Learning Potential-based Policies from Constrained Motion

Matthew Howard[1], Stefan Klanke[1], Michael Gienger[2], Christian Goerick[2] and Sethu Vijayakumar[1]

[1]*Institute for Perception, Action and Behaviour,*
University of Edinburgh, Scotland, UK.
{*matthew.howard, s.klanke, sethu.vijayakumar*}@*ed.ac.uk*

[2]*Honda Research Institute Europe GmbH,*
Offenbach/Main D-63073, Germany.
{*michael.gienger, christian.goerick*}@*honda−ri.de*

Abstract—**We present a method for learning potential-based policies from constrained motion data. In contrast to previous approaches to direct policy learning, our method can combine observations from a variety of contexts where different constraints are in force, to learn the underlying unconstrained policy in form of its potential function. This allows us to generalise and predict behaviour where novel constraints apply. As a key ingredient, we first create multiple simple local models of the potential, and align those using an efficient algorithm. We can then detect and discard unsuitable subsets of the data and learn a global model from a cleanly pre-processed training set. We demonstrate our approach on systems of varying complexity, including kinematic data from the ASIMO humanoid robot with 22 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. 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. Alternatively, in problems that involve control of contacts such as manipulating objects, the motion of fingers is constrained by the presence of the object [1]. In systems designed to be highly competent and adaptive, such as humanoid robots (Fig. 1), behaviour may be subject to a wide variety of constraints [2], usually non-linear in actuator space and often discontinuous. Consider running on uneven terrain: The leg movements of the runner are constrained by the impact of the feet on the ground in a dynamic, discontinuous and unpredictable way.

The focus in this paper is on modelling control policies subject to a certain class of constraints on motion, with the aim of finding policies that can *generalise between different constraints*. We take a direct policy learning approach (DPL) [3] whereby we attempt to learn non-parametric models of the policy from motion data (e.g. from human demonstrations). While DPL has been studied for a variety of control problems in recent years (for a review, see [4] and references therein), crucially these problems involved *unconstrained* policies or policies subject to *identical constraints in every observation* (in which case the constraints can be absorbed into the policy itself). The difference here is that we consider observations from policies subject to a set of dynamic, non-linear constraints, and that these constraints may change between observations, or even during the course of an observation. In general, learning (unconstrained) policies from constrained motion data is a formidable task. This is due to (i) *non-convexity* of observations induced by the constraints, and; (ii) *degeneracy* in the set of possible policies that could have
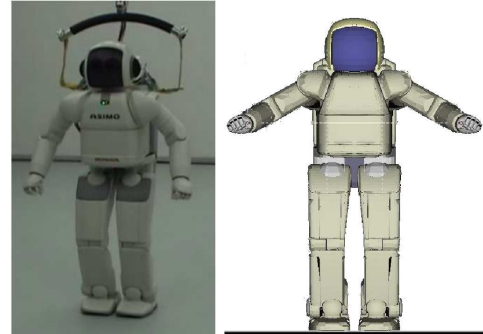

Fig. 1. ASIMO humanoid robot (left) and VRML rendering of a kinematically and dynamically accurate simulation model (right).

produced the movement under the constraint [5]. However we will show that despite these theoretical limits it is still possible to find a good approximation of the underlying policy given observations under the right conditions. We take advantage of recent work in local dimensionality reduction [6] to propose a method that (i) given observations under a sufficiently rich set of constraints reconstructs the fully unconstrained policy; (ii) given observations under an impoverished set of constraints learns a policy that generalises well to constraints of a similar class, and; (iii) given 'pathological' constraints will learn a policy that at worst reproduces behaviour subject to the same constraints. Our algorithm is fast, robust and scales to complex high-dimensional movement systems. Furthermore it can deal with constraints that are both *non-linear* and *discontinuous* in time and space.

II. PROBLEM FORMULATION

Following [3], we consider the learning of autonomous kinematic policies

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

where $\mathbf{x} \in \mathbb{R}^n$ is some appropriately[1] chosen state-space and $\dot{\mathbf{x}} \in \mathbb{R}^n$ is the desired change in state. The goal of DPL is to approximate the policy (1) as closely as possible [3]. It is usually formulated as a supervised learning problem where it is assumed that we have observations of $\dot{\mathbf{x}}(t)$, $\mathbf{x}(t)$ (often in

