2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

A Novel Method for Learning Policies from Constrained Motion

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 environment. Constraints are usually unobservable and frequently change between contexts. In this paper, we present a novel approach for learning (unconstrained) control policies from movement data, where observations come from movements under different constraints. As a key ingredient, we introduce a small but highly effective modification to the standard risk functional, allowing us to make a meaningful comparison between the estimated policy and constrained observations. We demonstrate our approach on systems of varying complexity, including kinematic data from the ASIMO humanoid robot with 27 degrees of freedom.**

I. Introduction

A wide variety of everyday human skills can be framed in terms of performing some task subject to constraints imposed by the physical environment [8]. Examples include opening a door, pulling out a drawer or stirring soup in a saucepan.

In a more generic setting, constraints may take a much wider variety of forms. For example in climbing a ladder, the constraint may be on the centre of mass or the tilt of the torso of the climber to prevent over-balancing. Or in contact control [9] problems such as manipulation or grasping a solid object, the motion of fingers is constrained during the grasp by the presence of the object. Also in systems designed to be highly competent and adaptive, such as humanoid robots, behaviour may be subject to a wide variety of constraints [3], usually non-linear in actuator space and often discontinuous. Consider the task of running or walking on uneven terrain: the cyclic movement of the legs of the runner is constrained by the impact of the feet on the ground in a dynamic, discontinuous and unpredictable way. A promising approach to providing robots with such skills as running and opening doors, is to take examples of motion from existing systems, such as humans, and attempt to learn a control policy that somehow captures the desired behaviour [2], [1], [13], [5]. An important component of this is the ability to deal with the effect of constraints and the apparent variability in the observed movement induced by these constraints. For example one wishes to learn a policy that allows one to open many doors of varying widths, or stir in saucepans of varying sizes.

The focus in this paper is on modelling control policies subject to generic constraints on motion, with the aim of finding policies that can generalise *over different constraints*. We take a direct policy learning (DPL) approach[1] [13] whereby we attempt to learn a continuous model of the policy from motion data. While DPL has been studied for a variety of control problems in recent years (for a review, see [1] and references therein), crucially these problems involved policies that are either directly observable from motion data, i.e. unconstrained policies, or policies subject to identical constraints in every observation (in which case the constraints can be absorbed into the policy itself) [2]. The difference here is that we consider observations from policies projected into the nullspace of a set of dynamic, non-linear constraints, and that these constraints may change between observations, or even during the course of a single observation.

In general, learning (unconstrained) policies from constrained motion data is a formidable task. This is due to (i) the *non-convexity* of observations under different constraints, and; (ii) *degeneracy* in the set of possible policies that could have produced the movement under the constraint. However, despite these hard analytical limits, we will show that it is still possible to find a good approximation of the unconstrained policy given observations under the right conditions. Our proposal is to reformulate the standard risk functional by introducing a projection of the estimated policy onto the observations before calculating errors. By making this simple, but significant alteration, we show that it is possible to model the unconstrained policy (i) with no explicit knowledge of the constraints, and; (ii) without explicit access to unconstrained policy vectors. Furthermore, we show that using this approach one can fully reconstruct the unconstrained policy given observations under a sufficiently rich set of constraints. To validate the approach we modify standard regression techniques to use the proposed objective function and demonstrate robust learning for several policies on complex, high-dimensional movement systems, subject to realistic constraints.

II. Learning from Constrained Policies

Here, we characterise the problem of direct policy learning when constraints are applied to motion. Following [13], [11], we consider the learning of autonomous policies

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

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$ are some appropriately chosen state and action vectors[2] and $\boldsymbol{\pi}$ is the policy mapping between the two.

The goal of DPL is to approximate the policy $\boldsymbol{\pi}$ as closely as possible [13], [11] given observations (often in the form of trajectories) of the states and actions $\mathbf{u}(t)$, $\mathbf{x}(t)$. In previous work this has been done by fitting parametrised models in

M. Howard, S. Klanke and S. Vijayakumar are with the Institute of Perception Action and Behaviour, University of Edinburgh, Scotland, UK. E-mail: `matthew.howard@ed.ac.uk`

M. Gienger and C. Goerick are with the Honda Research Institute Europe (GmBH), Offenbach, Germany. E-mail: `michael.gienger@honda-ri.de`

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

[2]For example in kinematic control, the state vector could be the joint angles, $\mathbf{x} \equiv \mathbf{q}$, and the action could be the velocities $\mathbf{u} \equiv \dot{\mathbf{q}}$, or in dynamic control a suitable state might be, $\mathbf{x} \equiv \mathbf{q}, \dot{\mathbf{q}}$, with actions corresponding to applied torques, $\mathbf{u} \equiv \boldsymbol{\tau}$.

