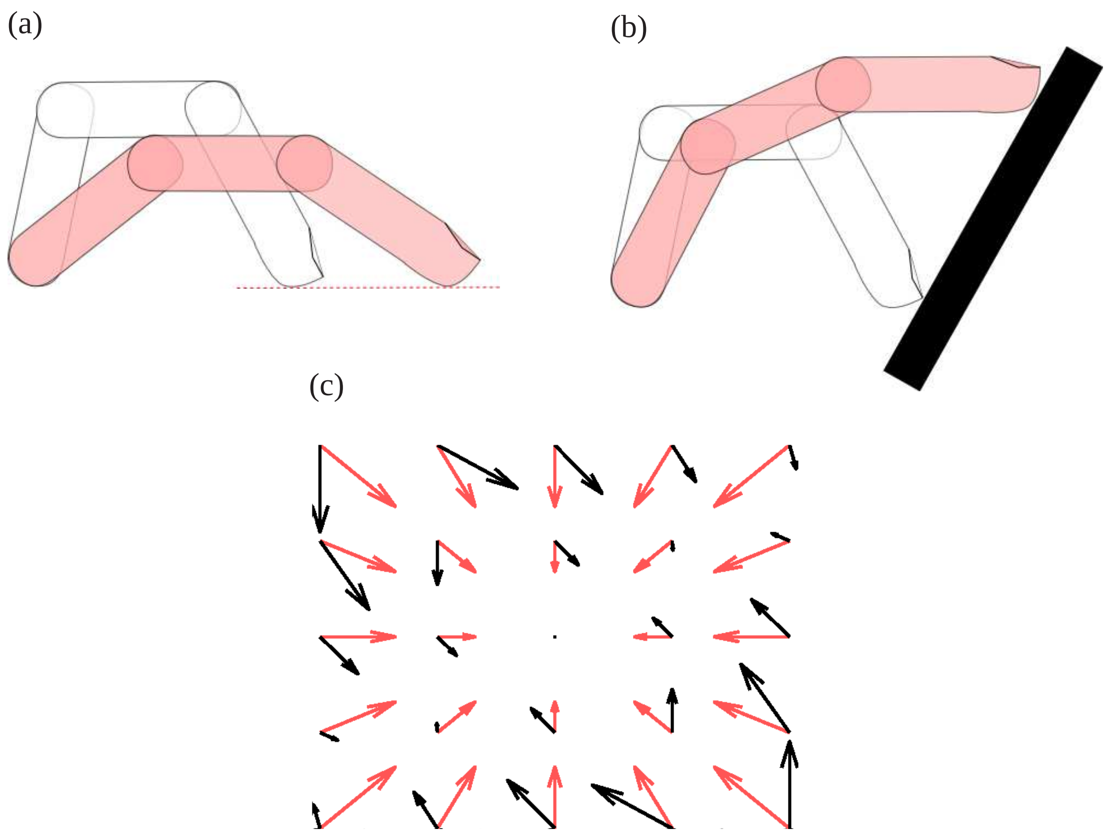


Figure 3.3: Illustration of two apparently different behaviours from the same policy: (a) unconstrained movement (b) movement constrained by an obstacle (black box) (c) vector field visualisation of the unconstrained (red) and constrained (black) policy for two of the finger joints as a function of their angles.

Mattikalli and Khosla, 1992). As a concrete example, consider again the finger extension policy, but this time with an immovable obstacle lying in the path of the finger, as shown in Fig. 3.3(b). Here, the obstacle acts as a stationary constraint since the finger cannot penetrate the obstacle surface. This alters the observations of the policy, as the finger moves along the surface due to the constraint. The vector field representation of this constrained behaviour is shown in black in Fig. 3.3(c).

In itself, a stationary constraint such as this applied to the policy does not cause problems for standard approaches to policy learning, *provided that the constraints are consistent across observations*. That is, if the constraint matrix is the same function of state $\mathbf{A}(\mathbf{x},t) \equiv \mathbf{A}(\mathbf{x})$ in all observations, the constraints can be absorbed into the policy model, i.e. we can learn the constrained policy mapping $\pi_{\mathbf{N}} : \mathbf{x} \mapsto \mathbf{u}$ where $\pi_{\mathbf{N}}(\mathbf{x}) \equiv \mathbf{N}(\mathbf{x})\pi(\mathbf{x})$. As already mentioned, this is exhibited by many existing studies where policy-based learning has been applied to problems where fixed constraints are implicit in the movement; ref. Ch. 2.

The difficulty, however, arises when there is the possibility of variation in the constraints appearing in the data, for example if the shape, position or orientation of the obstacle in Fig. 3.3(b) changed between or during observations. If this is the case, applying traditional approaches to learning in this scenario would result in one of two possibilities. The first is that if the observations are *labelled with respect to the constraint*, one could learn a separate policy model for the behaviour in each of the settings. In other words, we could learn a set of policies $\tilde{\pi}_{\mathbf{N}_i}$ for each constraint $\mathbf{N}_i$, $i \in \{1,\ldots,N\}$. However, this is unsatisfactory, since each model would only be valid for the specific setting (i.e. specific obstacle configuration), and we would need increasing numbers of models as we observed the policy under new constraints (obstacle configurations).

The second possibility arises when the data is either *unlabelled* with respect to the constraint or contains a mixture of observations under different constraints. In this case, one might try to perform regression directly on the observations, that is observations from both vector fields (cf. Fig. 3.3(c), black and red vectors). However, this presents the problem of *non-convexity* in the training data, which causes difficulties for many supervised learning algorithms.

The problem is illustrated in Fig. 3.4(a). There we show a policy π constrained in two different ways. In the first observation $\mathbf{u}_1$, the constraint prevents movement in the direction normal to the vertical plane[2]. For the second observation $\mathbf{u}_2$, the constraint only allows movement in the horizontal plane. To the learner, data from these two scenarios appears *non-convex*, in the sense that for any given point $\mathbf{x}$ in the input space multiple observed outputs $\mathbf{u}$ exist. Directly training on these observations with many supervised learning algorithms would result in *model-averaging*. Here, averaging of $\mathbf{u}_1, \mathbf{u}_2$ results in the prediction $\bar{\mathbf{u}}$ that is clearly a poor representation of the true policy π, both in terms of direction and magnitude of the predictions (ref. Fig. 3.4(b)). In order to avoid this, we need to explicitly consider the effect of constraints when learning.

A second problem that arises when training on data from constrained policies (3.11)–(3.12) is that of *degeneracy* in the data. This stems from the fact that, for any given set of projected (constrained) policy observations, there exist multiple can-

[2]It should be noted that in general the orientation of the constraint plane onto which the policy is projected may vary both with state position and time.

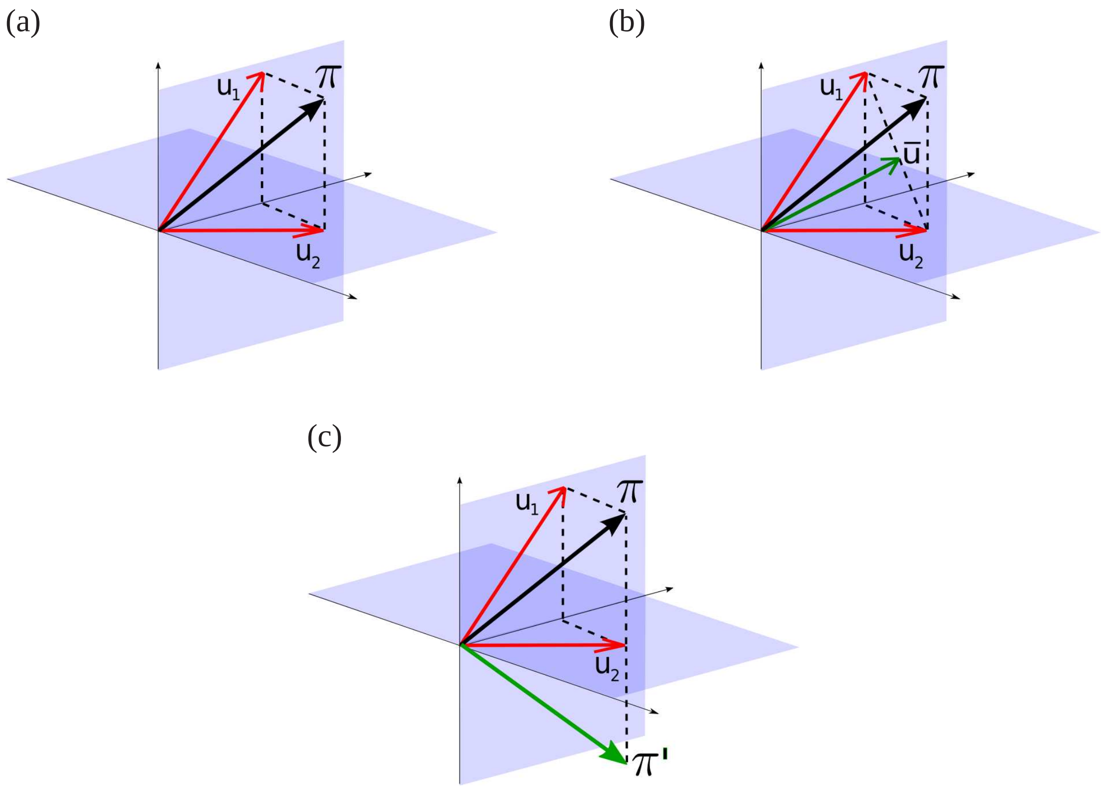


Figure 3.4: Illustration of the effect of constraints on the unconstrained policy, the averaging effect of standard DPL and the degeneracy problem. (a) Two constraints applied to the policy π result in projected observations $\mathbf{u}_1, \mathbf{u}_2$. (b) Direct regression results in averaging of the two movements $\bar{\mathbf{u}}$ in a way that cannot explain the observations. (c) Two policies π, π' that both may be constrained in such a way as to produce the observation $\mathbf{u}_2$.

didate policies that could have produced that movement. The cause for this is that the projection eliminates components of the unconstrained policy that are orthogonal to the image of $\mathbf{N}(\mathbf{x},t)$ so that the component of π in this direction is undetermined by the observation. For example, consider the constrained observation $\mathbf{u}_2$ in Fig. 3.4(c). There, the restriction of the motion in vertical direction implies that we do not observe the vertical component of π. Given only $\mathbf{u}_2$, we cannot determine if the policy π or an alternative, such as π' (ref. Fig. 3.4(c)) produced the observation. In effect we are not given sufficient information about the unconstrained policy to guarantee that it is fully reconstructed.

This problem cannot be avoided when dealing with stationary constraint systems. However, despite this, we wish to do the best we can with the data available. In the methods developed in this thesis, we adopt a strategy whereby we look for policies that are, as a minimum, consistent with the constrained observations $\mathbf{u}$. For example, returning to Fig. 3.4(c), if we only observe $\mathbf{u}_2$, (that is the policy under a single, specific constraint) the simplest (and safest) strategy would be to use that same vector as our prediction. In this way, we can at least accurately predict the policy under that constraint (albeit only under that particular constraint). If we are then given further observations under new constraints, we can recover more information about the unconstrained policy π. For instance, observing $\mathbf{u}_1$ eliminates the possibility that π' underlies the movements since it cannot project onto both $\mathbf{u}_1$ and $\mathbf{u}_2$. Applying this strategy for increasing numbers of observations, our model will not only generalise over the constraints seen, but also come closer to the unconstrained policy π.

Finally, it should be noted that if, in all observations, certain components of the policy are always constrained, then we can never hope to uncover those components. However, in such cases it is reasonable to assume that, if these components are always eliminated by the constraints, then they are not relevant for the scenarios in which movements were recorded.

In the following chapters, we propose several methods by which we can overcome these problems to learn a good model of the policy π, without need for explicit knowledge of the constraints $\mathbf{N}(\mathbf{x},t)$, and that is, at the very least, consistent with all the observations under the constraints seen in the data.

3.3.3 Forced-action Constraint Systems

The final class of constrained system that we consider is that of 'forced-action' constraints, i.e. the system (3.2)–(3.4):

$$\mathbf{A}(\mathbf{x},t)\pi(\mathbf{x}) = \mathbf{b}(\mathbf{x},t)$$

with policy observations

$$\mathbf{u}(t) = \mathbf{A}(\mathbf{x},t)^{\dagger}\mathbf{b}(\mathbf{x},t) + \mathbf{N}(\mathbf{x},t)\pi(\mathbf{x}).$$

Here, the effect of the constraints is two-fold. First, similar to the stationary constraint case, the policy π is restricted in the sense that the action components orthogonal to the image of $\mathbf{N}(\mathbf{x},t)$ are projected out of the observations. However, in this case there is the further complication that the observed actions contain an additional component that is independent of the policy, due to the additive term $\mathbf{A}(\mathbf{x},t)^{\dagger}\mathbf{b}(\mathbf{x},t)$. This term can

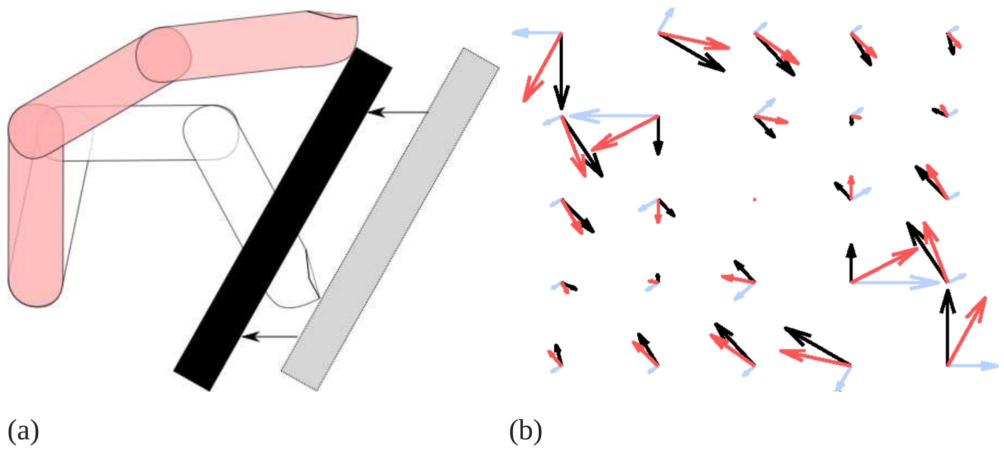


Figure 3.5: Finger extension under moving constraints. The fingertip is constrained to maintain contact with the obstacle as the obstacle moves. The observed policy (red) is now a combination of forced movement due to the motion of the obstacle (light blue) and constrained policy (black) components.

be thought of as a ‘forced action’, i.e. an action that must be taken in order to satisfy the constraints.

For example, consider again the finger-extension policy, with the constraint that the fingertip maintains contact with the surface of the obstacle, but this time with an external force moving the obstacle. This is illustrated in Fig. 3.5 for the same policy as that in Fig. 3.3. Here, the finger extends along the surface of the obstacle until it reaches the contact point furthest from the base of the finger (i.e. closest to the fully extended position). However, as the obstacle moves toward the finger, the finger is forced away from the fully extended position. In the vector field view, Fig. 3.5(b), the observed motion (red) contains a component due to the constrained policy (black)[3] and an additional component due to the motion of the obstacle (blue).

Constraints of this form are also commonly applied in the control of redundant manipulators (Sec. 3.2.1 and Sec. 3.2.2), where the forced-action component (also known as the task- or Operational Space component) is used to ensure that the system follows some desired trajectory (e.g. in end-effector space) or applies a desired set of forces.

With regards to learning, the same problems of *degeneracy* and *non-convexity* ap-

[3]This is identical to that of the policy subject to stationary constraints (compare black vectors in Fig. 3.5(b) and Fig. 3.3(c).

ply for this kind of constraint, as for the stationary constraint case (ref. Sec. 3.3.2). However, in addition to this, the added movement component induced by this kind of constraint results in ambiguity as to what parts of the observations belong to the policy and which to the constraint. If it is assumed that no prior knowledge about the constraints (i.e. $\mathbf{A}(\mathbf{x},t)$ and $\mathbf{b}(\mathbf{x},t)$) is given, then there is the problem of separating the two components of action before learning can proceed. To the author's knowledge, it remains an open problem whether the policy can be learnt under this kind of constraint.

3.4 Feasibility of Learning and Measuring Performance

In the preceding three sections, we saw how different classes of constraint affect our observations and the problems this causes when trying to model the underlying policy. In particular, we noted that in some cases, it may not be possible to fully reconstruct the unconstrained policy (due to the specific set of constraints contained in the data), but that we still wish to do the best we can with the data available. Given this to be the case, in this section we explore possible limitations of learning in two ways.

First, we look at the *feasibility of learning* and characterise the conditions necessary for accurate modelling in the ideal case. We present a geometric argument and derive the ideal set of observations necessary for exact reconstruction of the policy. We will see that, though ideal observations are unlikely in real data sets, this analysis still indicates that learning is feasible and even suggests some naive methods that may lead to a solution.

Second, we look at how we can *measure performance* in settings where the constraints do not permit exact reconstruction. Specifically, we define a pair of error measures by which we can judge (i) how well our policy models represent the observed (constrained) data, and; (ii) the extent to which our model generalises over constraints. In later chapters we will use these measures extensively to assess the quality of models learnt by the algorithms developed in this thesis.

3.4.1 Exact Geometric Reconstruction of Policies

Given the difficulties highlighted in the preceding sections, one may ask whether it is plausible to expect any learning algorithm to be able to reconstruct the policy from constrained observations, or whether the problem itself is ill-posed. However, as an indication of the feasibility, we can perform a geometric analysis to see how, under ideal conditions, it is possible to exactly reconstruct the policy from projected observations (Howard and Vijayakumar, 2007).

Theorem 3.4.1. *Exact Reconstruction of Projected Policies*

Given observations $\mathbf{u}_n = \mathbf{N}_n\pi$; $n = \{1, \ldots, N\}$ of a policy $\pi(\mathbf{x})$ projected into the nullspace of a set of N constraints at a point $\mathbf{x}$ in the state space; if the constraints are such that the observations span the action space, then the unconstrained policy can be exactly reconstructed as

$$\pi = \mathbf{u}^{\times} \tag{3.13}$$

where $\mathbf{u}^{\times}$ is the solution to the linear system

$$\mathbf{U}^T\mathbf{u}^{\times} = \mathbf{d} \tag{3.14}$$

where $\mathbf{U} \equiv (\mathbf{u}_1, \ldots, \mathbf{u}_N)$ and the elements of $\mathbf{d}$ are given by $d_n = \mathbf{u}_n^T\mathbf{u}_n$.

Proof. Consider that we observe a two dimensional policy, $\pi \in \mathbb{R}^2$, constrained by a number of constraints of the form

$$\mathbf{A}_n = (\alpha_1, \alpha_2)_n = \alpha_n. \tag{3.15}$$

Under the constraints, the observations that we see $\mathbf{u}_n = \mathbf{N}_n\pi$ will lie inscribed in a circle in the two-dimensional space with diameter equal to the norm of the (unconstrained) policy vector at that point, i.e. $\|\pi\|$, as illustrated in Fig. 3.6. Euclid's theorem states that any triangle inscribed in a semi-circle is a right-angle triangle. Hence, if we construct a line orthogonal to the *i*th observation $\mathbf{u_i}$ and find the appropriate triangle whose hypotenuse matches the diameter of the circle, then we can calculate the length of the vector along the hypotenuse and thus reconstruct the unconstrained policy vector. To do this in two-dimensional space, we can take two observations under different constraints $\mathbf{u}_1, \mathbf{u}_2$, find the equations of the two lines orthogonal to those observations and solve for the intersection point $\mathbf{u}^{\times}$, to exactly recover the unconstrained policy π, as illustrated in Fig. 3.6. In *m* dimensional space we can take a similar approach; there, the constrained observations lie on a hypersphere, so we construct hyperplanes normal to each observation and solve for the intersection $\mathbf{u}^{\times}$. This leads to the system (3.14) with the unconstrained policy given by (3.13). □

Theorem 3.4.1 also suggests the following corollary.

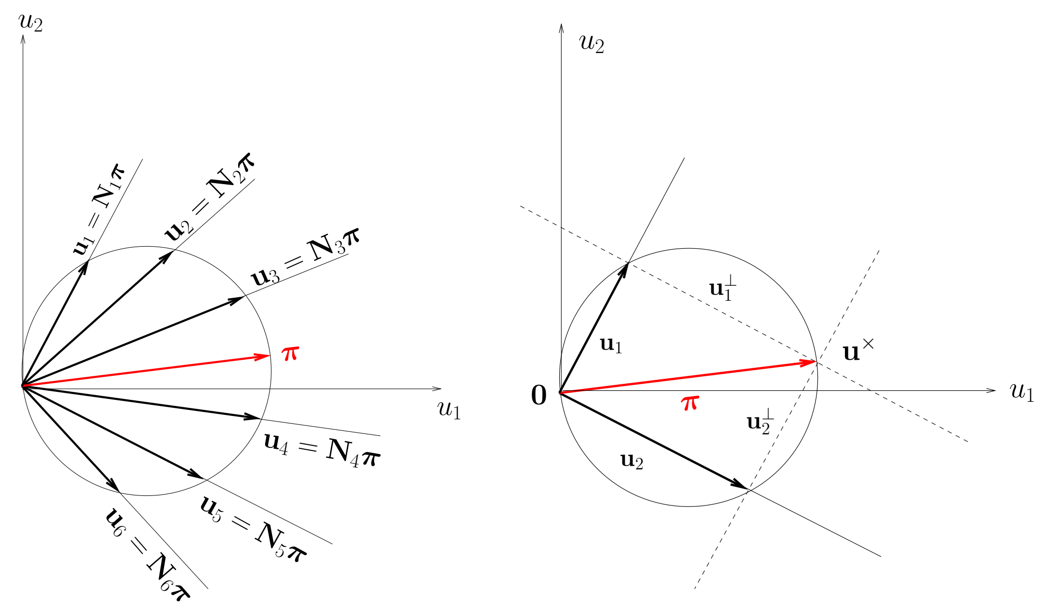

u_2
u_1
$\mathbf{u}_1 = \mathbf{N}_1\boldsymbol{\pi}$
$\mathbf{u}_2 = \mathbf{N}_2\boldsymbol{\pi}$
$\mathbf{u}_3 = \mathbf{N}_3\boldsymbol{\pi}$
$\boldsymbol{\pi}$
$\mathbf{u}_4 = \mathbf{N}_4\boldsymbol{\pi}$
$\mathbf{u}_5 = \mathbf{N}_5\boldsymbol{\pi}$
$\mathbf{u}_6 = \mathbf{N}_6\boldsymbol{\pi}$
u_2
u_1
$\mathbf{0}$
$\mathbf{u}_1^{\perp}$
$\mathbf{u}_1$
$\mathbf{u}^{\times}$
$\boldsymbol{\pi}$
$\mathbf{u}_2^{\perp}$
$\mathbf{u}_2$

Figure 3.6: Under each of the different constraints (3.15), the policy is projected onto a different manifold orthogonal to the constraint direction α_n. These constrained policy vectors $\mathbf{u}_n$ lie inscribed in a hypersphere in state-space (left). Euclid's Theorem can be used to exactly reconstruct π given observations under different constraints (right).

Corollary 3.4.1. *Given observations $\mathbf{u}_n = \mathbf{N}_n\pi$, $n = 1,\ldots,N$ of a policy π, at a point $\mathbf{x}$ in state space, the observation with the largest norm $\|\mathbf{u}_n\|$ lies closest to the unconstrained policy.*

Proof. By inspection of Fig. 3.6, or by considering that $\mathbf{N}(\mathbf{x},t)$ is a projection matrix, with k eigenvalues of value 0 and $d-k$ eigenvalues of value 1. Fewer constraints (smaller k) results in larger norms. $\square$

These results indicate that, at least under these ideal conditions, it is possible to reconstruct the policy from constrained observations. Furthermore, this gives us reason to believe that a learning algorithm can be formulated to deal with this problem. For example, corollary 3.4.1 immediately suggests a possible, (albeit rather data-intensive) solution: One could train in such a way that, as increasing observations under different constraints arrive, observations with smaller norms are down-weighted in the regression. As the amount of data presented to the learner increases, the model should come ever closer to the unconstrained policy. In fact, as we will see in later chapters, there are far more efficient ways to formulate the learning problem and find a good approximation of the policy with relatively small data sets.

3.4.2 Error Measures

In the preceding section we saw how exact reconstruction of the unconstrained policy at a given point in state space is possible under ideal conditions, namely, given a spanning set of constrained observations. In practice, however, such ideal observations will not, in general, be available and an approximation must therefore be made. In order to measure the quality of this approximation, we must define error measures that reflect our learning goals. In this thesis, the primary goals of learning are to (i) represent the movement in such a way as to be at least consistent with the (constrained) observations, and, (ii) where possible, fully reconstruct the unconstrained policy to generalise over constraints. To quantify these goals, we can define the following two metrics.

First, we define the normalised *unconstrained policy error* (nUPE) as

$$E_{upe}[\tilde{\pi}] = \frac{1}{N\sigma_\pi^2}\sum_{n=1}^{N} ||\pi(\mathbf{x}_n) - \tilde{\pi}(\mathbf{x}_n)||^2 \tag{3.16}$$

where N is the number of data points, $\pi(\mathbf{x}_n)$ and $\tilde{\pi}(\mathbf{x}_n)$ are the (unconstrained) true and learnt policy predictions at the points $\mathbf{x}_n$ and σ_π^2 is the variance in the true policy over those points. The nUPE measures the difference between samples of the (unconstrained) true and learnt policies, normalised by the variance. Since the primary goal of the approaches developed in this thesis is to find a good approximation of the unconstrained policy, a low nUPE indicates good performance. Note also that the nUPE also gives an indication of how well the learnt policy will generalise over different constraints, since if the learnt policy closely matches the true unconstrained policy, then it will also closely match the true policy under any arbitrary projection (constraint).

The second measure we define is the normalised *constrained policy error* (nCPE)

$$E_{cpe}[\tilde{\pi}] = \frac{1}{N\sigma_\pi^2}\sum_{n=1}^{N} ||\mathbf{N}_n\left(\pi(\mathbf{x}_n) - \tilde{\pi}(\mathbf{x}_n)\right)||^2 \tag{3.17}$$

where $\mathbf{N}_n$ denotes the constraint (projection) matrix for the n-th point. The nCPE measures the difference between the true and learnt policies under the projections $\mathbf{N}_n$. The significance of the nCPE is that it allows one to measure the accuracy of the learnt policy under a specific set of constraints (i.e. those encoded by the projections $\mathbf{N}_n$). For example, if we chose $\mathbf{N}_n$ as the set of projections corresponding to the constraints in force in the training data, then we can assess how well our model will perform under those same constraints. Alternatively, if we chose $\mathbf{N}_n$ corresponding to a set of novel, unseen constraints, we can directly measure how well the policy generalises to predict behaviour under those new constraints.

In later chapters we will use these measures extensively to compare the performance of our algorithms against each other and against existing policy learning methods. We will also see how, considering the goal of learning in terms of such measures leads to novel approaches to learning (ref. Ch. 5).

3.5 Conclusion

In this chapter, we outlined a formal system for the analysis of constrained motion systems. Based on recent work in analytical dynamics, the formalism provides a simple and intuitive way to deal with constraints both in terms of the kinematics and dynamics of a wide variety of systems. Further to this, we gave examples of its application in terms of well-known kinematic and force-based control schemes for redundant manipulators.

Using this model, we then discussed the implications that constraints have on the learning of policies from raw observation data. We saw that, while standard approaches to policy-based learning are effective for learning from unconstrained or consistently constrained data, they face difficulties when policies are subject to several different constraints. For stationary constraints, these can be termed the problems of *non-convexity* and *degeneracy*. For forced-action constraints, these problems are further complicated by additional components to the observed actions induced by the constraints.

Finally, we discussed our strategy for dealing with these constraints by looking for policy models that are *consistent* with the observations subject to the constraints. We noted that, with a geometrical analysis, exact reconstruction of the policy from constrained observations is possible under ideal conditions. This suggests that learning in this scenario is feasible despite the problems caused by constraints. In the next chapter, we discuss a method for doing this for the special case of potential-based policies, before going on to deal with the learning of generic policies in subsequent parts of the thesis.