---

[1]It should be noted that in all DPL approaches the choice of state-space is problem specific [3] 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, to correspond to the end-effector coordinates of the robot. Transfer of behaviour across non-isomorphic state spaces, for example if demonstrator and imitator have different embodiments, is also possible by defining appropriate state-action metrics [7].
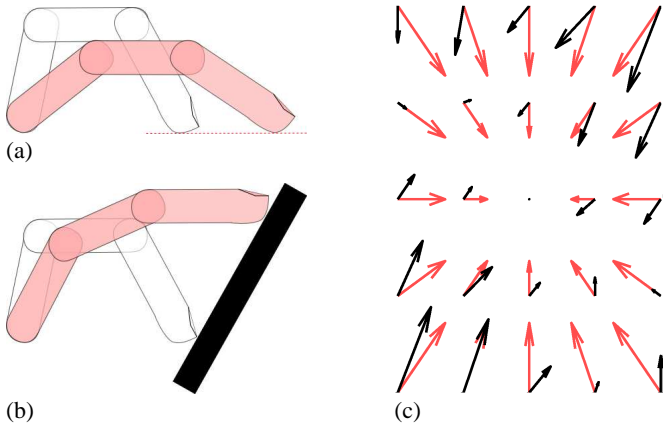
Fig. 2. Illustration of two apparently different behaviours from the same policy: (a) unconstrained movement (b) movement constrained such that the fingertip maintains contact with a surface (black box) (c) the unconstrained (red) and constrained (black) policy over two joints of the finger.

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 [8], non-parametric modelling [9], and probabilistic Bayesian approaches [10].

Consider now learning a simple policy to extend a jointed finger under variable constraints, such as when different obstacles lie in the path of the 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, constraining it so that it moves along the surface. The vector field representation of the two behaviours is shown in Fig. 2(c).

Given this learning task, the standard approach to DPL would be to perform regression on the vector field produced in the two settings [3], [8]. This would mean that, assuming the observations were labelled with respect to the constraint (position and orientation of the obstacle), one could learn a (separate) policy model for each of the settings. Clearly this is unsatisfactory since each model would only be valid for the specific setting, and we would need increasing numbers of models given observations under new constraints (e.g. different obstacles at different positions and orientations).

However, on closer inspection, we can avoid the need for multiple policy models, by making two observations. Firstly, we notice that some features of the policy are consistent across observed trajectories (here, the goal of the movements – 'extend the finger' – appear similar). Secondly, one can see that in different trajectories the movement is *restricted* in different ways (here, contact with the obstacle prevents the finger from moving in certain directions). Based on these observations then, we might reasonably suppose that the movement stems from some *single underlying* policy and that this policy has been sampled under *different constraints*. Viewed like this, instead of learning separate policies for each specific constraint, we would rather learn a policy that *generalises* over constraints.

A. Constraint Model

In this paper, we explore the problem of DPL when the policy under observation is subject to hard constraints on motion. Mathematically, we say given a set of k-dimensional constraints (where $k < n$)

$$\mathbf{A}(\mathbf{x},t)\dot{\mathbf{x}} = \mathbf{0} \qquad (2)$$

the policy is projected into the nullspace of those constraints

$$\dot{\mathbf{x}}(t) = \mathbf{N}(\mathbf{x},t)\boldsymbol{\pi}(\mathbf{x}(t)) \qquad (3)$$

where $\mathbf{A}(\mathbf{x},t) \in \mathbb{R}^{k \times n}$ is some (rank-k) matrix describing the constraint, $\mathbf{N}(\mathbf{x},t) \equiv (\mathbf{I} - \mathbf{A}^\dagger \mathbf{A}) \in \mathbb{R}^{n \times n}$ is in general a non-linear, time-varying projection operator and $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix[2]. Constraints of the form (2) commonly occur in interactions with solid objects, e.g. when manipulating tools [1], [11] and are also common in the control of redundant degrees of freedom (DOFs) in high-dimensional manipulators [12], [13]. As an example: Setting $\mathbf{A}$ to the Jacobian that maps from joint-space to end-effector position allows any motion in joint-space provided that the end-effector remains stationary.

If the policy is constrained (2)-(3), the best policy representation of the movements is the unconstrained policy $\boldsymbol{\pi}$, since this is gives *maximal information* about the behaviour. Knowing $\boldsymbol{\pi}$, or finding a good approximation of it, we can generalise over constraints (cf. Fig. 2(a)-(b)) simply by applying the desired constraint. However, learning the unconstrained policy from observations of constrained movement is a non-trivial task due to two analytical restrictions on what information can be recovered from the available data: The problems of *non-convexity* and *degeneracy* [5].

