Methods for Learning Control Policies from Variable-Constraint Demonstrations

Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar

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 not observable and frequently change between contexts. In this chapter, we explore the problem of learning control policies from data containing variable, dynamic and non-linear constraints on motion. We discuss how an effective approach for doing this is to learn the *unconstrained policy* in a way that is *consistent with the constraints*. We then go on to discuss several recent algorithms for extracting policies from movement data, where observations are recorded under variable, unknown constraints. We review a number of experiments testing the performance of these algorithms and demonstrating how the resultant policy models *generalise over constraints* allowing prediction of behaviour under unseen settings where new constraints apply.

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 [37], the task [6] 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

Matthew Howard, Stefan Klanke, and Sethu Vijayakumar
Institute of Perception Action and Behaviour, University of Edinburgh, Scotland, UK
e-mail: matthew.howard@ed.ac.uk

Michael Gienger and Christian Goerick
Honda Research Institute Europe (GmBH), Offenbach, Germany

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 253–291.
springerlink.com

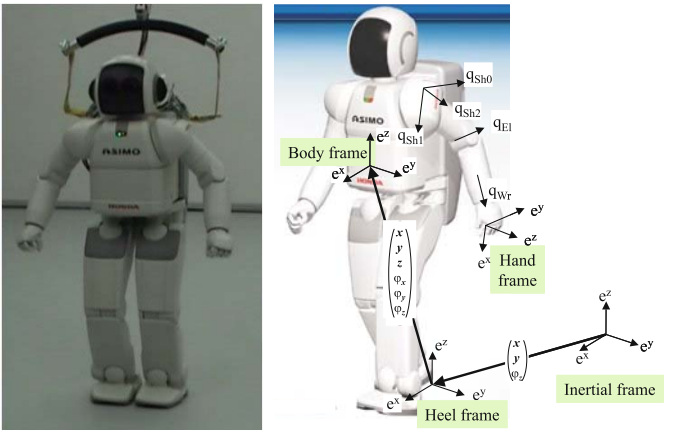

Fig. 1 ASIMO humanoid robot (left) and kinematic model used for whole body motion control (right) [15]. In our experiments 22 upper body degrees of freedom were used (2×7 DOF arms, 2 DOF head, 6 DOF torso), with the heel frame fixed.

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 [38] 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 overbalancing [27]. When manipulating or grasping solid objects the motion of ones fingers is constrained by the presence of the object [43]. Consider the task of running or walking on uneven terrain: the cyclic movement of the runner's legs is constrained by the impact of the feet on the ground in a dynamic, discontinuous and unpredictable way. In short, constraints that may be non-linear, spatially or temporally varying, or even discontinuous are ubiquitous in our everyday behaviour [50, 49, 15, 44, 48].

A promising approach to providing robots with abilities such as the above is to take examples of motion from existing skilled demonstrators (e.g. humans) and attempt to learn a control policy that somehow captures the desired behaviour [3, 4, 47]. Such techniques offer (i) a simple, intuitive interface for programming robots [4], (ii) effective methods for motion recognition and classification [26], and; (iii) accelerated optimisation of movements by using demonstration data to seed the solution [45]. However, while a wide variety of approaches for learning and representing movements have been proposed in recent years [3, 4, 47, 14], relatively 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).

In this chapter we will review recent works that deal precisely with this problem, that is learning from movement data that implicitly contains dynamic and uncertain constraints. We will primarily focus on two methods recently proposed [22, 19], that allow the effect of constraints to be dealt with in an appropriate way during learning. We will compare and contrast the two methods, consider their relative strengths, and in particular assess their suitability for improving existing policy learning methods that currently rely on traditional supervised learning techniques.

The novel techniques we consider are aimed at learning from demonstrated movements that are subject to variable, dynamic constraints with the goal of finding policies that can *generalise over constraints*. The two approaches both use a similar strategy to do this; namely they both 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 work in analytical dynamics (e.g. see [55]) where an effective and intuitive strategy for analytically solving constrained motion problems is to consider the effect constraints have in modifying the fundamental equations of motion of the unconstrained system. The difference here is that we attempt to do this in an automated, data-driven way.

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 [19]. We will discuss at length how these problems arise when learning in the constrained setting, and then look at how the two methods overcome them, first for the special case of potential-based policies, and later for learning generic, arbitrary policies. Using these *constraint-consistent* approaches to learning it has been shown [22, 19] that 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 (e.g. those that constrain the same part of the policy in all observations, effectively rendering it unobservable) we can learn a policy that at least 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 have been reported [22, 19] in order to validate the methods and to assess their performance. Here, we will compare and review some of these for learning on data from several policies on complex, high-dimensional movement systems, subject to various realistic

constraints, with a view to illustrating the utility of the approaches for transferring behaviour to robots such as the ASIMO humanoid (Fig. 1).

2 Effect of Variable Dynamic Constraints on Learning

In this section, we characterise the general problem of learning control policies from data, and discuss the problems encountered when variable constraints are applied to motion.

2.1 Learning Control Policies from Data

Following Schaal et al. [47], we consider the direct learning of autonomous policies

$$\mathbf{u}(t) = \boldsymbol{\pi}(\mathbf{x}(t)) \ , \qquad \boldsymbol{\pi} : \mathbb{R}^n \mapsto \mathbb{R}^d, \tag{1}$$

from data, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$ are appropriately[1] chosen state- and action-spaces, respectively. The goal of learning is to approximate the policy (1) as closely as possible [47] using a supervised learning approach, that is, we are given observations of $\mathbf{u}(t)$, $\mathbf{x}(t)$ (often in the form of trajectories) and from these we wish to learn the mapping $\boldsymbol{\pi}$. In previous work this has been done by fitting parametrised models in the form of dynamical systems [25, 24], non-parametric modelling [40, 6, 12], probabilistic Bayesian approaches [17, 16] and hidden Markov models [53, 26].

An implicit assumption found in policy learning approaches to date is that the data used for training comes from behavioural observations of some *unconstrained* or *consistently constrained* policy. By this it is meant that the policy is observed either under no constraint (e.g. movements in free space such as gestures or figure drawing), or under constraints consistent over observations (e.g. interacting with the same objects or obstacles in each case). However, in many everyday behaviours, there is variability in the constraints, such as when opening doors of varying sizes or walking on uneven terrain. This *variability in the constraints* cannot be accounted for by standard learning approaches.

[1] It should be noted that, as with all policy-based learning approaches, the choice of state- and action-space is problem specific [47] and, when used for imitation learning, depends on the *correspondence* between demonstrator and imitator. 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 $\mathbf{x}$ would 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 [1].

Example: Finger Extension with Contact Constraints

As an example, consider the learning of a simple policy to extend a jointed finger. In Fig. 2(a) the finger is unconstrained and the policy simply moves the joints towards the zero (outstretched) position. On the other hand, in Fig. 2(b), an obstacle lies in the path of the finger, so that the finger movement is constrained – it is not able to penetrate the obstacle, so moves along the surface. The vector field representation of the two behaviours is shown in Fig. 2(c).

Given the task of learning in this scenario, applying traditional learning approaches would result in one of two possibilities. The first is that if the observations are *labelled with respect to the constraint* (here, the orientation, position and shape of the obstacle) one could learn a separate policy model for the behaviour in each of the settings. However this is clearly unsatisfactory, since each model would only be valid for the specific setting, and we would need increasing numbers of models as we observed the policy under new constraints (for example different shaped obstacles at different positions and orientations). The second possibility is that the data is *unlabelled* with respect to the constraint. In this case, one might try to perform regression directly on the observations, that is observations from both vector fields (cf. Fig. 2(c), black and red vectors). However, this presents the problem that *model averaging* would occur across observations under different constraints, resulting in a poor representation of the movement in terms of the magnitude and direction of the predictions (see Sec. 2.3).

We can avoid the need for multiple policy models if we relax our assumptions on the form (1) of the observed commands, and allow for an additional transformation of $\boldsymbol{\pi}(\mathbf{x})$. We thus model both the red and black observations as stemming from the same policy ('extend the finger'), and attribute its different appearance to the transformations as induced by the constraints. With a restriction on the class of possible transformations, as will be detailed in the next section, we can model the unconstrained policy even if we only observed constrained movements, and we can apply new constraints to adapt the policy to novel scenarios.

2.2 Formal Constraint Model

In the remainder of this chapter we will focus on constraints which act as hard restrictions on the actions available to the policy. Specifically, we consider policies subject to a set of k-dimensional ($k \leq n$) Pfaffian constraints [55]

$$\mathbf{A}(\mathbf{x}, t)\mathbf{u} = \mathbf{0}. \tag{2}$$

Under these constraints, the policy is projected into the nullspace of $\mathbf{A}(\mathbf{x}, t)$:

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{N}(\mathbf{x}, t)\boldsymbol{\pi}(\mathbf{x}(t)) \tag{3}$$

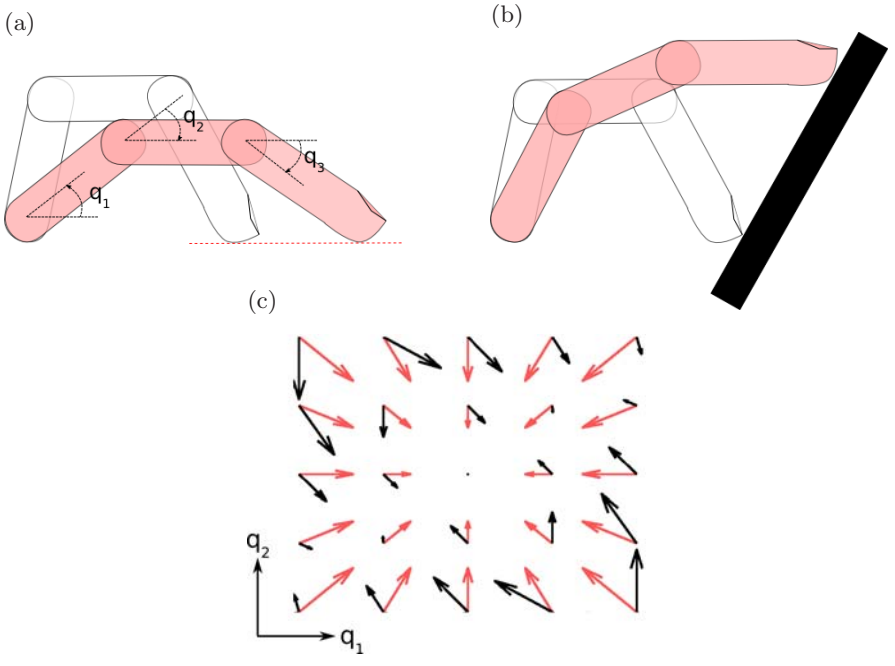

Fig. 2 Illustration of two apparently different behaviours from the same policy: (a) unconstrained movement (b) movement constrained by an obstacle (black box) (c) vector field visualisation of the unconstrained (red) and constrained (black) policy for two of the finger joints as a function of their angles.