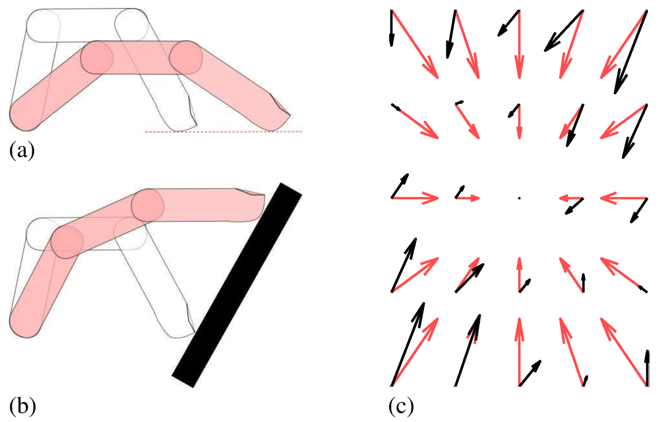

Fig. 1. Illustration of two apparently different behaviours from the same policy: (a) unconstrained movement (b) movement constrained by an obstacle (black box) (c) the unconstrained (red) and constrained (black) policy over two of the joints of the finger.

the form of dynamical systems [5], non-parametric modelling [11], and probabilistic Bayesian approaches [4].

An implicit assumption found in DPL approaches to date is that the data used for training comes from behavioural observations of some *unconstrained* or *consistently constrained* policy [2]. 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/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 DPL approaches.

As an example, consider the learning of a simple policy to extend a jointed finger. In Fig. 1(a) the finger is unconstrained and the policy simply moves the joints towards the zero (outstretched) position. On the other hand, in Fig. 1(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. 1(c).

In standard DPL [13], [5], these two apparently different behaviours would lead to the learning of two separate policies (i.e. the different coloured vector fields) for extending the finger in the two settings. However, the fact that the goals of the two policies are similar ('extend the finger') suggests that in fact the movement stems from the *same policy* under *different constraints*. Viewed like this, instead of learning two separate policies we would rather learn a single policy that *generalises* over the different constraints.

The best policy representation of the movements in Fig. 1 is that of the unconstrained policy $\boldsymbol{\pi}$, since this gives maximal information about the behaviour. Knowing $\boldsymbol{\pi}$, or finding a good approximation of it, we can (i) reproduce observed constrained behaviours (cf. Fig. 1(b)) simply by applying the same constraints, (ii) predict the unconstrained behaviour (cf. Fig. 1(a)) in parts of the space where only constrained movements have been seen, and; (iii) even predict behaviour in situations where novel constraints, unseen in the training data, apply.

A. Constraint Model

In this paper we consider constraints which act as hard restrictions on actions available to the policy. Mathematically, we say given a set of k-dimensional constraints

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

the policy is projected into the nullspace of those constraints

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

where $\mathbf{N}(\mathbf{x},t) \equiv (\mathbf{I} - \mathbf{A}^\dagger\mathbf{A}) \in \mathbb{R}^{d\times d}$ is in general a time-varying projection operator that is non-linear in state[3], $\mathbf{A}(\mathbf{x},t) \in \mathbb{R}^{k\times d}$ 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. Such constraints are also common in the control of redundant degrees of freedom in high-dimensional manipulators [7], [6], [10], where policies such as (3) are used, for example, to aid joint stabilisation under task constraints. As an example: Setting $\mathbf{A}$ to the Jacobian that maps from joint-space to end-effector position coordinates would allow any motion in the joint space provided that the end-effector remained stationary.

Learning the unconstrained policy $\boldsymbol{\pi}(\mathbf{x})$ from observations of the constrained actions $\mathbf{u}(\mathbf{x},t)$ is a non-trivial task for several reasons. Firstly, we commonly do not know exactly what constraints $\mathbf{A}(\mathbf{x},t)$ (and therefore $\mathbf{N}(\mathbf{x},t)$) are in force in different observations. For example we may not know the exact radius of an opening door. Secondly there is the problem of *non-convexity* of the training targets, that is the different projections $\mathbf{N}(\mathbf{x},t)$ cause the action vectors $\mathbf{u}$ to appear different under different constraints. For example compare the set of constrained (black) and unconstrained (red) vectors for the policy shown in Fig. 1(c). Finally, there is *degeneracy* in the sense that for any given observation $\mathbf{u}$, there may be multiple policies $\boldsymbol{\pi}$ that could be projected to produce that observation.