The *non-convexity* problem comes from the fact that between observations, or even during an observation, constraints may change. For example in Fig. 2(c) any given observation $\dot{\mathbf{x}}$ may come from the set of constrained (black) or unconstrained (red) vectors. At any given point in $\mathbf{x}$ there may be multiple observations $\dot{\mathbf{x}}$ under the different constraints. This causes problems for supervised learning algorithms, for example directly training on these observations may result in model-averaging.

The *degeneracy* problem stems from the fact that for any given constrained observation, there exist multiple policies that could have produced the 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 they are undetermined by the observation. In effect we are not given sufficient information about the unconstrained policy to guarantee that it is fully reconstructed.

However, despite these restrictions, we wish to do the best we can with the data available. In this paper we propose a method to deal with these problems, for the important special class of *potential-based* policies.

B. Potential-based Policies

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

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

Such policies can be thought of as greedily optimising the potential function at every time step [14] and thus encode attractor landscapes where the minima of the potential correspond to stable attractor points; in the finger example, the

---

[2]Throughout the paper $\mathbf{I}$ denotes the identity matrix of appropriate dimension and $\mathbf{A}^\dagger$ denotes the (unweighted) Moore-Penrose pseudoinverse of the matrix $\mathbf{A}$.

$\mathbf{x} = \mathbf{0}$ point would correspond to such a minimum. Other examples include reaching movements which may be represented by a potential, defined in hand space, with a minimum at the target. Furthermore decision-based behaviours may be encoded as potentials with multiple minima. For example the decision of a which hand to use for reaching may be represented by a potential with two minima, one corresponding to reaching with the right hand, the other to reaching with the left. The hand used would then be determined by relative offset of the minima (e.g. right-handedness would imply a lower minimum for that hand). Potential-based policies are also extensively used for null-space control of redundant manipulators [14].

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* [5] rather than modelling it directly. The advantage of this is twofold. Firstly, under the projection operator $\mathbf{N}(\mathbf{x}, t)$ the potential-based policy (4) can be locally estimated using numerical line integration [5]. Secondly, the potential function is a scalar function and thus gives a compact representation of the policy. This means that the non-convexity problem of reconciling conflicting n-dimensional vector observations is reduced to finding a function $\hat{\phi}(\mathbf{x})$ where the (1-dimensional) prediction is consistent at any given point $\mathbf{x}$.

III. LEARNING NULLSPACE POLICIES THROUGH LOCAL MODEL ALIGNMENT

A. Estimating the potential along single trajectories

As has been described in [5], it is possible to model the potential along sampled trajectories using a form of line integration. 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. This results in a tuple of points $\mathbf{X}_k = \mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,T\delta t}$ for each trajectory, which, for sufficiently high sampling rate, are related through the linear approximation

$$\mathbf{x}_{i+1} \approx \mathbf{x}_i + \delta t \, \mathbf{N}_i \boldsymbol{\pi}_i. \tag{5}$$

Using (5) we can integrate along trajectories using an appropriate numerical integration scheme. An example of such a scheme is Euler integration, which involves the first order approximation

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) + \frac{1}{\delta t}(\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathbf{N}_i \nabla_{\mathbf{x}}\phi(\mathbf{x}_i). \tag{6}$$

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 the correct time correspondence is kept. For steps $\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}$ that follow the projected policy (3) we can rearrange (5) with the scaled time coordinates, and substitute into (6) to yield

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2, \tag{7}$$

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 turn to this problem in the next section.

B. Constructing the global potential function

Let us assume we have K trajectories $\mathbf{X}_k = (\mathbf{x}_{k1}, \mathbf{x}_{k2} \ldots \mathbf{x}_{kN_k})$ and corresponding point-wise estimates of the potential $\hat{\boldsymbol{\Phi}}_k = (\hat{\phi}_{k1}, \hat{\phi}_{k2} \ldots \hat{\phi}_{kN_k})$, 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 $f_k(\mathbf{x}_i) \approx \hat{\phi}_{ki}$. Here, to keep things simple, we use nearest-neighbour (NN) regression, i.e.,

$$f_k(\mathbf{x}) = \Phi_{ki^*} \quad, \quad i^* = \arg\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2. \tag{8}$$

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 caclulate responsibilities $q_k(\mathbf{x})$ using Gaussian kernels, i.e.,

$$q_k(\mathbf{x}) = \frac{w_k(\mathbf{x})}{\sum_{i=1}^{K} w_i(\mathbf{x})}, \tag{9}$$

$$w_k(\mathbf{x}) = \exp\left[-\frac{1}{2\sigma^2}\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2\right]. \tag{10}$$

leading to 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. have shown how to align multiple local PCA models into a common low-dimensional space [6]. 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. The authors [6] retrieved the parameters of the optimal mappings $\mathbf{g}_k$ by minimising the objective function

$$E = \frac{1}{2}\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{j=1}^{K} q_{km}q_{jm}\|\mathbf{g}_{km} - \mathbf{g}_{jm}\|^2, \tag{11}$$

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.