Chapter 4

Learning Potential-based Policies from Constrained Motion

4.1 Introduction

In the previous chapter, we reviewed a formal framework for dealing with constraints in movement and discussed the problems such constraints cause for standard approaches to learning. We noted that direct regression of the observed actions and commands is unlikely to produce satisfactory results when there are constraints due to model averaging effects. Further, it was suggested that new algorithms be developed that can learn models that are consistent with the constrained observations.

In this chapter, we describe a method for doing this for a special class of policies, namely that of *potential-based policies*. The method can be classed as an indirect learning approach (cf. Sec. 2.2.3) in that it aims to represent policies in terms of their generating functions; in this case, the potential function underlying the policy.

In the following we first describe in detail the definition of a potential-based policy, and how dealing with these is a promising way to solve some of the problems with direct learning as outlined in in the previous chapter. We then go on to propose a method to learn potential-based policies for stationary constraint systems (ref. Sec. 3.3.2) through a local model alignment scheme. Finally, we look at the performance of the method on a number of constrained systems of varying size and complexity.

4.1.1 Potential-based Policies

Potential-based policies are a special subclass of generic policies that are conservative in the mathematical sense. Specifically, a generic policy $\pi(\mathbf{x}) \in \mathbb{R}^n \mapsto \mathbb{R}^d$ is a vector field $\mathbf{y} = \mathbf{f}(\mathbf{x}) \in \mathbb{R}^d$ on on the state-space $\mathbf{x} \in \mathbb{R}^n$, where n, d are the dimensions of the input/output state spaces, respectively.

According to the Helmholtz decomposition, any vector field $\mathbf{f}(\mathbf{x})$ may be comprised of rotational and divergent components

$$\mathbf{f}(\mathbf{x}) = \nabla_{\mathbf{x}} \times \mathbf{\Phi}(\mathbf{x}) + \nabla_{\mathbf{x}}\phi(\mathbf{x}) \tag{4.1}$$

where $\mathbf{\Phi}$ and ϕ are vector and scalar potentials, respectively. A potential-based (i.e. conservative) policy is one for which the first term in (4.1) is zero so that the policy can be fully represented by the potential function $\phi(\mathbf{x})$. This leads to the following definition:

Definition 4.1.1. *Conservative Policies*
A conservative (i.e. potential-based) policy is a policy defined through the gradient of a scalar potential function $\phi(\mathbf{x})$

$$\pi(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}), \quad \forall \mathbf{x} \in X \tag{4.2}$$

where X is the region of the input (state) space of interest. Note that policies may be globally or locally conservative, that is, there may be regions outside X where the vector field is non-conservative (Boas, 2006).

A necessary and sufficient condition for a policy to be conservative (potential-based) is that it has *zero curl* in the region X:

$$\nabla_{\mathbf{x}} \times \pi(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in X. \tag{4.3}$$

The curl of a vector field is a measure of rotational flow of the field, so if the curl is zero the policy contains no rotational components. In other words, potential-based policies do not contain periodic behaviour; for example, a limit cycle cannot be represented as a potential-based policy. Instead, potential-based policies represent divergent attractor landscapes where the minima of the potential correspond to stable attractors, and maxima correspond to repellors. They can be also be thought of as policies that greedily optimise the potential function at every time step (Nakamura, 1991). An example is given in Fig. 4.1 where a potential function with three maxima (repellors) and two

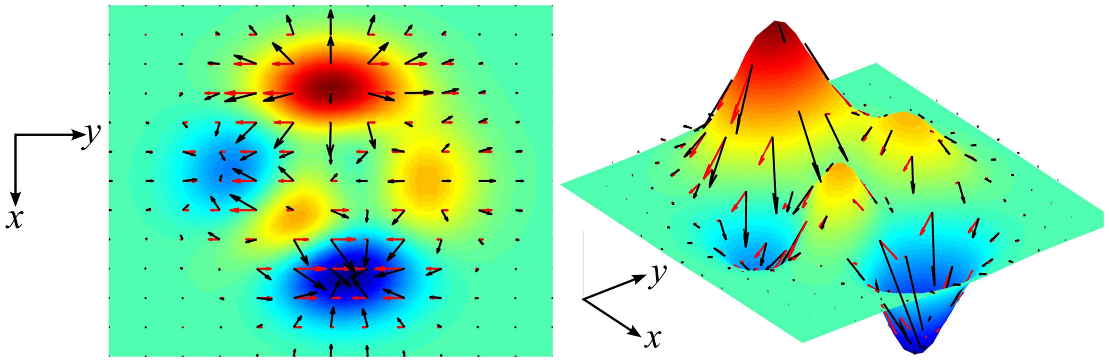


Figure 4.1: Potential function with three maxima (repellors) and two minima (attractors). Overlaid are the corresponding unconstrained policy vectors (black) and a set of constrained policy vectors (red).

minima (attractors) is shown, and the corresponding policy is overlaid (black vectors).

A wide variety of behaviours may be represented as potential-based. For example, reaching behaviours may be represented by a potential defined in hand space, with a single minimum at the target. Furthermore, decision-based behaviours may be encoded as potentials (Körding and Wolpert, 2004; Körding et al., 2004; Chajewska et al., 2001, 1998). For example, when reaching for an object, a potential may be defined with two minima, one corresponding to reaching with the right hand, the other reaching with the left. The decision of which hand to use for reaching would thus be determined by the start state (e.g. reach with the closest hand) and the relative offset of the two minima (e.g. right-handedness would imply a lower minimum for that hand). Potential-based policies are also extensively used as nullspace policies in control of redundant manipulators (Gienger et al., 2005; English and Maciejewski, 2000; Chaumette and Marchand, 2001; Choi and Kim, 2000; Nakamura, 1991; Yoshikawa, 1985), and for navigation and obstacle avoidance problems in mobile robotics (Ren et al., 2006; Conner et al., 2003; Rimon and Koditschek, 1992). Furthermore, in reinforcement learning and optimal control (Bertsekas, 2007; Sutton and Barto, 1998; Todorov, 2006), policies that are greedy with respect to the value function can be thought of as potential-based, in the sense that the policy does a gradient descent on the value function.

4.1.2 Learning from Constrained Potential-based Policies

If the policy under observation is potential-based, an elegant solution to solving the non-convexity and degeneracy problems is to model the policy's *potential function* (Howard et al., 2008a,b; Howard and Vijayakumar, 2007) rather than modelling it directly. This is due to a special property of constrained potential-based policies, namely that observations of the constrained movements give us information about the shape of the underlying potential, up to a translation in ϕ corresponding to constants of integration for the observations.

In Fig. 4.1 this is shown for a potential function defined over a two-dimensional state-space (top and 3-D perspective views). The potential function (colours) and unconstrained policy (black vectors) is shown, along with the policy subject to a constraint (red vectors). For the case of potential-based policies the policy vectors are given by the gradient vector of the potential (as expressed in (4.2)). This means that the (unconstrained) policy vectors point in the direction of steepest descent, with the magnitude equal to the slope in that direction (Fig. 4.1, black vectors).

Now, if a constraint is applied, the direction and magnitude of the vectors change. In the example in Fig. 4.1 the constraint prevents movement in one dimension (x dimension in Fig. 4.1, left) so that only motion corresponding to the second dimension (y dimension in Fig. 4.1, left) is observed. The vectors now point in the direction of steepest descent *subject to the constraint*, with magnitude equal to the slope of the potential in that direction, as can be seen from Fig. 4.1, right. In other words the projected vectors correspond to the *directional derivatives* of the potential, in the direction parallel to the observations.

This lends us the opportunity of modelling the unconstrained policy, by piecing together information about the slope of the potential in different directions. For each observation (e.g. $\mathbf{u}_1$ in Fig. 3.4) we get information about the directional derivative in that direction (i.e. the direction parallel to $\mathbf{u}_1$). This means we transform the problem of combining a set of n-dimensional vector observations (ref. Sec. 3.4.1) to one of 'piecing together' local estimates of the slope of the potential.

A convenient method for doing this for policies with a linear relationship between actions and state-changes (i.e. those for which $\mathbf{u} \equiv \dot{\mathbf{x}}$, such as kinematic policies) is to use line integration to accurately estimate the form of the potential along trajectories (Howard et al., 2008a,b; Howard and Vijayakumar, 2007; Howard et al., 2006) and then use these local estimates to build a global model of the potential. We outline a method for doing this in the next section.

4.2 Learning Policies Through Local Model Alignment

In the following we propose a method for modelling the potential from constrained motion data for a kinematic policy subject to stationary constraints. Specifically, we assume we are given a set of K observed trajectories $(\mathbf{x}_k(t), \mathbf{u}_k(t) \equiv \dot{\mathbf{x}}_k(t))$ that, during any given observation, may have been constrained (projected) $\mathbf{u}_k = \mathbf{N}_k(\mathbf{x},t)\pi(\mathbf{x}_k(t))$ (ref. (3.11)), with unknown projection matrix $\mathbf{N}_k(\mathbf{x},t)$. Our goal is to uncover the unconstrained policy $\pi(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x})$ by modelling the potential function $\phi(\mathbf{x})$. To do this, we first model the potential on a trajectory-wise basis using numerical line integration. We then consolidate these trajectory-wise models using results from recent work in dimensionality reduction (Verbeek, 2006; Verbeek et al., 2004) to ensure consistency. Finally, we use these consistent models to learn a global model of the potential function, and thus the policy, for use in control.

4.2.1 Estimating the Potential along Single Trajectories

As has been described in (Howard et al., 2008a,b; Howard and Vijayakumar, 2007; Howard et al., 2006), it is possible to model the potential along sampled trajectories using a form of line integration. Specifically, combining (3.12) and (4.2) for $\mathbf{u} \equiv \dot{\mathbf{x}}$, the (continuous time) state evolution of the system is given by

$$\dot{\mathbf{x}} = \mathbf{N}(\mathbf{x},t)\pi(\mathbf{x}) = -\mathbf{N}(\mathbf{x},t)\nabla_{\mathbf{x}}\phi(\mathbf{x}) \tag{4.4}$$

Let $\bar{\mathbf{x}}(t)$ be the solution of (4.4). If we line-integrate along $\bar{\mathbf{x}}(t)$ we have

$$\int_{\bar{\mathbf{x}}} (\nabla_{\mathbf{x}}\phi)^T \, d\mathbf{x} = \int_{t_0}^{t_f} (\nabla_{\mathbf{x}}\phi)^T \dot{\mathbf{x}} \, dt = -\int_{t_0}^{t_f} (\nabla_{\mathbf{x}}\phi)^T \mathbf{N}(\mathbf{x},t)\nabla_{\mathbf{x}}\phi(\mathbf{x}) \, dt, \tag{4.5}$$

where t_0 and t_f are the start and finishing instants of $\bar{\mathbf{x}}(t)$. We assume that we have recorded trajectories $\mathbf{x}(t), \dot{\mathbf{x}}(t)$ of length T sampled at some sampling rate $1/\delta t$ Hz so that for each trajectory we have a tuple of points $\mathbf{X}_k = \mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,T\delta t}$. Now, assuming the sampling rate to be sufficiently high, we can make a linear approximation to (4.4)

$$\mathbf{x}_{i+1} \approx \mathbf{x}_i + \delta t \, \mathbf{N}_i \pi_i = \mathbf{x}_i - \delta t \, \mathbf{N}_i \nabla_{\mathbf{x}}\phi(\mathbf{x}_i) \tag{4.6}$$

and (4.5) can be approximated using an appropriate numerical integration scheme. An example of such a scheme is Euler integration, which involves the first order approximation

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) + \frac{1}{\delta t}(\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathbf{N}_i \nabla_{\mathbf{x}} \phi(\mathbf{x}_i). \tag{4.7}$$

Since the effect of the time constant δt is simply to scale the discretised policy vectors, we can neglect it by scaling time units such that $\delta t = 1$. This comes with the proviso that for implementation on the imitator robot, the learnt policy may need to be scaled back to ensure that the correct time correspondence is kept. For steps $\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}$ that follow the projected policy (3.4) we can rearrange (4.6) with the scaled time coordinates, and substitute into (4.7) to yield

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2, \tag{4.8}$$

where the negative sign reflects our assumption (as expressed in (4.2)) that attractors are minima of the potential. We use this approximation to generate estimates $\breve{\phi}(\mathbf{x}_i)$ of the potential along any given trajectory $\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_N$ in the following way: We set $\breve{\phi}_1 = \breve{\phi}(\mathbf{x}_1)$ to an arbitrary value and then iteratively assign $\breve{\phi}_{i+1} := \breve{\phi}_i - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2$ for the remaining points in the trajectory.

Note that an arbitrary constant can be added to the potential function without changing the policy. Therefore, ‘local’ potentials that we estimate along different trajectories need to be *aligned* in a way that their function value matches in intersecting regions. We’ll turn to this problem in the next section.

4.2.2 Constructing the Global Potential Function

Let us assume we are given K trajectories $\mathbf{X}_k = (\mathbf{x}_{k1}, \mathbf{x}_{k2} \ldots \mathbf{x}_{kN_k})$ and corresponding point-wise estimates $\breve{\Phi}_k = (\breve{\phi}_{k1}, \breve{\phi}_{k2} \ldots \breve{\phi}_{kN_k})$ of the potential, as provided from the Euler integration just described. In a first step, we fit a function model $\tilde{\phi}_k(\mathbf{x})$ of the potential to each tuple $(\mathbf{X}_k, \breve{\Phi}_k)$, such that $\tilde{\phi}_k(\mathbf{x}_i) \approx \breve{\phi}_{ki}$. Although in principle any regression method could be applied here, our options are somewhat limited by the fact that these possibly non-linear models have to be acquired from the few data points available in each trajectory. To avoid unnecessary complications, we choose a nearest-neighbour (NN) regression model, i.e.,

$$\tilde{\phi}_k(\mathbf{x}) = \breve{\Phi}_{ki^*} \quad , \quad i^* = \arg\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2. \tag{4.9}$$

Since we wish to combine the models to a global potential function, we need to define some function for weighting the outputs of the different models. For the NN algorithm, we choose to use a Gaussian kernel

$$w_k(\mathbf{x}) = \exp\left[-\frac{1}{2\sigma^2}\min_i \|\mathbf{x}-\mathbf{x}_{ki}\|^2\right]. \tag{4.10}$$

From these weights we can calculate responsibilities

$$q_k(\mathbf{x}) = \frac{w_k(\mathbf{x})}{\sum_{i=1}^K w_i(\mathbf{x})} \tag{4.11}$$

and a (naive) global prediction $\tilde{\phi}(\mathbf{x}) = \sum_{k=1}^K q_k(\mathbf{x})\tilde{\phi}_k(\mathbf{x})$ of the potential at $\mathbf{x}$. However, as already stated, the potential is only defined up to an additive constant, and most importantly this constant can vary from one local model to another. This means that we first have to shift the models by adding some *offset* to their estimates of the potential, such that all local models are *in good agreement* about the global potential at any number of states $\mathbf{x}$.

Fortunately, a similar problem has already been tackled in the literature: In the field of non-linear dimensionality reduction, Verbeek et al. (2004) have shown how to align multiple local PCA models into a common low-dimensional space. In particular, they endowed each local PCA model with an additional affine mapping $\mathbf{g}_k(\mathbf{z}) = \mathbf{A}_k\mathbf{z}+\mathbf{b}_k$, which transformed the coordinates $\mathbf{z}_k$ of a data point *within* the k-th PCA model into the desired global coordinate system. Verbeek et al. (2004) retrieved the parameters of the optimal mappings $\mathbf{g}_k$ by minimising the objective function

$$E = \frac{1}{2}\sum_{m=1}^M\sum_{k=1}^K\sum_{j=1}^K q_{km}q_{jm}\|\mathbf{g}_{km}-\mathbf{g}_{jm}\|^2, \tag{4.12}$$

where $\mathbf{g}_{km}$ denotes the coordinate of the m-th data vector, as mapped through the k-th PCA model, and q_{km} is the corresponding responsibility of that model. The objective can easily be interpreted as the ‘disagreement’ between any two models, summed up over all data points, and weighted by the responsibilities of two models each. That is, the disagreement for any combination of m,k and j only really counts, if the responsibility of both the k-th and the j-th model is sufficiently high for the particular query point. Notably, E is convex and can be minimised by solving a generalised eigenvalue problem of moderate dimensions, that is, there are no local minima, and the solution can be found efficiently.

In analogy to the PCA-alignment method (Verbeek et al., 2004), we augment our local potential models $\tilde{\phi}_k(\cdot)$ by a scalar offset b_k and define the corresponding objective function as

$$E(b_1 \dots b_K) = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{j=1}^{K} q_k(\mathbf{x}_m) q_j(\mathbf{x}_m) \times \\ \left((\tilde{\phi}_k(\mathbf{x}_m) + b_k) - (\tilde{\phi}_j(\mathbf{x}_m) + b_j) \right)^2, \tag{4.13}$$

or, in a slightly shorter form,

$$E(\mathbf{b}) = \frac{1}{2} \sum_{m,k,j} q_{km} q_{jm} \left(\tilde{\phi}_{km} + b_k - \tilde{\phi}_{jm} - b_j \right)^2. \tag{4.14}$$

Here, $\sum_m$ denotes a summation over the complete data set, that is, over all points from all trajectories ($M = \sum_{k=1}^{K} N_k$). Using the symmetry in $j \leftrightarrow k$ and $\sum_k q_{kn} = 1$, we split (4.14) into terms that are constant, linear, or quadratic in b_k, yielding

$$\begin{aligned} E(\mathbf{b}) &= \sum_{m,k} q_{km} \tilde{\phi}_{km}^2 - \sum_{m,j,k} q_{km} q_{jm} \tilde{\phi}_{km} \tilde{\phi}_{jm} \\ &\quad + 2 \sum_{m,k} q_{km} \tilde{\phi}_{km} b_k - 2 \sum_{m,k} q_{km} q_{jm} \tilde{\phi}_{jm} b_k \\ &\quad + \sum_{m,k} q_{km} b_k^2 - \sum_{m,k,j} q_{km} q_{jm} b_k b_j \\ &= E_0 + 2\mathbf{a}^T \mathbf{b} + \mathbf{b}^T \mathbf{H} \mathbf{b}. \end{aligned} \tag{4.15}$$

Here, we introduced E_0 as a shortcut for the terms independent of $\mathbf{b}$, the vector $\mathbf{a} \in \mathbb{R}^K$ with elements $a_k = \sum_m q_{km} \tilde{\phi}_{km} - \sum_{m,j} q_{km} q_{jm} \tilde{\phi}_{jm}$, and the Hessian matrix $\mathbf{H} \in \mathbb{R}^{K \times K}$ with elements $h_{ij} = \delta_{ij} \sum_m q_{jm} - \sum_m q_{im} q_{jm}$. The objective function is quadratic in $\mathbf{b}$, so we retrieve the optimal solution by setting the derivatives to zero, which yields the equation $\mathbf{H}\mathbf{b} = -\mathbf{a}$.

However, note that a common shift of all offsets b_k does not change the objective (4.13), which corresponds to the shift-invariance of the global potential. Therefore, the vector $(1, 1, \dots, 1)^T$ spans the nullspace of $\mathbf{H}$, and we need to use the pseudo-inverse of $\mathbf{H}$ to calculate the optimal offset vector

$$\mathbf{b}_{opt} = -\mathbf{H}^{\dagger} \mathbf{a}. \tag{4.16}$$

Compared to aligning PCA models, the case we handle here is simpler in the sense that we only need to optimise for scalar offsets b_k instead of affine mappings. On the other hand, our local potential models are non-linear, have to be estimated from relatively little data, and therefore do not extrapolate well, as will be discussed in the following section.

4.2.3 Smoothing Parameter Selection and Outlier Detection

Since we restrict ourselves to using simple NN regression for the local potential models, the only open parameter of our algorithm is σ^2, i.e., the kernel parameter used for calculating the responsibilities (4.10). A too large choice of this parameter will oversmooth the potential, because the NN regression model basically predicts a locally constant potential, but at the same time trajectories will have relatively high responsibilities for even far apart points $\mathbf{x}$ in state space.

On the other hand, a too small value of σ^2 might lead to *weakly connected trajectories*: If a particular trajectory does not make any close approach to other trajectories in the set, the quick drop-off of its responsibility implies that it will not contribute to the alignment error (based on pairs of significant responsibility), which in turn implies that its own alignment – the value of its offset – does not matter much.

The same reasoning applies to groups of trajectories that are close to each other, but have little connection to the rest of the set. For the remainder of this chapter, we will refer to such trajectories as 'outliers', since like in classical statistics we need to remove these from the training set: If their influence on the overall alignment is negligible, their own alignment can be poor, and this becomes a problem when using the output of the optimisation (4.16) to learn a global model of the potential. To avoid interference, we only include trajectories if we are sure that their offset is consistent with the rest of the data[1].

Fortunately, outliers in this sense can be detected automatically by looking for small eigenvalues of $\mathbf{H}$: In the same way as adding the same offset to all trajectories leads to a zero eigenvalue, further very small eigenvalues and the corresponding eigenvectors indicate indifference towards a shift of some subset of trajectories versus the rest of the set. In practice, we look for eigenvalues $\lambda < 10^{-8}$, and use a recursive bi-partitioning algorithm in a way that is very similar to spectral clustering (Kannan et al., 2004). We then discard all trajectories apart from those in the largest 'connected' group. Please refer to Sec. 4.2.4 for details of this step.

Finally, with these considerations in mind, we select the smoothing parameter σ^2 to match the scale of typical distances in the data sets. In all of the experiments presented

[1]It should be noted that these trajectories are not outliers in the sense of containing corrupt data and could in fact be used for further training of the model. For example, one could take a hierarchical approach, where groups of strongly connected trajectories are aligned first to form models consisting of groups of trajectories with good alignment. We can then recursively repeat the process, aligning these larger (but more weakly connected) groups until all of the data has been included.

here we used the same heuristic selection. In particular, we first calculated the distances between any two trajectories $k, j \in \{1 \dots K\}$ in the set as the distances between their closest points

$$d_{kj} = \min \left\{ \|\mathbf{x}_{kn} - \mathbf{x}_{jm}\|^2 \mid n, m \in \{1 \dots N\} \right\}, \tag{4.17}$$

and also the distances to the closest trajectory

$$d_k^{min} = \min \left\{ d_{kj} \mid j \neq k \right\}. \tag{4.18}$$

We then consider three choices for σ^2, which we refer to as ‘narrow’, ‘wide’ and ‘medium’:

$$\sigma^2_{nar} = \text{median} \left\{ d_k^{min} \mid k \in \{1 \dots K\} \right\} \tag{4.19}$$

$$\sigma^2_{wid} = \text{median} \left\{ d_{jk} \mid j, k \in \{1 \dots K\}, j \neq k \right\} \tag{4.20}$$

$$\sigma^2_{med} = \sqrt{\sigma^2_{nar} \sigma^2_{wid}}. \tag{4.21}$$

In Sec. 4.3.1 we give a comparison of the learning performance for each of these choices of σ^2 for policies of varying complexity.

4.2.4 Recursive Bi-partitioning for Outlier Detection

In the following, we describe our mechanism for detecting trajectories (or groups thereof) that we need to discard from the training set before learning a global model of the potential. To this end, similarly to spectral clustering, we look at the eigenvectors belonging to all small eigenvalues of the Hessian $\mathbf{H}$ (4.15). Let

$$\mathbf{V} = (\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_n)^T \quad \text{where} \quad \lambda_i \mathbf{v}_i = \mathbf{H} \mathbf{v}_i \,, \;\; \lambda_i < 10^{-8}. \tag{4.22}$$

That is, if $\mathbf{H}$ was calculated from 100 trajectories and has $n = 7$ small eigenvalues, $\mathbf{V}$ would be a 7×100 matrix. We then cluster the columns of $\mathbf{V}$ into two centres $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$. Since each column of $\mathbf{V}$ represents a trajectory, we effectively partition the training data into two groups whose relative potential offset has negligible influence on the alignment objective function (4.15). For both groups, we repeat the process using corresponding sub-matrices of $\mathbf{H}$. That is, we recursively split up our trajectories into groups until there is only one zero eigenvalue left in each group (corresponding to $\mathbf{v} = \mathbf{1}$, the constant shift of all trajectories in that group). The process is visualised in Fig. 4.2.

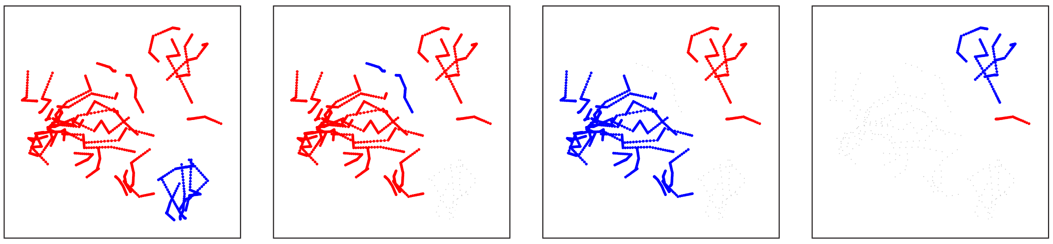


Figure 4.2: Illustration of our recursive outlier detection scheme. At any stage, we look for non-trivial small eigenvalues of the alignment Hessian, and if those exist, we split the trajectories into 2 independent groups (red and blue). From left to right: 1) top-level partitioning 2) splitting up the red group from step 1, 3) splitting the red group from step 2, 4) splitting the red group from step 3. The largest connected group consists of the blue trajectories from step 3, which we use for training the global model.

4.2.5 Learning the Global Model

After calculating optimal offsets $\mathbf{b}_{opt}$ and cleaning the data set from outliers, we can learn a global model $\tilde{\phi}(\mathbf{x})$ of the potential using any regression algorithm. Here, we choose Locally Weighted Projection Regression (LWPR) (Vijayakumar et al., 2005) because it has been demonstrated to perform well in cases where the data lies on low-dimensional manifolds in a high-dimensional space, which matches our problem of learning the potential from a set of trajectories. As the training data for LWPR, we use all non-outlier trajectories and their estimated potentials as given by the Euler integration *plus* their optimal offset, that is, the input-output tuples

$$\left\{ (\mathbf{x}_{kn}, \breve{\phi}_{kn} + b_k^{opt}) \mid k \in \mathcal{K}, n \in \{1 \dots N_k\} \right\}, \tag{4.23}$$

where $\mathcal{K}$ denotes the set of indices of non-outlier trajectories. Once we have learnt the model $\tilde{\phi}(\mathbf{x})$ of the potential, we can take derivatives to estimate the unconstrained policy $\tilde{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}} \tilde{\phi}(\mathbf{x})$. For convenience, the complete procedure is summarised in Algorithm 1.

4.3 Experiments

To explore the performance of the algorithm, experiments were performed on data from autonomous kinematic control policies (Schaal et al., 2003) applied[2] to different

[2]Since the goal of the experiments was to validate the proposed approach, we used policies known in closed form as a ground truth.

Algorithm 1 Local Potential Alignment

```
1: Estimate $\mathbf{X}_k$, $\breve{\Phi}_k, \{k = 1 \ldots K\}$ using Euler integration. Calculate $\sigma^2$.
2: Alignment:
     • Calculate prediction and responsibility of each local model $\tilde{\phi}_k$ on each data
       point $\mathbf{x}_m,\ m = 1 \ldots M$:
       $\tilde{\phi}_{km} = \tilde{\phi}_k(\mathbf{x}_m);\ q_{km} = w_k(\mathbf{x}_m)/\sum_i w_i(\mathbf{x}_m)$.
     • Construct $\mathbf{H}, \mathbf{a}$ with elements
       $h_{ij} = \delta_{ij} \sum_m q_{jm} - \sum_m q_{im} q_{jm};\ a_k = \sum_m q_{km} \tilde{\phi}_{km} - \sum_{m,j} q_{km} q_{jm} \tilde{\phi}_{jm}$.
     • Find optimal offsets $\mathbf{b}_{opt} = -\mathbf{H}^{\dagger}\mathbf{a}$.
3: Discard outliers ($\mathbf{H}$ eigenvalues, $\lambda < 10^{-8}$).
4: Train global model on data tuples $(\mathbf{x}_{kn}, \breve{\phi}_{kn} + b_k^{opt})$.
```

plants, including the whole body motion controller (WBM) of the humanoid robot ASIMO (Gienger et al., 2005). In this section, we first discuss results from an artificial toy problem controlled according to the same generic framework to illustrate the key concepts. We then discuss an example scenario in which the algorithm is used to enable ASIMO to learn a realistic bi-manual grasping task from observations of a constrained demonstrator. We then give a brief discussion on how the algorithm scales to policies in very high dimensional systems defined over 22 DOF of the ASIMO WBM controller (Gienger et al., 2005). Finally, we report the performance of the algorithm when learning from data containing a set of pathological constraints.