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 will have a non-linear dependence on state and time[2], $\mathbf{A}(\mathbf{x}, t) \in \mathbb{R}^{k \times d}$ is some matrix describing the constraint and $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix. Constraints of the form (2) commonly appear in scenarios where manipulators interact with solid objects, for example when grasping a tool or turning a crank or a pedal, that is, contact constraint scenarios [38, 35, 34]. Such constraints are also common in the control of redundant degrees of freedom in high-dimensional manipulators [33, 30, 39], where policies such as (3) are used, for example, to aid joint stabilisation [39], or to avoid joint limits [9], kinematic singularities [59] or obstacles [10, 29] under task constraints. As an example: Setting $\mathbf{A}$ to the Jacobian that maps from joint-space to end-effector position increments would allow any motion in the joint space provided that the end-effector remained stationary. The same formalism applies equally to policies controlling dynamic quantities such as forces and momentum, for example see [39] and [27] respectively for constrained control schemes where the formalism applies directly. It has also found more exotic

[2] Here, $\mathbf{A}^{\dagger}$ denotes the (unweighted) Moore-Penrose pseudoinverse of the matrix $\mathbf{A}$.

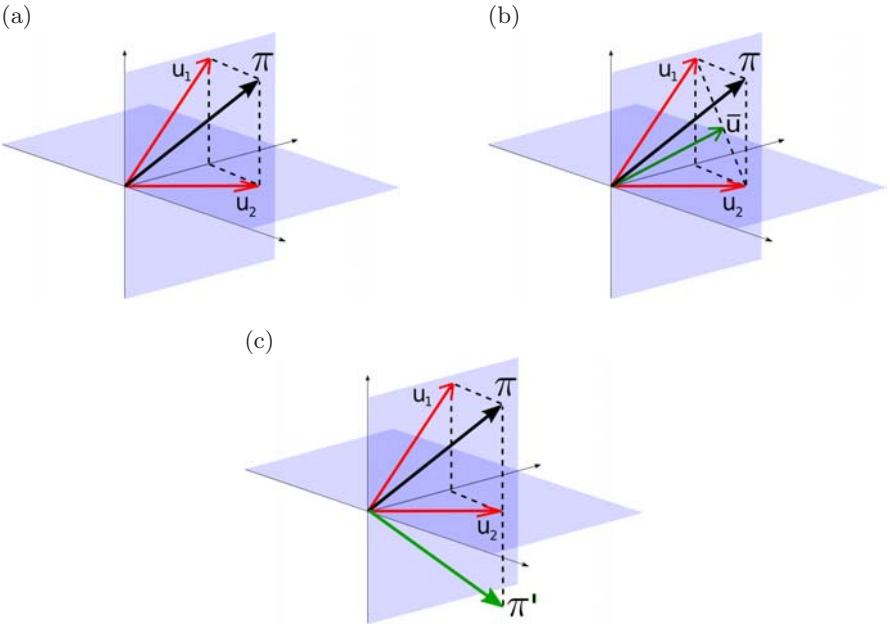

Fig. 3 Illustration of the effect of constraints on the unconstrained policy, the averaging effect of direct regression and the degeneracy problem. (a) Two constraints applied to the policy $\boldsymbol{\pi}$ result in projected observations $\mathbf{u}_1, \mathbf{u}_2$. (b) Direct regression results in averaging of the two movements $\bar{\mathbf{u}}$ in a way that cannot explain the observations. (c) Two policies $\boldsymbol{\pi}, \boldsymbol{\pi}'$ that both may be constrained in such a way as to produce the observation $\mathbf{u}_2$.

applications; for example Antonelli et al. [2] apply it to team coordination in mobile robots.

In general the effect of constraints (2)-(3) is to disallow commands in some sub-space of the system, specifically the space orthogonal to the image of $\mathbf{N}(\mathbf{x}, t)$. In essence these components of motion are *projected out* of the observed movement. For example, as illustrated in Fig. 3(a), a policy $\boldsymbol{\pi}$ is constrained in two different ways corresponding to two different projections of the unconstrained movement. In the first observation $\mathbf{u}_1$, the constraint prevents movement in the direction normal to the vertical plane[3]. For the second observation $\mathbf{u}_2$, the constraint only allows movement in the horizontal plane.

[3] Note that if the constraint has a non-linear dependence on time or state position the orientation of the constraint plane onto which the policy is projected will vary according to that dependence.

2.3 Learning from Constrained Motion Data

From the viewpoint of learning, constraints as described in the previous section present problems for traditional policy learning approaches. Specifically, there are two issues in particular that must be dealt with; that of *non-convexity* of observations and *degeneracy* between policies [20].

The *non-convexity* problem comes from the fact that between observations, or even during the course of a single observation, constraints may change. For example consider Fig. 3(b). There, the two policy observations under the different constraints, $\mathbf{u}_1$ and $\mathbf{u}_2$, appear different depending on the constraint. To the learner, this means that the data from the two scenarios will appear *non-convex*, in the sense that for any given point in the input ($\mathbf{x}$) space multiple outputs ($\mathbf{u}$) may exist. This causes problems for supervised learning algorithms, for example directly training on these observations may result in model-averaging. Here, averaging of $\mathbf{u}_1, \mathbf{u}_2$ results in the prediction $\bar{\mathbf{u}}$ that clearly does not match the unconstrained policy $\boldsymbol{\pi}$, either in direction or magnitude (ref. Fig. 3(b)).

The *degeneracy* problem stems from the fact that for any given set of projected (constrained) policy observations, there exist multiple candidate policies that could have produced that movement. This is due to the projection eliminating components of the unconstrained policy that are orthogonal to the image of $\mathbf{N}(\mathbf{x}, t)$ so that the component of $\boldsymbol{\pi}$ in this direction is undetermined by the observation. For example consider the constrained observation $\mathbf{u}_2$ in Fig. 3(c). There motion in the y direction is restricted, meaning that that component of $\boldsymbol{\pi}$ is not seen in this observation. Given only $\mathbf{u}_2$ we cannot determine if the policy $\boldsymbol{\pi}$ or an alternative, such as $\boldsymbol{\pi}'$ (ref. Fig. 3(c)) produced the observation. In effect we are not given sufficient information about the unconstrained policy to guarantee that it is fully reconstructed.

Despite these restrictions, we wish to do the best we can with the data available. 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, 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 given further observations under new constraints we can recover more information about the unconstrained policy $\boldsymbol{\pi}$. For instance, observing $\mathbf{u}_1$ eliminates the possibility that $\boldsymbol{\pi}'$ 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 $\boldsymbol{\pi}$.

Finally, it should be noted that, if in all observations, certain components of the policy are 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.

Despite the problems that such constraints cause, in recent studies two approaches have been proposed that make considerable progress in learning in a constraint-consistent way [22, 19]. The first of these provided a solution to the problem for the important special class of *potential-based* policies[4] [19]. Using this approach it was shown that learning consistent policies can be efficiently learnt from variable-constraint data using an indirect learning approach that models the policy's underlying potential function as its representation.

Following this, a second approach was proposed, aimed at removing some of the restrictive assumptions used by the earlier potential-based approach. The key to the second method was to use a small but significant modification to the empirical risk function used by standard regression techniques. It was found that using this approach policies of arbitrary complexity, including rotational policies (i.e. policies that cannot be described by a potential) can also be efficiently learnt [22]. In the next two sections we describe the details of the two approaches.

3 Constraint-Consistent Learning of Potential-Based Policies

In this section we discuss constraint-consistent learning for the special case that the policy is potential-based. We first give a precise definition of such policies and describe the kinds of behaviour that they can be used to represent. We then go on to discuss how such policies are particularly amenable to constraint-consistent learning and describe a method recently proposed for doing this [19].

3.1 Potential-Based Policies

A potential-based policy is a policy defined through the gradient of a scalar potential function $\phi(\mathbf{x})$

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}). \tag{4}$$

Such policies can be thought of as greedily optimising the potential function at every time step [36] and thus encode attractor landscapes where the minima of the potential correspond to stable attractor points. An example is given in Fig. 4 where a potential function with three maxima (repellors) and two 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

[4] We precisely define this class of policies in the next section.