In analogy to the PCA-alignment method [6], we augment our local potential models $f_k(\cdot)$ by a scalar offset b_k and define

the corresponding objective function as

$$E(b_1 \ldots b_K) = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{j=1}^{K} q_k(\mathbf{x}_m) q_j(\mathbf{x}_m) \times$$
$$\left((f_k(\mathbf{x}_m) + b_k) - (f_j(\mathbf{x}_m) + b_j) \right)^2, \quad (12)$$

or, in a slightly shorter form,

$$E(\mathbf{b}) = \frac{1}{2} \sum_{m,k,j} q_{km} q_{jm} \left(f_{km} + b_k - f_{jm} - b_j \right)^2. \quad (13)$$

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

$$E(\mathbf{b}) = E_0 + 2\mathbf{a}^T \mathbf{b} + \mathbf{b}^T \mathbf{H} \mathbf{b}. \quad (14)$$

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 (12), which corresponds to the shift-invariance of the global potential. Therefore, the vector $(1, 1, \ldots, 1)^T$ spans the nullspace of $\mathbf{H}$, and we need to use the pseudo-inverse of $\mathbf{H}$ to calculate the optimal offset vector

$$\mathbf{b}_{opt} = -\mathbf{H}^\dagger \mathbf{a}. \quad (15)$$

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.

C. Over-smoothing and Outlier Detection

Since we restrict ourselves to using simple NN regression for the local potential models in this paper, the only open parameter of our algorithm is σ^2, i.e., the kernel parameter used for calculating the responsibilities (9). Too large a 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 for even far apart points $\mathbf{x}$ in state space.

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

The same reasoning applies to groups of trajectories that are close to each other, but have little connection to the rest of the set. For the remainder of the paper, we will refer to such trajectories as 'outliers', since like in classical statistics we need to remove these from the training set: If their influence on the overall alignment is negligible, their own alignment can

be poor, and this becomes a problem when using the output of the optimisation (15) to learn a global model of the potential. To avoid interference, we only include trajectories if we are sure that their offset is consistent with the rest of the data.

Fortunately, outliers in this sense can be detected automatically by looking for small eigenvalues of $\mathbf{H}$: In the same way as adding the same offset to all trajectories leads to a zero eigenvalue, further very small eigenvalues and the corresponding eigenvectors indicate indifference towards a shift of some subset of trajectories versus the rest of the set. In practice, we look for eigenvalues $\lambda < 10^{-8}$, and use a recursive bi-partitioning[3] algorithm in a way that is very similar to spectral clustering. We then discard all trajectories apart from those in the largest 'connected' group.

D. 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. Here, we choose Locally Weighted Projection Regression (LWPR) [15] because it performs well in cases where the data lies on low-dimensional manifolds in a high-dimensional space, which matches our problem of learning the potential from a set of trajectories. As the training data for LWPR, we use all non-outlier trajectories and their estimated potentials as given by the Euler integration *plus* their optimal offset, that is, the input-output tuples

$$\left\{ (\mathbf{x}_{kn}, \hat{\phi}_{kn} + b_k^{opt}) \mid k \in \mathcal{K}, n \in \{1 \ldots N_k\} \right\}, \quad (16)$$

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.

Algorithm 1 PolicyAlign

1: Estimate $\mathbf{X}_k, \hat{\boldsymbol{\Phi}}_k, \{k = 1 \ldots K\}$ using Euler integration.
2: Alignment:
- Calculate prediction and responsibility of each local model f_k on each data point $\mathbf{x}_m$, $m = 1 \ldots M$:
  $f_{km} = f_k(\mathbf{x}_m)$; $q_{km} = w_k(\mathbf{x}_m) / \sum_i w_i(\mathbf{x}_m)$
- 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 $(\mathbf{x}_{kn}, \hat{\phi}_{kn} + b_k^{opt})$

IV. EXPERIMENTS

To explore the performance of our algorithm, we performed experiments on data from autonomous kinematic control policies [3] applied[4] to different plants, including the whole body motion controller (WBM) of the humanoid robot ASIMO [2]. In this section, we first discuss results from an artificial toy

---

[3] Partitioning the set into separate groups can be stopped as soon as there is only one zero-eigenvalue left.