4.3.1 Toy Example

The toy example consists of a two-dimensional system with a policy defined by a quadratic potential, subject to discontinuously switching constraints. Specifically, the potential is given by

$$\phi(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_c)^T \mathbf{W} (\mathbf{x} - \mathbf{x}_c) \tag{4.24}$$

where $\mathbf{W}$ is some square weighting matrix which we set to $0.05\mathbf{I}$ and $\mathbf{x}_c$ is a vector defining the location of the attractor point, here chosen to be $\mathbf{x}_c = \mathbf{0}$. Data was collected by recording trajectories generated by the policy from a start state distribution X_0.

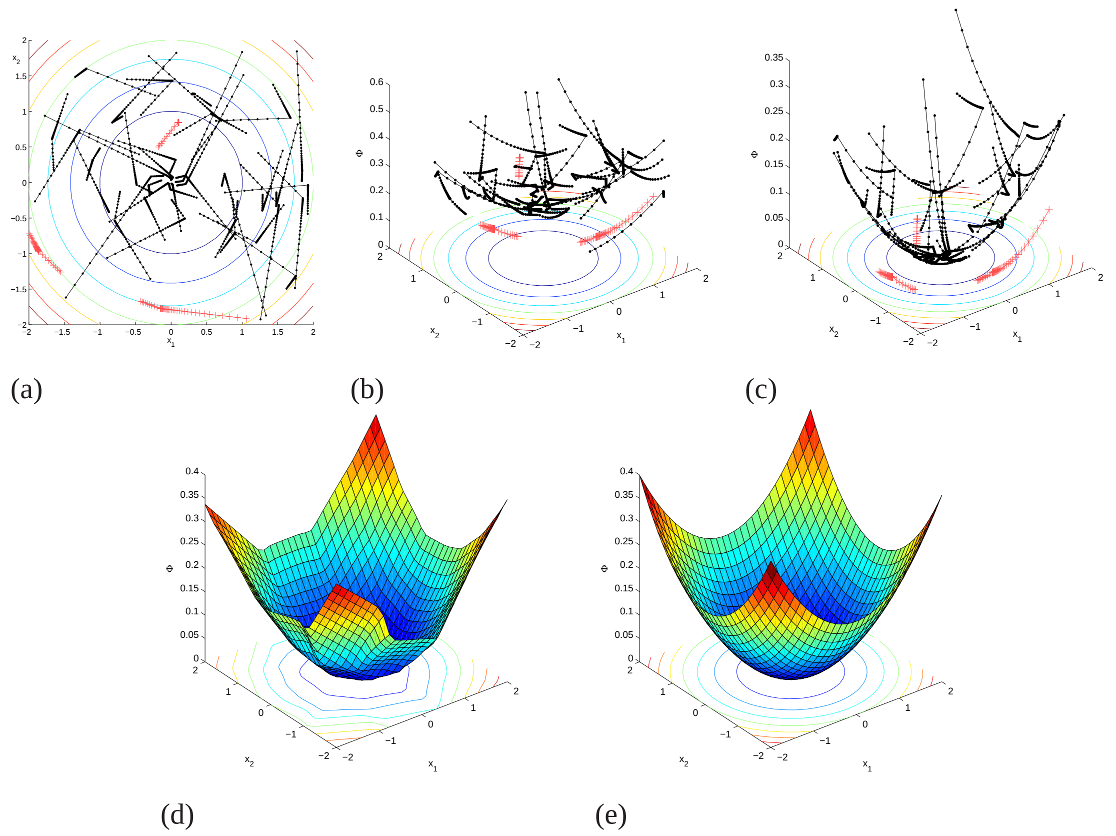


Figure 4.3: Top: (a) Toy data (trajectories (2-D) and contour of true potential. Estimated potential along the trajectories before (b) and after (c) alignment. Trajectories detected as difficult to align 'outliers' are shown by light crosses. Bottom: Learnt (d) and true (e) potential function after training on the aligned trajectories.

During the movement, the policy was subjected to random constraints

$$\mathbf{A}(\mathbf{x},t) = (\alpha_1, \alpha_2) \equiv \alpha \tag{4.25}$$

where the $\alpha_{1,2}$ were drawn from a normal distribution, $\alpha_i = \mathcal{N}(0,1)$. The constraints mean that motion is constrained in the direction orthogonal to the vector α in state space. To increase the complexity of the problem, the constraints were randomly switched during trajectories by re-sampling α twice at regular intervals during the trajectory. This switches the direction in which motion is constrained as can be seen by the sharp turns in the trajectories (ref. Fig. 4.3(a)).

Figure 4.3 shows an example of our algorithm at work for a set of $K = 40$ trajectories of length $N = 40$ for the toy system. The raw data, as a set of trajectories through the two-dimensional state space, is shown in panel (a), whereas panel (b) additionally depicts the local potential models as estimated from the Euler integration prior to alignment. Each local model has an arbitrary offset against the true potential so there are inconsistencies between the predictions from each local model. Figure 4.3(c) shows the trajectories after alignment, already revealing the structure of the parabola.

At this point, the outlier detection scheme has identified three trajectories as being weakly connected to the remaining set. In Fig. 4.3(a), we can see that the outliers are indeed the only trajectories that do not have any intersection with neighbouring trajectories. At the ‘narrow’ length scale determined by the smoothing parameter (4.19), they are hard to align properly, and need to be discarded before learning the global model. Finally, Fig. 4.3(d) shows the global model $f(\mathbf{x})$ of the potential that was trained on the aligned trajectories, which is clearly a good approximation of the true parabolic potential shown in Fig. 4.3(e).

For a more thorough evaluation, we repeated this experiment on 100 data sets and evaluated

- the nMSE of the **aligned potential**, which measures the difference between $\breve{\phi}_{kn} + b_k$ and the true potential ϕ, i.e.,

$$E_{align}[\mathbf{b}] = \frac{1}{N\sigma_\phi^2} \sum_{n=1}^{N} \left(\breve{\phi}(\mathbf{x}_n) - \phi(\mathbf{x}_n) - \nu\right)^2, \quad \nu = \frac{1}{N} \sum_{n=1}^{N} \left(\breve{\phi}(\mathbf{x}_n) - \phi(\mathbf{x}_n)\right), \quad (4.26)$$

  where the notation $\breve{\phi}(\mathbf{x}_n)$ is understood to already include the proper offset, that is, $\breve{\phi}(\mathbf{x}_n) = \breve{\phi}_{kn'} + b_k$ and where σ_ϕ^2 denotes the variance of the true potential;

- the nMSE of the **learnt potential**, measuring the difference between $\tilde{\phi}(\cdot)$ and $\phi(\cdot)$, i.e.,

$$E_{pot}[\tilde{\phi}] = \frac{1}{N\sigma_\phi^2} \sum_{n=1}^{N} \left(\tilde{\phi}(\mathbf{x}_n) - \phi(\mathbf{x}_n) - \mu\right)^2, \quad \mu = \frac{1}{N} \sum_{n=1}^{N} \left(\tilde{\phi}(\mathbf{x}_n) - \phi(\mathbf{x}_n)\right), \quad (4.27)$$

- the normalised **unconstrained policy error**, (3.16), quantifying the difference between $\tilde{\pi} = \nabla\tilde{\phi}$ and $\pi = \nabla\phi$;

- the normalised **constrained policy error**, (3.17), which is the discrepancy between $\mathbf{N}\tilde{\pi}$ and $\mathbf{N}\pi$, and finally;

- the percentage of trajectories discarded as outliers

on a subsample of the data held out for testing. Please note that in (4.26) and (4.27) we subtract the mean difference ν and μ, respectively, between the two quantities to

Setup	σ^2	Potential nMSE	nUPE	nCPE	Disc. (%)
Parabola	narrow	0.0052 ± 0.0024	0.0486 ± 0.0211	0.0235 ± 0.0092	17.55 ± 15.96
$K = 40$	medium	0.0195 ± 0.0203	0.0859 ± 0.0486	0.0224 ± 0.0074	0.48 ± 1.11
$N = 40$	wide	0.3143 ± 0.1045	0.5758 ± 0.2726	0.1135 ± 0.0371	0 ± 0
Sinusoidal	narrow	0.0026 ± 0.0019	0.1275 ± 0.1125	0.0535 ± 0.0353	50.18 ± 14.37
$K = 40$	medium	0.0522 ± 0.0645	0.1399 ± 0.0422	0.0376 ± 0.0097	1.03 ± 3.99
$N = 40$	wide	0.5670 ± 0.1363	0.8373 ± 0.2188	0.2464 ± 0.0638	0 ± 0
Sinusoidal	narrow	0.0014 ± 0.0004	0.0657 ± 0.0142	0.0308 ± 0.0065	25.46 ± 11.42
$K = 100$	medium	0.0019 ± 0.0017	0.0628 ± 0.0089	0.0284 ± 0.0044	1.25 ± 3.33
$N = 100$	wide	0.2137 ± 0.1000	0.4262 ± 0.1367	0.1554 ± 0.0483	0 ± 0

Table 4.1: Error and outlier statistics (mean$\pm$std.dev. over 100 data sets) for the experiment on 2-D toy data. Here, the 'narrow', 'medium' and 'wide' choices of σ^2 were calculated according to (4.19), (4.21) and (4.20), respectively. For brevity, we did not include the figures for the alignment nMSE. These were only marginally different from the potential nMSE.

remove the irrelevant global offset of the potentials. We did so for our three different choices of σ^2 given in (4.19)-(4.21). We also repeated the experiment using a sinusoidal potential function

$$\phi(\mathbf{x}) = 0.1 \sin(x_1) \cos(x_2) \tag{4.28}$$

with the same amount of data, as well as while using $K = 100$ trajectories of length $N = 100$ for each data set.

Table 4.1 summarises the results. Firstly, we can see that the 'wide' choice for σ^2 leads to large error values which are due to over-smoothing. Using the narrow σ^2, we retrieve very small errors at the cost of discarding quite a lot of trajectories[3], and the medium choice seems to strike a reasonable balance especially with respect to the nUPE and nCPE statistics. Further to this, Fig. 4.4(a) depicts how the nUPE and nCPE evolve with increasing size of the training set, showing a smooth decline (please note the logarithmic scale).

Secondly, when comparing the results for the parabolic and sinusoidal potentials, we can see that the latter, more complex potential (with multiple sinks) requires much more data. With only 40 trajectories and 40 points each, most of the data sets are too

[3]Please note that we also discard the outliers for evaluating the error statistics – we can hardly expect to observe good performance in regions where the learnt model $\tilde{\phi}(\mathbf{x})$ has seen no data.

disrupted to learn a reasonable potential model. While at the narrow length scale (4th row), on average more than half of the data set is discarded, even the medium length scale (5th row) over-smooths the subtleties of the underlying potential.

Finally, the nCPE is always much lower than the nUPE, which follows naturally when training on data containing those very movement constraints. Still, with a reasonable amount of data, even the unconstrained policy can be modelled with remarkable accuracy.

As a final test, we also performed experiments to assess the noise robustness of the proposed approach. For this, we again used data from the quadratic potential and but this time contaminated the observed states $\mathbf{x}_n$ with Gaussian noise, the scale of which we varied to match up to 20% of the scale of the data. The resulting nUPE roughly follows the noise level, as is plotted in Fig. 4.4(b).

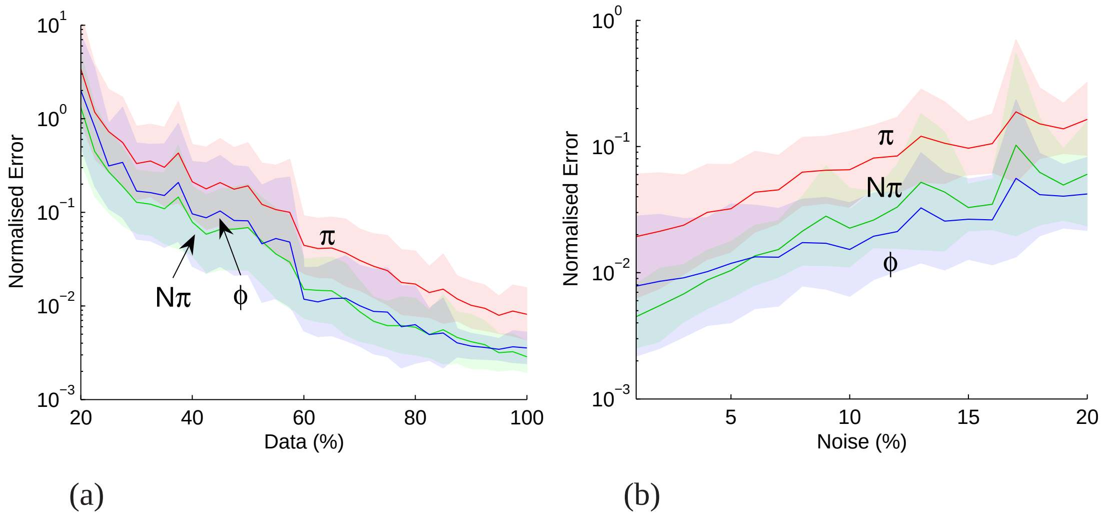


Figure 4.4: Learning performance on the quadratic potential (4.24) with varying data set sizes and noise levels. (a) Potential nMSE, nCPE and nUPE versus data set size as a percentage of the full $K = 40$ trajectories of length $N = 40$. (b) Potential nMSE, nCPE and nUPE for increasing noise levels in the observed $\mathbf{x}_n$.

4.3.2 Reaching for a Ball

The two goals of our second set of experiments were (i) to characterise how well the algorithm scaled to more complex, realistic constraints and policies and (ii) to assess how well the learnt policies generalised over different constraints. For this, we set up a demo scenario in which a set of trajectories demonstrating the task of reaching

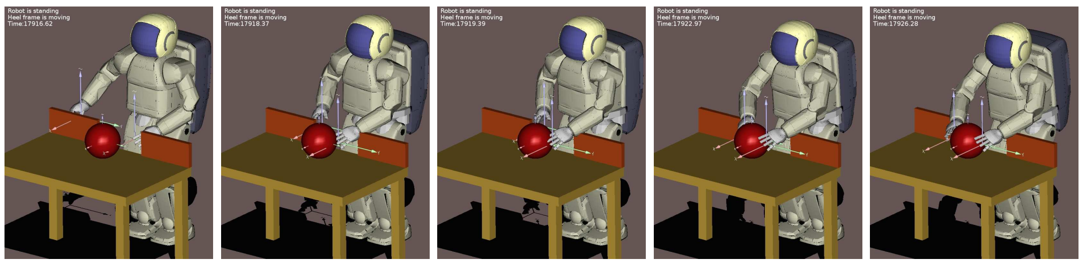


Figure 4.5: Example constrained trajectory used as training data in the ball-reaching experiment. Starting with hands at the sides, the demonstrator robot reaches between the barriers to get the ball. Note that the width of the gap in the barriers was randomly altered for each trajectory recorded.

for a ball on a table were given. Furthermore, it was assumed that trajectories were recorded under different contexts where different constraints applied. The goal was then to uncover a policy that both accurately reproduced the demonstrated behaviour and furthermore *generalised* to novel contexts with unseen constraints.

The example scenario was implemented using the whole body motion (WBM) controller of the 27-DOF humanoid robot ASIMO (for details see Gienger et al. 2005). For this, data was recorded from a 'demonstrator' robot that, for ease of comparison with the 2-D system, was defined by the same quadratic potential (4.24), i.e.,

$$\pi(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_c)^T\mathbf{W}(\mathbf{x}-\mathbf{x}_c), \tag{4.29}$$

this time with the target point $\mathbf{x}_c \in \mathbb{R}^6$ corresponding to a grasping position, with the two hands positioned on either side of the ball. The state-space of the policy was defined as the Cartesian position of the two hands, corresponding to 6 DOFs[4] (hereafter, the 'task space'). Following the policy (4.29) with this set of parameters, the demonstrator was able to reach the ball under each of the constraints considered in this experiment (see below). Inverse kinematics via the WBM controller was used to map the desired task space policy motion into the appropriate joint-space velocity commands for sending to the robot.

The demonstrator's movements were constrained by the presence of a barrier on the table with a gap in it, placed so that the demonstrator robot had to reach through the gap to get the ball (ref. Fig. 4.5). The barriers acted as inequality constraints on each of the hands so that motion in the direction normal to the barrier surface was

[4] 3 DOFs per hand $\times$ 2 hands.

prevented if a hand came too close. Specifically, the constraints took the form

$$\mathbf{A}(\mathbf{x},t) = \left(\begin{array}{c|c} \mathbf{A}_{[1,1]} & \mathbf{0} \\ \hline \mathbf{A}_{[1,2]} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}_{[2,1]} \\ \hline \mathbf{0} & \mathbf{A}_{[2,2]} \end{array}\right) \tag{4.30}$$

where

$$\begin{aligned} \mathbf{A}_{[i,j]}(\mathbf{x},t) &= \hat{\mathbf{n}}_j^T \;; \quad d_{i,j} \le d_{min} \;\; and \;\; \hat{\mathbf{u}}_{[i]}^T \hat{\mathbf{n}}_j > 0 \\ \mathbf{A}_{[i,j]}(\mathbf{x},t) &= \mathbf{0} \quad ; \quad otherwise. \end{aligned}$$

Here, $d_{i,j}$ is the distance of the ith hand (where $i \in \{1,2\}$, i.e. left and right hands respectively) to the closest point on the jth barrier (where $j \in \{1,2\}$, i.e. left and right barriers respectively), $\hat{\mathbf{n}}_j \in \mathbb{R}^3$ is the normal to the barrier surface[5] at that point and $\hat{\mathbf{u}}_{[i]} \in \mathbb{R}^3$ is the normalised command for the ith hand (i.e. the ith 3-vector block of the command vector $\mathbf{u}$ corresponding to that hand; for example, for the right hand ($i=2$) this was $\mathbf{u}_{[2]} \equiv (u_4,u_5,u_6)^T$ with $\hat{\mathbf{u}}_{[2]} = \mathbf{u}_{[2]}/|\mathbf{u}_{[2]}|$). Here, the full constraint matrix $\mathbf{A}(\mathbf{x},t) \in \mathbb{R}^{4\times 6}$ was constructed by assigning 3-vectors to the appropriate matrix blocks $\mathbf{A}_{[i,j]}$, according to the system state. For example, if the left hand ($i=1$) approached the left barrier ($j=1$) to a distance of $d_{1,1} < d_{min}$, and if the next commanded movement would bring the hand toward that barrier (i.e. $\hat{\mathbf{u}}_{[1]}^T \hat{\mathbf{n}}_1 > 0$), then the elements of the constraint matrix corresponding to that hand/barrier pair were updated (in this example the first row of the matrix would be updated, $\mathbf{A}_{1,:} = (\hat{\mathbf{n}}_1^T,0,0,0)$, constraining the left hand). Note that under this setup the constraints are highly nonlinear (due to the complex dependence on state) and have discontinuously switching dimensionality (i.e. the rank of $\mathbf{A}(\mathbf{x},t)$ switches) when either of the hands approaches or recedes from the barrier.

Data was collected by recording $K=100$ trajectories of length $2s$ at 50 Hz, (i.e. $N=100$ points per trajectory) from the demonstrator following the policy (5.10) under the constraints (4.30). Start states were sampled from a Gaussian distribution over joint configurations $\mathbf{q} \sim \mathcal{N}(\mathbf{q}_0, 0.1\mathbf{I})$ (where $\mathbf{q}_0$ corresponds to the default standing position) and using forward kinematics to calculate the corresponding hand positions. The joint vector $\mathbf{q}$ was clipped where necessary to avoid joint limits and self collisions, and to ensure the start postures looked natural.

[5]Note that in order to ensure smooth, natural-looking trajectories the barriers were modelled as planes with smooth ‘swept-sphere’ edges, similar to those described in Sugiura et al. (2007).

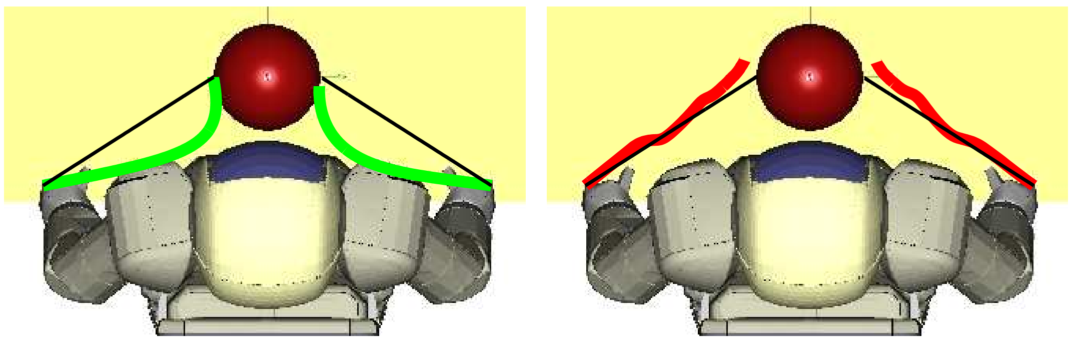


Figure 4.6: Unconstrained reaching movement for the expert policy (black), the policy learnt with the naive approach (green) and that learnt with the policy alignment algorithm (red).

Constraint	Naive	Pot. Align.
Training	0.1298 ± 0.0113	0.1691 ± 0.0289
Unseen Barrier	0.5108 ± 0.0327	0.2104 ± 0.0357
Unconstrained	0.8766 ± 0.0589	0.2277 ± 0.0386

Table 4.2: Constrained policy nMSEfor unseen constraints on the ball-grasping task. Values are mean±s.d. over 50 data sets.

For each trajectory, the constraints were varied by randomly changing the width of the gap in the barriers. The gap widths were sampled from a Gaussian distribution $d_{gap} \sim \mathcal{N}(\mu_{gap}, \sigma_{gap})$ where $\mu_{gap} = 0.25m$, $\sigma_{gap} = 0.1m$ and the diameter of the ball was $0.15m$. The hand-barrier distance at which the constraints came into force was fixed at $d_{min} = 0.05m$. Fig. 4.5 shows an example trajectory under this set-up.

We used our algorithm to perform learning on 50 such data sets using the 'narrow' choice of smoothing parameter σ^2. For comparison, we also repeated the experiment on the same data, using a naive approach that learnt $\tilde{\pi}_{naive} : \mathbf{x} \rightarrow \mathbf{u} \in \mathbb{R}^n \mapsto \mathbb{R}^n$ by training directly on the tuples $(\mathbf{x}_i, \mathbf{u}_i), i = 1, \ldots K \times N$ and used LWPR to learn the global model. This is in contrast to the proposed alignment scheme where we learn the 1-dimensional potential function and use the gradient of the learnt function as the policy prediction.

For this task, our algorithm achieved a very low alignment error of $6.95 \pm 0.09 \times 10^{-4}$, with $0.48 \pm 0.84\%$ of the trajectories discarded, resulting in an nMSE in the learnt potential of $7.85 \pm 0.56 \times 10^{-4}$ (mean±s.d. over 50 data sets). In Table 4.2

we give the errors in predicting the policy subject to (i) the training data constraints (nCPE), (ii) no constraints (nUPE), and (iii) a novel constraint, unseen in the training data, on a set of test data. For the latter, a barrier was placed centrally between the robot and the ball, so that the robot had to reach around the barrier to reach the ball.

The remarkably low alignment error can be attributed to the fact that in most of the observations the grasping task was achieved successfully despite the constraints forcing the hands to take alternative routes to the ball. This meant many of the trajectories closely approached the minimum of the potential, making the alignment easier around this point. This is further indicated by the low percentage of trajectories discarded.

The key result, however, can be seen by examining the policy errors (ref. Table 4.2). Comparing the two approaches, both achieve a similar nCPE, with the naive approach in fact performing slightly better. This indicates that the two methods both do equally well in modelling the movement under the training constraint to approximately the same level of accuracy.

However, when comparing the errors for the unconstrained policy, and the policy subject to the unseen constraint, a different picture emerges. Using the model learnt by the alignment approach, the unconstrained policy predictions, and the predictions under the unseen constraint, maintain a similar level of error to that of the constrained policy. In stark contrast to this, the naive approach fares very poorly, with a large jump in error when predicting the policy under the new barrier constraint and when predicting the unconstrained behaviour.

The difference in the two approaches is highlighted if we compare trajectories generated by the two policies. In Fig. 4.6 we show example trajectories for the unconstrained reaching movement produced by the expert (black), and the policies learnt by (i) the naive approach (green), and (ii) the alignment approach (red). In the former the hands take a curved path to the ball, reproducing the average behaviour of the demonstrated (constrained) trajectories – the naive method is unable to extract the underlying task (policy) from the observed paths around the obstacles. In contrast, the policy learnt with the alignment approach better predicts the unconstrained policy, enabling it to take a direct route to the ball that closely matches that of the expert (Fig. 4.6, right).

4.3.3 Learning from High-dimensional Joint-space Data

In our next experiment, we tested the scalability of our approach for learning in very high dimensions. For this, we again used the quadratic potential (4.24) where now the state vector $\mathbf{x}$ corresponded to the 22-dimensional joint configuration of the upper body of the ASIMO humanoid robot (ref. Fig. 1.1). In this case the policy (4.24) represents an attractor in joint space that pulls the robot into a desired posture at $\mathbf{x}_c$. For the experiments, $\mathbf{x}_c$ was chosen to correspond to a reaching posture with both arms outstretched and we chose $\mathbf{W} = 0.05\mathbf{I}$.

In this experiment, the policy was constrained such that in each trajectory one of the hands of the robot was constrained to lie in a plane of random orientation. Specifically, the constraint matrix $\mathbf{A}(\mathbf{x},t) \in \mathbb{R}^{1\times 22}$, took the form

$$\mathbf{A}(\mathbf{x},t) = \hat{\mathbf{n}}_s^T \mathbf{J}_i(\mathbf{x}) \tag{4.31}$$

where $\hat{\mathbf{n}}_s \in \mathbb{R}^3$ is the normal to the plane and $\mathbf{J}_i(\mathbf{x}) \in \mathbb{R}^{2\times 27}$ is the Jacobian mapping from joint-space to the Cartesian ith hand velocity (with $i \in \{1,2\}$, i.e. left and right hands respectively), The constraints were alternated between the left and right hands for successive trajectories, so that the left hand was constrained for half of the trajectories, and the right hand was constrained for the remainder. The plane orientation $\hat{\mathbf{n}}_s$ was drawn from a uniform random distribution. Similar constraints such as these occur in a variety of behaviours where contact must be maintained with a surface; for example, when writing on a whiteboard or when wiping a window (Park and Khatib, 2006).

We ran the experiment on 50 data sets of $K = 100$ trajectories of length $N = 100$, with start states selected using the same process as described in the preceding section. Using the narrow setting of the smoothing parameter the algorithm achieved an alignment error of $1.6 \pm 0.3 \times 10^{-3}$ with just $0.02 \pm 0.14\%$ of the trajectories discarded. Learning on this data with LWPR, we achieved an nMSE in the learnt potential of $1.5 \pm 0.4 \times 10^{-3}$, nCPE of 0.065 ± 0.014 and nUPE of 0.157 ± 0.047. We consider this to be remarkably good performance given the high dimensionality of the input space and the relatively small size of the data set.

4.3.4 Degeneracy due to Constraints

In our final set of experiments, we briefly explore the limitations in performance of our algorithm for reconstructing the (unconstrained) policy when, due to the particular set of constraints found in the data, there is degeneracy in the possible solutions (see discussion in Sec. 3.3.2). We found an illustrative example of this can be found when considering the movement of a constrained planar three-link arm.

The experimental set up was as follows. Data was collected from a simulated planar arm with revolute joints and unit link lengths, moving according to the quadratic potential (4.24) (with $\mathbf{x}_c = \mathbf{0}$ and $\mathbf{W} = 0.05\mathbf{I}$) from a random distribution of start states. The movement of the arm was restricted by constraining the end-effector to move along a line. Mathematically the constraint matrix was

$$\mathbf{A}(\mathbf{x},t) = \hat{\mathbf{n}}^T \mathbf{J}_{hand}(\mathbf{x}) \tag{4.32}$$

where $\hat{\mathbf{n}}$ is a unit vector normal to the hand-space plane and $\mathbf{J}_{hand}(\mathbf{x})$ is the hand Jacobian. The constraint was varied by altering the orientation of the plane by drawing $\hat{\mathbf{n}}$ from a uniform random distribution $U_{\hat{\mathbf{n}}}$ at the start of each trajectory.