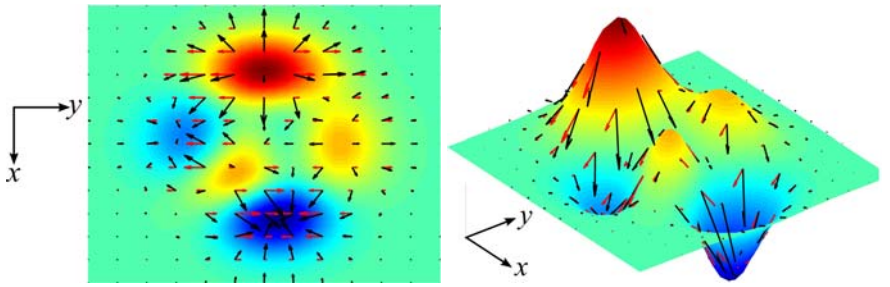

Fig. 4 Potential function with three maxima (repellors) and two minima (attractors). Overlaid are the corresponding unconstrained policy vectors (black) and a set of constrained policy vectors (red).

behaviours may be encoded as potentials [32, 31, 7, 8]. 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 [15, 13, 9, 10, 36, 59], and for navigation and obstacle avoidance problems in mobile robotics [41, 11, 42]. Furthermore, in reinforcement learning and optimal control [52, 54], 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.

3.2 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* [20, 23] 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 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)). This means that the (unconstrained) policy vectors point

in the direction of steepest descent, with magnitude equal to the slope in that direction (Fig. 4, black vectors).

Now, if a constraint is applied, the direction and magnitude of the vectors change. In the example in Fig. 4 the constraint prevents movement in one dimension (x dimension in Fig. 4, left) so that only motion corresponding the the second dimension (y dimension in Fig. 4, 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, 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) 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 these n-dimensional vector observations (ref. Fig. 3) to one of 'piecing together' local estimates of the slope of the potential.

A convenient method for doing this in the case of constrained kinematic policies is to use line integration to accurately estimate the form of the potential along trajectories [20, 23] and then use these local estimates to build a global model of the potential. We outline this approach in the next section.

3.3 Learning the Potential through Local Model Alignment

In the following we describe a method for modelling the potential from constrained motion data. Given observations of constrained trajectories, 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 [56, 57] 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.

Estimating the Potential Along Single Trajectories

As has been described in [20, 23], it is possible to model the potential along sampled trajectories from a constrained kinematic policy ($\mathbf{u} \equiv \dot{\mathbf{x}}$) using a form of line integration. Specifically, combining (3) and (4), the (continuous time) state evolution of the system under such policies is given by

$$\dot{\mathbf{x}} = \mathbf{N}(\mathbf{x}, t)\boldsymbol{\pi}(\mathbf{x}) = -\mathbf{N}(\mathbf{x}, t)\nabla_{\mathbf{x}}\phi(\mathbf{x}) \tag{5}$$

Let $\bar{\mathbf{x}}(t)$ be the solution of (5). If we line-integrate along $\bar{\mathbf{x}}(t)$ we have

$$\int_{\bar{\mathbf{x}}} (\nabla_{\mathbf{x}}\phi)^T \, d\mathbf{x} = \int_{t_0}^{t_f} (\nabla_{\mathbf{x}}\phi)^T \dot{\mathbf{x}} \, dt = -\int_{t_0}^{t_f} (\nabla_{\mathbf{x}}\phi)^T \mathbf{N}(\mathbf{x},t) \nabla_{\mathbf{x}}\phi(\mathbf{x}) \, dt, \quad (6)$$

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 (5)

$$\mathbf{x}_{i+1} \approx \mathbf{x}_i + \delta t \, \mathbf{N}_i \boldsymbol{\pi}_i = \mathbf{x}_i - \delta t \, \mathbf{N}_i \nabla_{\mathbf{x}}\phi(\mathbf{x}_i) \quad (7)$$

and (6) can be approximated using an appropriate numerical integration scheme. An example of such a scheme is Euler integration, which involves the first order approximation

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) + \frac{1}{\delta t}(\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathbf{N}_i \nabla_{\mathbf{x}}\phi(\mathbf{x}_i). \quad (8)$$

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 $\mathbf{x}_i \to \mathbf{x}_{i+1}$ that follow the projected policy (3) we can rearrange (7) with the scaled time coordinates, and substitute into (8) to yield

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2, \quad (9)$$

where the negative sign reflects our assumption (as expressed in (4)) that attractors are minima of the potential. We use this approximation to generate estimates $\hat{\phi}(\mathbf{x}_i)$ of the potential along any given trajectory $\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_N$ in the following way: We set $\hat{\phi}_1 = \hat{\phi}(\mathbf{x}_1)$ to an arbitrary value and then iteratively assign $\hat{\phi}_{i+1} := \hat{\phi}_i - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2$ 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.

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 $\hat{\boldsymbol{\Phi}}_k = (\hat{\phi}_{k1}, \hat{\phi}_{k2} \ldots \hat{\phi}_{kN_k})$ of the potential, as provided from the Euler integration just described. In a first step, we fit a function model $f_k(\mathbf{x})$ of the potential to each tuple $(\mathbf{X}_k, \hat{\boldsymbol{\Phi}}_k)$, such

that $f_k(\mathbf{x}_i) \approx \hat{\phi}_{ki}$. 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.,

$$f_k(\mathbf{x}) = \Phi_{ki^*} \quad , \quad i^* = \arg\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2. \tag{10}$$

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

$$w_k(\mathbf{x}) = \exp\left[-\frac{1}{2\sigma^2}\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2\right]. \tag{11}$$

From these weights we can calculate responsibilities

$$q_k(\mathbf{x}) = \frac{w_k(\mathbf{x})}{\sum_{i=1}^K w_i(\mathbf{x})} \tag{12}$$

and a (naive) global prediction $f(\mathbf{x}) = \sum_{k=1}^K q_k(\mathbf{x}) f_k(\mathbf{x})$ 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. [57] 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. [57] retrieved the parameters of the optimal mappings $\mathbf{g}_k$ by minimising the objective function

$$E = \frac{1}{2}\sum_{m=1}^M \sum_{k=1}^K \sum_{j=1}^K q_{km} q_{jm} \|\mathbf{g}_{km} - \mathbf{g}_{jm}\|^2, \tag{13}$$

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 [57], we augment our local potential models $f_k(\cdot)$ by a scalar offset b_k and define the corresponding objective function as

$$E(b_1 \dots b_K) = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{j=1}^{K} q_k(\mathbf{x}_m) q_j(\mathbf{x}_m) \times \left((f_k(\mathbf{x}_m) + b_k) - (f_j(\mathbf{x}_m) + b_j) \right)^2, \tag{14}$$

or, in a slightly shorter form,

$$E(\mathbf{b}) = \frac{1}{2} \sum_{m,k,j} q_{km} q_{jm} \left(f_{km} + b_k - f_{jm} - b_j \right)^2. \tag{15}$$

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 (15) into terms that are constant, linear, or quadratic in b_k, yielding

$$\begin{aligned} E(\mathbf{b}) &= \sum_{m,k} q_{km} f_{km}^2 - \sum_{m,j,k} q_{km} q_{jm} f_{km} f_{jm} \\ &\quad + 2 \sum_{m,k} q_{km} f_{km} b_k - 2 \sum_{m,k} q_{km} q_{jm} f_{jm} b_k \\ &\quad + \sum_{m,k} q_{km} b_k^2 - \sum_{m,k,j} q_{km} q_{jm} b_k b_j \\ &= E_0 + 2\mathbf{a}^T \mathbf{b} + \mathbf{b}^T \mathbf{H} \mathbf{b}. \end{aligned} \tag{16}$$

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 $a_k = \sum_m q_{km} f_{km} - \sum_{m,j} q_{km} q_{jm} f_{jm}$, 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 (14), 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

$$\mathbf{b}_{opt} = -\mathbf{H}^{\dagger} \mathbf{a}. \tag{17}$$

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.

Smoothing Parameter Selection

Since we restrict ourselves to using simple NN regression for the local potential models, the only open parameter of the algorithm is σ^2, i.e., the kernel parameter used for calculating the responsibilities (11). A too large choice of this parameter will over-smooth the potential, because the NN regression model basically predicts a locally constant potential, but at the same time trajectories will have relatively high responsibilities even for 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.

Such trajectories can cause problems when attempting to learn a global model of the potential using the output of the optimisation (17), since if their influence on the overall alignment is negligible, their own alignment can be poor. Fortunately, this situation 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 [28] (please refer to [19] for details on this step). We can then either discard all trajectories apart from those in the largest 'connected' group (treating the weakly connected trajectories as outliers) or recursively repeat the alignment process on the larger groups of aligned trajectories.

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 the experiments presented in [19] this parameter was selected heuristically by first calculating the distances between any two trajectories $k, j \in \{1 \dots K\}$ in the set as the distances between their closest points

$$d_{kj} = \min \left\{ \|\mathbf{x}_{kn} - \mathbf{x}_{jm}\|^2 \mid n, m \in \{1 \dots N\} \right\}, \tag{18}$$