However, despite these restrictions, by reformulating the standard DPL learning problem, we will show that it is still possible to learn a good model of the policy $\boldsymbol{\pi}$, without need for explicit knowledge of the constraints $\mathbf{N}(\mathbf{x},t)$, and that is, as a minimum, consistent with all constrained observations. We turn to this in the next section.

III. Method

Our method works on data that is given as tuples $(\mathbf{x}_n, \mathbf{u}_n)$ of observed states and constrained actions. We assume that all commands $\mathbf{u}$ are generated from the same underlying policy $\boldsymbol{\pi}(\mathbf{x})$, which for a particular observation might have been constrained, that is $\mathbf{u}_n = \mathbf{N}_n\boldsymbol{\pi}(\mathbf{x}_n)$ for some projection matrix $\mathbf{N}_n$. Furthermore 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.

With only $\mathbf{x}_n$ and $\mathbf{u}_n$ given, one may be tempted to simply minimise

$$E_{naive}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2. \tag{4}$$

[3]Here, $\mathbf{A}^\dagger$ denotes the Moore-Penrose pseudoinverse of the matrix $\mathbf{A}$

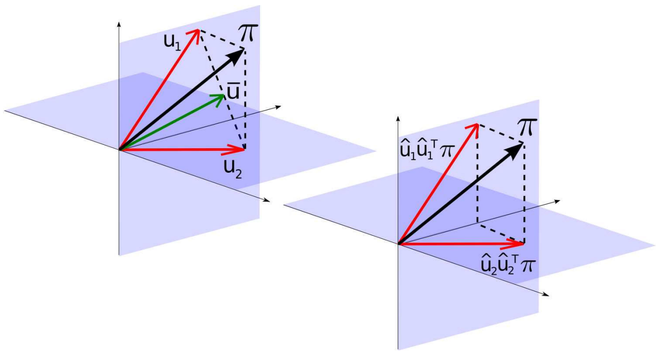

Fig. 2. Illustration of our learning scheme. Left: Naive regression on constrained commands $\mathbf{u}_1, \mathbf{u}_2$ results in averaging of the observations $\bar{\mathbf{u}}$ in a way that cannot explain the observed commands. Right: The projection of the correct policy $\boldsymbol{\pi}$ onto the observations matches those observations.

However this would ignore that constraints might have been in force and correspond to a naive averaging of commands from different circumstances (cf. Fig. 2).

If we had access to samples of either (i) the (unconstrained) policy $\boldsymbol{\pi}_n = \boldsymbol{\pi}(\mathbf{x}_n)$, or (ii) the projection matrices $\mathbf{N}_n$, we could use standard regression techniques to estimate a policy $\tilde{\boldsymbol{\pi}}(\mathbf{x})$ by minimising an appropriate risk functional. Specifically in the former case we could minimise

$$E_{upe}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\boldsymbol{\pi}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2. \tag{5}$$

In the latter case, we could minimise

$$E_{cpe}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{N}_n\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2, \tag{6}$$

which we refer to as the *unconstrained policy error* (UPE) and *constrained policy error* (CPE), respectively. However, since by assumption samples of $\boldsymbol{\pi}_n$ and $\mathbf{N}_n$ are not available these functionals cannot be used to estimate the policy.

Instead, we aim to estimate a policy $\tilde{\boldsymbol{\pi}}(\cdot)$ that is *consistent* with our observed $\mathbf{u}_n$. That is we wish to reconstruct the policy, knowing that it is 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*. That is, 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. 2, right). 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 replace $\mathbf{N}_n$ in (6) by a projection onto $\mathbf{u}_n$ and minimise the *inconsistency* which we define as the functional

$$E_i[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \hat{\mathbf{u}}_n\hat{\mathbf{u}}_n^T\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 = \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\right)^2 \tag{7}$$

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 the norm of the difference between true and predicted policy. The same argument holds for the inconsistency error (7) 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

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

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. 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 can be used in conjunction with many standard regression techniques. However, for the experiments in this paper, we restrict ourselves to two classes of function approximator for learning the (unconstrained) policy to demonstrate how the risk functional can be used. The example function approximators we use are (i) simple parametric models with fixed basis functions (Sec. III-A), and (ii) locally linear models (Sec. III-B). In the next section we describe how these two models can be reformulated to take advantage of the new risk functional.