We ran experiments on 50 such data sets each containing $K = 100$ trajectories of length $N = 100$. For this learning problem, the algorithm achieved nUPE of 0.3524 ± 0.1626 and nCPE of 0.0455 ± 0.0276. The nMSE in the learnt potential was 0.1739 ± 0.1424 with $10.28 \pm 8.25\%$ trajectories discarded. In comparison the naive approach to learning achieved nUPE of 0.8008 ± 0.0274 and nCPE of 0.0105 ± 0.0023.

The reason for the comparatively high nUPE here becomes clear if we analyse the effect of the constraints on the movement of the arm (see Fig. 4.7). In Fig. 4.7(a) the training data trajectories are plotted over the three joints of the arm. It can be seen that the trajectories do not reach the point attractor at $\mathbf{x} = \mathbf{0}$, but rather move to a line in joint space (shown in black). This 'line attractor' represents the minimum of the potential that can be reached without breaking the constraints. No trajectories travel in the direction parallel to this line. Furthermore, away from this line there are few points where trajectories come close to one another or intersect. The effect of this is that the algorithm gets little or no information about how the potential changes in the direction parallel to the line.

This is confirmed by comparing how the nUPE and nCPE change as we move along the line attractor, and radially outward from it. In Fig. 4.7 we show the potential nMSE, nUPE and nCPE on data contained within different regions of the state space.

First, we evaluated the error on data points contained between two planes normal to the line attractor at distance d from the point attractor $\mathbf{x} = \mathbf{0}$ (Fig 4.7(b), dashed lines), and plotted it with increasing d (Fig 4.7(d)). We can see that close to $\mathbf{x} = \mathbf{0}$, the potential nMSE and nUPE start low but increase rapidly for large d. On the other hand the nCPE stays approximately constant over the entire set.

Second, we looked at how the errors change as we move radially outward. For this, we evaluated errors on data contained within a cylinder of radius r centred on the line

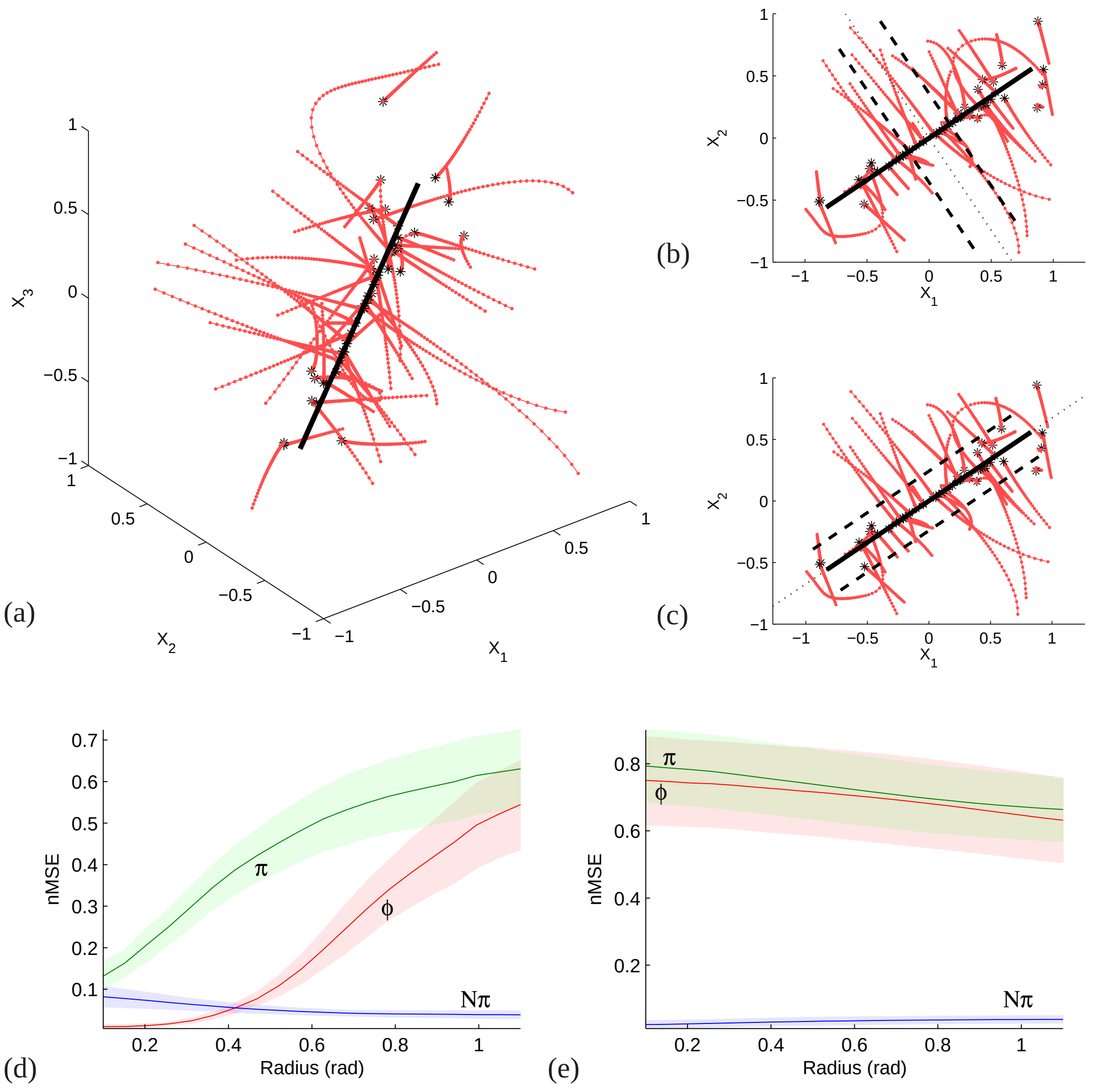


Figure 4.7: (a) Trajectories in state-space for the three link arm subject to random planar constraints on the hand. (b) and (c) show projections onto the first two joints of the arm, and also indicate the line attractor (solid black line). We sampled the nMSE at increasing distances along the line (b) and radially outward from it (c). Plots (d) and (e) depict the cumulative nMSE of the potential ϕ, policy π, and constrained policy $(\mathbf{N}\pi)$ as a function of the distance measures from (b) and (c), respectively.

attractor (Fig 4.7(c), dashed lines). Fig 4.7(e) shows the change in error with increasing radius r. Again the nCPE remains constant. This time, however, the potential nMSE and nUPE are high even at small r. This indicates that the points at the two ends of the line are contributing most of the error.

Clearly in this example, the adverse constraints in the training data prevent our algorithm from fully reconstructing the unconstrained policy. The constraints prevent motion parallel to the line attractor so we cannot recover the form of the potential along that direction. However, the good performance in terms of the nCPE indicates that, at the very least, the algorithm is able to reconstruct the policy under the same constraints despite these adverse conditions.

4.4 Conclusion

In this chapter, we explored the learning of policies from constrained motion data, for the special case of potential-based policies. We gave a formal definition of a potential-based policy in terms of the curl properties of the vector field described by the policy, and characterised the kinds of behaviour such a policy may encode; that is, discrete movements such as goal-oriented reaching.

We then went on to discuss why this class of policy is amenable to constraint-consistent learning. We noted the special property of potential-based policies; namely that the projected gradient vectors give us directional derivative information that can be used to recover the shape of the potential. We then proposed a novel method to exploit this property, allowing us to learn kinematic policies subject to stationary constraints indirectly, by modelling the underlying potential function. The proposed method is fast and data-efficient, and it scales to complex constraints in high-dimensional movement systems. The core ingredient is an algorithm for *aligning local models of the potential*, which leads to a convex optimisation problem.

Given the difficulties in learning that we predicted in Ch. 3, this method performs remarkably well. Ultimately, the ability to learn the potential depends on the constraints: Given a pathological set of constraints, one can never hope to recover the potential. However, using this method, motion data under different constraints can be combined to learn a potential that is consistent with the observations. With a reasonably rich set of constraints, we can recover the unconstrained policy with high accuracy, and we can *generalise to predict behaviour under different constraints*.

Having proven then, the principle that learning policies from constrained motion data is feasible, at least for this restricted class of problems (i.e. kinematic trajectory data from potential-based policies), we are now in a position to look for ways to tackle more generic policy learning problems. In particular, it is desirable that we remove the restriction to potential-based policies, to enable us to learn more generic movements, including rotational or periodic movements; for example, stirring soup or turning a crank or pedal. Furthermore, in order to extend the method to more generic, non-kinematic policies (e.g. force-based policies) and non-stationary constraints (ref. Sec. 3.3.3) it will be necessary to remove the need to bootstrap the learning along observed trajectories (ref. Sec. 4.2.1). In the next chapter, we will see how re-examining the problem in terms of the objective functions used for learning leads to an alternative approach to learning that no longer suffers from the restrictions of the potential-based approach.

Chapter 5

Learning Generic Policies from Constrained Motion

5.1 Introduction

In the preceding chapter we saw how, for the special case of kinematic, potential-based policies it is possible to accurately learn the unconstrained policy without need for explicit knowledge of the constraints. This was done using trajectory data collected under different constraints to find a model that was consistent with the observations by seeking the underlying potential function. We saw that this approach is a great improvement over the standard approach to direct policy learning, out-performing direct regression in a number of experiments. However, it still suffers from several limitations. Essentially these are due to the assumptions (i) that the policy is potential-based (i.e. irrotational in the sense of having zero curl; ref. Sec. 4.1.1) (ii) the data is kinematic (i.e. $\mathbf{u} = \dot{\mathbf{x}}$) and (iii) the data takes the form of trajectories through the state-space.

In this chapter, we explore ways to remove these limitations and learn generic policies from observed state-action pairs for stationary constraint systems. We will show that it is still possible to learn a good model of the policy π, without need for explicit knowledge of the constraints $\mathbf{N}(\mathbf{x},t)$, and without the need for the restrictive assumptions outlined above. In order to do this, the key to our approach will be to reconsider the *risk function* used for modelling the policy.

An outline of the chapter is as follows. First we will look at different risk functions that may be used to optimise our model with respect to the data given. We will assess the suitability of several candidate error measures, including the standard risk, the

UPE and CPE (as defined in Sec. 3.4.2). We will then go on to propose a novel risk function for learning, based on *maximising consistency* with the constraints based on projections of the unconstrained policy vector. We will outline how this new functional can be used in combination with two example policy models, that is parametric and local linear policy models. Finally we will test the performance of this approach on several constrained systems of varying size and complexity, similar those described in the preceding chapter.

5.2 Learning Policies by Minimising Inconsistency

In this section we consider several candidate risk functions that could be used for learning and assess their suitability with respect to the data we are assumed given. We will then propose a novel risk function (Howard et al., 2009b,a) that both satisfies our original assumptions, and promises to be effective for learning from variable constraint data.

5.2.1 Optimisation of the Standard Risk, UPE and CPE

As outlined in Sec. 3.3, throughout this thesis we target problems where we are given data in the form of tuples $(\mathbf{x}_n, \mathbf{u}_n)$ of observed states and constrained actions. We assume that all commands $\mathbf{u}$ are generated from the same underlying policy $\pi(\mathbf{x})$, which for a particular observation might have been constrained. For the stationary constraint problem (ref. Sec. 3.3.2), this means that we observe $\mathbf{u}_n = \mathbf{N}_n \pi(\mathbf{x}_n)$ for some projection matrix $\mathbf{N}_n$. We assume that the projection matrix for any given observation is not explicitly known, i.e. our data is unlabelled with respect to the constraints in force at the time of observation.

Given this data, the first possibility that springs to mind is to perform direct least-squares regression for learning. In this approach one would attempt to estimate the policy $\tilde{\pi}(\cdot)$ by minimising the *standard risk*

$$E_{direct}[\tilde{\pi}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \tilde{\pi}(\mathbf{x}_n)\|^2. \tag{5.1}$$

As already mentioned in Ch. 3, this is an effective approach for learning from unconstrained data (ref. Sec. 3.3.1) or data where the same constraint appears in all observations (i.e. the constraint matrix $\mathbf{A}(\mathbf{x})$ is the same static function of state for all observations). In the former case, one would obtain the best fit to the *unconstrained policy*, and in the latter one would find the best fit to the *constrained policy under that particular set of constraints*. For example, if one had several observations of a system opening some particular door and in every observation the door was the same, then optimisation of (5.1) would be effective for learning.

The problem with this approach, however, is that it cannot handle data where commands are observed under variable constraints. As also mentioned in Sec. 3.3.2, if we consider an example where multiple observations are given under different constraints, optimisation of (5.1) would result in a naive averaging of commands from different circumstances (cf. Fig. 3.4, centre). In terms of our door opening example, if we observed the agent opening a new door and attempted to incorporate that into our policy model, we would either get the average door opening action, or have to start a new policy model for the new door. We can therefore rule out (5.1) for learning in this setting, since it does not meet our requirements for accuracy and generalisation.

An alternative approach then, might be to directly target the error measures that we use to measure performance (ref. Sec. 3.4.2). For example, we could attempt to optimise our model with respect to the *unconstrained policy error*

$$E_{upe}[\tilde{\pi}] = \sum_{n=1}^{N} \|\pi_n - \tilde{\pi}(\mathbf{x}_n)\|^2. \tag{5.2}$$

Optimising (5.2) would clearly give us the best fit to the policy, and in the case that no constraints were in force, would correspond to direct regression on the policy observations. This would also satisfy our accuracy and generalisation requirements since, as discussed in Sec. 3.4.2, we could project our policy model in any arbitrary way and still hope to get a good nCPE. However, the problem here is that, by assumption, we do not have access to samples of the (unconstrained) policy $\pi_n = \pi(\mathbf{x}_n)$ so (5.2) is not available for learning.

Alternatively, we could try optimising for the *constrained policy error*

$$E_{cpe}[\tilde{\pi}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{N}_n\tilde{\pi}(\mathbf{x}_n)\|^2. \tag{5.3}$$

Optimising (5.3) would give the fit that is *most consistent with the constrained observations*, i.e., it would minimise the error in the components of the policy along the dimensions left unconstrained in the observations. Compared to optimising the UPE, it would not give such a tight fit, since the projections $\mathbf{N}_n$ eliminate components of the policy that could potentially contain errors. However, despite this, we could still hope to get good generalisation given sufficient variability in the constraints.

Unfortunately, there are a number of problems with using (5.3) for learning. In most problems of interest, while it may be relatively easy to identify *when a constraint is in force*, it is usually much more difficult to determine *what that constraint is*. In most cases constraints are not directly observable and there is often *ambiguity* in what features of motion are due to constraints and what are implicit in the policy itself.

For example, consider a contact control scenario such as wiping a window. There, we can identify the surface of the window as an *environmental constraint*[1] preventing the wiping hand from penetrating the surface. We may also identify a task constraint preventing the hand from lifting from the surface, since contact must be maintained for successful wiping. This is one reasonable analysis of the system and, assuming it to be correct, we may go on to estimate the constraint. For example, we may model the shape of the surface being wiped, assume constraints of a particular form and then estimate the corresponding projections $\mathbf{N}_n$. This would then allow us to use (5.3) to estimate the policy.

However, the difficulty here is that it is not clear how, in general, such identification of the constraints can be done. For example, while the above analysis seems relatively straight-forward to the expert human, it requires a rather detailed understanding of the interaction between wiping hand and surface, and of the requirements of the wiping task. Such an analysis, if available, would considerably ease the learning problem (allowing us to use (5.3) for higher-accuracy predictions). However, it is not clear at the present time how such an analysis may be automated in a simple way.

An additional, and perhaps more critical, problem, however, is that of ambiguities in the observations that may lead to different analyses and predictions of the true constraints. For example, it may be that the *unconstrained policy itself* exactly encodes a wiping movement parallel to the surface, so that the presence of the surface is incidental. Alternatively, there could be an *additional task constraint* applied that prevents the hand from pressing hard against the surface. Note that we cannot directly determine which is the correct analysis simply by observing the given movement: If the window surface (environmental constraint) was removed in both of these cases the wiping would still appear to go on exactly as before. In this example then, there is ambiguity

[1]Note that would in fact be an inequality constraint since only movement into the surface is restricted, while movement away is unconstrained.

in what features of movement are due to the policy, what are due to the constraints, and exactly what constraints (if any) are in force.

To avoid these problems, in this thesis we take a different approach. In Ch. 4 we saw how, for the special case of potential-based policies, it was possible to achieve high modelling accuracy *without explicit knowledge of the constraints* (i.e. without using the projections $\mathbf{N}_n$). In this chapter then, we will look for a similar approach, again with the assumption that the constraints $\mathbf{N}_n$ are unknown. While this may result in poorer accuracy as compared to methods that explicitly use knowledge of the projections, it has the great benefit that the problems of modelling the constraints in the data are avoided. To do this, in the next section we will look at an alternative risk function that satisfies our assumptions while still promising high accuracy and good generalisation over constraints.

5.2.2 Optimisation of the Inconsistency

Having ruled out the use of (5.2)-(5.3) for learning in this setting we must look for alternative approaches. Our aim is to try to estimate a policy $\tilde{\pi}(\cdot)$ that is *consistent* with our observed $\mathbf{u}_n$, only using quantities that we can derive from the data. That is, we wish to reconstruct the policy, knowing that it may be projected in some way by the constraints. At this point a key observation can be made: in order to uncover the unconstrained policy we must find a policy model that can be *projected in such a way that the observed commands are recovered*. In other words, we require

$$\mathbf{u}(\mathbf{x}) := \mathbf{P}\pi(\mathbf{x})$$

for an appropriate projection matrix $\mathbf{P}$, that either projects onto the same space as the (unknown) $\mathbf{N}(\mathbf{x})$ (i.e. the image of $\mathbf{N}$), or an (even smaller) subspace of that. One such projection, which we know to lie within this subspace, is the 1-D projection onto the observed command itself, that is $\mathbf{P} = \hat{\mathbf{u}}\hat{\mathbf{u}}^T$, with $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|$ (ref. Fig. 5.1). Furthermore, since $\mathbf{u}$ is given, we have all the information we need to calculate this projection and use it for learning, neatly side-stepping the need to explicitly model the full constraint matrix $\mathbf{N}$.

With this as motivation, we propose to replace $\mathbf{N}_n$ in (5.3) by a projection onto $\mathbf{u}_n$ and minimise the *inconsistency* which we define as the functional

$$E_i[\tilde{\pi}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \hat{\mathbf{u}}_n\hat{\mathbf{u}}_n^T\tilde{\pi}(\mathbf{x}_n)\|^2 = \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T\tilde{\pi}(\mathbf{x}_n)\right)^2 \tag{5.4}$$

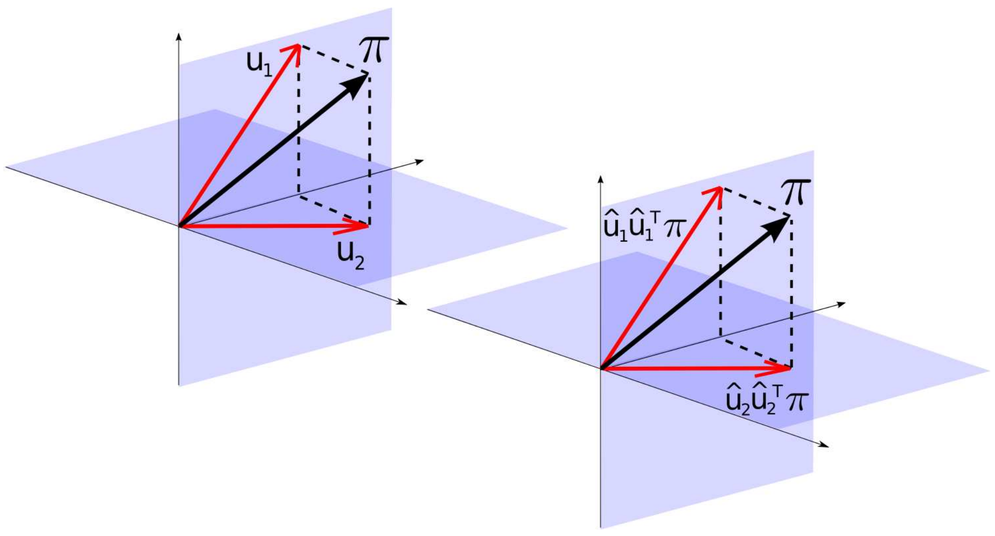


Figure 5.1: Illustration of our learning scheme. The projection of the correct policy π onto the observations matches those observations.

with $r_n = \|\mathbf{u}_n\|$, $\hat{\mathbf{u}}_n = \frac{\mathbf{u}_n}{r_n}$. Since $\mathbf{u}_n = \mathbf{N}_n \pi_n$ we can write $\|\mathbf{u}_n - \mathbf{N}_n \tilde{\pi}(\mathbf{x}_n)\|^2 = \|\mathbf{N}_n(\pi_n - \tilde{\pi}(\mathbf{x}_n))\|^2$ and recognise that the CPE is always less than or equal to the UPE, because the projections $\mathbf{N}_n$ can only decrease the norm of the difference between true and predicted policy. The same argument holds for the inconsistency error (5.4) where the projection onto the 1-D subspace spanned by $\hat{\mathbf{u}}_n$, possibly takes away even more of the error[2]. So we can establish the inequality

$$E_i[\tilde{\pi}] \leq E_{cpe}[\tilde{\pi}] \leq E_{upe}[\tilde{\pi}].$$

Naturally, for estimating the correct policy, we would rather like to minimise an *upper bound* of E_{upe}, but it is unclear how such a bound could be derived from the data we are assumed given (we will revisit this issue in Ch. 6). However, note that by framing our learning problem as a risk minimisation task, we can apply standard regularisation techniques such as adding suitable penalty terms to prevent over-fitting due to noise.

The proposed risk functional (5.4) can be used in conjunction with many standard regression techniques. In principle, provided that there is sufficient variability in the constraints, policies of arbitrary complexity can be learnt, limited only by the representational power of the underlying regression model. (Note also that since the constraints

[2]Note that, in the approach described in Ch. 4, since we look for models that are consistent along the direction of movement of the trajectories along the direction, we also effectively minimise the error in the same sub-space.

do not explicitly enter into the risk calculation, the complexity of the constraints does not affect the performance of learning.) However, for the experiments in the remainder of this chapter, we focus to two classes of function approximator for learning the (unconstrained) policy to demonstrate how the risk functional can be used. The example function approximators we use are (i) simple parametric models with fixed basis functions (Sec. 5.2.3), and (ii) locally linear models (Sec. 5.2.4). In the next section we describe how the two models can be reformulated to take advantage of the new risk functional.

5.2.3 Example: Parametric Policy Models

A particularly convenient model of the policy is given by $\tilde{\pi}(\mathbf{x}) = \mathbf{W}\mathbf{b}(\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{d\times M}$ is a matrix of weights, and $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^M$ is a vector of fixed basis functions. This notably includes the case of (globally) linear models where we set $\mathbf{b}(\mathbf{x}) = \bar{\mathbf{x}} = (\mathbf{x}^T, 1)^T$, or the case of normalised radial basis functions (RBFs) $b_i(\mathbf{x}) = \frac{K(\mathbf{x}-\mathbf{c}_i)}{\sum_{j=1}^M K(\mathbf{x}-\mathbf{c}_j)}$ calculated from Gaussian kernels $K(\cdot)$ around M pre-determined centres $\mathbf{c}_i$, $i = 1\ldots M$. With this model, the *inconsistency* error from (5.4) becomes

$$\begin{aligned} E_i(\mathbf{W}) &= \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T \mathbf{W}\mathbf{b}(\mathbf{x}_n)\right)^2 \\ &= \sum_{n=1}^{N} \left(r_n - \mathbf{v}_n^T \mathbf{w}\right)^2 = E_i(\mathbf{w}), \end{aligned}$$

where we defined $\mathbf{w} \equiv vec(\mathbf{W})$ and $\mathbf{v}_n \equiv vec(\hat{\mathbf{u}}_n \mathbf{b}(\mathbf{x}_n)^T) = \mathbf{b}(\mathbf{x}_n) \otimes \hat{\mathbf{u}}_n$ in order to retrieve a simpler functional form. Since our objective function is quadratic in $\mathbf{w}$, we can solve for the optimal weight vector easily:

$$\begin{aligned} E_i(\mathbf{w}) &= \sum_n r_n^2 - 2\sum_n r_n \mathbf{v}_n^T \mathbf{w} + \mathbf{w}^T \sum_n \mathbf{v}_n \mathbf{v}_n^T \mathbf{w} \\ &= E_0 - 2\mathbf{g}^T \mathbf{w} + \mathbf{w}^T \mathbf{H}\mathbf{w} \end{aligned}$$

yielding

$$\mathbf{w}^{opt} = \arg\min E_i(\mathbf{w}) = \mathbf{H}^{-1}\mathbf{g} \tag{5.5}$$

with $\mathbf{H} = \sum_n \mathbf{v}_n \mathbf{v}_n^T$ and $\mathbf{g} = \sum_n r_n \mathbf{v}_n$. For regularisation, we use a simple weight-decay penalty term, that is, we select $\mathbf{w}_{reg}^{opt} = \arg\min(E_i(\mathbf{w}) + \lambda \|\mathbf{w}\|^2)$. This only requires modifying the Hessian to $\mathbf{H}^{reg} = \sum_n \mathbf{v}_n \mathbf{v}_n^T + \lambda \mathbf{I}$.

Please note that the projection onto $\mathbf{u}$ introduces a coupling between the different components of $\tilde{\pi}$, which prevents us from learning those independently as is common in normal regression tasks. For the same reason, the size of the Hessian scales with $O(d^2M^2)$. For convenience, pseudocode for the learning is given in Algorithm 2.

Algorithm 2 Inconsistency Optimisation

```
1: Initialise policy model (e.g., allocate RBF centres $\mathbf{c}_i$ and kernel size $\sigma^2$).
2: Pre-calculation of terms:
     • Find $r_n = \|\mathbf{u}_n\|$, $\hat{\mathbf{u}}_n = \mathbf{u}_n/r_n$, and $\mathbf{v}_n = \mathbf{b}(\mathbf{x}_n) \otimes \hat{\mathbf{u}}$ for each data point.
3: Optimisation:
     • Build Hessian $\mathbf{H} = \sum_n \mathbf{v}_n \mathbf{v}_n^T$ and linear term $\mathbf{g} = \sum_n r_n \mathbf{v}_n^T$.
     • (Optional: Replacing $\mathbf{H}$ with $\mathbf{H}^{reg} = \mathbf{H} + \bar{\lambda}\mathbf{I}$ and assign regularisation parameter $\bar{\lambda}$ that minimises $E_i[\tilde{\pi}]$ on validation data subset.)
     • Find optimal weights $\mathbf{w}^{opt} = \mathbf{H}^{-1}\mathbf{g}$.
     • Reshape $\mathbf{W}^{opt} = vec^{-1}(\mathbf{w}^{opt})$ for prediction.
```

5.2.4 Example: Locally Linear Policy Models

The basis function approach quickly becomes non-viable in high-dimensional input spaces. Alternatively, we can fit multiple locally weighted linear models $\tilde{\pi}_m(\mathbf{x}) = \mathbf{B}_m\bar{\mathbf{x}} = \mathbf{B}_m(\mathbf{x}^T, 1)^T$ to the data, learning each local model independently (Schaal and Atkeson, 1998). For a linear model centred at $\mathbf{c}_m$ with an isotropic Gaussian receptive field with variance σ^2, we would minimise

$$\begin{aligned} E_i(\mathbf{B}_m) &= \sum_{n=1}^{N} w_{nm} \left(r_n - \hat{\mathbf{u}}_n^T \mathbf{B}_m \bar{\mathbf{x}}_n\right)^2 && (5.6) \\ &= \sum_{n=1}^{N} w_{nm} \left(r_n - \mathbf{v}_n^T \mathbf{b}_m\right)^2 = E_i(\mathbf{b}_m), && (5.7) \end{aligned}$$

where we defined $\mathbf{b}_m = vec(\mathbf{B}_m)$ and $\mathbf{v}_n \equiv vec(\hat{\mathbf{u}}_n \bar{\mathbf{x}}_n^T)$ similarly to the parametric case. The factors

$$w_{nm} = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_n - \mathbf{c}_m\|^2)$$

weight the importance of each observation $(\mathbf{x}_n, \mathbf{u}_n)$, giving more weight to nearby samples. The optimal slopes $\mathbf{B}_m$ in vector form are retrieved by

$$\mathbf{b}_m^{opt} = \arg\min E_i(\mathbf{b}_m) = \mathbf{H}_m^{-1}\mathbf{g}_m \tag{5.8}$$

with $\mathbf{H}_m = \sum_n w_{nm}\mathbf{v}_n\mathbf{v}_n^T$ and $\mathbf{g}_m = \sum_n w_{nm} r_n \mathbf{v}_n$.