[4] Since the goal of the experiments was to validate the proposed approach, we used policies known in closed form as a ground truth. In the follow-up paper we apply our method to human motion capture data.

problem controlled according to the same generic framework to illustrate the key concepts. We then discuss an example scenario in which the algorithm is used to enable ASIMO to learn a realistic bi-manual grasping task from observations of a constrained demonstrator. Finally we briefly discuss how our algorithm scales to policies in very high dimensional systems such as the 22 DOF of the ASIMO WBM controller [2].

A. Selection of smoothing parameter

For simplicity, in all our experiments we used the same heuristics for selecting the smoothing parameter σ^2 to match the scale of typical distances in the data sets. In particular, we first calculated the distances between any two trajectories $k, j \in \{1 \ldots K\}$ as the distances between their closest points $d_{kj} = \min\{\|\mathbf{x}_{kn} - \mathbf{x}_{jm}\|^2 \mid n, m \in \{1 \ldots N\}\}$, and also the distances to the closest trajectory $d_k^{min} = \min\{d_{kj} \mid j \neq k\}$. We then consider three choices for σ^2, which we refer to as 'narrow', 'wide' and 'medium':

$$\sigma^2_{nar} = \text{median}\left\{d_k^{min} \mid k \in \{1 \ldots K\}\right\} \quad (17)$$

$$\sigma^2_{wid} = \text{median}\left\{d_{jk} \mid j, k \in \{1 \ldots K\}, j \neq k\right\} \quad (18)$$

$$\sigma^2_{med} = \sqrt{\sigma^2_{nar}\sigma^2_{wid}}. \quad (19)$$

B. Toy Example

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

$$\phi(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_c)^T \mathbf{W} (\mathbf{x} - \mathbf{x}_c) \quad (20)$$

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

$$\mathbf{A}(\mathbf{x}, t) = (\alpha_1, \alpha_2) \equiv \boldsymbol{\alpha} \quad (21)$$

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. To increase the complexity of the problem, the constraints were 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 as can be seen by sharp turns in the trajectories. Figure 3 shows an example of our algorithm at work for a set of $K = 40$ trajectories of length $N = 40$ for the toy system. The raw data as a set of trajectories through the two-dimensional state space is shown in panel (a), whereas panel (b) additionally depicts the local potential models as estimated from the Euler integration prior to alignment. Each local model has an arbitrary offset against the true potential so there are inconsistencies between the predictions from each local model. Figure 3(c) shows the trajectories after alignment, already revealing the structure of the parabola.

At this point, the outlier detection scheme has identified three trajectories as being weakly connected to the remaining set. In Fig. 3(a) we can see that the outliers are indeed the only trajectories that do not have any intersection with neighbouring trajectories. At the 'narrow' length scale determined by the smoothing parameter (17), they are hard to align properly,
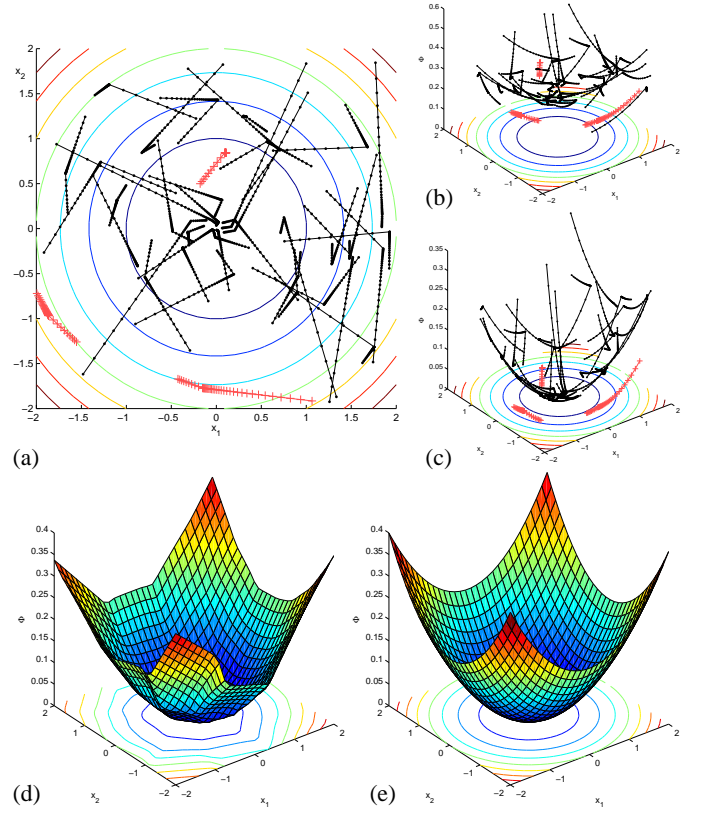