A. *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 $\mathbf{c}_i$, $i = 1\ldots M$. With this model, the *inconsistency* error from (7) 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

$$\mathbf{w}^{opt} = \arg\min E_i(\mathbf{w}) = \mathbf{H}^{-1}\mathbf{g} \tag{8}$$

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 $O(d^2M^2)$.

B. Locally linear policy models

The basis function approach quickly becomes nonviable 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 [12]. 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

$$\mathbf{b}_m^{opt} = \arg\min E_i(\mathbf{b}_m) = \mathbf{H}_m^{-1}\mathbf{g}_m \tag{9}$$

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

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

IV. Experiments

To explore the performance of our algorithm, we performed experiments on data from autonomous kinematic control policies [13] applied to three simulated plants. In our first set of experiments we illustrate the concepts involved on an artificial two-dimensional toy system[4]. We then demonstrate our algorithm applied to higher dimensional plants including a physically realistic simulation of the 7-DOF DLR lightweight arm, and to whole body motion control of the 27-DOF humanoid robot ASIMO [3].

A. Toy Example

Our first experiment demonstrates the learning of unconstrained policies from constrained trajectories in a simple toy example consisting of a two-dimensional system with discontinuously switching motion constraints. As an example policy, we used a limit cycle attractor (Fig. 3, left) of the form

$$\dot{r} = r(\rho - r^2), \qquad \dot{\theta} = \omega \tag{10}$$

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 we set $\rho = 0.5\ m$ and $\omega = 1\ rad\ s^{-1}$ with a sampling rate of 50 Hz. Data was collected by recording 40 trajectories of length 40 time steps each, generated by the policy from a random start state. During the trajectories the policy was subjected to random 1-D constraints

$$\mathbf{A}(\mathbf{x}, t) = (\alpha_1, \alpha_2) \equiv \boldsymbol{\alpha} \tag{11}$$

where the $\alpha_{1,2}$ were drawn from a normal distribution, $\alpha_i = N(0, 1)$. The constraints mean that motion is constrained in the direction orthogonal to the vector $\boldsymbol{\alpha}$ in state space. These were randomly switched by generating a new $\boldsymbol{\alpha}$ twice at regular intervals during the trajectory, inducing sharp turns which can be seen in Fig. 3 (right).

We used a parametric model to learn the policy through minimisation of the inconsistency (7) as described in Sec. III-A. We included the regularisation term and picked the parameter λ by minimising the inconsistency on a validation subset. For this toy problem, we chose our function model as a set of 36 normalised RBFs centred on a 6×6 grid, and we simply fixed the kernel width to yield suitable overlap. We repeated this experiment on 100 data sets and evaluated the normalised UPE, CPE and the inconsistency[5], that is, the functionals from (5), (6) and (7) divided by the number of data points and the variance of the policy $\boldsymbol{\pi}_n$ on a subset held out for testing. For comparison, we repeated the experiment using a naive approach that attempted to perform regression with the same RBF model directly on the constrained observations. That is, the naive approach attempted to minimise the functional (4).

Figure 3 shows the true policy, the trajectories we trained on, the policies learnt using our and the naive approach, and finally the error statistics below the plots. With an average nUPE of 0.0027, our method outperforms the naive approach by orders of magnitude. Notably, even with only 4 trajectories (Fig. 3, column 2) the reconstructed policy already resembles the limit cycle, although large errors still persist in some parts of the state space (e.g. the lower right corner). Further to this, the top panel of Fig. 4 depicts how the nUPE and nCPE evolve with increasing size of the training set, showing a smooth decline (please note the log. scale). In order to further explore the performance of our algorithm, we contaminated the observed commands $\mathbf{u}_n$ with Gaussian noise, the scale of which we varied to match up to 20% of the scale of the data. The resulting nUPE roughly follows the noise level, as is plotted in Fig. 4 (bottom).

B. Generalisation over unseen constraints

The two goals of our second set of experiments were to characterise (i) how well the algorithm scaled to more complex, realistic constraints and (ii) how well the learnt policies generalised over unseen constraints. For this we used a kinematic simulation of the 7-DOF DLR lightweight robot (LWR-III). The experimental procedure was as follows: We generated a random initial posture by drawing 7 joint angles uniformly from half the range of each joint, that is $x_i \sim U[-0.5x_i^{max}; 0.5x_i^{max}]$, where for example $x_1^{max} = 170°$. We set up a joint limit avoidance type policy as $\boldsymbol{\pi}(\mathbf{x}) = -0.05\nabla\Phi(\mathbf{x})$, with the potential given by $\Phi(\mathbf{x}) = \sum_{i=1}^{7}|x_i|^p$ for $p = 1.5, p = 1.8$, or $p = 2.0$. We then generated 100 trajectories with 100 points each, following the policy under

[4]In fact even these 'simplified' problems are relevant to constrained policies in low dimensional task spaces, such as end-effector space.