For predicting the global policy, we combine the local linear models using the convex combination

$$\tilde{\pi}(\mathbf{x}) = \frac{\sum_{m=1}^M w_m \mathbf{B}_m \bar{\mathbf{x}}}{\sum_{m=1}^M w_m}$$

where $w_m = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{c}_m\|^2\right)$. For implementation, the pseudocode in Algorithm 2 can be used for each local model simply by making appropriate substitutions to incorporate the weighting factors w_m in the calculation of the Hessian $\mathbf{H}_m$ and linear term $\mathbf{g}_m$.

5.3 Experiments

To explore the performance of the new algorithm, we performed experiments on data from autonomous control policies of varying size and complexity, similar to those reported in Sec. 4.3. In this section, we first discuss results from an illustrative toy problem, this time focusing on a on a rotational (i.e. non-potential-based) policy. We then demonstrate how the method generalises across constraints on kinematic data from the 7-DOF DLR lightweight arm (Sec. 5.3.2). Next, we repeat the ball-reaching experiment (ref. Sec. 4.3.2) using WBM control of the humanoid robot ASIMO (Gienger et al., 2005) and that of learning in the full 22-DOF ASIMO upper body joint space (cf. Sec. 4.3.3). After validating the approach on these artificial systems where the ground truth is known, we then explore the utility of the new approach for learning in a real imitation learning setting: We demonstrate an application of our approach to enable the ASIMO robot to learn a car washing task from observed human movements (Sec. 5.3.5). Finally, in Sec. 5.3.6, the performance of the new approach is compared with that of the previous alignment-based approach of the preceding chapter, using identical data sets and with similar policy models.

5.3.1 Toy Example

Our first experiment demonstrates the learning of rotational policies from constrained trajectories in a simple toy example consisting of a two-dimensional system with discontinuously switching motion constraints. As an example policy, we used a limit cycle attractor of the form

$$\dot{r} = r(\rho - r^2), \qquad \dot{\theta} = \omega \tag{5.9}$$

where r, θ are the polar representation of the Cartesian state space coordinates (i.e. $x_1 = r\sin\theta$, $x_2 = r\cos\theta$), ρ is the radius of the attractor and $\dot{\theta}$ is the angular velocity (see Fig. 5.2 (a)). For the experiments we set $\rho = 0.5\ m$ and $\omega = 1\ rad\ s^{-1}$ with a sampling rate of 50 Hz. Data was collected by recording 40 trajectories of length 40 time steps each, generated by the policy from a random start state distribution X_0.

During the trajectories the policy was subjected to random switching constraints, similar to those described in Sec. 4.3.1, i.e. constraints of the form

$$\mathbf{A}(\mathbf{x},t) = (\alpha_1, \alpha_2) \equiv \alpha$$

where the $\alpha_{1,2}$ were drawn from a normal distribution $\alpha_i = N(0,1)$. The constraints mean that motion is constrained in the direction orthogonal to the vector α in state space. As before, these were randomly switched by generating a new α twice at regular intervals during the trajectory, inducing sharp turns which can be seen in Fig. 5.2 (b-d).

We used a parametric model to learn the policy through minimisation of the inconsistency (5.4) as described in section 5.2.3. We included the regularisation term and picked the parameter $\bar{\lambda}$ by minimising the inconsistency on a validation subset. For this toy problem, we chose our function model as a set of 36 normalised RBFs centred on a 6×6 grid, and we simply fixed the kernel width to yield suitable overlap. We repeated this experiment on 100 data sets and evaluated the normalised UPE, CPE (ref. Appendix 3.4.2) and the inconsistency[3] on a subset held out for testing. For comparison, we repeated the experiment using a naive approach that attempted to perform regression with the same RBF model directly on the constrained observations, i.e., the naive approach attempted to minimise the functional (5.1).

Figure 5.2 shows the true policy, the trajectories we trained on, the policies learnt using our and the naive approach, and finally the error statistics below the plots. With an average nUPE of 0.0027, our method outperforms the naive approach by orders of magnitude. Notably, even with only 4 trajectories (Fig. 5.2(b)), the reconstructed policy already resembles the limit cycle, although large errors still persist in some parts of the state space (e.g., the lower right corner). Further to this, the top panel of Fig. 5.3 depicts how the nUPE and nCPE evolve with increasing size of the training set, showing a smooth decline (please note the log. scale).

In order to further explore the performance of our algorithm, we contaminated the observed commands $\mathbf{u}_n$ with Gaussian noise, the scale of which we varied to match up

[3]Actually, for $\mathbf{u} \in \mathbb{R}^2$ the inconsistency is exactly equivalent to the CPE, since both necessarily involve the same 1-D projection.

to 20% of the scale of the data. The resulting nUPE roughly follows the noise level, as is plotted in Fig. 5.3 (bottom).

5.3.2 Generalisation Over Unseen Constraints

The two goals of our second set of experiments were (i) to characterise how well the algorithm scaled to more complex, realistic constraints and (ii) to characterise how well the learnt policies generalised over unseen constraints. For this, we used kinematic data from the 7-DOF DLR lightweight robot (LWR-III) (Fig. 1.1). The experimental procedure was as follows: We generated a random initial posture by drawing 7 joint angles uniformly from half the range of each joint, that is $x_i \sim U[-0.5x_i^{max}; 0.5x_i^{max}]$, where, for example, $x_1^{max} = 170^\circ$. We set up a joint limit avoidance type policy as $\pi(\mathbf{x}) = -0.05\nabla\phi(\mathbf{x})$, with the potential given by $\phi(\mathbf{x}) = \sum_{i=1}^{7} |x_i|^p$ for $p = 1.5, p = 1.8$, or $p = 2.0$. We then generated 100 trajectories with 100 points each, following the policy under 4 different constraints, which we refer to as 1-2-3, 4-5-6, 1-3-5, and 2-4-6. Here, the three numbers denote which end-effector coordinates in task space[4] we kept fixed, that is, 1-2-3 means we constrained the end-effector position, but allowed arbitrary changes in the orientation (here, orientation was represented as yaw, pitch and roll angles in the inertial frame). Similarly, 2-4-6 means we constrained the y-coordinate and the orientation around the x- and z-axis, while allowing movement in x-z position and around the y-axis. For all 4 constraint types, we estimated the policy from a training subset, and evaluated it on test data from the same constraint, as well as on trajectories from the complementary constraint (e.g., 2-4-6 is complementary to 1-3-5).

For learning in the 7-D state space, we selected locally linear models as described in Sec. 5.2.4, where we chose rather wide receptive fields (fixing $\sigma^2 = 3$) and placed the centres $\{\mathbf{c}_m\}$ of the local models such that every training sample $(\mathbf{x}_n, \mathbf{u}_n)$ was weighted within at least one receptive field with $w_m(\mathbf{x}_n) \geq 0.7$. On average, this yielded about 50 local models.

While the linear policy $\pi(\cdot)$ corresponding to $p = 2.0$ was learnt almost perfectly (all normalised errors were in the order of 10^{-9}), the less linear policies ($p = 1.8$ and especially $p = 1.5$) turned out to be a much harder problem. This can be seen when comparing both the nUPE and nCPE for the two policies (ref. Table 5.1). Still, we

[4]The numbers can also be read as row indices of the 6×7 Jacobian matrix.

Potential	Constr.	nUPE	nCPE	Compl. nCPE	Norm. Incon.
$p = 1.5$	1-2-3	64.338 ± 32.030	2.917 ± 0.368	15.951 ± 6.473	0.755 ± 0.067
	4-5-6	34.753 ± 19.125	2.491 ± 0.228	15.478 ± 7.755	0.388 ± 0.036
	1-3-5	16.179 ± 3.813	3.204 ± 0.276	5.108 ± 1.079	0.706 ± 0.067
	2-4-6	10.355 ± 1.827	2.723 ± 0.237	4.749 ± 0.956	0.401 ± 0.039
$p = 1.8$	1-2-3	8.096 ± 5.766	0.477 ± 0.088	2.278 ± 1.133	0.112 ± 0.011
	4-5-6	5.364 ± 2.961	0.352 ± 0.038	2.221 ± 0.984	0.051 ± 0.006
	1-3-5	2.275 ± 0.645	0.455 ± 0.041	0.773 ± 0.171	0.098 ± 0.011
	2-4-6	1.421 ± 0.314	0.401 ± 0.042	0.729 ± 0.174	0.058 ± 0.007

Table 5.1: Normalised UPE, CPE on the training constraints, CPE on complementary constraints and inconsistency error, for data from the DLR arm (Fig. 1.1). All errors normalised by the variance of the policy. We report (mean $\pm$ s.d.)$\times 10^{-2}$ over 100 trials with different data sets.

recovered the constrained policy in all cases to good accuracy (ref. Table 5.1, 4th column), with good generalisation to the complementary constraints (ref. Table 5.1, 5th column). We can also see that constraining the end-effector position (1-2-3) made it more difficult to recover the unconstrained policy compared to constraining the orientation (4-5-6), or using mixed constraints (1-3-5 and 2-4-6). It should also be noted that running the same experiment using the naive approach (ref. Sec. 5.3.1) gave consistently poor results; for example, when training on data under the (1-2-3) constraint, the naive approach gave nUPE of $83.44 \pm 1.20 \times 10^{-2}$ for the $p = 1.5$ policy, $80.94 \pm 1.37 \times 10^{-2}$ for $p = 1.8$ and $79.62 \pm 1.39 \times 10^{-2}$ for $p = 2.0$.

5.3.3 Reaching for a Ball

The goal of our next set of experiments was to illustrate the utility of our approach for learning from observations of an everyday task with realistic constraints. For this, we re-visited the ball reaching experiment (ref. Sec. 4.3.2), in which we are given a set of observations of a demonstrator reaching for a ball on a table and the task is to learn a policy that reproduces this movement. As before, the learning problem is complicated by the presence of barriers on the table that constrain the possible movements and force the demonstrator to reach between the barriers to get to the ball. The goal is to uncover a policy that accurately predicts the demonstrator's behaviour and generalises across

Constraint	Naive	Non-naive
Training	0.1940 ± 0.0153	0.0056 ± 0.0022
Unseen Barrier	0.4678 ± 0.0264	0.0057 ± 0.0023
Unconstrained	0.7014 ± 0.0430	0.0058 ± 0.0023

Table 5.2: Normalised policy errors for predicting the policy under three constraint conditions from the ball-reaching data for the naive and non-naive methods. Values are mean$\pm$s.d. over 50 data sets.

constraints.

To simulate this scenario we again used the WBM controller of the ASIMO humanoid (for details see Gienger et al. 2005). We collected data from a 'demonstrator' robot, this time following a policy $\mathbf{u} = \dot{\mathbf{x}} = \pi(\mathbf{x})$ defined by an inverted Gaussian potential

$$\pi(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = \alpha\left(1 - e^{\|\mathbf{x}-\mathbf{x}_c\|^2/2\sigma^2}\right), \tag{5.10}$$

where $\mathbf{x} \in \mathbb{R}^6$ corresponds to the Cartesian position of the two hands and we chose $\sigma^2 = 2$, $\alpha = 0.25$ and the target point $\mathbf{x}_c \in \mathbb{R}^6$ to correspond to a reaching position, with the two hands positioned on either side of the ball. Similar to the quadratic potential-based policy (4.29), with this set of parameters, the demonstrator was able to reach the ball under each of the constraints considered in this experiment (see below). However, note that here, unlike the potential-based policy, the policy is a non-linear function of state, and thus represents a more difficult learning task.

The demonstrator's movements were constrained by the same constraints described in Sec. 4.3.2, i.e., (4.30), with the width of the gap randomly changed at the start of each demonstrated trajectory according to a Gaussian distribution $d_{gap} \sim \mathcal{N}(\mu_{gap}, \sigma_{gap})$ where $\mu_{gap} = 0.25m$, $\sigma_{gap} = 0.1m$. Under this set up we collected $K = 100$ trajectories of length $2s$ at 50 Hz, (i.e. $N = 100$ points per trajectory). Start states were sampled from a Gaussian distribution over joint configurations, $\mathbf{q} \sim \mathcal{N}(\mathbf{q}_0, 0.1\mathbf{I})$ (where $\mathbf{q}_0$ corresponds to the default standing position) and using forward kinematics to calculate the corresponding hand positions. The joint vector $\mathbf{q}$ was again clipped where necessary to avoid joint limits and self collisions, and to ensure the start postures looked natural.

Learning was performed on 50 such data sets using 150 local linear models, with centres placed using k-means. For comparison, the experiment was also repeated on the same data with the same local linear model (i.e., same number and placement of centres), but using the naive approach for training (i.e. training on $(\mathbf{x}_i, \mathbf{u_i} \equiv \dot{\mathbf{x}}_i), i = 1, \ldots K \times N$ directly, using the risk functional (5.1)).

To assess the performance for both methods we evaluated the errors in predicting the policy subject to (i) the training data constraints (nCPE), (ii) no constraints (nUPE), and (iii) a novel constraint, unseen in the training data, on a set of test data. For the latter, a barrier was placed centrally between the robot and the ball, so that the robot had to reach around the barrier to reach the ball (see Fig. 5.5).

As expected, learning using the proposed risk functional (5.4) (the 'non-naive' approach) performed much better than the naive approach in terms of the numerical error measures (ref. Table 5.2), similar to our results with the potential based approach (ref. Table 4.2). To further confirm this, we also compared the trajectories generated by the two policies under the different constraint settings to see if the effect of generalisation over constraints was reproduced with the new method.

In Fig. 5.4 we show example trajectories for the *unconstrained* reaching movements produced by the demonstrator ('expert'), and the policies learnt by (i) the naive approach, and; (ii) the non-naive approach; from a number of start states. We see that, for the former, the hands always take a curved path to the ball (Fig. 5.4, left), reproducing the average behaviour of the (constrained) demonstrations (and similar to what we saw when learning with the naive approach and LWPR, ref. Sec. 4.3.2). However, in contrast, the policy learnt with the new approach better predicts the unconstrained policy, and takes a direct route to the ball that closely matches that of the demonstrator (Fig. 5.4, right). Similar to the potential-based approach then, the new method extracts the essential unconstrained grasping movement despite training exclusively on data containing constraints.

Secondly, Fig. 5.5 shows example trajectories when the learnt policies are again constrained. Figure 5.5 (top) shows the movement from the non-naive policy under a similar constraint as in the training data. Under this constraint both naive and non-naive policies take a similar path as the demonstrator: The hands move in first, then forward to the ball. Note that under this constraint the movement of the naive policy is noticeably slower due to the model averaging effect (ref. Sec. 3.3.2).

Finally, under the unseen barrier constraint, there is a marked difference in behaviour. Under this constraint, the demonstrator (still following the policy (5.10)) reaches around the barrier to get the ball. This behaviour is reproduced by the policy learnt with the new approach (Fig. 5.5, middle). In contrast, however, the naive policy does not generalise to the new constraint and gets trapped behind the barrier, eventually dislodging it[5] (Fig. 5.5, bottom). The behaviour of the three policies (demonstrator, naive and non-naive policies) can be examined in detail in the accompanying video.

5.3.4 Learning from High-dimensional Joint-space Data

To test the scalability of the new approach for learning in very high dimensions, we also re-visited the experiment on learning from ASIMO joint space data (ref. Sec. 4.3.3). For this, we again used a policy based on a quadratic potential in joint space

$$\pi(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_c)^T\mathbf{W}(\mathbf{x}-\mathbf{x}_c),$$

where $\mathbf{x}_c \in \mathbb{R}^{27}$ is a target posture and $\mathbf{W}$ is a weighting matrix. The policy represents an attractor in joint space that pulls the robot into a desired posture at $\mathbf{x}_c$. For the experiments, $\mathbf{x}_c$ was chosen to correspond to a reaching posture with both arms outstretched (ref. Fig. 5.6, right) and we chose $\mathbf{W} = 0.05\mathbf{I}$. Note that, in contrast to the experiment described in Sec. 4.3.3), to increase the difficulty of the learning task, 5 additional DOFs (corresponding to the Cartesian heel positions, the torso height and torso lateral orientation) were included in the state vector.

During data collection, the policy was constrained by the presence of obstacles which took the form of a vertical wall placed directly in front of the robot at different orientations and distances (ref. Fig. 5.6, left). Specifically, the constraint matrix, $\mathbf{A}(\mathbf{x},t) \in \mathbb{R}^{2\times 27}$, took the form

$$\begin{aligned} \mathbf{A}_i(\mathbf{x},t) &= \mathbf{0} \qquad\quad ; \quad d_i > 0 \\ \mathbf{A}_i(\mathbf{x},t) &= \hat{\mathbf{n}}^T\mathbf{J}_i(\mathbf{x}) \;\; ; \quad otherwise. \end{aligned} \tag{5.11}$$

Here, $\hat{\mathbf{n}} \in \mathbb{R}^2$ is the normal[6] to the wall surface, d_i is the perpendicular distance of the ith hand from the wall surface (with $i \in \{1,2\}$, i.e. left and right hands respectively), $\mathbf{J}_i(\mathbf{x}) \in \mathbb{R}^{2\times 27}$ is the Jacobian mapping from joint-space to the lateral (i.e. horizontal planar) coordinates of that hand and $\mathbf{A}_i(\mathbf{x},t) \in \mathbb{R}^{1\times 27}$ is the corresponding row of the constraint matrix. At the start of each trajectory, the orientation of the wall was drawn

[5]Note that the collision of the hands with the barrier in fact violates the constraint. The reason for this is that on the real robot, under this constraint, the naive policy forces the robot into a self-collision (of the robot's arms with the torso). To prevent damage to the robot, an on-board safety mechanism then kicks in and pushes the hands away from the body, causing collision with the barrier.

[6]Note that since the wall was vertical in all example trajectories (and thus did not affect vertical movements) only the normal in the horizontal plane is relevant to calculation of the constraints.

from a uniform random distribution $\theta \sim U[-\theta^{max}, \theta^{max}]$ where θ is the angle of the wall with respect to the left-right axis of the robot heel frame (horizontal axis in Fig. 5.6, left), and we chose $\theta^{max} = 27^\circ$. The distance of the wall was adjusted at the start of each trajectory to ensure that the the hands were a minimum distance of $0.15m$ from the wall before the onset of movement.

The effect of the constraints was to restrict the movement of the hands when they approached the wall. This constraint was projected back into the joint space where the policy was operating via the Jacobian. This causes the policy to appear highly complex and non-linear in the state space (joint space), with discontinuous changes to the dimensionality of the constraints as the hands of the robot approached the wall.

Using the formalism from Sec. 5.2.3 with $\mathbf{b}(x) = \bar{\mathbf{x}}$, we fitted linear models to 100 data sets, each consisting of 100 trajectories of 100 data points. Despite the high dimensionality, the new method reached a normalised UPE of $0.291 \pm 0.313 \times 10^{-2}$. It is important to point out that this result can not only be explained by our choice of a linear model where we knew that the true policy was also linear: Direct (naive) linear regression on the observed commands resulted in a normalised UPE of $63.9 \pm 3.1 \times 10^{-2}$ (nCPE was $7.98 \pm 0.66 \times 10^{-2}$), which again is orders of magnitude higher, similar to our results on the lower dimensional data in the preceding sections.

5.3.5 Washing a Car

Having validated our approach on data where the ground truth (true unconstrained policy) was known, in this section, we report experiments on learning from human demonstrations for seeding the robot motion. For this experiment, we chose to investigate the problem of learning to wash a car. This is an example of a task which can be intuitively described in terms of a simple movement policy (‘wiping’) subject to contact constraints that vary depending on the different surfaces of the car to be wiped. Due to the different shapes and orientations of the car surfaces, complex, non-linear constraints are imposed on the motion. The resultant trajectories appear periodic, but are perturbed in different ways by the constraints. The goal of our experiments was to learn a policy that captured the periodic nature of the movements, while eliminating artifacts induced by the constraints.

The experimental setup was as follows. Seven demonstrations of a human wiping different surfaces with a sponge were given to the robot. To simulate observations of washing different surfaces of the car, the wiping was performed on a perspex sheet placed at different tilts and rotations with respect to the robot (see Fig. 5.7). Specifically, the sheet was oriented to be flat (horizontal), tilted $\pm 16^\circ$ and $\pm 27^\circ$ about the x-axis (horizontal axis pointing directly ahead from the robot) and $\pm 16^\circ$ about the y-axis (horizontal right-left axis). The three-dimensional coordinates of the sponge were tracked using the on-board stereo cameras of the ASIMO robot at a rate of 20 frames per second (for details on the ASIMO vision system please see Bolder et al. 2007). The recorded trajectories are shown in Fig. 5.8 (left).

The policy was modelled as the $\mathbb{R}^3 \mapsto \mathbb{R}^3$ mapping from hand (sponge) positions to velocities. Since this is a relatively low-dimensional problem, and for ease of comparison with the toy problem (Sec. 5.3.1), we used RBFs to model the policy. For each of the experiments described below, we used a set of 300 RBFs with centres placed by k-means as our policy model.

Since the ground truth (i.e. the true unconstrained policy and the exact constraints in force) is not known for the human data, performance was evaluated on a behavioural level. In particular, we looked at how the movements produced by the learnt policies compared with those of the human when subject to (what we assumed to be) a similar set of constraints. For this, we implemented the learnt policies on the ASIMO humanoid robot and applied constraints that approximated[7] those contained in the demonstrations.

Specifically, we assumed the constraints in the car wash task to arise from two sources, namely (i) environmental (i.e. physical) constraints and (ii) constraints self-imposed by the demonstrator to ensure task success. In this experiment, the former can be clearly identified as an inequality constraint preventing the hand from penetrating the wiping surface, i.e.

$$\mathbf{A}(\mathbf{x},t) = \hat{\mathbf{n}}_s(\mathbf{x}) \quad ; \quad d = 0 \;\; \textit{and} \;\; \hat{\mathbf{u}}^T \hat{\mathbf{n}}_s(\mathbf{x}) > 0 \tag{5.12}$$

where d is the distance of the hand from the surface and $\hat{\mathbf{n}}_s(\mathbf{x})$ is the normal to the surface s at point $\mathbf{x}$. In addition, we can also identify a self-imposed constraint in force. In the car wash setting, successful performance of the task (i.e. wiping) requires the sponge to maintain contact with the surface at all times so that motion of the hand away from the surface (i.e. lifting the sponge) is not permitted. To capture this, we therefore assumed a further constraint of the form

$$\mathbf{A}(\mathbf{x},t) = \hat{\mathbf{n}}_s(\mathbf{x}) \quad ; \quad d = 0 \;\; \textit{and} \;\; \hat{\mathbf{u}}^T \hat{\mathbf{n}}_s(\mathbf{x}) < 0. \tag{5.13}$$

[7] Please note that for training the policy models, the constraints were not explicitly modelled.

Note that in combination, the effect of the two constraints (5.12)-(5.13), when considered on the wiping surface ($d = 0$), amounts to the single equality constraint

$$\mathbf{A}(\mathbf{x},t) = \hat{\mathbf{n}}_s(\mathbf{x}) \quad ; \quad d = 0. \tag{5.14}$$

This constraint was applied to the learnt policies as a reasonable approximation of the true constraints contained in the data, in order to compare the demonstrated and reproduced movements for any given surface s and assess the generalisation across constraints.

Under this set-up, we first compared learning with our approach against learning with the naive approach. For this, we trained two RBF models on the full data set of seven demonstrations (i.e. wiping data for each of the surfaces). The first model was trained with the approach described in Sec. 5.2.3, the second with the standard (naive) approach to regression. We then used the policies learnt by the two approaches to reproduce the movements under each of the surface constraints (i.e. constraint (5.14) for $s = 1, \cdots, 7$). The results are shown in Fig. 5.8, where we show the demonstrated trajectories (left), those produced by the non-naive policy (centre) and those learnt by the naive approach (right) under the different constraints (tilts of the surface).

Looking at the learnt policies, we see that our approach learns a smooth policy that resembles the limit cycle of Section 5.3.1. The trajectories under each of the constraints are smooth periodic movements, similar to those of the human. These were implemented on the ASIMO robot to produce natural wiping movements (see Fig. 5.9). The policy learnt with the naive approach also captures the periodicity to some extent. However, it appears highly irregular in several regions and the trajectories are unstable, with some spiralling in to the centre, and others diverging to other parts of the state space. By attempting to learn all of the artifacts induced by the constraints, the naive approach learns an unstable policy that cannot be safely used for movement reproduction on the robot[8].

Finally, to confirm that our approach is able to generalise well over unseen constraints, we repeated the experiment, but this time training the model on a subset of the data containing one set of constraints, then testing on a different subset containing different constraints. Specifically, we used our approach to train a model on the three demonstrations corresponding to the surface tilted by $0°$, $+16°$ and $+27°$ about the x-axis (Fig. 5.10, left). We then took the demonstrated movements for the surface tilted

[8]The behaviour produced by the two methods can be examined in detail in the second accompanying video.

at -16° and -27° about the x-axis (Fig. 5.10, right) as our test set and compared the movement reproduction.

In Fig. 5.10 we show the demonstrated (grey) and reproduced (black) trajectories for the training data constraints (left) and the test data constraints (right). Though we train on a smaller data set here, the policy learnt by our approach again produces a smooth wiping movement that reproduces the human movement well, both under the training data constraints and under the unseen test constraints.

5.3.6 Direct Comparison with the Potential-based Approach

As a final test, we performed several experiments to directly compare the alignment approach described in Ch. 4 and the new approach based on optimising the inconsistency. For simplicity we analysed the learning of several two-dimensional policies of varying complexity. Specifically, we tested our approaches for learning the policy derived from the quadratic potential (4.24), that derived from the sinusoidal potential (4.28), and the limit cycle policy (5.9).

The experimental procedure was as follows. We sampled 40 trajectories from the three policies with random start states and at a rate of 50 Hz, resulting in 40 data points per trajectory. During each trajectory the policy was subject to the same random switching 1-D constraints described in Sec. 4.3.1. We trained models of the policy (i) using direct regression on the state-action tuples $(\mathbf{x}_n, \mathbf{u}_n)$, (ii) using the alignment approach of Ch. 4, and (iii) optimising the inconsistency (5.4) for 50 such data sets. In each case we used the same policy model for learning, that is we used a set of 36 normalised Gaussian RBFs placed on a 6×6 grid, and selected the kernel widths to yield a suitable overlap. Note that for the alignment approach the RBF model was used in place of LWPR to learn the potential function from the preprocessed data, i.e. it was trained on the tuples $(\mathbf{x}_{kn}, \breve{\phi}_{kn} + b_k^{opt})$, where $k \in \mathcal{K}$ the non-outlier trajectories and $n \in \{1 \ldots N_k\}$ (ref. Sec. 4.2.5).

The results are summarised in Table 5.3, where we see the following trends. First, the direct learning approach performs the worst both in terms of the nUPE and nCPE. This approach is naive to the constraints so is unable to find a consistent model. Looking at the errors for potential-based (quadratic and sinusoidal) policies, the alignment approach does approximately an order of magnitude better than the direct approach both in terms of nUPE and nCPE. However, as expected, it performs poorly for the limit cycle policy since this a rotational (i.e. non-zero curl) policy, and cannot be

Policy	Alg.	nUPE	nCPE
Quad. Pot.	direct	0.54727 ± 0.06218	0.10732 ± 0.02010
	align.	0.01158 ± 0.01561	0.00443 ± 0.00588
	incon.	0.00001 ± 0.00001	0.00001 ± 0.00001
Sin. Pot	direct	0.40478 ± 0.04789	0.12354 ± 0.01097
	align.	0.05020 ± 0.05395	0.02162 ± 0.02536
	incon.	0.00003 ± 0.00003	0.00001 ± 0.00004
Lim. Cyc.	direct	0.43225 ± 0.06599	0.10034 ± 0.01678
	align.	2.91233 ± 1.56180	1.26902 ± 0.80364
	incon.	0.00024 ± 0.00040	0.00003 ± 0.00002

Table 5.3: Normalised CPE and UPE for the direct, alignment- and inconsistency-based approaches when learning policies based on a quadratic and sinusoidal potential, and a limit cycle policy. All errors are mean$\pm$s.d. on 50 data sets.