and also the distances to the closest trajectory

$$d_k^{min} = \min \left\{ d_{kj} \mid j \neq k \right\}. \tag{19}$$

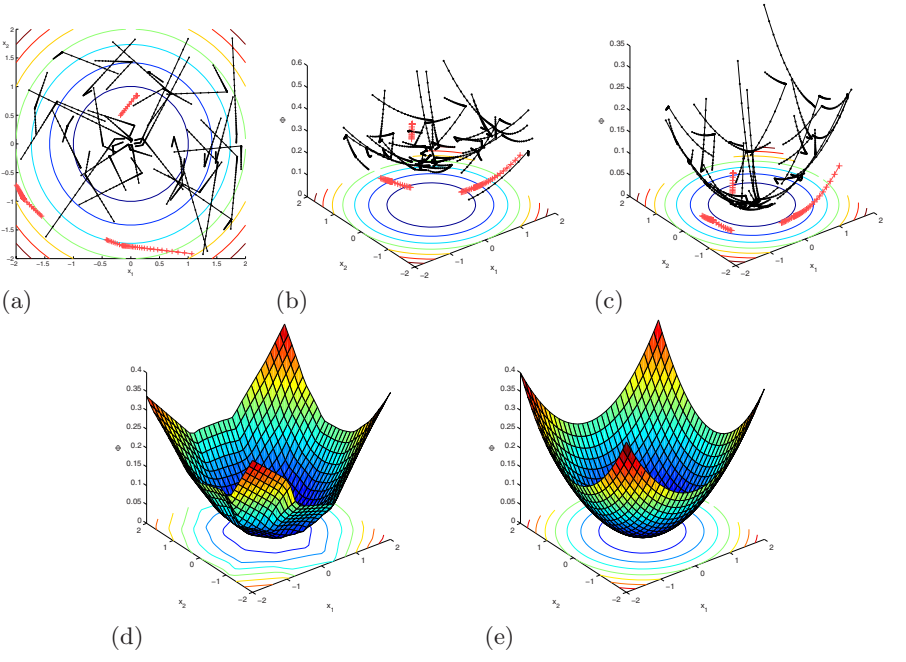

Fig. 5 The alignment algorithm at work for a set of $K = 40$ trajectories of length $N = 40$ sampled from a parabolic potential ($\phi(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x}$, $\mathbf{W} = 0.05\mathbf{I}$) with randomly switching constraints ($\mathbf{A}(\mathbf{x}, t) = (\alpha_1, \alpha_2) \equiv \boldsymbol{\alpha}$, $\alpha_i = N(0, 1)$). (a) Raw data (constrained trajectories through the two-dimensional state space) and contour of the potential. (b) Local potential models estimated by Euler integration prior to alignment. (c) Local potential models after alignment, already revealing the structure of the parabola. (d) Global model $f(\mathbf{x})$ trained on the aligned trajectories (here, trained with LWPR [58]). (e) True parabolic potential shown for comparison. The weakly connected 'outlier' trajectories (here, discarded prior to learning the global model) are shown in red.

Then three choices for σ^2, were considered, referred to as 'narrow', 'wide' and 'medium':

$$\sigma^2_{nar} = \text{median}\left\{d^{min}_k \mid k \in \{1 \dots K\}\right\} \tag{20}$$

$$\sigma^2_{wid} = \text{median}\left\{d_{jk} \mid j, k \in \{1 \dots K\}, j \neq k\right\} \tag{21}$$

$$\sigma^2_{med} = \sqrt{\sigma^2_{nar}\sigma^2_{wid}}. \tag{22}$$

As shown by experiment [19, 20], the choice σ^2_{med} usually represents a reasonable balance between smoothing and alignment performance.

Algorithm 1. PolicyAlign

1: Estimate $\mathbf{X}_k$, $\hat{\mathbf{\Phi}}_k, \{k = 1 \dots K\}$ using Euler integration. Calculate σ^2.

2: Alignment:

- Calculate prediction and responsibility of each local model f_k on each data point $\mathbf{x}_m,\ m = 1 \dots M$:
  $f_{km} = f_k(\mathbf{x}_m);\ \ q_{km} = w_k(\mathbf{x}_m) / \sum_i w_i(\mathbf{x}_m)$
- Construct $\mathbf{H}, \mathbf{a}$ with elements
  $h_{ij} = \delta_{ij} \sum_m q_{jm} - \sum_m q_{im} q_{jm};\ \ a_k = \sum_m q_{km} f_{km} - \sum_{m,j} q_{km} q_{jm} f_{jm}$
- 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 $(\mathbf{x}_{kn}, \hat{\phi}_{kn} + b_k^{opt})$

Learning the Global Model

After calculating optimal offsets $\mathbf{b}_{opt}$ and cleaning the data set from outliers, we can learn a global model $f(\mathbf{x})$ of the potential using any regression algorithm. For example, in the experiments presented in Sec. 5, we will use Gaussian Radial Basis Function (RBF) models, and in [19, 20] Locally Weighted Projection Regression (LWPR) [58] was used. As the training data for these models, 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

$$\left\{ (\mathbf{x}_{kn}, \hat{\phi}_{kn} + b_k^{opt}) \mid k \in \mathcal{K}, n \in \{1 \dots N_k\} \right\}, \tag{23}$$

where $\mathcal{K}$ denotes the set of indices of non-outlier trajectories. Once we have learnt the model $f(\mathbf{x})$ of the potential, we can take derivatives to estimate the unconstrained policy $\hat{\boldsymbol{\pi}}(\mathbf{x}) = -\nabla_{\mathbf{x}} f(\mathbf{x})$. For convenience, the complete procedure is summarised in Algorithm 1 and illustrated pictorially in Fig. 5 for an example parabolic potential with randomly switching constraints.

4 Constraint-Consistent Learning of Generic Policies

In the previous section we outlined a method for learning in a constraint-consistent manner based on indirect modelling of the policy through its potential. As discussed in Sec. 3.2, potential-based policies are particularly amenable to learning in the constrained setting since observations under different constraints correspond to the directional derivatives of the potential in the different (unconstrained) directions. This allows us to model the shape of the potential to find a policy that is consistent with the observations.

While this approach has been shown to greatly outperform direct learning approaches in an number of experiments on constrained systems [19], it still suffers from some restrictions due to the assumption of a potential-based policy. While this is not a problem when learning from systems such as those described in Sec. 3.1, it can cause difficulties when the policy under observation encodes periodic or rotational behaviour (precisely speaking, when the *curl* of the observed policy is non-zero).

In order to avoid this restriction an alternative approach to learning must be taken. In [22], a new method was proposed that enables learning of generic policies from variable constraint data. This method was based on a small but significant modification of the empirical risk function used for learning. In the following 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 discuss the novel risk function proposed in [22] that both satisfies our original assumptions, and has been shown to be effective for learning from variable constraint data [22].

4.1 Optimisation of the Standard Risk, UPE and CPE

As already outlined in Sec. 2.3, throughout this chapter we are targeting problems where we are given data in the form of tuples $(\mathbf{x}_n, \mathbf{u}_n)$ of observed states and constrained actions, where we assume that all commands $\mathbf{u}$ are generated from some underlying policy $\boldsymbol{\pi}(\mathbf{x})$, which for a particular observation might have been constrained. For constrained systems (2)-(3), this means that we observe $\mathbf{u}_n = \mathbf{N}_n\boldsymbol{\pi}(\mathbf{x}_n)$ 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 $\tilde{\boldsymbol{\pi}}(\cdot)$ by minimising the *standard risk*

$$E_{direct}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2. \tag{24}$$

As already mentioned in Sec. 2 this is an effective approach for learning from unconstrained data or data where the same constraint appears in all observations (i.e. the constraint matrix $\mathbf{A}(\mathbf{x}, t)$ 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 an system opening some particular door and

in every observation the door was the same, then optimisation of (24) would be effective for learning the policy for that particular door.

The problem with this approach, however, is that it cannot handle data where commands are observed under variable constraints. As discussed in Sec. 2.3, if we consider an example where multiple observations are given under different constraints, optimisation of (24) would result in a naive averaging of commands from different circumstances (cf. Fig. 3(b)). 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 (24) for learning in this setting, since it does not meet our requirements for accuracy and generalisation.

An alternative approach then, might be to target error measures that directly measure performance in terms of our objectives. For example, we could attempt to optimise our model with respect either to the *unconstrained policy error* (UPE)

$$E_{upe}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\boldsymbol{\pi}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 \tag{25}$$

or the *constrained policy error* (CPE)

$$E_{cpe}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{N}_n\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2. \tag{26}$$

Optimising the former would directly give us the best fit to the policy, while optimising the latter would give the best fit that is consistent with the constrained observations. The optimal model with respect to either of these would satisfy our accuracy and generalisation requirements. For example in the former case we could apply any projection (constraint) to the model and still achieve a good CPE under the new constraint.

However, the problem here is that by assumption we do not have access to samples of either (i) the (unconstrained) policy $\boldsymbol{\pi}_n = \boldsymbol{\pi}(\mathbf{x}_n)$, or (ii) the projection matrices $\mathbf{N}_n$ needed for calculating these quantities. This is because in most problems of interest, *constraints are not directly observable* and there is *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*[5] 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. However, while this is one reasonable analysis of the system, there also exist other

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

possibilities. 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 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. Since none of these can be resolved by the given observations, we cannot in general use either of these functionals for learning.

4.2 Learning Generic Policies by Minimising Inconsistency

Having ruled out the use of (24)-(26) for learning in this setting we must look for alternative approaches. Our aim is to try to estimate a policy $\tilde{\boldsymbol{\pi}}(\cdot)$ 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

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

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. 6). 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 (26) by a projection onto $\mathbf{u}_n$ and minimise the *inconsistency* which we define as the functional

$$E_i[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \hat{\mathbf{u}}_n\hat{\mathbf{u}}_n^T\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 = \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\right)^2 \qquad (27)$$

with $r_n = \|\mathbf{u}_n\|$, $\hat{\mathbf{u}}_n = \frac{\mathbf{u}_n}{r_n}$. Since $\mathbf{u}_n = \mathbf{N}_n\boldsymbol{\pi}_n$ we can write $\|\mathbf{u}_n - \mathbf{N}_n\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 = \|\mathbf{N}_n(\boldsymbol{\pi}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n))\|^2$ and recognise that the CPE is always less than or equal to the UPE, because the projections $\mathbf{N}_n$ can only decrease

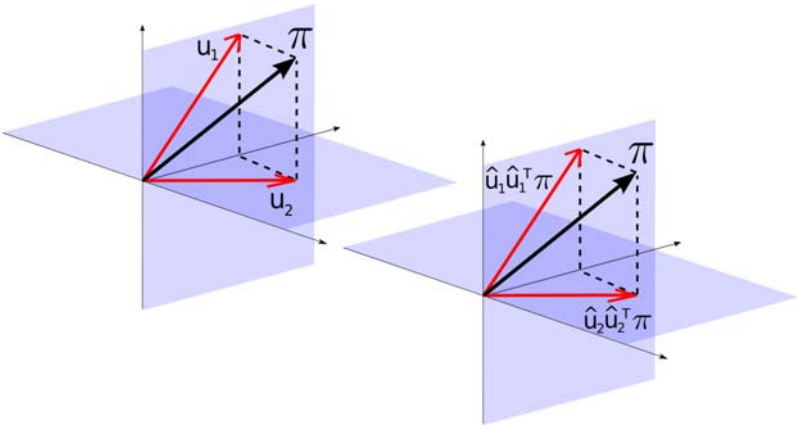

Fig. 6 Illustration of our learning scheme. The projection of the correct policy $\boldsymbol{\pi}$ onto the observations matches those observations.

the norm of the difference between true and predicted policy. The same argument holds for the inconsistency error (27) where the projection onto the 1-D subspace spanned by $\hat{\mathbf{u}}_n$, possibly takes away even more of the error. So we can establish the inequality

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

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. 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 (27) can be used in conjunction with many standard regression techniques. In the following we derive training rules for 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. 4.3), and (ii) locally linear models (Sec. 4.4). We turn to this in the next section.