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

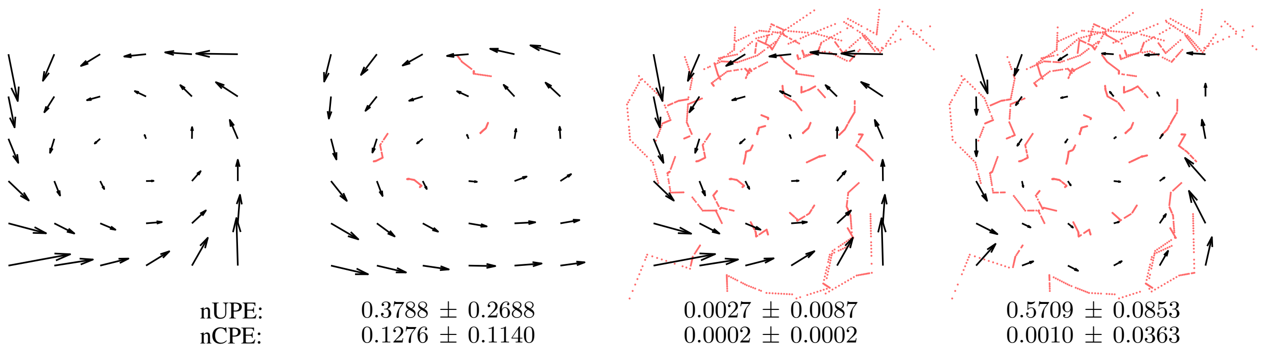

Fig. 3. From left to right: 1) true limit cycle policy, 2) learnt policy trained on 4 constrained trajectories, 3) learnt policy from 40 constr. traj., 4) policy resulting from naive regression on observed commands. Trajectories are shown as dotted lines, the policy is depicted by black arrows. The normalised CPE and UPE (mean±s.d. over 100 data sets) are given below the figures.

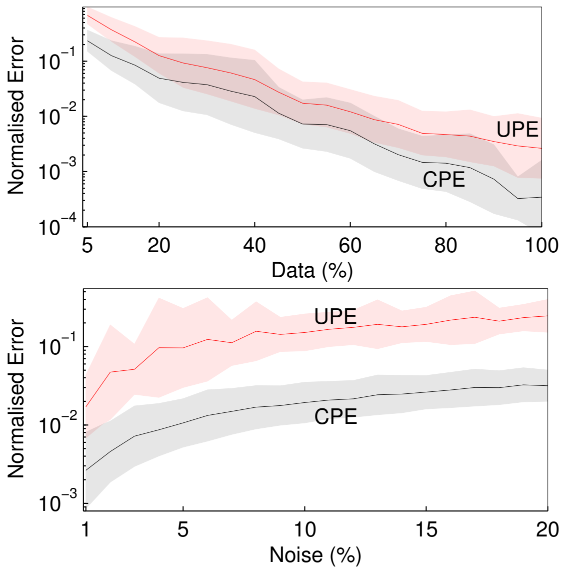

Fig. 4. Top: Normalised UPE and CPE versus data set size as a percentage of the full $K=40$ trajectories of length $N=40$. Bottom: Normalised UPE and CPE for increasing noise levels in the observed $\mathbf{u}_n$.

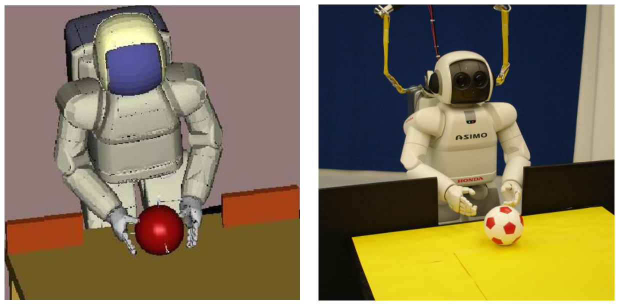

Fig. 5. Example constrained reaching movement demonstrated by the expert policy. Starting with hands at the sides, the teacher robot reaches between the barriers to grasp the ball.