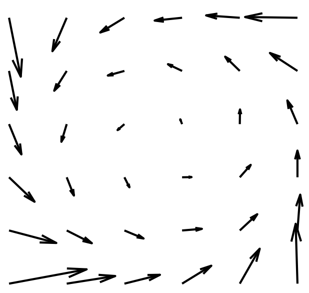
represented well by a potential function. Finally, looking at the errors for the new approach based on optimising the inconsistency, the errors are several orders of magnitude smaller than even the alignment approach. We attribute this to the build up of error from several sources in the alignment approach; for example, errors in the alignment and errors in modelling the aligned data.

5.4 Conclusion

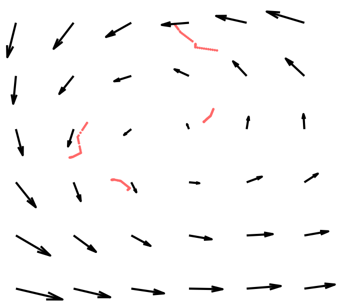
In this chapter, we introduced a novel approach to learning that enabled us to model policies subject to stationary constraints. Having considered several possible risk functions, we settled on a small but very effective modification in the calculation of the standard risk that satisfied our assumptions on the data assumed given. Similar to the potential-based approach reported in Ch. 4, this allowed us to recover the unconstrained policy from arbitrarily constrained observations, without the need for explicit knowledge of the constraints. However, unlike that approach, the new method does not rely on the somewhat restrictive assumptions of kinematic, potential-based policies, and data in the form of trajectories. The effectiveness of the new method was demonstrated using parametric (RBF) and locally linear function approximators to learn policies for problems of varying size and complexity, and in many cases the new method also out-performed the potential-based approach.

While the new approach avoids many of the restrictions of the potential-based method, it still suffers from several problems. In particular, as discussed in Sec. 5.2.2 there is the problem that the inconsistency error is a lower-bound on error. This is due to the fact that the assumed projection $\mathbf{P}$ is a loose approximation of the true projection $\mathbf{N}$ induced by the constraints. This can result in poor and possibly unstable learning in certain cases. In the next chapter, we discuss these problems in detail and discuss methods to alleviate them.

(a) True policy



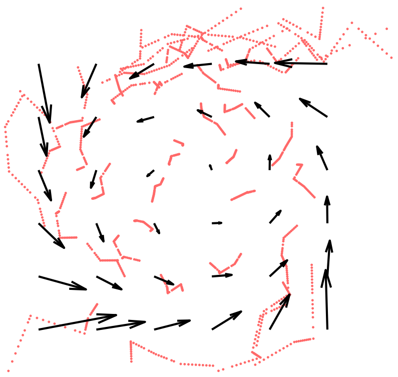
(b) Our method, trained on 4 traj.



nUPE: 0.3788 ± 0.2688

nCPE: 0.1276 ± 0.1140

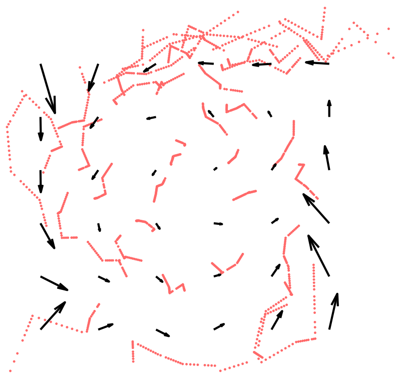
(c) Our method, 40 trajectories



nUPE: 0.0027 ± 0.0087

nCPE: 0.0002 ± 0.0002

(d) Naive method, 40 trajectories



nUPE: 0.5709 ± 0.0853

nCPE: 0.0010 ± 0.0363

Figure 5.2: Results on 2D toy data. (a) true limit cycle policy, (b) learnt policy trained on 4 constrained trajectories, (c) learnt policy from 40 constrained trajectories, (d) policy resulting from naive regression on observed commands. Trajectories are shown as dotted lines, the policy is depicted by black arrows. The normalised CPE and UPE (mean±s.d. over 100 data sets) are given below the figures.

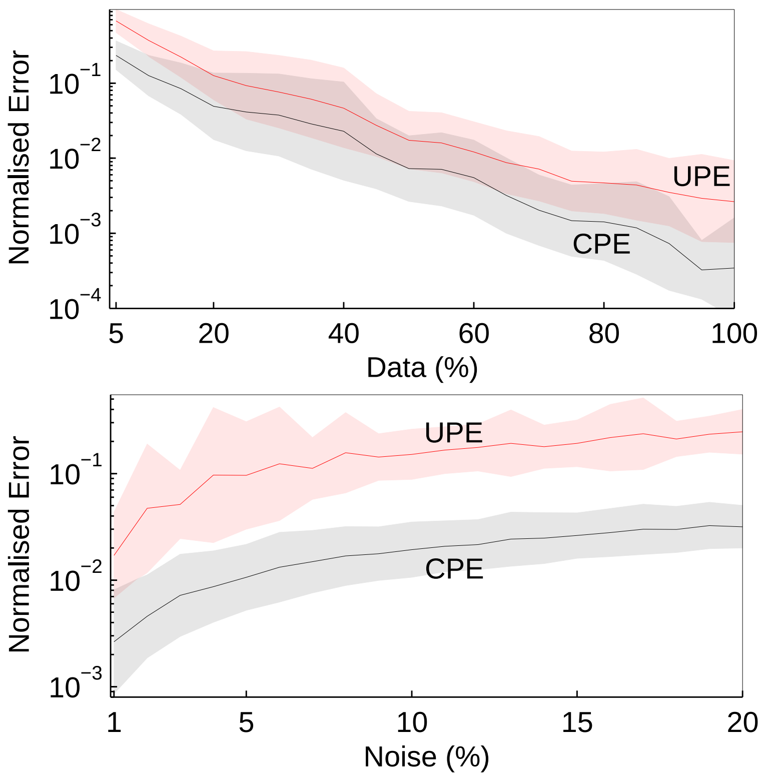


Figure 5.3: Learning performance on the limit-cycle policy (5.9) with varying data set sizes and noise levels. Top: Normalised UPE and CPE versus data set size as a percentage of the full $K = 40$ trajectories of length $N = 40$. Bottom: Normalised UPE and CPE for increasing noise levels in the observed $\mathbf{u}_n$. For clarity, we do not report the (consistently high) errors of the naive method.

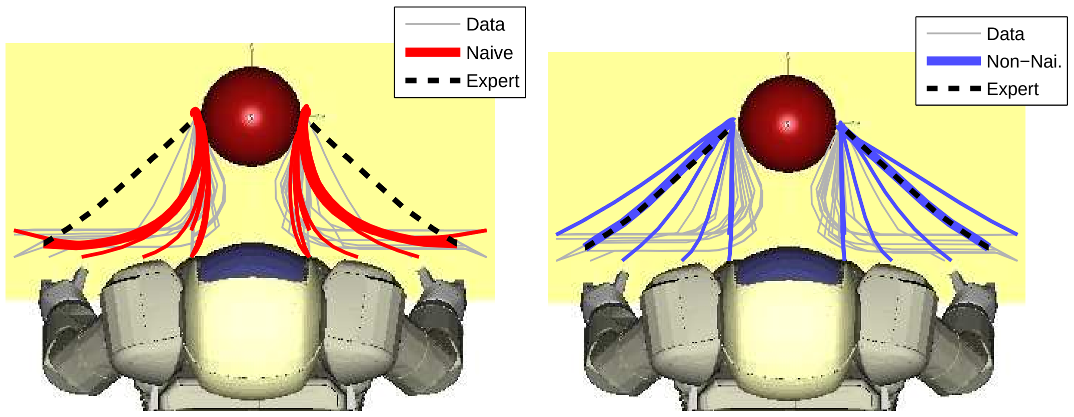


Figure 5.4: Reaching movements produced by the policies learnt by the naive approach (top) and by optimisation of the inconsistency (bottom) when unconstrained. Shown are trajectories of the hands from five start states, with one example highlighted (thick line). The expert trajectory corresponding to the highlighted example is overlaid (black dashed line). Twenty example training data trajectories are also shown (thin grey lines).

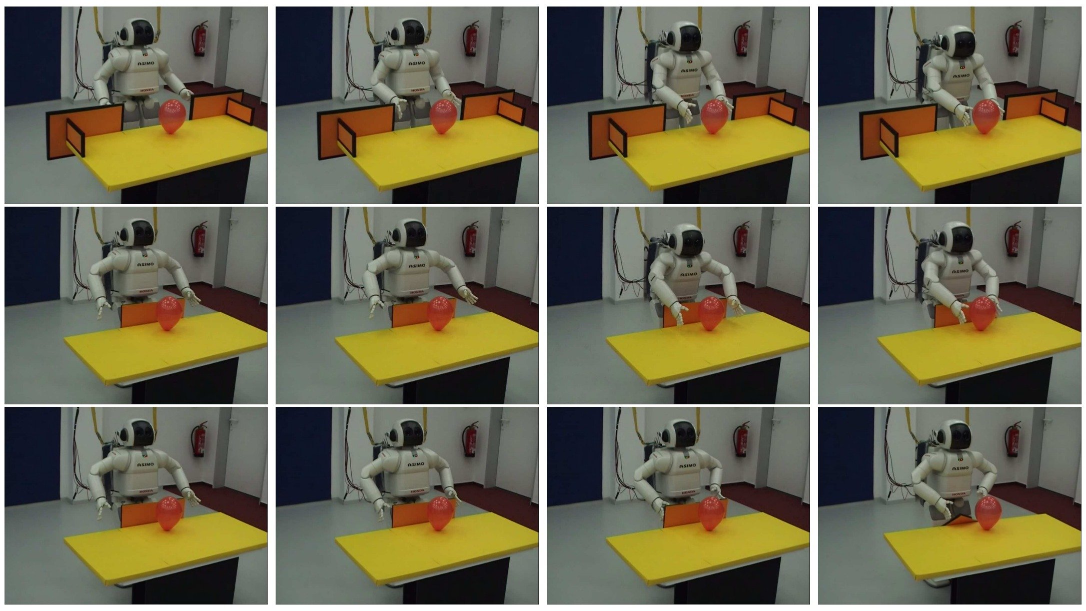


Figure 5.5: Reaching movements produced by the learnt policies under different constraints. Shown are trajectories from (i) the non-naive policy under a similar constraint as in the training data (top row); (ii) the non-naive policy under a new, unseen barrier constraint (middle row), and; (iii) the naive policy under the new constraint.

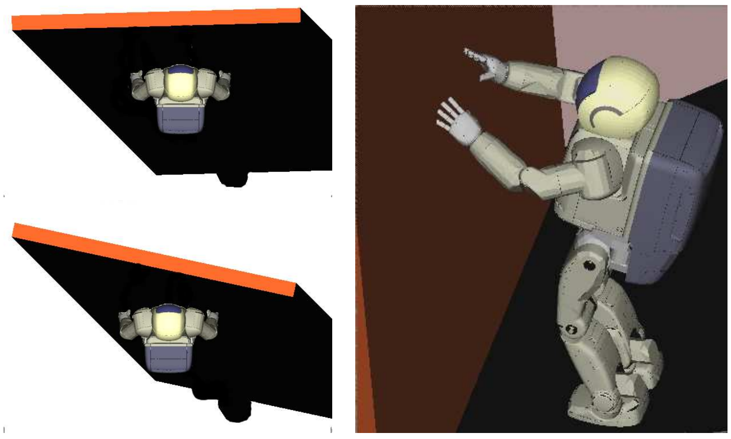


Figure 5.6: Data collection for the joint space policy under wall constraints. Left: Start states for two example reaching movements with the wall at different distances and orientations with respect to the robot. Right: Side view after reaching.

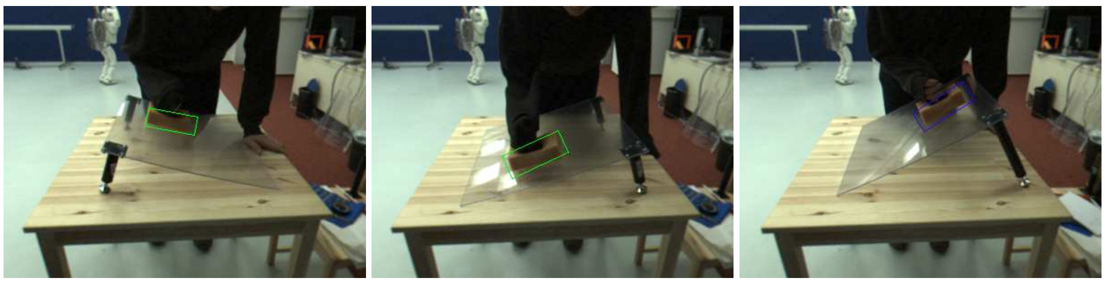


Figure 5.7: Human wiping demonstrations on surfaces of varying tilt and rotations. The ASIMO stereo vision system was used to track the 3-D coordinates of the sponge (coloured rectangles show the estimated position). Tilts of $\pm 16^o$ and $+27^o$ about the x-axis are shown.

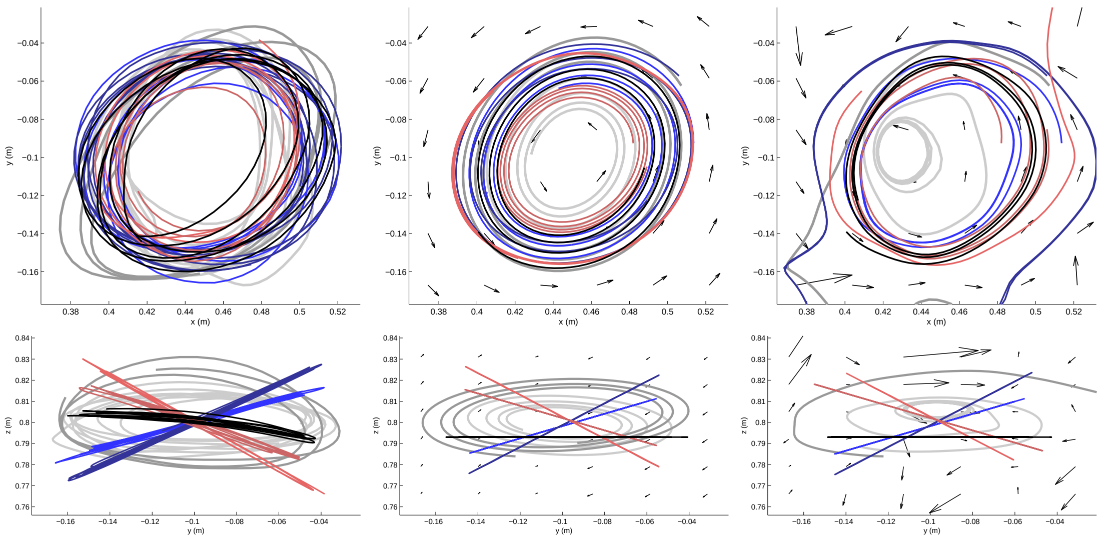


Figure 5.8: Learning from human wiping demonstrations. Left: Trajectories of the sponge when wiping on the surface when flat (black), tilted $+16^\circ$ and $+27^\circ$ about the x-axis (red), -16° and -27° about the x-axis (blue), and $\pm16^\circ$ about the y-axis (grey). Centre and right: Reproduced trajectories using the policies (black arrows) learnt with the non-naive and naive approaches respectively. In each case the same example trajectory is highlighted (thick black). The top and front views are shown (top and bottom rows).

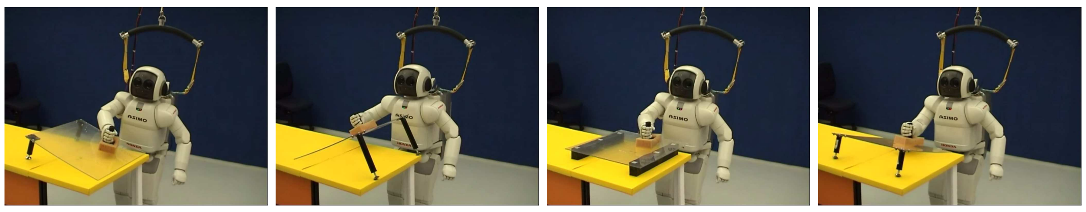


Figure 5.9: Reproduced movements on the ASIMO robot for the surface tilted 0°, $+16^\circ$, -27° about the x-axis, and $+16^\circ$ about the y-axis.

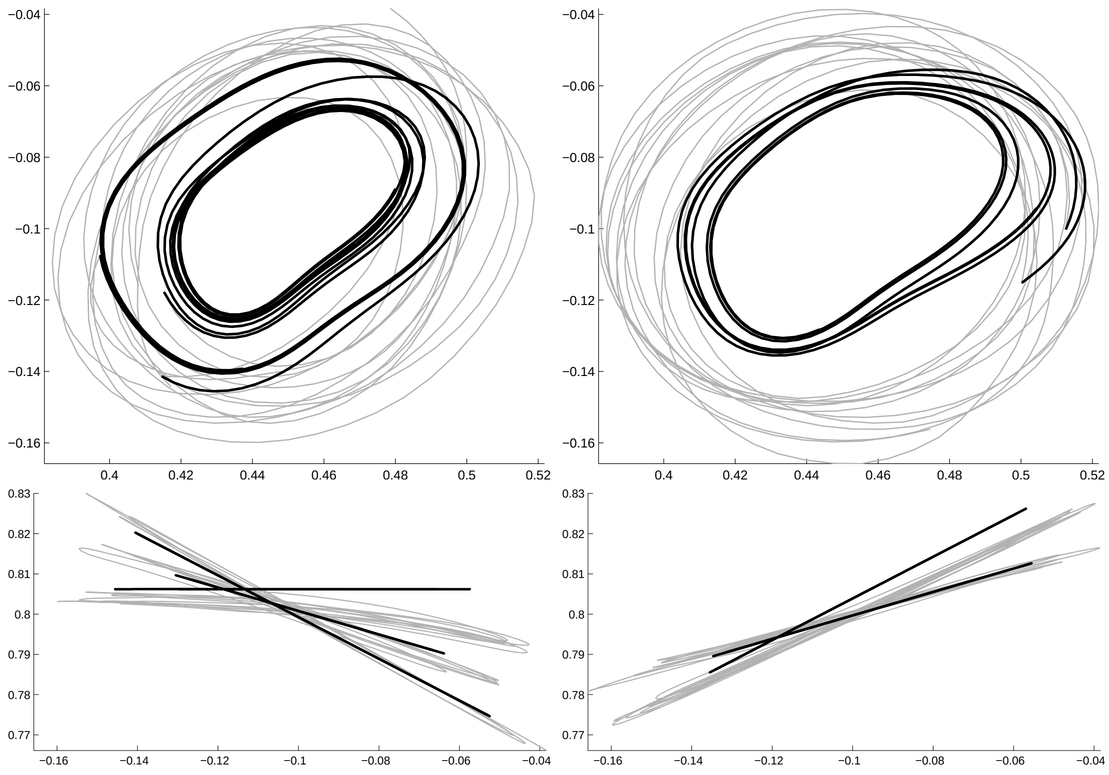


Figure 5.10: Generalisation over constraints when learning from human wiping data. Left: Three demonstrated trajectories with surface tilt 0°, $+16^\circ$ and $+27^\circ$ (grey lines) used to train the model. Right: Two trajectories with tilt -16° and -27° (grey lines) held out for testing. Reproduced trajectories from the learnt policy under the corresponding constraints (both train and test) are overlaid in black.

Chapter 6

Improving Robustness for Constraint-consistent Learning

6.1 Introduction

In the preceding chapter, we explored a novel reformulation of the risk functional used to optimise our policy model. This proved to be highly effective for reconstructing policies from stationary constraint systems without explicit knowledge of the constraints for generic policies of arbitrary complexity.

However, while this method performed well in the various experiments considered in the preceding chapter, in its most basic form it has several limitations. In particular, its effectiveness is highly dependent on the ‘richness’ of the data, in terms of the number of different constraints seen (specifically, the extent to which the action space is spanned by the observations). In fact, if the data contains very little variability in the constraints, for example, if the data is unconstrained or contains a highly correlated constraints, then the approach of optimising the reformulated risk (5.4) alone can result in poor performance. This is because the inconsistency error tends to explain all variations in the observations as variations in constraints rather than as variations in the policy itself.

In this chapter, we propose an extension to the method to deal with this problem. As a key ingredient, we suggest a partitioning of the model optimisation into two parts. The primary part uses the same inconsistency objective function (5.4) to deal with the effect of variable constraints in the data. However, we then propose a *secondary optimisation* scheme to tighten the fit to the data *in regions where there is little variation* *in the constraints*. By extending the method in this way, we will see it is possible to seamlessly blend constraint-consistent learning with optimisation of more standard risk functionals.

In the following, we first explain the model degeneracy problem that can lead degraded performance in certain cases. We then describe how the parameter null space of our models can be utilised for secondary optimisation of additional criteria to tighten the fit. We derive the appropriate learning rules for the example policy models discussed in the preceding chapter, i.e. parametric and local linear models. Finally, we present experiments where the pure inconsistency-based approach has difficulties in learning, and show that the two-step optimisation approach eliminates these problems, selectively providing the best aspects of both standard direct learning and constraint-consistent learning.

6.2 Model Degeneracy for Correlated Constraints

Optimisation of the inconsistency (5.4) has been demonstrated to be effective when learning from data containing high variability in the constraints for systems of varying size and complexity (ref. Ch.. 5, Howard et al. 2009b,a). However, in the simple form outlined so far, it can suffer from the problem of *degeneracy in the set of models* that are optimal with respect to (5.4) when the data contains little variability in the constraints. Because the observations **u** influence the estimated policy in a more complex way than in direct regression, small variations in the observations may result in large variations of the learnt policy[1], which can become catastrophic when the method is given data with insufficient variability in the constraints to disambiguate the best policy models.

To illustrate the problem, Fig. 6.1 shows three candidate policy models $\tilde{\pi}_1$, $\tilde{\pi}_2$ and $\tilde{\pi}_3$ as well as data under a single constraint (right) and two different constraints (left). Consider that we have to select one of these candidates based on the available data. For the multiple (i.e. variable) constraint case (Fig. 6.1, left), optimising the inconsistency (5.4) clearly determines the best model given the available data: In this case we would choose $\tilde{\pi}_1$, since this has the lowest inconsistency error, $E_i[\tilde{\pi}_1] < E_i[\tilde{\pi}_2] < E_i[\tilde{\pi}_3]$.

However, when there is less variability in the constraints, for example, we only see an observation under a single constraint (Fig. 6.1, right), there may be little difference in the inconsistency for the three models (here, $E_i[\tilde{\pi}_1] = E_i[\tilde{\pi}_2] = E_i[\tilde{\pi}_3]$) resulting in

[1]In machine learning terms, the pure inconsistency-based estimator has high variance.

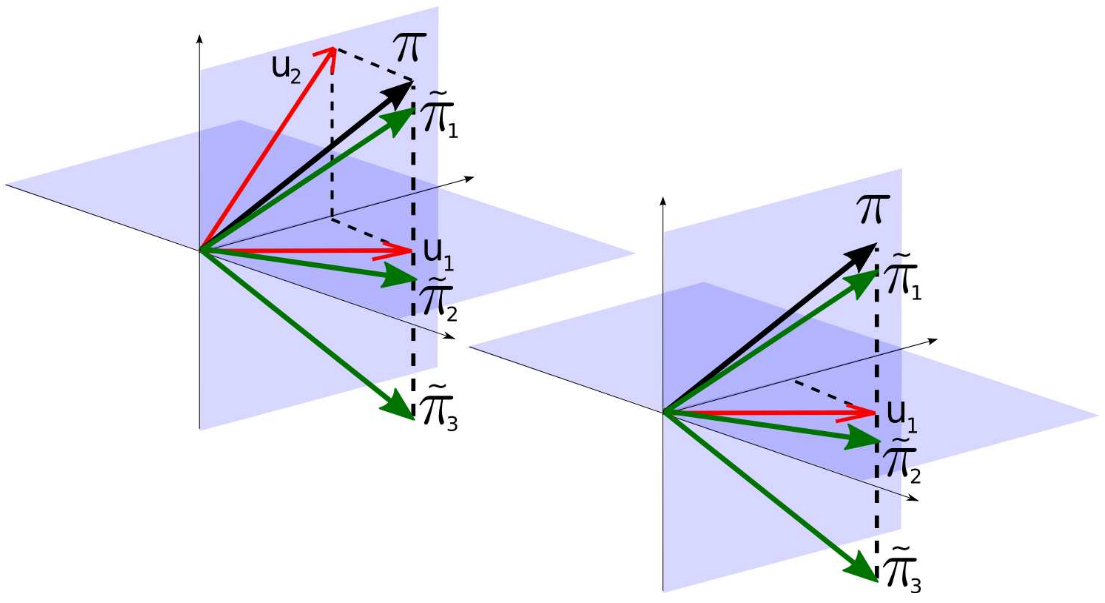


Figure 6.1: Illustration of the model degeneracy problem. Shown are three different models with equal inconsistency with respect to the observation $\mathbf{u}_1$. Left: Given observations under different constraints, e.g. $\mathbf{u}_2$, the inconsistency error disambiguates between the three candidate models selecting that which is consistent with both observations (i.e. $\tilde{\pi}_1$). Right: Given only observations under a single constraint there is ambiguity in which is the best model since we cannot be sure about the policy components in the vertical dimension.

ambiguity as to which model to choose. This is a critical problem, since if we select the wrong model, e.g. $\tilde{\pi}_3$, then it may significantly degrade performance both in terms of prediction of the unconstrained policy (compare π and $\tilde{\pi}_3$ in Fig. 6.1) and also the constrained policy (consider the projection of $\tilde{\pi}_3$ onto the vertical plane, and compare with $\mathbf{u}_2$). Note also that this is a manifestation of the fact that E_i is a lower bound on both the unconstrained policy error (UPE) and the constrained policy error (CPE) (ref. Sec. 5.2.2), since it is precisely the fact that these components of the policy that are projected out in the calculation of the inconsistency error that leads to this model degeneracy problem.

6.3 Secondary Optimisation of the Standard Risk

In order to deal with this problem, our proposal is to perform an additional *secondary optimisation* to select between models. For this, we propose to optimise the secondary objective

$$E_2[\tilde{\pi}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \tilde{\pi}(\mathbf{x}_n)\|^2 \quad s.t. \quad \tilde{\pi} \in \arg\min_{\pi'} \left\{E_i[\pi']\right\}. \tag{6.1}$$

In other words, we optimise the standard risk *subject to the model being consistent with the constrained observations*[2].

By performing this additional secondary optimisation we tighten our fit to the available data and avoid models that are not strongly supported by the inconsistency. For example, in Fig. 6.1 (right), optimisation of (6.1) will result in model $\tilde{\pi}_2$ being chosen since this has the lower E_2. Since we have no information about the vertical component of the policy here, choosing this model is the safest strategy since there is little support for $\tilde{\pi}_1$ or $\tilde{\pi}_3$ based on the available data. In effect, this acts like a safety guarantee on the model performance: In the case, that observations are given under an impoverished set of constraints, the model will at worst reproduce the behaviour under those same constraints[3].

Similar to the pure inconsistency optimisation approach (Ch. 5), we can apply the extended approach to many standard regression techniques. However, for the experiments in the remainder of the chapter, we again restrict ourselves to two example classes of function approximator (i) simple parametric models with fixed basis functions (Sec. 6.3.1), and (ii) locally linear models (Sec. 6.3.2). In the following we describe how these two models can be reformulated to take advantage of the new approach.