4.3 Parametric Policy Models

A particularly convenient model of the policy is given by $\tilde{\boldsymbol{\pi}}(\mathbf{x}) = \mathbf{W}\mathbf{b}(\mathbf{x})$, 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 $\mathbf{c}_i$, $i = 1 \dots M$. With this model, the *inconsistency* error from (27) 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

$$\mathbf{w}^{opt} = \arg\min E_i(\mathbf{w}) = \mathbf{H}^{-1}\mathbf{g} \tag{28}$$

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 $\tilde{\boldsymbol{\pi}}$, 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 $O(d^2 M^2)$.

4.4 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 $\tilde{\boldsymbol{\pi}}_m(\mathbf{x}) = \mathbf{B}_m \bar{\mathbf{x}} = \mathbf{B}_m (\mathbf{x}^T, 1)^T$ to the data, learning each local model independently [46]. For a linear model centred at $\mathbf{c}_m$ with an isotropic Gaussian receptive field with variance σ^2, we would minimise

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

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

$$\mathbf{b}_m^{opt} = \arg\min E_i(\mathbf{b}_m) = \mathbf{H}_m^{-1}\mathbf{g}_m \tag{29}$$

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

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

where $w_m = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{c}_m\|^2\right)$. For extensive experiments assessing the performance of learning with parametric and local linear models using the novel risk function, we refer the reader to the original experiments reported in [22, 21].

5 Constraint-Consistent Learning Performance

To explore the performance of the two approaches, a number of experiments have been performed, learning on data from autonomous kinematic control policies from different plants, including learning from human demonstration data to enable the ASIMO humanoid robot to learn a realistic car washing task [22, 19]. In this section, we briefly review some of these results to provide the reader with a view of the comparative performance the two approaches. For this, we first discuss learning on data from a simple, two-dimensional system controlled according to the framework outlined in Sec. 2. 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 from a constrained demonstrator. We then give a brief discussion on how constraint-consistent learning has been applied for learning from human demonstration data for transferring skills to the ASIMO humanoid.

5.1 Two-Dimensional Constrained System

In this section we compare the performance of the constraint-consistent learning approaches described in Sec. 3 and Sec. 4 on a simple two-dimensional system $(\mathbf{x}, \mathbf{u} \equiv \dot{\mathbf{x}} \in \mathbb{R}^2$ with policies subject to discontinuously switching constraints. Specifically, the constraints are given by

$$\mathbf{A}(\mathbf{x}, t) = (\alpha_1, \alpha_2) \equiv \boldsymbol{\alpha} \tag{30}$$

where the $\alpha_{1,2}$ are drawn from a normal distribution, $\alpha_i = N(0,1)$. Here, the constraints mean that motion is constrained in the direction orthogonal to the vector $\boldsymbol{\alpha}$ in state space. To increase the complexity of the problem, the constraints are randomly switched during trajectories by re-sampling $\boldsymbol{\alpha}$ twice at regular intervals during the trajectory. This switches the direction in which motion is constrained, causing sharp turns in the trajectories (for example, see Fig. 5(a)).

To compare the methods, we investigate learning on data from three policies of differing complexity. These were (i) a policy defined by a quadratic potential function

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi_q(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_c)^T\mathbf{W}(\mathbf{x}-\mathbf{x}_c) \tag{31}$$

where we chose $\mathbf{x}_c = 0$ and $\mathbf{W} = \alpha\mathbf{I}$; (ii) a sinusoidal potential

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi_s(\mathbf{x}) = \alpha\sin(x_1)\cos(x_2), \tag{32}$$

where we set $\alpha = 0.1$ and (iii) a limit cycle policy

$$\dot{r} = r(\rho - r^2), \quad \dot{\theta} = \omega. \tag{33}$$

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. For the experiments, the latter two parameters were set to $\rho = 0.5\ m$ and $\omega = 1\ rad\ s^{-1}$. Here, the two potential-based policies act as attractor landscapes with, for example, a single point attractor at $\mathbf{x}_c$ for the quadratic potential and multiple point attractors for the sinusoidal potential. Note that the limit cycle policy is a rotational policy and therefore cannot be defined by a potential.

Data was collected by sampling $K = 40$ trajectories at a rate of 50 Hz from the policies, starting from random states. This resulted in $N = 40$ data points per trajectory. We then attempted to learn on this data using (i) the alignment approach (ref. Sec. 3), (ii) optimisation of the inconsistency (ref. Sec. 4), and (iii) direct regression (i.e. training directly on the tuples $(\mathbf{x}_i, \mathbf{u}_i), i = 1, \ldots K \times N$ and optimising the risk function (24)). For each of the methods we learnt models consisting of a set of normalised Gaussian RBFs with centres arranged on a 6×6 grid, and with the kernel widths fixed to yield suitable overlap. For the latter two approaches the model represented the mapping $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \mathbf{u} \in \mathbb{R}^2 \mapsto \mathbb{R}^2$ and for the alignment approach it represented the mapping $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \phi \in \mathbb{R}^2 \mapsto \mathbb{R}$. For a thorough comparison, we learnt models of each of the three policies using the three different learning approaches. We repeated this experiment on 100 data sets and evaluated the normalised UPE and CPE, that is, the functionals from (25) and (26) divided by the number of data points and the variance of the policy $\boldsymbol{\pi}_n$ on a subset held out for testing.

Table 1 Normalised UPE and CPE for learning the three policies from the toy example using (i) direct regression, (ii) the alignment approach and (iii) the inconsistency approach. All errors are (mean±s.d.)$\times 10^{-2}$ over 100 data sets.

Policy	Alg.	nUPE	nCPE
Quad. Pot.	direct	54.727 ± 6.218	10.732 ± 2.010
	align.	1.158 ± 1.561	0.443 ± 0.588
	incon.	0.001 ± 0.001	0.001 ± 0.001
Sin. Pot	direct	40.478 ± 4.789	12.354 ± 1.097
	align.	5.020 ± 5.395	2.162 ± 2.536
	incon.	0.003 ± 0.003	0.001 ± 0.004
Lim. Cyc.	direct	43.225 ± 6.599	10.034 ± 1.678
	align.	291.233 ± 156.180	126.902 ± 80.364
	incon.	0.024 ± 0.040	0.003 ± 0.002

Table 1 summarises the results. Firstly, looking at the results for using direct regression to learning the three policies, we see uniformly poor performance both in terms of the normalised UPE and CPE. Because the direct approach to learning is naive to the effect of the constraints, model averaging results. This causes the predictions for each of the three polices to be poor, even under the training constraints.

In contrast to this, looking at the results for the potential-based policies, the alignment approach performs approximately an order of magnitude better both in terms of the UPE and the CPE. Comparing errors for the quadratic and sinusoidal potential-based policies we also see that the latter, more complex potential (with multiple sinks) is more difficult to learn with a data set of this size. However, as expected, the alignment approach performs very badly when training on the limit cycle data: The potential-based representation is not appropriate in this case since the policy is rotational.

Looking at the errors for the inconsistency approach, however, we see improved performance on all three policies, including the rotational limit cycle data. Comparing results for the three policies we see that the sinusoidal potential-based policy and the limit-cycle policy are more difficult to learn due to their increased complexity. However, despite this, the increase in error for this approach is still relatively small.

Finally, in all of the results, the nCPE is always much lower than the nUPE, which follows naturally from the fact that the projection induced by the constraints projects out some of the error in the models, as discussed in Sec. 4. Still, the results show that with a reasonable amount of data, the unconstrained policy can be modelled with remarkable accuracy using the two constraint-consistent approaches.

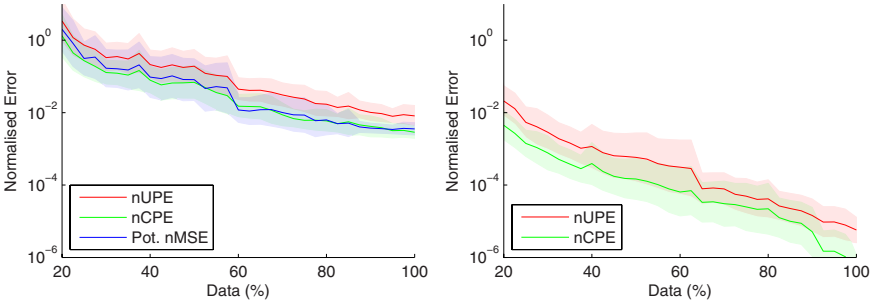

Fig. 7 Learning performance on the quadratic potential (31) with varying data set sizes for the alignment approach (left) and the inconsistency approach (right). Normalised CPE and UPE versus data set size as a percentage of the full $K = 40$ trajectories of length $N = 40$ are shown. For the alignment approach the normalised error in the potential is also plotted.