Fig. 3. Top: (a) Toy data (trajectories (2-D) and contour of true potential. Estimated potential along the trajectories before (b) and after (c) alignment. Trajectories detected as difficult to align 'outliers' are shown by light crosses. Bottom: Learnt (d) and true (e) potential function after training on the aligned trajectories.

and need to be discarded before learning the global model. Finally, Fig. 3(d) shows the global model $f(\mathbf{x})$ of the potential that was trained on the aligned trajectories, which is clearly a good approximation of the true parabolic potential shown in Fig. 3(e). For a more thorough evaluation, we repeated this experiment on 100 data sets and evaluated

- the nMSE of the aligned potential, which measures the difference between $\hat{\phi}_{kn} + b_k$ and the true potential ϕ,
- the nMSE of the learnt potential, measuring the difference between $f(\cdot)$ and $\phi(\cdot)$,
- the normalised unconstrained policy error (nUPE), quantifying the difference between $\hat{\boldsymbol{\pi}} = \nabla f$ and $\boldsymbol{\pi} = \nabla \phi$,
- the normalised constrained policy error (nCPE), which is the discrepancy between $\mathbf{N}\hat{\boldsymbol{\pi}}$ and $\mathbf{N}\boldsymbol{\pi}$, and finally
- the percentage of trajectories discarded as outliers

on a subsample of the data held out for testing. We did so for our three different choices of σ^2 given in (17-19). We also repeated the experiment using a sinusoidal potential function

$$\phi_s(\mathbf{x}) = 0.1 \sin(x_1) \cos(x_2) \quad (22)$$

with the same amount of data, and then using $K = 100$ trajectories of length $N = 100$ for each data set.

Table I summarises the results. Firstly, we can see that the 'wide' choice for σ^2 leads to large error values which are due to over-smoothing. Using the narrow σ^2, we retrieve very small errors at the cost of discarding quite a lot of trajectories[5],

---

[5]Please note that we discard the outliers both for training and evaluating the error statistics.

Setup	σ^2	Alignment nMSE	Potential nMSE	nUPE	nCPE	Discarded (%)
Parabola	narrow	0.0047 ± 0.0026	0.0052 ± 0.0024	0.0486 ± 0.0211	0.0235 ± 0.0092	17.55 ± 15.96
$K = 40$	medium	0.0204 ± 0.0211	0.0195 ± 0.0203	0.0859 ± 0.0486	0.0224 ± 0.0074	0.48 ± 1.11
$N = 40$	wide	0.3542 ± 0.1089	0.3143 ± 0.1045	0.5758 ± 0.2726	0.1135 ± 0.0371	0 ± 0
Sinusoidal	narrow	0.0017 ± 0.0022	0.0026 ± 0.0019	0.1275 ± 0.1125	0.0535 ± 0.0353	50.18 ± 14.37
$K = 40$	medium	0.0534 ± 0.0647	0.0522 ± 0.0645	0.1399 ± 0.0422	0.0376 ± 0.0097	1.03 ± 3.99
$N = 40$	wide	0.6259 ± 0.1330	0.5670 ± 0.1363	0.8373 ± 0.2188	0.2464 ± 0.0638	0 ± 0
Sinusoidal	narrow	0.0005 ± 0.0002	0.0014 ± 0.0004	0.0657 ± 0.0142	0.0308 ± 0.0065	25.46 ± 11.42
$K = 100$	medium	0.0011 ± 0.0017	0.0019 ± 0.0017	0.0628 ± 0.0089	0.0284 ± 0.0044	1.25 ± 3.33
$N = 100$	wide	0.2892 ± 0.1198	0.2137 ± 0.1000	0.4262 ± 0.1367	0.1554 ± 0.0483	0 ± 0

TABLE I

ERROR AND OUTLIER STATISTICS (MEAN±STD.DEV. OVER 100 DATA SETS) FOR THE EXPERIMENT ON 2-D TOY DATA.

and the medium choice seems to strike a reasonable balance especially with respect to the nUPE and nCPE statistics.

Secondly, when comparing the results for the parabolic and sinusoidal potentials, we can see that the latter, more complex potential (with multiple sinks) requires much more data. With only 40 trajectories and 40 points each, most of the data sets are too disrupted to learn a reasonable potential model. While at the narrow length scale (4th row), on average more than half of the data set is discarded, even the medium length scale (5th row) over-smooths the subtleties of the underlying potential.