4 different constraints, which we refer to as 1-2-3, 4-5-6, 1-3-5, and 2-4-6. Here, the three numbers denote which end-effector coordinates in task space[6] we kept fixed, that is, 1-2-3 means we constrained the end-effector position, but allowed arbitrary changes in the orientation. Similarly, 2-4-6 means we constrained the y-coordinate and the orientation around the x- and z-axis, while allowing movement in x-z position and around the y-axis. For all 4 constraint types, we estimated the policy from a training subset, and evaluated it on test data from the same constraint, as well as on trajectories from the complementary constraint (e.g., 2-4-6 is complementary to 1-3-5).

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

While the linear policy $\boldsymbol{\pi}(\cdot)$ corresponding to $p=2.0$ was learnt almost perfectly (all normalised errors in the order of 10^{-9}), the less linear policies ($p = 1.8$ and especially $p=1.5$) turned out to be a much harder problem. This can be seen when comparing both the nUPE and nCPE for the two policies (ref. Table I). Still, we recovered the constrained policy in all cases to good accuracy (ref. Table I, 4th column), with good generalisation to the complementary constraints (ref. Table I, 5th column). We can also see that constraining the end-effector position (1-2-3) made it more difficult to recover the unconstrained policy compared to constraining the orientation (4-5-6), or using mixed constraints (1-3-5 and 2-4-6). It should also be noted that running the same experiment using the naive approach (ref. Sec.IV-A) gave consistently poor results; for example, when training on data under the (1-2-3) constraint, the naive approach gave nUPE of $83.44 \pm 1.20 \times 10^{-2}$ for the $p = 1.5$ policy, $80.94 \pm 1.37 \times 10^{-2}$ for $p = 1.8$ and $79.62 \pm 1.39 \times 10^{-2}$ for $p=2.0$.

C. Grasping a Ball

The goal of our next set of experiments was to illustrate the utility of our approach for learning from observations of an everyday task with realistic constraints. For this we chose 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. 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 using the whole

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

Policy	Constr.	nUPE	nCPE	Compl. nCPE
$p=1.5$	1 - 2 - 3	64.33 ± 32.03	2.91 ± 0.36	15.95 ± 6.47
	4 - 5 - 6	34.75 ± 19.12	2.49 ± 0.22	15.47 ± 7.75
	1 - 3 - 5	16.17 ± 3.81	3.20 ± 0.27	5.10 ± 1.07
	2 - 4 - 6	10.35 ± 1.82	2.72 ± 0.23	4.74 ± 0.95
$p=1.8$	1 - 2 - 3	8.09 ± 5.76	0.47 ± 0.08	2.27 ± 1.13
	4 - 5 - 6	5.36 ± 2.96	0.35 ± 0.03	2.22 ± 0.98
	1 - 3 - 5	2.27 ± 0.64	0.45 ± 0.04	0.77 ± 0.17
	2 - 4 - 6	1.42 ± 0.31	0.40 ± 0.04	0.72 ± 0.17

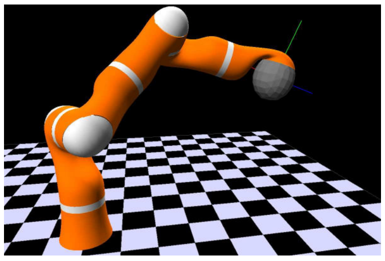

TABLE I

NORMALISED UPE, CPE ON THE TRAINING CONSTRAINTS, CPE ON COMPLEMENTARY CONSTRAINTS AND INCONSISTENCY ERROR, FOR DATA FROM THE DLR ARM (RIGHT). ALL ERRORS NORMALISED BY THE VARIANCE OF THE POLICY. WE REPORT (MEAN $\pm$ S.D.)$\times 10^{-2}$ OVER 100 TRIALS WITH DIFFERENT DATA SETS.

body motion (WBM) controller of the 27-DOF humanoid robot ASIMO (see [3] for details). We set up an 'expert' demonstrator robot from which observations were recorded. For simplicity, the expert's policy was defined by an inverted Gaussian potential

$$\pi(\mathbf{x}) = \nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = \alpha\left(1 - e^{\|\mathbf{x}-\mathbf{x}_c\|^2/2\sigma^2}\right), \quad (12)$$

where we chose $\sigma^2 = 2$, $\alpha = 0.25$ and the target point $\mathbf{x}_c \in \mathbb{R}^n$ to correspond to a grasping position, with the two hands positioned on either side of the ball. The state-space of the policy was defined as the Cartesian position of the two hands, corresponding to 6 DOFs in state and action space (hereafter, the 'task space'). In order to realise the task space policy motion, joint-space control was performed using inverse kinematics via the WBM controller.