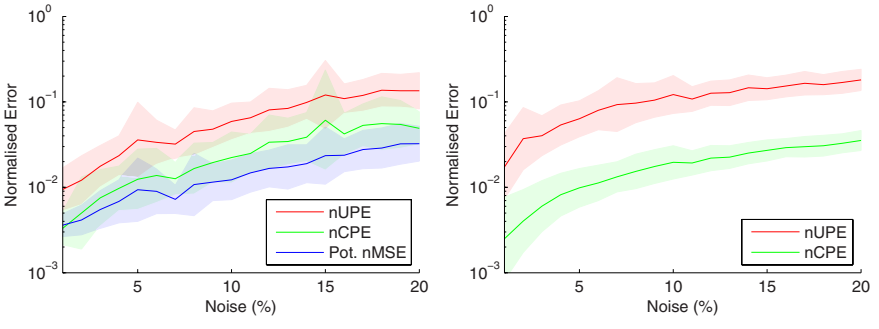

Fig. 8 Learning performance on the quadratic potential (31) with varying noise levels for the alignment approach (left) and the inconsistency approach (right). Normalised CPE and UPE versus noise in the observed $(\mathbf{x}_n, \mathbf{u}_n)$ as a percentage of the variance of the data are shown. For the alignment approach the normalised error in the potential is also plotted.

As a further test, we can also look at the performance in terms of the amount of data required to find a good approximation. In Fig. 7 we show the error curves for the two constraint-consistent learning approaches when learning on different-sized subsets of the data from the quadratic potential (31). As can be seen (ref. Fig. 7), the performance of the two approaches improves with increasing quantities of data in terms of UPE and CPE, with the inconsistency approach generally achieving a lower error than the alignment approach for this data set.

Finally, in Fig. 8 we characterise the noise robustness of the two constraint-consistent approaches when learning, again using the same data, but this time with the the observed states $\mathbf{x}_n$ and actions $\mathbf{u}_n$ contaminated with Gaussian

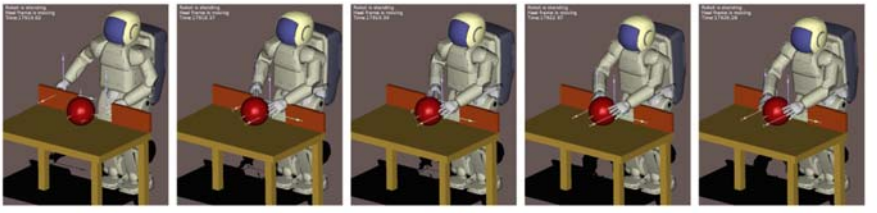

Fig. 9 Example constrained trajectory used as training data in the ball-reaching experiment. Starting with hands at the sides, the demonstrator robot reaches between the barriers to get the ball. Note that the width of the gap in the barriers was randomly altered for each trajectory recorded.

noise, the scale of which was varied to match up to 20% of the scale of the data. We see that the performance of the two approaches approximately follows the noise levels in the data (ref. Fig. 8), although there is slightly more variance in the performance of the alignment approach. This latter effect can be explained by the fact that the alignment uses the nearest neighbour trajectories for the alignment, the measurement of which becomes increasingly unreliable as the noise in $\mathbf{x}_n$ increases. However, despite this, the performance of the two approaches can be seen to decline smoothly as the amount of noise increases.

5.2 Reaching for a Ball

In this section we characterise (i) how well the two approaches scale to more complex, realistic constraints and policies and (ii) how well the policies learnt with these approaches generalise over different constraints. For this, we use data from an example scenario, in which a set of observations of a demonstrator performing the task of reaching for a ball on a table are given, and the student is expected to learn a policy to enable it to reproduce this task [22, 19]. The learning problem is complicated however, by the presence of different obstacles on the table for each of the example trajectories, constraining the possible motions of the hands. The goal is to uncover a policy that accurately predicts the demonstrator's (unconstrained) behaviour and generalises to predict the behaviour under novel constraints.

The example scenario was implemented [22, 19] using the whole body motion (WBM) controller of the 27-DOF humanoid robot ASIMO (for details on the controller see [15]). For this, data was recorded from a 'demonstrator' robot that followed a policy defined by an inverted Gaussian potential

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = \alpha\left(1 - e^{\|\mathbf{x}-\mathbf{x}_c\|^2/2\sigma^2}\right), \tag{34}$$

where $\mathbf{x} \in \mathbb{R}^6$ corresponds to the Cartesian position of the two hands (hereafter, the 'task space') and the actions $\mathbf{u} = \dot{\mathbf{x}} = \boldsymbol{\pi}(\mathbf{x})$ correspond to the hand velocities. 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. Following the policy (34) 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. 9). The barriers acted as inequality constraints on each of the hands so that motion in the direction normal to the barrier surface was prevented if a hand came too close. Specifically, the constraints took the form

$$\mathbf{A}(\mathbf{x}, t) = \left(\begin{array}{c|c} \mathbf{A}_{[1,1]} & \mathbf{0} \\ \hline \mathbf{A}_{[1,2]} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}_{[2,1]} \\ \hline \mathbf{0} & \mathbf{A}_{[2,2]} \end{array}\right) \tag{35}$$

where

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

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[6] 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.

[6] 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 [51].

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 (34) under the constraints (35). Start states were sampled from a Gaussian distribution over joint configurations $\mathbf{q} \sim N(\mathbf{q}_0, 0.1\mathbf{I})$ (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. 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 $d_{gap} \sim N(\mu_{gap}, \sigma_{gap})$ 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. 9 shows an example trajectory under this set-up.

We used the three algorithms (the direct, alignment and inconsistency approaches) to perform learning on 50 such data sets using 150 Gaussian RBF models, with centres placed using k-means. For comparison, we repeated the experiment on the same data with the same model (i.e. same number and placement of centres) with the three approaches. Please note that (similar to the experiments in the preceding section) the model for the direct and inconsistency approaches corresponded to the $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \mathbf{u} \in \mathbb{R}^6 \mapsto \mathbb{R}^6$ mapping, whereas for the alignment approach it represented the mapping $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \phi \in \mathbb{R}^6 \mapsto \mathbb{R}$.

To assess the performance for the 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. 11). Specifically, the constraint took a form similar to (35) but this time with only one barrier present (i.e. $j \equiv 1$), so that the constraint matrix $\mathbf{A}(\mathbf{x}, t) \in \mathbb{R}^{2\times 6}$ had attained a maximum rank of $k = 2$ when both hands approached the barrier. The width of the new barrier was fixed at $0.5m$.

Comparing the numerical errors (ref. Table 2) for the two constraint-consistent learning methods (i.e. the alignment and inconsistency approaches) with those of the direct approach we see that the former perform several orders of magnitude better under each of the constraint settings considered, with the inconsistency approach performing marginally better. However, the real difference between the constraint-consistent learning methods and the direct approach is best highlighted if we compare trajectories generated by the policies under different constraint settings.

Firstly, Fig. 10 shows example trajectories for the *unconstrained* reaching movements produced by the demonstrator ('expert'), and the policies learnt by (i) the direct approach, (ii) the alignment approach, and (iii) the

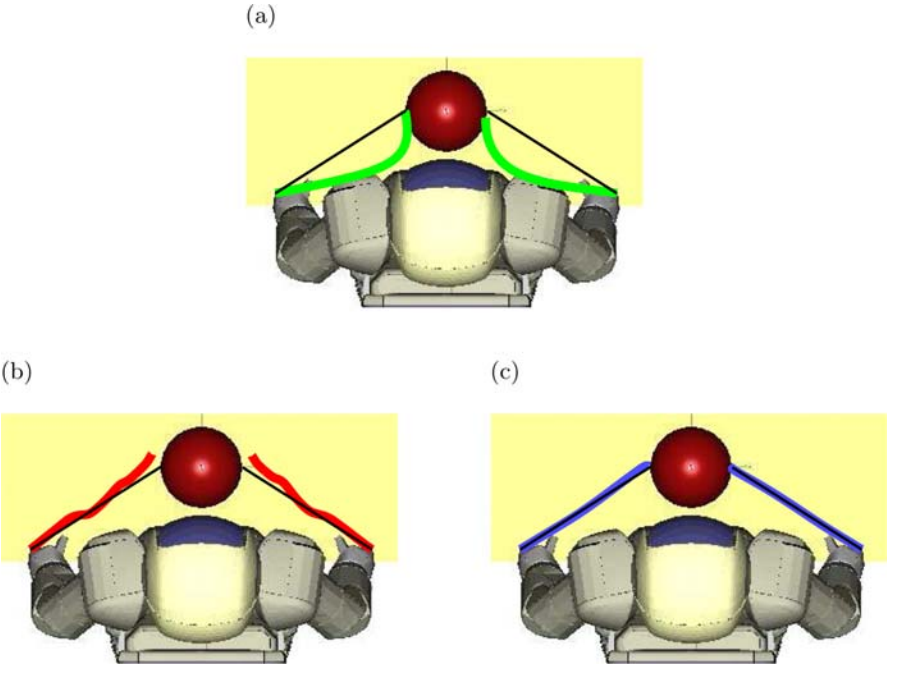

Fig. 10 Unconstrained reaching movement for the policies learnt with (a) direct regression (green) (b) the alignment approach (red), and (c) the inconsistency approach (blue). In each figure the demonstrator's movement is overlaid in black.

Table 2 Normalised errors for policies learnt by the three methods, evaluated on (i) training constraints, (ii) no constraints, and (iii) an unseen test constraint on the ball-reaching task. Values are mean±s.d. over 50 data sets.

Constraint	Direct	Align.	Incon.
Training	0.0531 ± 0.0068	0.0092 ± 0.0021	0.0052 ± 0.0022
Unseen Barrier	0.4630 ± 0.0350	0.0101 ± 0.0023	0.0052 ± 0.0022
Unconstrained	0.9216 ± 0.0625	0.0106 ± 0.0024	0.0052 ± 0.0022

inconsistency approach. In the former the hands always take a curved path to the ball (Fig. 10(a)), reproducing the average behaviour of the (constrained) demonstrated trajectories. The direct approach, being naive to the effect of the constraints is unable to extract the underlying task (policy) from the observed paths around the obstacles. In contrast, the policies learnt with the constraint-consistent approaches better predict the unconstrained policy, enabling them to take a direct route to the ball that closely matches that of the demonstrator (Fig. 10(b),(c)).