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

C. Grasping a Ball

The two goals of our second set of experiments were (i) to characterise how well the algorithm scaled to more complex, realistic constraints and policies and (ii) to assess how well the learnt policies generalised over different constraints. For this we set up a demo scenario in which a set of trajectories demonstrating the task of reaching for a ball on a table were given. Furthermore, it was assumed that trajectories were recorded in contexts where different constraints applied. The goal was to uncover a policy that both accurately reproduced the demonstrated behaviour and generalised to novel contexts with unseen constraints.

For this, we set up an 'expert' demonstrator from which observations were recorded. For ease of comparison with the 2-D system, the expert's policy was defined by the same quadratic potential (20) this time with the target point $\mathbf{x}_c$ corresponding to a grasping position, with the two hands positioned on either side of the ball. The state-space of the policy was defined as the Cartesian position of the two hands, corresponding to 6 DOFs in state and action space (hereafter, the 'task space'). The task space policy motion was realised using the ASIMO WBM controller (see [2] for details).

The policy was constrained by placing barriers on the table between the robot and the ball, so that the robot had to reach through a gap in the barriers to get the ball. These acted as constraints on the hands restricting motion in the direction normal to the barrier surface if a hand came too close (cf. [16]). The constraints are nonlinear in the state space and have discontinuously switching dimensionality when the hands approach or recede from the barriers. The constraints were varied by randomly changing the width of the gap for each trajectory. The gap widths were sampled from a 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. 4 shows the experimental set-up.
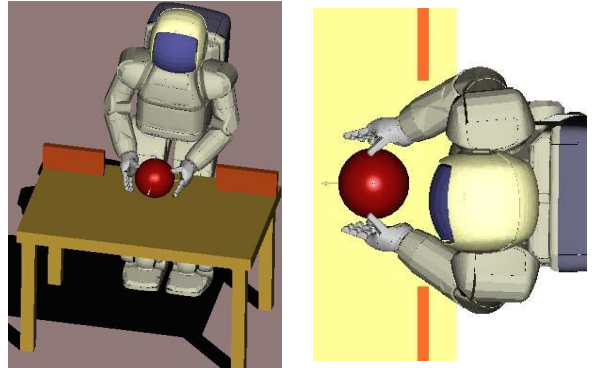

Fig. 4. Experimental set-up for the ball grasping experiment. Starting with hands at the sides, the teacher robot reaches between the barriers to grasp the ball.

Data was collected by recording $K = 100$ trajectories of length $N = 100$. 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 calculating the hand positions using forward kinematics. The joint vector $\mathbf{q}$ was clipped where necessary to avoid joint limits and self collisions.

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

For this task, our algorithm achieved a very low alignment error of $6.95 \pm 0.09 \times 10^{-4}$ and an nMSE in the learnt potential of $7.85 \pm 0.56 \times 10^{-4}$ with $0.48 \pm 0.84\%$ trajectories discarded (mean±s.d. over 50 data sets). In Table II we give the errors in predicting the policy subject to (i) the training data constraints, (ii) no constraints, and (iii) a novel, unseen constraint. For the latter, a barrier was placed centrally between the robot and the ball, so the robot had to reach around the barrier to grasp it.

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

The key result, however, can be seen by examining the policy errors (ref. Table II). Comparing the two approaches, both achieve a similar nCPE, with the naive approach in fact
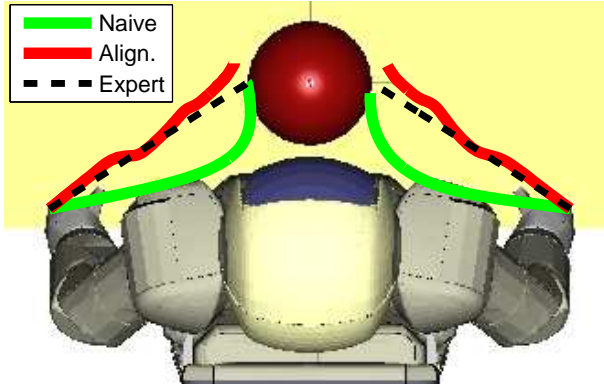
Fig. 5. Unconstrained reaching movement of the expert policy (black) and policies learnt with the naive approach (green) alignment algorithm (red).

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

TABLE II
CONSTRAINED POLICY nMSE FOR UNSEEN CONSTRAINTS ON THE
BALL-GRASPING TASK. VALUES ARE MEAN±S.D. OVER 50 DATA SETS.

performing slightly better. This indicates that the two methods both do equally well in modelling the constrained movements to approximately the same level of accuracy. However, when comparing the errors for the unconstrained policy, and the policy subject to the unseen constraint, a different picture emerges. Using the model learnt by the alignment approach, the unconstrained policy predictions, and the predictions under the unseen constraint, maintain a similar level of error to that of the constrained policy. In stark contrast to this, the naive approach fares very poorly, with a large jump in error when predicting the policy under the new barrier constraint and predicting the unconstrained behaviour.