6.3.1 Parametric Policy Models

As described in Sec. 5.2.3, for the parametric policy model we assume a model of the form $\tilde{\pi}(\mathbf{x}) = \mathbf{W}\mathbf{b}(\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{d\times M}$ is a matrix of weights, and $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^M$ is a vector of fixed basis functions. With this model, the *inconsistency* error from (5.4) becomes

$$\begin{aligned} E_i(\mathbf{W}) &= \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T \mathbf{W}\mathbf{b}(\mathbf{x}_n)\right)^2 \\ &= \sum_{n=1}^{N} \left(r_n - \mathbf{v}_n^T \mathbf{w}\right)^2 = E_i(\mathbf{w}), \end{aligned}$$

[2]It should also be noted that in principle we may choose alternative secondary optimisation functions depending on the application. For example, we may wish to bias solutions toward a particular dynamic behaviour, e.g. stabilising movements, subject to consistency with the demonstrated observations.

[3]This is similar to the minimum performance guarantee reported in (Howard et al., 2008a) for the special case of potential-based policies, now extended to the learning of any arbitrary policy.

where $\mathbf{w} \equiv vec(\mathbf{W})$ and $\mathbf{v}_n \equiv vec(\hat{\mathbf{u}}_n \mathbf{b}(\mathbf{x}_n)^T) = \mathbf{b}(\mathbf{x}_n) \otimes \hat{\mathbf{u}}_n$. Since our objective function is quadratic in $\mathbf{w}$, we can rearrange to give

$$\begin{aligned} E_i(\mathbf{w}) &= \sum_n r_n^2 - 2\sum_n r_n \mathbf{v}_n^T \mathbf{w} + \mathbf{w}^T \sum_n \mathbf{v}_n \mathbf{v}_n^T \mathbf{w} \\ &= E_0 - 2\mathbf{g}^T \mathbf{w} + \mathbf{w}^T \mathbf{H} \mathbf{w} \end{aligned}$$

with $\mathbf{H} = \sum_n \mathbf{v}_n \mathbf{v}_n^T$ and $\mathbf{g} = \sum_n r_n \mathbf{v}_n$. Now, to solve for the optimal weight vector, in the pure inconsistency approach we would take the direct inverse

$$\mathbf{w}_1 = \arg\min E_i(\mathbf{w}) = \mathbf{H}^{-1}\mathbf{g}$$

as described in Sec. 5.2.3. However, this ignores degeneracy in the solutions and may result in over-fitting. To avoid this, here we only optimise on elements of the weight vector that make a significant contribution to the inconsistency error E_i. For this, we perform an eigendecomposition for the inversion

$$\mathbf{w}_1 = \mathbf{V}_1 \Lambda^{-1} \mathbf{V}_1^T \mathbf{g} \tag{6.2}$$

where Λ is a diagonal matrix containing the large eigenvalues of $\mathbf{H}$ (i.e. eigenvalues above some minimum threshold $\lambda \geq \lambda_t$) and the columns of $\mathbf{V}_1$ are the corresponding eigenvectors.

In the part of the parameter space spanned by the remaining small eigenvectors ($\lambda < \lambda_t$) we then perform the secondary optimisation. For the parametric model, we wish to minimise

$$E_2(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{W}\mathbf{b}(\mathbf{x}_n)\|^2 \tag{6.3}$$

subject to the solution being optimal with respect to the inconsistency. We therefore look for a solution that has the form

$$\mathbf{w} = \mathbf{w}_1 + \mathbf{V}_2 \mathbf{z}. \tag{6.4}$$

where the columns of $\mathbf{V}_2$ contain the remaining eigenvectors of $\mathbf{H}$ and $\mathbf{z}$ is a vector. Using solution of this form means that our optimisation of the model with respect to the secondary objective does not affect the primary optimisation of the inconsistency error.

Rearranging (6.3), we have

$$E_2(\mathbf{W}) = \sum_n \mathbf{u}_n^T \mathbf{u}_n - 2\sum_n \mathbf{u}_n^T \mathbf{W}\mathbf{b}_n + \sum_n \|\mathbf{W}\mathbf{b}_n\|^2 \tag{6.5}$$

which can be written in terms of $\mathbf{w}$ as

$$\begin{aligned} E_2(\mathbf{w}) &= \sum_n \mathbf{u}_n^T \mathbf{u}_n - 2\sum_n (\mathbf{b}_n \otimes \mathbf{u}_n^T)\mathbf{w} \\ &\quad + \mathbf{w}^T \left(\sum_n \mathbf{b}_n \mathbf{b}_n^T \otimes \mathbf{I} \right) \mathbf{w} \\ &= E_{0,2} - 2\mathbf{g}_2^T \mathbf{w} + \mathbf{w}^T \mathbf{H}_2 \mathbf{w}. \end{aligned} \tag{6.6}$$

where $E_{0,2} = \sum_n \mathbf{u}_n^T \mathbf{u}_n$, $\mathbf{g}_2 \equiv \sum_n (\mathbf{b}_n \otimes \mathbf{u}_n^T)^T = vec(\mathbf{U}\mathbf{B}^T)$ and $\mathbf{H}_2 \equiv \left(\sum_n \mathbf{b}_n \mathbf{b}_n^T \otimes \mathbf{I}\right) = \mathbf{B}\mathbf{B}^T \otimes \mathbf{I}$.

Substituting (6.4) and differentiating, we can then retrieve the optimal $\mathbf{z}$:

$$\mathbf{z}^{opt} = (\mathbf{V}_2^T \mathbf{H}_2 \mathbf{V}_2)^{-1} \mathbf{V}_2^T (\mathbf{g}_2 - \mathbf{H}_2 \mathbf{w}_1). \tag{6.7}$$

We then combine (6.2) and (6.7) to find the optimal weights for our model

$$\mathbf{w}^{opt} = \mathbf{V}_1 \Lambda^{-1} \mathbf{V}_1^T \mathbf{g} + \mathbf{V}_2 \mathbf{z}^{opt}. \tag{6.8}$$

Finally, in order to automatically select the minimum eigenvalue threshold λ_t we perform a line search, repeating the above optimisation for a series of values of λ_t on a subset of the data, and picking the λ_t which minimises the quantity

$$E_\lambda[\tilde{\pi}] = E_i[\tilde{\pi}] + \alpha E_2[\tilde{\pi}].$$

Here α is a weighting factor that reflects our prior belief on whether the data contains variable constraints. For example, one would choose a very low α for data containing very high variance in the constraints. For convenience, pseudocode for the learning is given in Algorithm 3.

6.3.2 Locally Linear Policy Models

For multiple local linear policy models $\tilde{\pi}_m(\mathbf{x}) = \mathbf{B}_m \bar{\mathbf{x}} = \mathbf{B}_m(\mathbf{x}^T, 1)^T$, the derivation follows similar lines. For a linear model centred at $\mathbf{c}_m$ with an isotropic Gaussian receptive field with variance σ^2, the inconsistency error is given by

$$\begin{aligned} E_i(\mathbf{B}_m) &= \sum_{n=1}^N w_{nm} \left(r_n - \hat{\mathbf{u}}_n^T \mathbf{B}_m \bar{\mathbf{x}}_n \right)^2 \\ &= \sum_{n=1}^N w_{nm} \left(r_n - \mathbf{v}_n^T \mathbf{b}_m \right)^2 = E_i(\mathbf{b}_m) \end{aligned}$$

Algorithm 3 Hybrid Optimisation

```
1: Initialise policy model (e.g., allocate RBF centres $\mathbf{c}_i$ and kernel size $\sigma^2$). Select $\alpha$.
2: Pre-calculation of terms:
     • Find $r_n = \|\mathbf{u}_n\|$, $\hat{\mathbf{u}}_n = \mathbf{u}_n / r_n$, and $\mathbf{v}_n = \mathbf{b}(\mathbf{x}_n) \otimes \hat{\mathbf{u}}$ for each data point.
     • Construct primary Hessian $\mathbf{H} = \sum_n \mathbf{v}_n \mathbf{v}_n^T$, linear term $\mathbf{g} = \sum_n r_n \mathbf{v}_n^T$ and constant term $E_0 = \sum_n r_n^2$. Find eigenvalues $\lambda \in \{\lambda_1, \cdots, \lambda_M\}$ and eigenvectors $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_M]$ of $\mathbf{H}$.
     • Construct secondary Hessian $\mathbf{H}_2 = \mathbf{B}\mathbf{B}^T \otimes \mathbf{I}$, linear term $\mathbf{g}_2 = vec(\mathbf{U}\mathbf{B}^T)$ and constant term $E_{0,2} = \sum_n \mathbf{u}_n^T \mathbf{u}_n$.
3: Optimisation:
     • Repeat for $\lambda_t \in \{\lambda_{min}, \cdots, \lambda_{max}\}$.
         1. Build eigenvalue matrix $\Lambda$ containing all $\lambda \geq \lambda_t$. Split $\mathbf{V}$ into $\mathbf{V}_1$ and $\mathbf{V}_2$ according to eigenvalues.
         2. Find $\mathbf{z}^{opt}$ and $\mathbf{w}_1$. Calculate $\mathbf{w}^{opt} = \mathbf{V}_1 \Lambda^{-1} \mathbf{V}_1^T \mathbf{g} + \mathbf{V}_2 \mathbf{z}^{opt}$ for this $\lambda_t$.
         3. Evaluate $E_\lambda[\tilde{\pi}] = E_i[\tilde{\pi}] + \alpha E_2[\tilde{\pi}]$ on validation data subset.
4: Return weights $\mathbf{w}^{opt}$ that minimise $E_\lambda[\tilde{\pi}]$.
```

where $\mathbf{b}_m = vec(\mathbf{B}_m)$ and $\mathbf{v}_n \equiv vec(\hat{\mathbf{u}}_n \bar{\mathbf{x}}_n^T)$ as described in Sec. 5.2.4. The factors $w_{nm} = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_n - \mathbf{c}_m\|^2)$ weight the importance of each observation $(\mathbf{x}_n, \mathbf{u}_n)$, giving more weight to nearby samples.

The optimal slopes $\mathbf{B}_m$ with respect to (5.7) can again be retrieved using an eigen-decomposition:

$$\mathbf{b}_{1,m} = \arg\min E_i(\mathbf{b}_m) = \mathbf{V}_{1,m} \Lambda_m^{-1} \mathbf{V}_{1,m}^T \mathbf{g}_m \tag{6.9}$$

where Λ_m and $\mathbf{V}_{1,m}$ are the large eigenvalues and corresponding eigenvectors of the Hessian $\mathbf{H}_m = \sum_n w_{nm} \mathbf{v}_n \mathbf{v}_n^T$ for the mth local model and $\mathbf{g}_m = \sum_n w_{nm} r_n \mathbf{v}_n$. We select the number of eigenvalues used for the primary optimisation of the inconsistency using a subset-validation approach similar to the parametric case.

The secondary objective for this model is

$$\begin{aligned} E_2(\mathbf{B}_m) &= \sum_{n=1}^{N} w_{nm}\|\mathbf{u}_n - \mathbf{B}_m\bar{\mathbf{x}}_n\|^2 \\ &= E_{0,2} - 2\mathbf{g}_{2,m}^T\mathbf{b}_m + \mathbf{b}_m^T\mathbf{H}_{2,m}\mathbf{b}_m = E_2(\mathbf{b}_m) \end{aligned}$$

where $E_{0,2} = \sum_n w_{nm}\mathbf{u}_n^T\mathbf{u}_n$, $\mathbf{g}_{2,m} \equiv \sum_n w_{nm}(\bar{\mathbf{x}}_n \otimes \mathbf{u}_n^T)^T$ and $\mathbf{H}_{2,m} \equiv \left(\sum_n w_{nm}\bar{\mathbf{x}}_\mathbf{n}\bar{\mathbf{x}}_\mathbf{n}{}^T \otimes \mathbf{I}\right)$. Similar to the parametric case, we look for a solution of the form $\mathbf{b}_m = \mathbf{b}_{1,m} + \mathbf{V}_{2,m}\mathbf{z}_m$. This yields optimal weights

$$\mathbf{b}_m^{opt} = \mathbf{V}_{1,m}\Lambda_m^{-1}\mathbf{V}_{1,m}^T\mathbf{g}_m + \mathbf{V}_{2,m}\mathbf{z}_m^{opt} \tag{6.10}$$

with

$$\mathbf{z}_m^{opt} = (\mathbf{V}_{2,m}^T\mathbf{H}_{2,m}\mathbf{V}_{2,m})^{-1}\mathbf{V}_{2,m}^T(\mathbf{g}_{2,m} - \mathbf{H}_{2,m}\mathbf{b}_{1,m}). \tag{6.11}$$

Finally, for predicting the global policy, we combine the local linear models using the convex combination

$$\tilde{\pi}(\mathbf{x}) = \frac{\sum_{m=1}^{M} w_m\mathbf{B}_m\bar{\mathbf{x}}}{\sum_{m=1}^{M} w_m}; \quad w_m = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{c}_m\|^2\right).$$

For implementation, the pseudocode in Algorithm 3 can be used for each local model with appropriate substitutions to incorporate the weighting factors w_m in the calculation of objective function terms (i.e., the primary and secondary Hessian $\mathbf{H}_m, \mathbf{H}_{2,m}$, linear terms $\mathbf{g}_m, \mathbf{g}_{2,m}$ and constant terms $E_0, E_{0,2}$).

6.4 Experiments

In this section we report experiments exploring the performance of the new approach when learning on data from systems of varying complexity and size. First, in order to illustrate the concepts involved, we apply our method to data from a simulated 2-D toy system. We then test the scalability of the method to higher dimensional systems with more complex constraints using data from the joint-space of the 7-DOF DLR lightweight arm (Fig. 1.1). Finally we re-visit the car-washing experiment (ref. Sec. 5.3.5) in order to demonstrate the utility of our approach.

6.4.1 Toy Example

Our first experiment demonstrates the robustness of the new approach for learning unconstrained policies from variable-constraint data. For this, we re-used the toy example

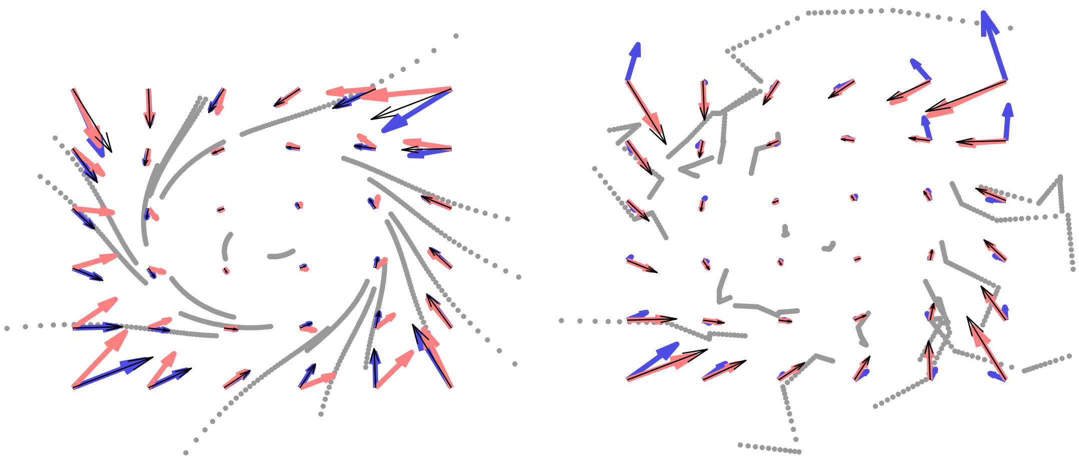


Figure 6.2: Policy learnt with the direct approach (blue) and pure inconsistency approach (red) when training on unconstrained (left) and randomly constrained (right) data. The true policy (thin black arrows) and training data (grey trajectories) are overlaid.

from Sec. 5.3.1, i.e. the simple two-dimensional limit-cycle system with discontinuously switching motion constraints. However, here, in addition to collecting data from the policy subject to random 1-D constraints (ref. Sec. 5.3.1), we also recorded trajectories from the unconstrained policy from the same start states. In Fig. 6.2, examples of the unconstrained (left) and constrained (right) trajectories are shown in grey.

We used a parametric model to learn the policy through the hybrid optimisation approach as described in Sec. 6.3.1. For this toy problem, we chose our function model as a set of 36 normalised RBFs centred on a 6×6 grid, and we simply fixed the kernel width to yield suitable overlap. We repeated this experiment on 100 data sets and evaluated the normalised UPE and CPE (ref. Appendix 3.4.2) and the inconsistency, divided by the number of data points and the variance of the policy π_n on a subset held out for testing. For comparison, we repeated the experiment using (i) direct regression on the observations (i.e. minimising (5.1)) and (ii) optimisation of the inconsistency alone (i.e. minimising the functional (5.4) without the secondary optimisation step) with the same RBF model.

Table 6.1 shows the results of learning with the different methods under the different constraint settings. Looking at the first row, we see that the direct regression approach is effective for learning on unconstrained data, but performs poorly on data containing random constraints. This is in line with expectations since for the former

Method	Constr.	nUPE	nCPE	Norm. Incon.
Direct	None	0.034 ± 0.044	0.034 ± 0.044	0.026 ± 0.039
	Rand.	58.338 ± 9.556	8.596 ± 2.813	8.596 ± 2.813
Incon.	None	26.640 ± 52.737	26.640 ± 52.737	0.014 ± 0.031
	Rand.	0.118 ± 0.162	0.007 ± 0.010	0.007 ± 0.010
Hybrid	None	0.065 ± 0.268	0.065 ± 0.268	0.042 ± 0.143
	Rand.	0.373 ± 1.109	0.011 ± 0.017	0.011 ± 0.017

Table 6.1: Error for the direct, inconsistency and hybrid optimisation approaches when learning on $K = 40$ trajectories of length $N = 40$ points, sampled from the limit cycle policy. All values given as (mean$\pm$s.d.)$\times 10^{-2}$

the data is unaffected by constraints and is thus already consistent (i.e. a unique output is observed at each point in the input space), whereas for the latter the variability in the constraints causes model averaging. In contrast, looking at the second row we see that optimisation of the inconsistency is highly effective for learning the unconstrained policy when there is high variation in the constraints. However, on the unconstrained data, though the normalised inconsistency (5th column) is low, the policy errors are relatively large. The pure inconsistency approach *misinterprets the variation in the policy as variation in the constraints*, and fits an incorrect model (shown in red in Fig. 6.2).

In contrast, the proposed hybrid approach achieves very low errors both on the unconstrained and the constrained data. With this approach we get the best of both of the other approaches: For data that is already self-consistent it benefits from the tight fit offered by direct least-squares regression. Conversely, if data contains variable constraints a model that is consistent with the observations under the different constraints is learnt.

To further test this, we repeated the experiment on data containing several levels of variability in the constraints. For this we again sampled a set of $K = 40$ trajectories of length $N = 40$ points from the limit cycle policy, however this time we applied the constraints

$$\mathbf{A}(\mathbf{x},t) = \mathbf{I} - \hat{\alpha}_\pi^T \hat{\alpha}_\pi \tag{6.12}$$

where $\hat{\alpha}_\pi \equiv \alpha_\pi / \|\alpha_\pi\|$, $\alpha_\pi \equiv \mathbf{R}(\theta)\pi(\mathbf{x})$ and $\mathbf{R}(\theta)$ is a rotation matrix with rotation angle θ. The latter was drawn uniform-randomly with increasing angular range, that is $\theta \sim U[-\theta^{max}, \theta^{max}]$ for increasing θ^{max}. This constraint was chosen since it allows us to

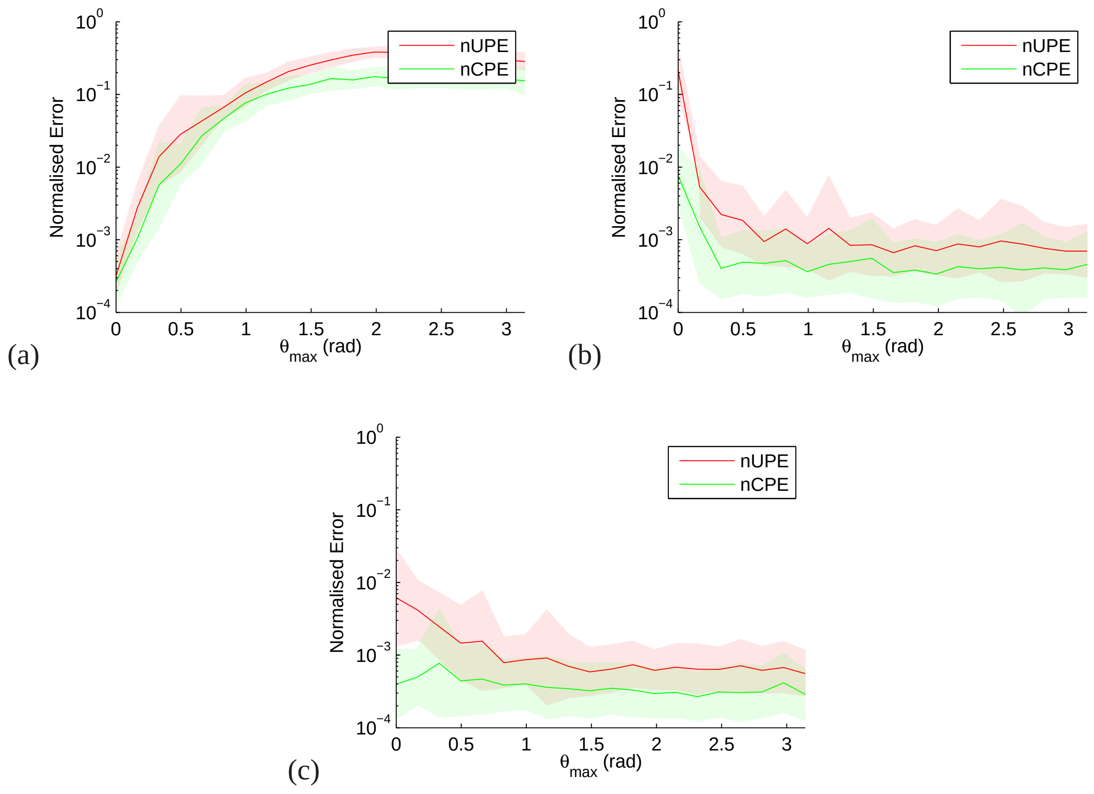


Figure 6.3: Normalised UPE and CPE versus variance in the constraints for learning with the (a) direct, (b) pure inconsistency and (c) hybrid optimisation approaches.

smoothly vary the effect of the constraints on the observations. For example, for $\theta = 0$ the direction of the constraint is exactly orthogonal to the policy at that point so that the resultant projection has no effect on the policy. As the range of θ increases however, the observations of the unconstrained policy are increasingly corrupted by the projections induced by the constraints.

Fig. 6.3 depicts how the UPE and CPE evolve with increasing constraint variance (i.e. increasing θ^{max}) for the direct, pure inconsistency and hybrid optimisation approaches (please note the log. scale). For the direct approach, the UPE and CPE are low when the constraint variance is low, but rapidly increase as the variance grows due to increased model-averaging. In contrast, the pure inconsistency approach deals well with constraints of high variance since this increases the span of the observations, resulting in most of the components of the policy being picked up by the inconsistency error. However, when the variance in constraints decreases, the pure inconsistency approach misinterprets the remaining variability in the observations (due to variation in the policy) as variation in the constraints, causing an increase in error.

Finally, the proposed hybrid approach achieves consistently low errors irrespective of the variance in the constraints, by automatically selecting the direct least-squares fit for low-variance constraints, and increasingly using the constraint-consistent fit for high-variance constraints. This automatic selection is fairly robust across the range of variances in the constraints seen. However, comparing the error for the very low variance constraints for the direct and hybrid methods (near $\theta_{max} = 0$ in Fig. 6.3(a) and (c)) the error is somewhat higher for the hybrid approach. We attribute this to a slight tendency to favour the constraint-consistent fit in cases where the data is ambiguous as to whether it is constrained or not, causing an increase in the average error over the 100 trials. This effect may be removed with an improved model selection method.

6.4.2 Higher Dimensional Policies and Constraints

The goal of our second set of experiments was to evaluate the scalability of the hybrid approach to higher dimensional systems with constraints of varying dimensionality. This is important when considering systems with many degrees of freedom and where the dimensionality of constraints may switch; for example, when switching between control of the position of an end-effector to control of the combined position and orientation. It is also the case that with increasing numbers of dimensions there are increasing numbers of ways in which the system can be constrained, in terms both of the different dimensionalities of the constraints (i.e. rank of the constraint matrix) and the ways in which constraints can be combined.

For this experiment, kinematic data from the 7-DOF DLR lightweight robot (Fig. 1.1) was again used. Similar to the experiment in Sec. 5.3.2, data was collected in the form of 100 trajectories of 100 points each, starting from random initial postures (drawn uniform-randomly from half the range of each joint, i.e., $x_i \sim U[-0.5x_i^{max}; 0.5x_i^{max}]$) following the $p = 1.8$ joint limit avoidance policy, i.e.,

$$\pi(\mathbf{x}) = -0.05\nabla\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = \sum_{i=1}^{7} |x_i|^{1.8}$$

under different constraints. This time trajectories were collected under 6 different constraints of differing dimensionality, which we refer to as 1, 1-2, 1-2-3, etc., where again the numbers denote which end-effector coordinates in task space were kept fixed. For example, 1-2-3 means the end-effector position was constrained, but arbitrary changes in the orientation were allowed. Similarly, 1-2-3-4 means the end-effector position and the orientation around the x-axis were constrained, while movement around the y and z axes was permitted. For all constraint types, the policy was estimated from a training subset, and the normalised CPE on test data from the same constraint, as well as the normalised UPE were evaluated.

For learning in the 7-D state space, we selected locally linear models as described in Sec. 6.3.2, where we chose rather wide receptive fields (fixing $\sigma^2 = 3$) and placed the centres $\{\mathbf{c}_m\}$ of the local models such that every training sample $(\mathbf{x}_n, \mathbf{u}_n)$ was weighted within at least one receptive field with $w_m(\mathbf{x}_n) \geq 0.7$. On average, this yielded about 50 local models.

The results are shown in Table 6.2 where we can see the following trends. First, as the constraint dimension increases, learning with the direct approach yields increasingly poor performance in terms of UPE and roughly consistent performance in terms of CPE. This is to be expected since, being naive to the effect of constraints, the direct approach attempts to find the closest fit to the constrained observations. Further, as the number of constraints increases the difference between the constrained and unconstrained policy vectors increases (since the number of components of the unconstrained policy projected out by the constraints increases). As a result the directly learnt model, while fitting the constrained policy closely, performs increasingly poorly in terms of UPE.

Second, for the pure inconsistency approach, we see that the CPE is worse for the 1-D constraint compared to the direct approach, but much better for the higher dimensional constraints. We also see much better performance in terms of the UPE for the intermediate constraints, but very large errors for the 6-D constraint. For the hybrid approach the UPE is uniformly better, and the CPE lower in all but the 1-D constraint case.

The improved UPE performance for these methods may be surprising given that the same constraint is applied for each observation. This would suggest that certain components of the policy are undetermined by the observations since they are never unconstrained. However, here the constraint matrix (i.e the Jacobian) is state-dependent, yielding some *spatial variability* in the constraints, and thereby sufficient information to improve the reconstruction of the unconstrained policy.

Looking at the inconsistency and hybrid approaches, we see that performance (especially in terms of CPE) increases with constraint dimensionality which can be explained by the approximation of the projection (as discussed in Sec. 5.2.2) becoming increasingly accurate. In fact, for the 6-D constraint the approximation is exact.

However, for this latter constraint, we see a huge increase in UPE for the pure inconsistency approach which is not seen for the hybrid approach. We attribute this to the combined spatial variation in the policy and the constraints in this particular case, to which the inconsistency approach is overly sensitive. On inspection we noted that the Hessian matrices of the local models had become ill-conditioned in this case. The secondary optimisation in the hybrid approach avoids this problem and emphatically outperforms the two other approaches.

6.4.3 Car Washing Experiment Revisited

Having validated our approach on data where the true unconstrained policy and constraints in force (i.e. the ground truth) were known, in this section we report experiments applying the hybrid approach to learning from human demonstration data. For this, we chose to re-visit the car-washing experiment described in Sec. 5.3.5.

Seven demonstrations of a human wiping different surfaces with a sponge were used to train a local linear model as described in Sec. 6.3.2. For learning we used a fixed kernel width of $\sigma^2 = 0.025$, and centres placed so that every data point was weighted with at least $w_m(\mathbf{x}_n) \geq 0.7$. For this data set this yielded about 22 local models.