Secondly, Fig. 11 shows example trajectories when the policies are again constrained. Figure 11 (top) shows the movement from the policy learnt by the inconsistency approach under a similar constraint as in the training data. Under this constraint the policies learnt by the three methods all take a similar path to that of the demonstrator: The hands move in first, then forward to the ball. Note that under this constraint the movement of the directly learnt policy is noticeably slower due to averaging of the constrained observations.

Finally, under the unseen barrier constraint, there is a marked difference in behaviour. Under this constraint, the demonstrator (still following the policy (34)) reaches around the barrier to get the ball. This behaviour is reproduced by the policy learnt with the two constraint-consistent approaches (Fig. 11, middle row, shows the movement for the policy learnt by the inconsistency approach). In contrast however, the directly learnt policy does not generalise to the new constraint and gets trapped behind the barrier, eventually dislodging it[7] (Fig. 11, bottom).

5.3 Washing a Car

In this section we discuss an application of constraint-consistent learning to the the problem of learning to wash a car from human demonstration data [22]. 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 these surfaces, complex, non-linear constraints are imposed on the motion. While the resultant trajectories remain periodic, they are perturbed in different ways by the constraints. The goal of this experiment then, was to learn a policy that captured the periodic nature of the movements, while eliminating artifacts induced by the constraints.

In [22] an experiment was performed to evaluate the performance of constraint-consistent learning on data from human demonstrations of wiping. In this experiment a set of demonstrations of wiping on different surfaces (i.e. on surfaces with different tilts and orientations, see Fig. 12) were presented to the ASIMO humanoid robot by a human demonstrator. The robot used on-board stereo cameras to track the three-dimensional coordinates of the sponge (for details on the ASIMO vision system please see [5]) and the resultant data was used for constraint-consistent learning. The resultant data was used to train a policy model representing the $\mathbb{R}^3 \mapsto \mathbb{R}^3$ mapping from hand

[7] 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 directly learnt 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.

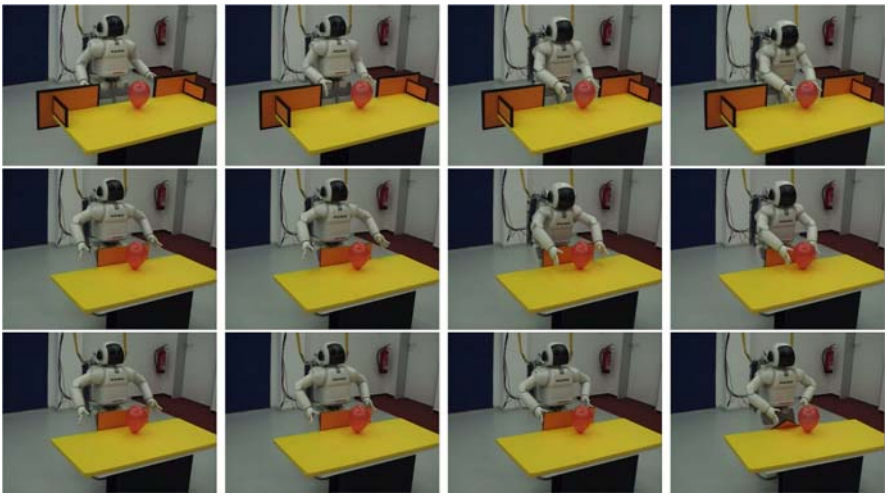

Fig. 11 Reaching movements produced by the learnt policies under different constraints. Shown are trajectories from (i) the policy learnt by the inconsistency approach under a similar constraint as in the training data (top row); (ii) the same policy under a new, unseen barrier constraint (middle row), and; (iii) the policy learnt with direct regression under the new constraint.

Fig. 12 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 $\pm 16^{\circ}$ and $+27^{\circ}$ about the x-axis are shown.

(sponge) positions to velocities, consisting of a set of 300 Gaussian RBFs with centres placed by k-means.

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. For this, the policies were implemented on the ASIMO humanoid robot and an approximation of the human's constraints based on an analysis of the hand-sponge system (for full details on the constraints used to approximate the human constraints, please refer to [22]) were applied to the robot during movement reproduction. Under this set-up,

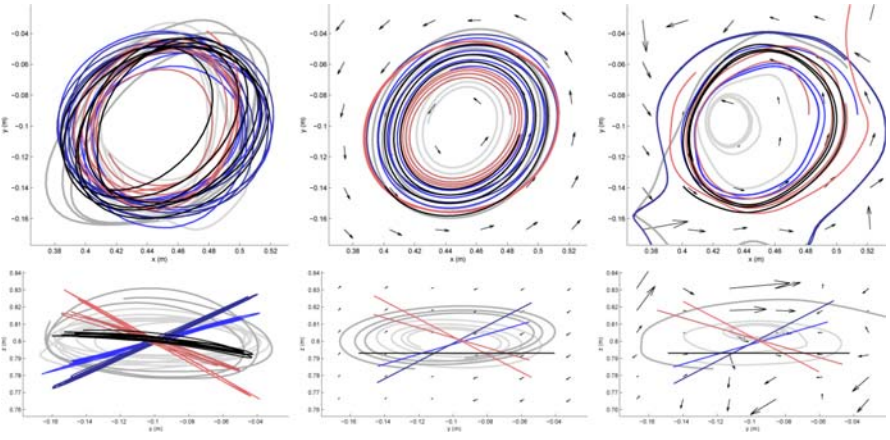

Fig. 13 Learning from human wiping demonstrations. Left: Trajectories of the sponge when wiping on the surface when flat (black), tilted $+16^\circ$ and $+27^\circ$ about the x-axis (red), -16° and -27° about the x-axis (blue), and $\pm 16^\circ$ about the y-axis (grey). Centre and right: Reproduced trajectories using the policies (black arrows) learnt with the inconsistency and direct approaches, respectively. In each case the same example trajectory is highlighted (thick black). The top and front views are shown (top and bottom rows).

constraint-consistent learning with the inconsistency approach was compared to that of direct regression (since in this case the task is clearly periodic, the inconsistency approach is the appropriate choice of constraint-consistent method.)

The results are shown in Fig. 13, where we show the demonstrated trajectories (left), those produced by the constraint-consistent policy (centre) and those learnt by direct regression (right) under the different constraints (tilts of the surface). Looking at the learnt policies, we see that the constraint-consistent approach learns a smooth policy and the trajectories under each of the constraints are smooth periodic movements, similar to those of the human. On the ASIMO robot these produced smooth, natural wiping movements (see Fig. 14).

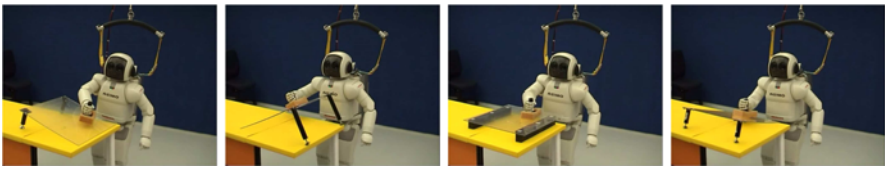

Fig. 14 Reproduced movements on the ASIMO robot for the surface tilted 0°, $+16^\circ$, -27° about the x-axis, and $+16^\circ$ about the y-axis.

The policy learnt with direct regression also captured 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 direct approach, naive to the constraints, learns an unstable policy that cannot be safely used for movement reproduction on the robot.

6 Discussion

In this chapter, we reviewed recent work in the area of policy-based learning of demonstrated movements, where those movements are subject to variable, dynamic, non-linear constraints. We discussed the problems encountered when learning in this setting and showed, through analysis and experimentation, how approaches based on standard supervised learning techniques come into difficulties when presented with data containing variable constraints. We then went on to describe two new learning methods for learning in a constraint-consistent manner. The first, earlier approach solved the learning problem for constrained potential-based policies. The second approach, based on a modification in the calculation of an empirical risk, was shown to be effective for arbitrary policies, including those with a rotational component. It was seen that both methods were capable of recovering the unconstrained policy from arbitrarily constrained observations, without the need for explicit knowledge of the constraints. This allows us to learn policies that generalise over constraints, including novel constraints, unseen in the training data. Furthermore, the comparative performance of the methods was reviewed for learning policies on systems of varying size and complexity.

How far do the policies generalise?

The results presented here and in [22, 19] clearly show the efficacy of learning from constrained demonstrations using the two approaches, and then applying the resultant policies to new constraint scenarios. However, in terms of lessons learnt from these studies there are also some bigger issues raised. One such issue is the question of when, faced with a new constraint, the learnt policy will fail at the desired task. For example, in the ball grasping experiment, under certain configurations of the constraints (e.g. if the barriers were placed exactly on either side of the ball, or a much larger barrier was placed between the robot and the ball) the learnt policy would fail at the task of grasping. This may be due to several factors, for instance if the control vector happens to be orthogonal to the nullspace of the constraint, deadlock would occur (this is similar to the problem of local minima in many gradient-based controllers, e.g. see [11]). While problems such as these are in general unavoidable when dealing with constrained systems, one of the nice properties of the

constraint-consistent approaches is that they learn policies that are successful *under the same constraints that the demonstrator is successful*. So, although the learnt policy for the grasping task is not guaranteed to successfully get the ball in the presence of any arbitrary barrier (constraint), it successfully reaches the ball whenever (i.e. with whatever barriers) the demonstrator does. In some sense we can say the *robustness* of the demonstrator's policy against different constraints was transferred to the learner.

Adaptation to constraints: Re-planning vs. re-using policies?