This difference is highlighted if we compare trajectories generated by the two policies. In Fig. 5 example unconstrained reaching trajectories produced by the expert (black), and the policies learnt by (i) the naive approach (green), and (ii) the alignment approach (red) are shown. In the former the hands take a curved path to the ball, reproducing the average behaviour of the demonstrated (constrained) trajectories – the naive method is unable to extract the underlying task (policy) from the observed paths around the obstacles. Consequently, it cannot generalise and find its way around the unseen barrier. In contrast, the policy learnt with the alignment approach better predicts the unconstrained policy, enabling it to take a direct route to the ball that closely matches that of the expert.

D. Learning from high-dimensional ASIMO data

In our final set of experiments we tested the scalability of our approach for learning in very high dimensions. For this we again used the quadratic potential (20) where now the state vector $\mathbf{x}$ corresponded to the 22-dimensional joint configuration of the upper body of the ASIMO robot. In this experiment, the policy was constrained such that hands of the robot were restricted to lie on a plane of random orientation. Such constraints occur in a variety of surface contact behaviours, for example when wiping windows [1].

We ran the experiment on 50 data sets of $K = 100$ trajectories of length $N = 100$. Using the narrow setting of

the smoothing parameter the algorithm achieved an alignment error of $1.6 \pm 0.3 \times 10^{-3}$, an nMSE in the learnt potential of $1.5 \pm 0.4 \times 10^{-3}$, nCPE of 0.0654 ± 0.0140 and nUPE of 0.1568 ± 0.0474, with just $0.02 \pm 0.14\%$ of the trajectories discarded. We consider this to be remarkably good performance given the high dimensionality of the input space and the relatively small size of the data set.

V. CONCLUSION

We have proposed a novel approach to direct learning of potential-based policies from constrained motion data. Our method is fast and data-efficient, and it scales to complex constraints in high-dimensional movement systems. The core ingredient is an algorithm for aligning local models of the potential, which leads to a convex optimisation problem.

Under the analytical limitations of what can be learnt in this setting, our method performs remarkably well. Given an impoverished set of motion observations from a pathological set of constraints, one can never hope to recover the fully unconstrained policy. However, using our method, motion data under different constraints can be combined to learn a potential that is consistent with the observations. With a reasonably rich set of constraints, we can recover the policy with high accuracy, and we can generalise to predict behaviour under different constraints.

Work is currently ongoing to transfer our results to the ASIMO hardware and also to apply our method to learning from human motion capture data.

REFERENCES

[1] J. Park and O. Khatib, "Contact consistent control framework for humanoid robots," in *IEEE Int. Conf. Robotics and Automation*, 2006.
[2] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *IEEE Int. Conf. on Humanoid Robots*, 2005.
[3] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Phil. Trans.: Biological Sciences*, vol. 358, pp. 537–547, 2003.
[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration," in *Handbook of Robotics*. MIT Press, 2007.
[5] M. Howard and S. Vijayakumar, "Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators," in *W.S. Robotics and Mathematics*, 2007.
[6] J. Verbeek, S. Roweis, and N. Vlassis, "Non-linear CCA and PCA by alignment of local models," in *Adv. Neural Information Processing Systems*, 2004.
[7] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Correspondence mapping induced state and action metrics for robotic imitation," *IEEE Trans. Sys., Man, and Cybernetics*, vol. 37, no. 2, pp. 299–307, 2007.
[8] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Adv. Neural Information Processing Systems*, 2003.
[9] J. Peters and S. Schaal, "Learning to control in operational space," *Int. J. Robotics Research*, vol. 27, pp. 197–212, 2008.
[10] D. Grimes, D. Rashid, and R. Rao, "Learning nonparametric models for probabilistic imitation," in *Adv. Neural Information Processing Systems*, 2007.
[11] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.
[12] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," in *IEEE Trans. Sys., Man, and Cybernetics*, 1977.
[13] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal, "A unifying framework for robot control with redundant DOFs," *Autonomous Robots J.*, vol. 24, pp. 1–12, 2008.
[14] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison Wesley, 1991.
[15] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Comp.*, vol. 17, pp. 2602–2634, 2005.
[16] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time collision avoidance with whole body motion control for humanoid robots," in *IROS*, 2007.