We evaluated performance on a behavioural level by implementing the resultant policy on the DLR Lightweight arm (see Fig. 6.4). A simple Resolved Motion Rate Control (ref. Sec. 3.2.1) inverse kinematics controller (Liégeois, 1977; Whitney, 1969) was used to realise the policy motion in end-effector space and, similar to Sec. 5.3.5, we assumed constraints of the form $\mathbf{A}_j(\mathbf{x},t) = \hat{\mathbf{n}}_j$ where $\hat{\mathbf{n}}_j$ is the normal to the jth surface. That is, the constraints ensured that the sponge did not penetrate the surface and would not be lifted from the surface. Similar to our previous result using the inconsistency approach (ref. Sec. 5.3.5), the policy learnt by the hybrid approach produced a smooth, periodic trajectory closely resembling that of the human (see accompanying video).

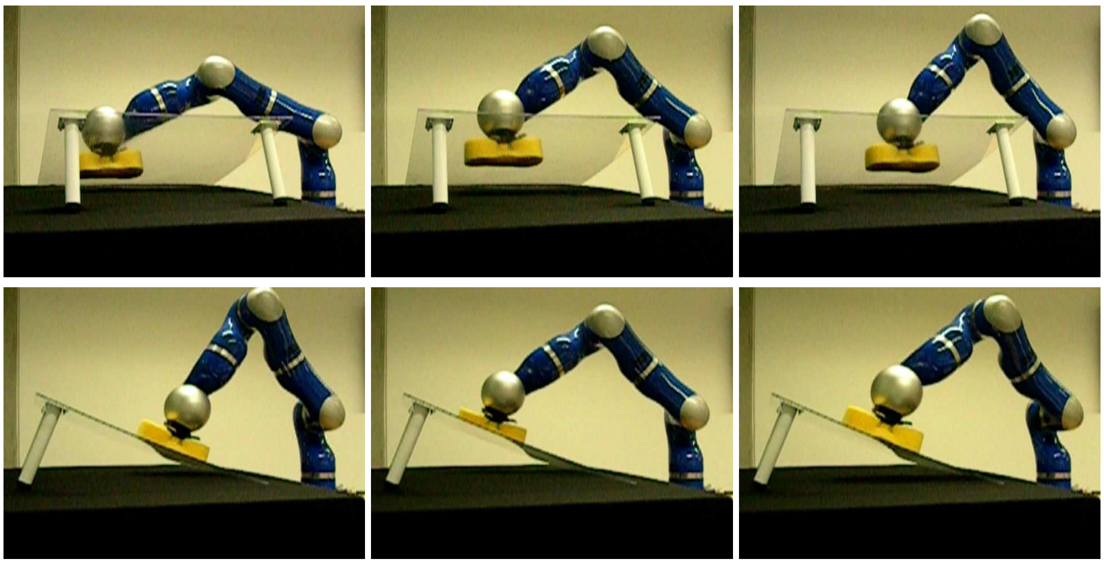


Figure 6.4: Reproduction of the car washing movement on the DLR Lightweight arm on a training constraint (top row) and an unseen test constraint (bottom row).

6.5 Conclusion

In this chapter, we extended the method proposed in Ch. 5 to improve robustness when learning policies from constrained observations. Building upon that method, we introduced a two-stage optimisation approach which seamlessly combines standard direct policy learning with the idea of fitting a model that is consistent with variable constraint data. Although the previous approach could handle cases where demonstrated movements are subject to variable, dynamic, non-linear and even discontinuous constraints, it suffered from poor performance on data containing highly correlated constraints or purely unconstrained data. The novel approach proposed here avoids these problems as demonstrated in our experiments.

In the next chapter, we summarise and give conclusions on the work undertaken in this thesis and suggest directions for future work.

Method	Constr.	nUPE	nCPE
Direct	1	26.94 ± 3.02	3.63 ± 0.54
	1 - 2	70.51 ± 2.22	5.72 ± 0.66
	1 - 2 - 3	80.70 ± 1.59	4.09 ± 0.33
	1 -. . . - 4	86.63 ± 1.36	4.66 ± 0.44
	1 -. . . - 5	91.47 ± 0.91	3.59 ± 0.39
	1 -. . . - 6	96.78 ± 0.78	1.85 ± 0.27
Incon.	1	18.30 ± 5.46	14.53 ± 5.08
	1 - 2	6.53 ± 2.90	1.04 ± 0.37
	1 - 2 - 3	6.93 ± 2.79	0.50 ± 0.11
	1 -. . . - 4	4.57 ± 2.49	0.27 ± 0.02
	1 -. . . - 5	5.28 ± 3.40	0.16 ± 0.02
	1 -. . . - 6	233.37 ± 136.97	0.04 ± 0.01
Hybrid	1	10.54 ± 4.56	6.98 ± 3.90
	1 - 2	5.85 ± 1.94	1.00 ± 0.30
	1 - 2 - 3	18.17 ± 8.00	0.55 ± 0.14
	1 -. . . - 4	8.04 ± 4.16	0.28 ± 0.03
	1 -. . . - 5	8.98 ± 5.25	0.18 ± 0.03
	1 -. . . - 6	41.30 ± 3.93	0.05 ± 0.01

Table 6.2: Normalised UPE and CPE for the three methods when training on data from the DLR arm. All errors normalised by the variance of the policy. We report (mean $\pm$ s.d.)$\times 10^{-2}$ over 50 trials with different data sets.

Chapter 7

Conclusions

In this thesis, we have explored the problem of learning control policies from constrained movement data with the aim of behaviour imitation and transfer from humans to robots. We have discussed several examples of human skills that can be framed in terms of performing some task subject to variable constraints, and shown that in many cases these constraints are unobservable from the data and frequently change between contexts.

In Chapter 2 we reviewed several state of the art methods for learning from movement data and their suitability for learning in this setting. In particular, we showed that few of these methods explicitly consider the effect of constraints on observed data, and commonly unconstrained or consistently constrained data is used when evaluating these methods. Furthermore, we saw that in the studies that do explicitly consider movement constraints, these usually only consider data containing the same consistent constraints in all observations, and cannot handle the effect of *constraint variability*.

In Chapter 3 (Howard et al., 2006; Howard and Vijayakumar, 2007) we outlined a model for constrained motion based on recent work in analytical dynamics. In the light of this model, we analysed the way in which constraints affect the kinematics and dynamics of movement and discussed the implications this has for learning under several different classes of constraint. In the remaining chapters we then went on to propose several methods for learning from variable constraint data for the class of stationary movement constraints.

In Chapter 4, (Howard et al., 2008b,a) we showed that an effective method for representing constrained movements is to learn the underlying unconstrained policy. We discussed how this can be done without need for explicit knowledge of the constraints by looking for a model that is consistent with the observations under the constraints. Furthermore, we proposed a method for doing this for the special case of potential-based policies from constrained data, based on forming local models of the potential and aligning these for global prediction. In our experiments we demonstrated robust learning on data containing variable, non-linear and even discontinuous constraints for several problems of varying size and complexity.

In Chapter 5 (Howard et al., 2009a,b) we then extended this approach by removing the restriction to potential-based policies. We proposed a novel method for learning generic policies from constrained observations based on a small, but very effective modification of the standard risk. This enabled us to learn arbitrary policies from constrained data, again without explicit knowledge of the constraints. We tested the performance of the approach on various systems, including learning from human demonstration data. The novel approach showed a significant improvement in performance over standard direct regression techniques, and also outperformed the potential-based approach.

Finally, in Chapter 6 (Howard et al., 2009c) we identified several situations where the method proposed in Ch. 5 has difficulties in learning, in particular when data contains invariant or highly correlated constraints. We then presented an extension to the method aimed at improving robustness in these situations (Howard et al., 2009c), based on a two-step optimisation approach. By applying this extension we were able to seamlessly integrate constraint-consistent learning with standard direct regression approaches, eliminating the problem of invariant constraints.

Outlook & Future Work

There exist a number of directions in which the work presented in this thesis may be extended in future work.

Constrained Dynamics

In all of the experiments presented in this thesis data was used from constrained kinematic policies, that is, mappings from positions to velocities either in joint space or Cartesian space. This was partly in order to keep the explanations and analysis simple, and partly due to technical limitations, e.g. a lack of force control or sensing on our robotic platforms. However, as discussed in Ch. 3, the constraint formalism used in this thesis is generic and can be applied to a wide variety of other systems (Udwadia and Kalaba, 1996). Example systems include higher-order kinematic control policies (i.e. control of accelerations or jerk), dynamic control policies (i.e. control of forces and torques) and the passive dynamics of several systems. An interesting direction of future work then, could be to apply the approaches developed here to learning from such systems. For example, in the window wiping task, one might also use information from the normal forces applied to the wiping surface for learning the policy.

Alternative Constraint Types

Each of the algorithms presented in this thesis deals successfully with the problem of variable, non-linear stationary constraints in the data, i.e. those that can be described in terms of the formalism outlined in Sec. 3.3.2. However, there are a number of alternative constraint types that they cannot currently handle. For example, systems with moving or 'forced action' constraints can cause additional difficulties as described in Sec. 3.3.3. Another example is that of constraints on movement duration (as distinct from constraints that are time-dependent in the sense of changing during the movement, ref. Ch. 3), which may have different effects on the observed actions. For example, a stringent constraint on the time permitted for task execution may mean that commands are scaled up to produce a quicker movement. For such alternative constraint types new learning methods may be developed to complement the approaches proposed in this thesis.

Improved Learning Theory

The learning algorithms proposed in this thesis, in particular those presented in Ch. 5 and Ch. 6 constitute a non-standard form of regression. While the experiments are testament to their good performance, there are still a number of open issues remaining in terms of theoretical predictions of performance. In particular, an interesting direction of future work could be to attempt to derive error bounds or confidence intervals on the learnt policies. Such bounds could used to improve the strategy for selecting between learning approaches (e.g. constraint-consistent learning versus direct regression, cf. Ch. 6). Another possibility could be to reformulate the current strategy based on least-squares optimisation into the full Bayesian framework.

Extensive Experiments on Human Data

In Ch. 5 and Ch. 6, we presented experiments that illustrated how the proposed approaches can be used for transferring behaviour from human demonstration to generate movements on the robot. We investigated the task of wiping under different constraints as induced by the shape and orientation of the wiping surface. We found that our methods enabled us to transfer the wiping to two robot platforms in a way that generalised over the constraints. However, while the resultant motions were qualitatively similar to the human demonstrations, there is still much to be done in terms of quantitative evaluations. In particular, an interesting direction of future work would be to perform extensive experiments on human data for a series of different everyday constrained tasks, such as opening doors, stirring soup in a pan and grinding coffee in a coffee grinder.

Modelling Adaptation to Constraints

In Ch. 2 we mentioned how, under the assumption of an invariant set of constraints, models of human and robotic adaptation have been developed. A further direction of future work could be to look at when and how adaptation may proceed in the presence of uncertain and variable constraints. For example, the inconsistency error (5.4) can be used as a distance metric by which to measure differences in behaviour (as represented by policies), up to a difference in constraints. Given a model of behaviour in one context (e.g. walking in an office environment) we could potentially use that model to measure how much, and in what ways that behaviour differs (i.e., is adapted) under new constraints (e.g. walking on a paved street versus walking in a rough, ploughed field). This could potentially provide better insight into how learnt behaviours can be transferred from one constraint setting to another, avoiding the need to completely re-plan the behaviour from scratch.

Bibliography

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*.

Abbeel, P. and Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. In *International Conference on Machine Learning*.

Aleotti, J. and Caselli, S. (2006). Robust trajectory learning and approximation for robot programming by demonstration. *Robotics and Autonomous Systems*, 54(5):409–413. The Social Mechanisms of Robot Programming from Demonstration.

Alissandrakis, A., Nehaniv, C., and Dautenhahn, K. (2002). Imitation with alice: learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man and Cybernetics*, 32(4):482–496.

Alissandrakis, A., Nehaniv, C., and Dautenhahn, K. (2007). Correspondence mapping induced state and action metrics for robotic imitation. *IEEE Transactions on Systems, Man and Cybernetics*, 37(2):299–307.

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2005). The null-space-based behavioral control for soccer-playing mobile robots. In *IEEE International Conference Advanced Intelligent Mechatronics*, pages 1257–1262.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2008). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, In Press, Corrected Proof:–.

Asfour, T., Gyarfas, F., Azad, P., and Dillmann, R. (2006). Imitation learning of dual-arm manipulation tasks in humanoid robots. In *IEEE International Conference on Humanoid Robots*.

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113.

Atkeson, C. G. and Schaal, S. (1997). Robot learning from demonstration. In *International Conference on Machine Learning*.

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control*. Athena Scientific, 3 edition.

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2007). Robot programming by demonstration. In *Handbook of Robotics*, volume chapter 59. MIT Press.

Boas, M. L. (2006). *Mathematical Methods in the Physical Sciences*. Wiley, 3 edition.

Bolder, B., Dunn, M., Gienger, M., Janssen, H., Sugiura, H., and Goerick, C. (2007). Visually guided whole body interaction. In *IEEE International Conference on Robotics and Automation*, pages 3054–3061.

Brillinger, D. R. (2007). Learning a potential function from a trajectory. *IEEE Signal Processing Letters*, 14(11):867–870.

Bruyninckx, H. and Khatib, O. (2000). gauss' principle and the dynamics of redundant and constrained manipulators. In *IEEE International Conference on Robotics and Automation*, pages 2563–2568.

Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods.

Calinon, S. and Billard, A. (2007). Learning of gestures by imitation in a humanoid robot. In *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*.

Calinon, S. and Billard, A. (2008). A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *IEEE International Conference on Intelligent Robots and Systems*.

Calinon, S., Guenter, F., and Billard., A. (2005). Goal-directed imitation in a humanoid robot. In *IEEE International Conference on Robotics and Automation*.

Casti, J. (1980). On the general inverse problem of optimal control theory. *Journal of Optimization Theory and Applications*, 32(4):491–497.

Chajewska, U., Getoor, L., Norman, J., and Shahar, Y. (1998). Utility elicitation as a classification problem. In *Uncertainty in Artificial Intelligence*, pages 79–88, San Francisco. Morgan Kaufmann Publishers.

Chajewska, U., Koller, D., and Ormoneit, D. (2001). Learning an agent's utility function by observing behavior. In *International Conference on Machine Learning*.

Chalodhorn, R., Grimes, D. B., Maganis, G. Y., Rao, R. P., and Asada, M. (2006). Learning humanoid motion dynamics through sensory-motor mapping in reduced dimensional space. In *IEEE International Conference on Robotics and Automation*.

Chaumette, F. and Marchand, A. (2001). A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. *IEEE Trans. Robotics and Automation*, 17(5):719–730.

Chiacchio, P., Chiaverini, S., Sciavicco, L., and Sicilian, B. (1991). Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *The International Journal of Robotics Research*, 10(4):410–425.

Choi, S. I. and Kim, B. K. (2000). Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, 18(2):143–151.

Conner, D., Rizzi, A., and Choset, H. (2003). Composition of local potential functions for global robot control and navigation. In *IEEE International Conference on Intelligent Robots and Systems*, volume 4, pages 3546–3551.

Degallier, S., Santos, C. P., Righetti, L., and Ijspeert, A. (2006). Movement generation using dynamical systems: A humanoid robot performing a drumming task. In *IEEE International Conference on Humanoid Robots*.

Delson, N. and West, H. (1993). Robot programming by human demonstration: Subtask compliance controller identification. In *IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 33–41.

Delson, N. and West, H. (1994a). Robot programming by human demonstration: the use of human inconsistency in improving 3d robot trajectories. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1248–1255.

Delson, N. and West, H. (1994b). Robot programming by human demonstration: the use of human variation in identifying obstacle free trajectories. In *IEEE International Conference on Robotics and Automation*, pages 564–571.

Delson, N. and West, H. (1996). Robot programming by human demonstration: adaptation and inconsistency in constrained motion. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 30–36.

Dietterich, T. (2002). Machine learning for sequential data: A review. In Caelli, T., editor, *Lecture Notes in Computer Science*. Springer-Verlag.

English, J. and Maciejewski, A. (2000). On the implementation of velocity control for kinematically redundant manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 30(3):233–237.

Freire da Silva, V., Reali Costa, A., and Lima, P. (2006). Inverse reinforcement learning with evaluation. In *IEEE International Conference on Robotics and Automation*, pages 4246–4251.

Gienger, M., Janssen, H., and Goerick, C. (2005). Task-oriented whole body motion for humanoid robots. In *IEEE International Conference on Humanoid Robots*, pages 238–244.

Grimes, D. B., Chalodhorn, R., and Rao, R. P. N. (2006). Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Robotics: Science and Systems*, Cambridge, MA. MIT Press.

Grimes, D. B., Rashid, D. R., and Rao, R. P. N. (2007). Learning nonparametric models for probabilistic imitation. In *Advances in Neural Information Processing Systems*, Cambridge, MA. MIT Press.

Grochow, K., Martin, S. L., Hertzmann, A., and Popovic, Z. (2004). Style-based inverse kinematics. In *ACM Transactions on Graphics*.

Guenter, F., Hersch, M., Calinon, S., and Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *RSJ Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521–1544.

Hersch, M., Guenter, F., Calinon, S., and Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*. Accepted.

Howard, M., Gienger, M., Goerick, C., and Vijayakumar, S. (2006). Learning utility surfaces for movement selection. In *IEEE International Conference on Robotics and Biomimetics*.

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2008a). Behaviour generation in humanoids by learning potential-based policies from constrained motion. *Applied Bionics and Biomechanics*, 5(4):195–211. (in press).

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2008b). Learning potential-based policies from constrained motion. In *IEEE International Conference on Humanoid Robots*.

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009a). A novel method for learning policies from constrained motion. In *IEEE International Conference on Robotics and Automation*.

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009b). A novel method for learning policies from variable constraint data. *Autonomous Robots*. (submitted).

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. (2009c). Robust constraint-consistent learning. In *IEEE International Conference on Intelligent Robots and Systems*.

Howard, M. and Vijayakumar, S. (2007). Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In *Workshop on Robotics and Mathematics*.

Hsu, E., Pulli, K., and Popović, J. (2005). Style translation for human motion. In *SIGGRAPH*, pages 1082–1089, New York, NY, USA. ACM.

Ijspeert, A., Nakanishi, J., and Schaal, S. (2001). Trajectory formation for imitation with nonlinear dynamical systems. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 752–757.

Ijspeert, A. J. (2001). A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 84:331–348.

Ijspeert, A. J. and Cabelguen, J.-M. (2006). Gait transition from swimming to walking: Investigation of salamander locomotion control using nonlinear oscillators. In *Adaptive Motion of Animals and Machines*, pages 177–188. Springer Tokyo.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002a). Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE International Conference on Intelligent Robots and Systems*, pages 958–963.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002b). Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation*, pages 1398–1403. ICRA2002 best paper award.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, pages 1523–1530, Cambridge, MA. MIT Press.

Inamura, T., Kojo, N., Sonoda, T., Sakamoto, K., Okada, K., and Inaba, M. (2005). Intent imitation using wearable motion capturing system with on-line teaching of task attention. In *IEEE International Conference on Humanoid Robots*, pages 469–474.

Inamura, T. and Nakamura, Y. (2003). Acquiring motion elements for bidirectional computation of motion recognition and generation. In *Springer Tracts in Advanced Robotics: Experimental Robotics VIII*, pages 372–381. Springer.

Inamura, T., Nakamura, Y., and Simozaki, M. (2002). Associative computational model of mirror neurons that connects missing link between behaviors and symbols. In *IEEE International Conference on Intelligent Robots and Systems*.

Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4):363–377.

Itiki, C., Kalaba, R., and Udwadia, F. (1996). Inequality constraints in the process of jumping. *Applied Mathematics and Computation*, 78:163–173.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In *IEEE Int. Conf. on Intelligent Robots and Systems*.

Kalman, R. (1964). When is a linear system optimal? *ASME Journal of Basic Engineering*, 86:51–60.

Kannan, R., Vempala, S., and Vetta, A. (2004). On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 428–436.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE Journal of Robotics and Automation*, RA-3(1):43–53.

Khatib, O., Warren, J., De Sapio, V., and Sentis, L. (2004). Human-like motion from physiologically-based potential energies. In Lenarcic, J. and Galletti, C., editors, *On Advances in Robot Kinematics*, pages 149–163. Kluwer Academic Publishers.

Kolter, J. Z., Abbeel, P., and Ng, A. (2008). Hierarchical apprenticeship learning with application to quadruped locomotion. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, Cambridge, MA. MIT Press.

Körding, K., Fukunaga, I., Howard, I., Ingram, J., and Wolpert, D. (2004). A neuroeconomics approach to inferring utility functions in sensorimotor control. *PLoS Biolology*, 2(10):330.

Körding, K. and Wolpert, D. (2004). The loss function of sensorimotor learning. In *Proceedings of the National Academy of Sciences*, volume 101, pages 9839–42.

Lee, D. and Nakamura, Y. (2006). Motion recognition with a monocular vision system based on mimesis scheme. In *24th Annual Conference of the Robotics Society of Japan*.

Lee, D. and Nakamura, Y. (2007). Mimesis scheme using a monocular vision system on a humanoid. In *IEEE International Conference on Robotics and Automation*.

Liégeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Sys., Man and Cybernetics*, 7:868–871.

Maire, F. and Bulitko, V. (2005). Apprenticeship learning for initial value functions in reinforcement learning. In *International Joint Conference on Artificial Intelligence*.

Mattikalli, R. and Khosla, P. (1992). Motion constraints from contact geometry: Representation and analysis. In *IEEE International Conference on Robotics and Automation*.

Morita, M. (1996). Memory and learning of sequential patterns by nonmonotone neural networks. *Neural Networks*, 9(8):1477–1489.

Morita, M. and Murakami, S. (1997). Recognition of spatiotemporal patterns by nonmonotone neural networks. In *International Conference on Neural Information Processing*.

Murray, R. M., Li, Z., and Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*. Addison Wesley, Reading, MA.

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., and Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79–91. Robot Learning from Demonstration.

Neu, G. and Szepesvári, C. (2007). Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Uncertainty in Artificial Intelligence*, pages 295–302.

Ng, A. Y. (2006). Reinforcement learning and apprenticeship learning for robotic control. In Ng, A. Y., editor, *Algorithmic Learning Theory*, pages 29–31. Springer Berlin / Heidelberg.

Ng, A. Y. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann, San Francisco, CA.

Ogawara, K., Takamatsu, J., Kimura, H., and Ikeuchi, K. (2002). Generation of a task model by integrating multiple observations of human demonstrations. In *IEEE International Conference on Robotics and Automation*.

Ohnishi, N. and Imiya, A. (2007). Corridor navigation and obstacle avoidance using visual potential for mobile robot. In *Canadian Conference on Computer and Robot Vision*, pages 131–138.

Ohta, K., Svinin, M., Luo, Z., Hosoe, S., and Laboissiere, R. (2004). Optimal trajectory formation of constrained human arm reaching movements. *Biological Cybernetics*, 91:23–36.

Park, D.-H., Hoffmann, H., Pastor, P., and Schaal, S. (2008). Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *IEEE International Conference on Humanoid Robots*.

Park, J. and Khatib, O. (2006). Contact consistent control framework for humanoid robots. In *IEEE International Conference on Robotics and Automation*.

Peters, J., Mistry, M., Udwadia, F., Cory, R., Nakanishi, J., and Schaal, S. (2005). A unifying methodology for the control of robotic systems. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1575–1582.

Peters, J., Mistry, M., Udwadia, F. E., Nakanishi, J., and Schaal, S. (2008). A unifying framework for robot control with redundant dofs. *Autonomous Robots*, 24:1–12.

Peters, J. and Schaal, S. (2008a). Learning to control in operational space. *The International Journal of Robotics Research*, 27(2):197–212.

Peters, J. and Schaal, S. (2008b). Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190.

Ramachandran, D. and Amir, E. (2006). Bayesian inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence*.

Ratliff, N., Bagnell, J., and Zinkevich, M. (2006a). Maximum margin planning. In *International Conference on Machine Learning*.

Ratliff, N., Bagnell, J., and Zinkevich, M. (2007). (online) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*.

Ratliff, N., Bradley, D., Bagnell, J. A., and Chestnutt, J. (2006b). Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*.

Ren, J., McIsaac, K. A., and Patel, R. V. (2006). Modified Newton's method applied to potential field-based navigation for mobile robots. *IEEE Transactions on Robotics*.

Rimon, E. and Koditschek, D. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518.

Russell, S. (1998). learning agents for uncertain environments. online (extended abstract).

Sapio, V. D., Khatib, O., and Delp, S. (2006). Task-level approaches for the control of constrained multibody systems. *Multibody System Dynamics*, 16:73–102.

Sapio, V. D., Warren, J., Khatib, O., and Delp, S. (2005). simulating the task-level control of human motion: a methodology and framework for implementation. *The Visual Computer*, 21(5):289–302.

Schaal, S. (1997). Learning from demonstration. In Mozer, M., Jordan, M., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, pages 1040–1046, Cambridge, MA. MIT Press.

Schaal, S. (2006). Dynamic movement primitives - a framework for motor control in humans and humanoid robotics. In Kimura, H., Tsuchiya, K., Ishiguro, A., and Witte, H., editors, *Adaptive Motion of Animals and Machines*, pages 261–280. Springer Tokyo.

Schaal, S. and Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084.

Schaal, S., Ijspeert, A., and Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transactions: Biological Sciences*, 358(1431):537–547.

Schaal, S., Mohajerian, P., and Ijspeert, A. (2007). Dynamics systems vs. optimal control - a unifying view. *Progress in Brain Research*, 168(165):425–445.

Sentis, L. and Khatib, O. (2004). Task-oriented control of humanoid robots through prioritization. In *IEEE International Conference on Humanoid Robots*.

Sentis, L. and Khatib, O. (2005). Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4):505–518.

Sentis, L. and Khatib, O. (2006). A whole-body control framework for humanoids operating in human environments. In *IEEE International Conference on Robotics and Automation*.

Sontag, E. D. (1998). *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Number 6 in Textbooks in Applied Mathematics. Springer, second edition.

Sugiura, H., Gienger, M., Janssen, H., and Goerick, C. (2007). Real-time collision avoidance with whole body motion control for humanoid robots. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2053–2058.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Svinin, M., Odashima, T., Ohno, S., Luo, Z., and Hosoe, S. (2005). An analysis of reaching movements in manipulation of constrained dynamic objects. In *IEEE International Conference on Intelligent Robots and Systems*.

Takano, W., Yamane, K., Sugihara, T., Yamamoto, K., and Nakamura, Y. (2006). Primitive communication based on motion recognition and generation with hierarchical mimesis model. In *IEEE International Conference on Robotics and Automation*.

Tani, J. and Yamamoto, J. (2002). On the dynamics of robot exploration learning. *Cognitive Systems Research*, 3(3):459–470.

Todorov, E. (2006). Optimal control theory. In Doya, K., editor, *Bayesian Brain*. MIT Press.

Ude, A. (1993). Trajectory generation from noisy positions of object features for teaching robot paths. *Robotics and Autonomous Systems*, 11(2):113–127.

Udwadia, F. E. (2003). A new perspective on the tracking control of nonlinear structural and mechanical systems. In *Proceedings of the Royal Society of London*, volume 459, pages 1783–1800.

Udwadia, F. E. (2008). Optimal tracking control of non-linear dynamical systems. *Proceedings of the Royal Society*, 464:2341–2363.

Udwadia, F. E. and Kalaba, R. E. (1996). *Analytical Dynamics: A New Approach*. Cambridge University Press.

Urtasun, R., Fleet, D. J., and Fua, P. (2006). 3d people tracking with gaussian process dynamical models. In *CVPR*.

Verbeek, J. (2006). Learning non-linear image manifolds by combining local linear models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 28(8):1236–1250.

Verbeek, J., Roweis, S., and Vlassis, N. (2004). Non-linear cca and pca by alignment of local models. In *Advances in Neural Information Processing Systems*.

Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634.

Vijayakumar, S., D'Souza, A., Shibata, T., Conradt, J., and Schaal, S. (2002). Statistical learning for humanoid robots. *Autonomous Robots*, 12(1):55–69.

Wang, J., Fleet, D., and Hertzmann, A. (2006). Gaussian process dynamical models. In *Advances in Neural Information Processing Systems*.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man, Mach. Sys.*, MMS-10(22):47–53.

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. NIPS 2007: Workshop on Robotics Challenges for Machine Learning.