A second, related issue concerns the role of adaptation of policies in response to constraints. Clearly there are circumstances in which it is desirable to re-plan the policy to cope with certain sets of constraints, especially if the learner's existing policy (here, learnt from demonstration) fails under those constraints (and in some cases the learner may even take advantage of certain types of constraint to improve performance). However, here a balance must be struck. On the one hand re-planning the policy will likely improve performance under any given set of constraints; but on the other hand the adapted policy will also become more specialised to that particular set of constraints (and may even lead to degraded performance for other constraints). In other words we lose the *generalisation to other constraints* that here we attempt to extract from the demonstrator. Furthermore, due to the inherent uncertainty in the constraints in most real world problems, it may not be feasible to explicitly incorporate all of the constraints when re-planning. For example consider planning a policy for walking on uneven terrain; to explicitly incorporate the constraints involved here would require a detailed model of the terrain, which is rarely available. The constraint-consistent approaches, however, allow us to sidestep this, providing a shortcut to uncovering the policy used by the demonstrator[8] (who, if observed to use the same policy under a number of constraint settings, presumably finds it sufficiently successful for those and similar settings). Therefore in this sense, with these approaches, we may now envisage a move away from the traditional approach of planning explicitly with respect to all possible constraints that is typically only possible in highly controlled, structured environments.

References

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

[8] It should also be noted that our approach may also be combined with adaptive methods, for example using the policy learnt from demonstration to initialise further optimisation of the policy, similar to e.g.,[18].

[2] Antonelli, G., Arrichiello, F., Chiaverini, S.: The null-space-based behavioral control for soccer-playing mobile robots. In: IEEE International Conference Advanced Intelligent Mechatronics, pp. 1257–1262 (2005)
[3] Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. In: Robotics and Autonomous Systems (2008) (in press) (Corrected Proof)
[4] Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Handbook of Robotics, ch. 59. MIT Press, Cambridge (2007)
[5] Bolder, B., Dunn, M., Gienger, M., Janssen, H., Sugiura, H., Goerick, C.: Visually guided whole body interaction. In: IEEE International Conference on Robotics and Automation, pp. 3054–3061 (2007)
[6] Calinon, S., Billard, A.: Learning of gestures by imitation in a humanoid robot. In: Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions (2007)
[7] Chajewska, U., Koller, D., Ormoneit, D.: Learning an agent's utility function by observing behavior. In: International Conference on Machine Learning (2001)
[8] Chajewska, U., Getoor, L., Norman, J., Shahar, Y.: Utility elicitation as a classification problem. In: Uncertainty in Artificial Intelligence, pp. 79–88. Morgan Kaufmann Publishers, San Francisco (1998)
[9] Chaumette, F., Marchand, A.: A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. IEEE Trans. Robotics and Automation 17(5), 719–730 (2001)
[10] Il Choi, S., Kim, B.K.: Obstacle avoidance control for redundant manipulators using collidability measure. Robotica 18(2), 143–151 (2000)
[11] Conner, D.C., Rizzi, A.A., Choset, H.: Composition of local potential functions for global robot control and navigation. In: IEEE International Conference on Intelligent Robots and Systems, October 27-31, vol. 4, pp. 3546–3551 (2003)
[12] D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: IEEE International Conference on Intelligent Robots and Systems (2001)
[13] English, J.D., Maciejewski, A.A.: On the implementation of velocity control for kinematically redundant manipulators. IEEE Transactions on Systems, Man and Cybernetics 30(3), 233–237 (2000)
[14] Fumagalli, M., Gijsberts, A., Ivaldi, S., Jamone, L., Metta, G., Natale, L., Nori, F., Sandini, G.: Learning how to exploit proximal force sensing: A comparison approach. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 149–167. Springer, Heidelberg (2010)
[15] Gienger, M., Janssen, H., Goerick, C.: Task-oriented whole body motion for humanoid robots. In: IEEE International Conference on Humanoid Robots, December 5, pp. 238–244 (2005)
[16] Grimes, D.B., Chalodhorn, R., Rajesh, P.N.R.: Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Robotics: Science and Systems. MIT Press, Cambridge (2006)
[17] Grimes, D.B., Rashid, D.R., Rajesh, P.N.R.: Learning nonparametric models for probabilistic imitation. In: Advances in Neural Information Processing Systems. MIT Press, Cambridge (2007)

[18] Guenter, F., Hersch, M., Calinon, S., Billard, A.: Reinforcement learning for imitating constrained reaching movements. RSJ Advanced Robotics, Special Issue on Imitative Robots 21(13), 1521–1544 (2007)
[19] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Behaviour generation in humanoids by learning potential-based policies from constrained motion. Applied Bionics and Biomechanics 5(4), 195–211 (2008) (in press)
[20] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Learning potential-based policies from constrained motion. In: IEEE International Conference on Humanoid Robots (2008)
[21] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: A novel method for learning policies from constrained motion. In: IEEE International Conference on Robotics and Automation (2009)
[22] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: A novel method for learning policies from variable constraint data. In: Autonomous Robots (submitted, 2009)
[23] Howard, M., Vijayakumar, S.: Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In: Workshop on Robotics and Mathematics (September 2007)
[24] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: IEEE International Conference on Robotics and Automation, pp. 1398–1403 (2002); ICRA 2002 best paper award
[25] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, pp. 1523–1530. MIT Press, Cambridge (2003)
[26] Inamura, T., Toshima, I., Tanie, H., Nakamura, Y.: Embodied symbol emergence based on mimesis theory. The International Journal of Robotics Research 23(4), 363–377 (2004)
[27] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In: IEEE Int. Conf. on Intelligent Robots and Systems (2003)
[28] Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, bad and spectral. Journal of the ACM 51(3), 497–515 (2004)
[29] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 428–436 (1985)
[30] Khatib, O.: 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 (1987)
[31] Körding, K.P., Fukunaga, I., Howard, I.S., Ingram, J.N., Wolpert, D.M.: A neuroeconomics approach to inferring utility functions in sensorimotor control. PLoS Biolology 2(10), 330 (2004)
[32] Körding, K.P., Wolpert, D.M.: The loss function of sensorimotor learning. Proceedings of the National Academy of Sciences 101, 9839–9842 (2004)
[33] Liégeois, A.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Trans. Sys., Man and Cybernetics 7, 868–871 (1977)

[34] Mattikalli, R., Khosla, P.: Motion constraints from contact geometry: Representation and analysis. In: IEEE International Conference on Robotics and Automation (1992)
[35] Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
[36] Nakamura, Y.: Advanced Robotics: Redundancy and Optimization. Addison Wesley, Reading (1991)
[37] Ohta, K., Svinin, M., Luo, Z., Hosoe, S., Laboissiere, R.: Optimal trajectory formation of constrained human arm reaching movements. Biological Cybernetics 91, 23–36 (2004)
[38] Park, J., Khatib, O.: Contact consistent control framework for humanoid robots. In: IEEE International Conference on Robotics and Automation (May 2006)
[39] Peters, J., Mistry, M., Udwadia, F.E., Nakanishi, J., Schaal, S.: A unifying framework for robot control with redundant dofs. Autonomous Robots 24, 1–12 (2008)
[40] Peters, J., Schaal, S.: Learning to control in operational space. The International Journal of Robotics Research 27(2), 197–212 (2008)
[41] Ren, J., McIsaac, K.A., Patel, R.V.: Modified Newton's method applied to potential field-based navigation for mobile robots. In: IEEE Transactions on Robotics (2006)
[42] Rimon, E., Koditschek, D.E.: Exact robot navigation using artificial potential functions. IEEE Transactions on Robotics and Automation 8(5), 501–518 (1992)
[43] De Sapio, V., Khatib, O., Delp, S.: Task-level approaches for the control of constrained multibody systems (2006)
[44] De Sapio, V., Warren, J., Khatib, O., Delp, S.: Simulating the task-level control of human motion: a methodology and framework for implementation. The Visual Computer 21(5), 289–302 (2005)
[45] Schaal, S.: Learning from demonstration. In: Mozer, M.C., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems, pp. 1040–1046. MIT Press, Cambridge (1997)
[46] Schaal, S., Atkeson, C.G.: Constructive incremental learning from only local information. Neural Computation 10, 2047–2084 (1998)
[47] Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philosophical Transactions: Biological Sciences 358(1431), 537–547 (2003)
[48] Sentis, L., Khatib, O.: Task-oriented control of humanoid robots through prioritization. In: IEEE International Conference on Humanoid Robots (2004)
[49] Sentis, L., Khatib, O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. International Journal of Humanoid Robotics 2(4), 505–518 (2005)
[50] Sentis, L., Khatib, O.: A whole-body control framework for humanoids operating in human environments. In: IEEE International Conference on Robotics and Automation (May 2006)
[51] Sugiura, H., Gienger, M., Janssen, H., Goerick, C.: Real-time collision avoidance with whole body motion control for humanoid robots. In: IEEE International Conference on Intelligent Robots and Systems, pp. 2053–2058 (2007)

[52] Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge (1998)
[53] Takano, W., Yamane, K., Sugihara, T., Yamamoto, K., Nakamura, Y.: Primitive communication based on motion recognition and generation with hierarchical mimesis model. In: IEEE International Conference on Robotics and Automation (2006)
[54] Todorov, E.: Optimal control theory. In: Doya, K. (ed.) Bayesian Brain. MIT Press, Cambridge (2006)
[55] Udwadia, F.E., Kalaba, R.E.: Analytical Dynamics: A New Approach. Cambridge University Press, Cambridge (1996)
[56] Verbeek, J.: Learning non-linear image manifolds by combining local linear models. IEEE Transactions on Pattern Analysis & Machine Intelligence 28(8), 1236–1250 (2006)
[57] Verbeek, J., Roweis, S., Vlassis, N.: Non-linear cca and pca by alignment of local models. In: Advances in Neural Information Processing Systems (2004)
[58] Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation 17(12), 2602–2634 (2005)
[59] Yoshikawa, T.: Manipulability of robotic mechanisms. The International Journal of Robotics Research 4(2), 3–9 (1985)