The expert's movements were constrained by the presence of a barrier on the table with a gap in it. The constraints acted on each of the hands so that motion in the direction normal to the barrier surface was prevented if a hand came too close (cf. [14]). The barrier was placed such that the expert robot had to reach through the gap to get the ball (ref. Fig. 5). Such state-dependent constraints are both nonlinear in the state space and have discontinuously switching dimensionality when either hand 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). 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.

The constraints were varied by randomly changing the width of the gap for each trajectory. 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 $0.15m$. Fig. 5 shows the experimental set-up.

We used our algorithm to perform learning on 50 such data sets using 150 local linear models, with centres placed using k-means. For comparison, we also repeated the experiment on the same data, using the naive approach (as described in Sec. IV-A). That is, the same local linear models were used, but were trained directly on the tuples $(\mathbf{x}_i, \mathbf{u_i} \equiv \dot{\mathbf{x}}_i), i = 1, \ldots K \times N$ using the risk functional (4).

To assess the performance for both methods we evaluated the errors in predicting the policy subject to (i) the training

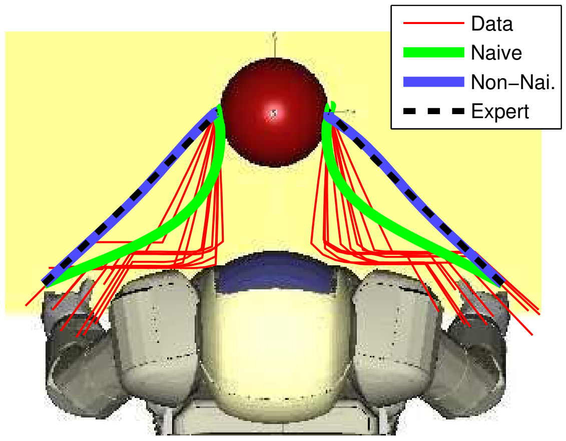

Fig. 6. Unconstrained reaching movement for the expert policy (black dashed), policy learnt by optimisation of the inconsistency (blue), and the naive approach (green). Ten example constrained reaching movements used for training are shown in red.

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

TABLE II

NORMALISED POLICY ERRORS FOR PREDICTING THE POLICY UNDER THREE CONSTRAINT CONDITIONS FROM THE BALL-GRASPING DATA FOR THE NAIVE AND NON-NAIVE METHODS. VALUES ARE MEAN$\pm$S.D. OVER 50 DATA SETS.

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 grasp the ball.

As expected, learning using the proposed risk functional (7) (the 'non-naive' approach) performed several orders of magnitude better than the naive approach in terms of the numerical error measures (ref. Table II). However, the real difference in the methods is best highlighted if we compare trajectories generated by the two policies. In Fig. 6 we show example trajectories for the unconstrained reaching movement produced by the expert (black), and the policies learnt by (i) optimising E_{naive} (green), and (ii) optimising E_i (blue). In the former the hands take a curved path to the ball, reproducing the average behaviour of the (constrained) demonstrated trajectories – the naive method is unable to extract the underlying task (policy) from the observed paths around the obstacles. In contrast, the policy learnt with the non-naive approach better predicts the unconstrained policy, enabling it to take a direct route to the ball that closely matches that of the expert. The behaviour of the expert and learnt policies can be examined in detail in the accompanying video.

D. Learning from High-dimensional Joint-space Data

In our final experiment we tested the scalability of our approach for learning in very high dimensions. For this we chose a policy defined by a quadratic potential in the joint space (i.e. $\mathbf{x} \equiv \mathbf{q} \in \mathbb{R}^{27}$)

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_c)^T \mathbf{W}(\mathbf{x} - \mathbf{x}_c), \quad (13)$$

where $\mathbf{x}_c \in \mathbb{R}^{27}$ is a target posture and $\mathbf{W}$ is a weighting matrix. The policy (13) represents an attractor in joint space that pulls the robot into a desired posture at $\mathbf{x}_c$. For the experiments, $\mathbf{x}_c$ was chosen to correspond to a reaching posture with both arms outstretched and $\mathbf{W} = 0.05\mathbf{I}$.

During data collection, the policy was constrained by the presence of a wall placed directly in front of the robot at random orientations and distances. The wall restricted the movement of the hands, and this constraint was projected back into the joint space, where the policy was operating. This causes the policy to appear highly complex and non-linear in the state space (joint space), with discontinuous changes to the dimensionality of the constraints as the arms of the robot approached the wall.

Using the formalism from Sec. III-A with $\mathbf{b}(x) = \bar{\mathbf{x}}$, we fitted linear models to 100 data sets, each consisting of 100 trajectories of 100 data points. Despite the high dimensionality, our method reached a normalised UPE of $0.291 \pm 0.313 \times 10^{-2}$. It is important to point out that this result can not only be explained by our choice of a linear model where we knew that the true policy (13) was also linear: Direct (naive) linear regression on the observed commands resulted in a normalised UPE of $63.9 \pm 3.1 \times 10^{-2}$, which again is orders of magnitude higher similar to our results on toy data.

V. Conclusion

In this work, we introduced a novel approach to direct policy learning in cases where demonstrated movements are subject to variable, dynamic, non-linear constraints. Due to a small but very effective modification in the calculation of an empirical risk, our method can recover 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. We demonstrated our method using parametric and locally linear function approximators to learn policies for problems of varying size and complexity.

In future work we aim to apply our approach to seed robot skill acquisition from variable-constraint human motion capture data. For our initial experiments, data collected using the Vicon motion capture system[7] will be used to investigate the constrained ball grasping task. Example trajectories are shown in Fig. 7. Please note the striking resemblance to those generated by the artificial policy (cf. Fig. 6).

[7]http://www.vicon.com

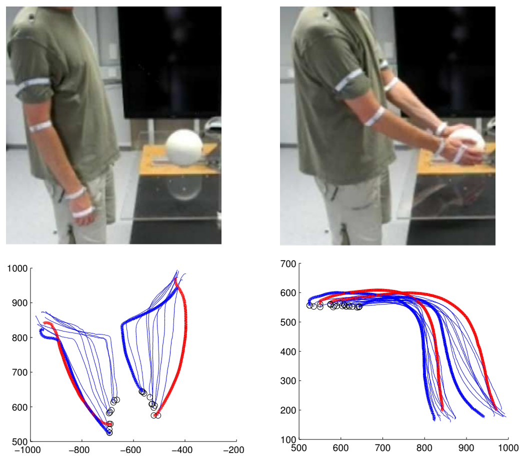

Fig. 7. Human motion capture experiments. Top: Example ball grasping movement. Bottom: Example trajectories of the two hands when viewed from above (left) and the side (right). Constrained trajectories are shown in blue, with an example unconstrained trajectory shown in red.

A further area of future work will focus on tightening the existing bounds on error for the UPE and CPE (ref. Sec. III). This should provide an increased level of accuracy and robustness for example in scenarios where different regions are either unconstrained or contain consistent or highly correlated constraints.

VI. Acknowledgements

The authors gratefully acknowledge the contribution of C. Towell for collecting human motion data and J. Steil and others at the CoR-lab at Bielefeld University for providing access to the Vicon system.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot Programming by Demonstration. In *Handbook of Robotics*. MIT Press, 2007.
[2] S. Calinon, F. Guenter, and A. Billard. Goal-directed imitation in a humanoid robot. In *ICRA*, 2005.
[3] M. Gienger, H. Janssen, and C. Goerick. Task-oriented whole body motion for humanoid robots. In *Humanoids*, 2005.
[4] D. Grimes, D. Rashid, and R. Rao. Learning nonparametric models for probabilistic imitation. In *NIPS*, 2007.
[5] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *NIPS*, 2003.
[6] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Robotics and Automation*, RA-3(1):43–53, 1987.
[7] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. In *IEEE Trans. Sys., Man, and Cybernetics*, 1977.
[8] K. Ohta, M. Svinin, Z. Luo, S. Hosoe, and R. Laboissiere. Optimal trajectory formation of constrained human arm reaching movements. *Biol. Cybern.*, 91:23–36, 2004.
[9] J. Park and O. Khatib. Contact consistent control framework for humanoid robots. In *ICRA*, 2006.
[10] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant DOFs. *Autonomous Robots J.*, 24:1–12, 2008.
[11] J. Peters and S. Schaal. Learning to control in operational space. *Int. J. Robotics Research*, 27:197–212, 2008.
[12] S. Schaal and C. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084, 1998.
[13] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Phil. Trans.: Biological Sciences*, 358:537–547, 2003.
[14] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *IROS*, 2007.