Learning Utility Surfaces for Movement Selection

Matthew Howard*, Michael Gienger†, Christian Goerick† and Sethu Vijayakumar*
*School of Informatics, University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom.
Email: {matthew.howard, sethu.vijayakumar}@ed.ac.uk
†Honda Research Institute Europe GmbH, Offenbach/Main D-63073, Germany.
E-mail: {michael.gienger, christian.goerick}@honda-ri.de

***Abstract*— Humanoid robots are highly redundant systems with respect to the tasks they are asked to perform. This redundancy manifests itself in the number of degrees of freedom of the robot exceeding the dimensionality of the task. Traditionally this redundancy has been utilised through optimal control in the null-space. Some cost function is defined that encodes secondary movement goals and movements are optimised with respect to this function, subject to fulfilment of task constraints. Until now design of cost functions has been carried out on an ad-hoc basis and has required time-consuming hand-tuning to ensure that the desired (or acceptable) behaviour is realised. Here we present a novel approach for designing cost functions for optimal control in the null-space by exploiting recent advances in statistical machine learning. The behaviour of a (kinematically or dynamically controlled) mechanical system performing some task is observed and separated into task- and null-space components. The null-space component is then modelled as a first order differential equation with the cost as the independent variable. Numerical solution of this equation provides training data for a statistical learning algorithm that is used to build an open-form model of the cost function. Results are presented in which the reconstructed function is used to replace that of the original control scheme and the resultant behaviour, for the same set of tasks, is compared.**

***Index Terms*— Redundancy, Null-space control, Dynamic and Kinematic control, Learning.**

I. Introduction

Humanoid robots such as Honda's ASIMO robot (Fig. 1) are by nature highly redundant systems both at the kinematic control level and the task execution/planning level. This redundancy manifests itself in the high numbers of redundant degrees of freedom (DOFs), large space of possible configurations, large range of motion capabilities and availability of multiple effectors.

In order to resolve this redundancy and make choices about the planning or execution of movements, an intuitive approach is to specify some cost or utility function as a metric by which to measure a movement's optimality. Cost functions can be defined over the state-space of the problem (or any subspace thereof) and provided they contain some unique global optimum, they can be used to specify a unique preferred strategy for movement. Additionally if motor learning is considered as the process of optimisation of movements [1], then cost functions can be considered the goals of the learning process.

A. Cost Functions for Motor Control

Research into cost functions in motor control seems to focus around two main threads; the design of cost functions for robotic control and the search for cost functions that provide a theory of human movement selection.

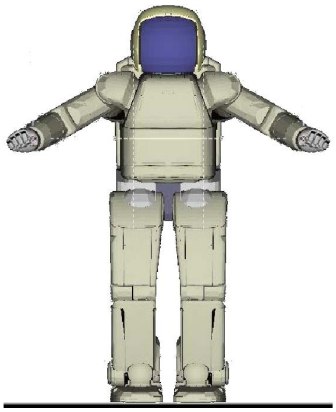

Fig. 1. Simulation of the ASIMO humanoid robot.

In the engineering domain, the cost functional approach has been successfully applied to problems in redundancy resolution in robotic manipulators [2]. Several optimisation criteria have been designed for movement planning (see [3] for a review) with notable ones including joint-limit avoidance [4], singularity avoidance [5] and obstacle avoidance [6], [7] criteria. In practice such criteria are frequently used as 'heuristics' for redundancy resolution – offering secondary constraints on actuation assuming the desired task-space movement is achieved – by mapping their effects onto the null-space of the movement [8].

In biological research, the idea of optimisation of some cost function has been popular in the attempt to understand the stereotypical features of human motion (see [9] for a review). The goal here is to find some underlying principle that explains how movement selection takes place in animals or humans. Many possible cost functions have been suggested such as the well-known minimum jerk [10], minimum torque-change [11] and minimum end-point variance [12] criteria and have been successfully used to predict invariant features of movements such as hand trajectories and velocity profiles or the two-thirds power law [13], [14].

However, the approaches taken so far share one common feature that limits their general applicability, namely, the fact that the cost functions defined are usually pre-specified, hand-constructed functions designed for a specific application. For example when designing optimisation criteria for robot

platforms, though they are constructed in a principled way, frequently there are open parameters and functions that need to be hand-tuned in order to realise desired behaviour. Also when trying to determine the optimisation criteria used by humans, the approach has often been to hand-design a function whose optimisation reproduces some property of human movements [9] or to empirically determine functions from experiments designed to elicit preferences in behaviour [1], [15]. The resultant criteria explain many features of human motion in certain types of movements well (e.g. single hand point-to-point reaching) but their extension to more complex scenarios (e.g. the choice between walking and leaning when reaching to a point out of range) is not clear.

Here, we propose to take a new approach to the problem whereby optimisation criteria are directly modelled from observed behaviour using statistical machine learning techniques. The principle is similar to that of inverse reinforcement learning [16], [17] whereby reward (cf. cost) functions are inferred for Markov decision processes from known action policies or from the trajectories generated by some unknown policy. We tailor our approach to movement selection in high-dimensional movement systems, i.e. for problems where multiple actions (in terms of choices of actuation) will achieve a given goal and where the task is to develop some criterion for choosing *which* to realise. We take a constructive approach whereby open-form, non-parametric models of cost functions are built incrementally. Once constructed these models can be used for optimal control in the context of optimal feedback control, gradient-based trajectory generation and planning, and null-space movement optimisation.

II. Problem Formulation

The problem we wish to address is that of determining how, given some task, redundant DOFs are utilised by some movement system in an optimal way with respect to some metric (i.e. cost function). Formally:

Given: (i) Some general set of control variables $\mathbf{c} \in \Re^n$; (ii) Some task formalised as a set of partial constraints on $\mathbf{c}$, i.e. $\mathbf{h}(\mathbf{c}) \in \Re^k$, where $k < n$; (iii) Observations of a system performing the task in a way such as to optimise some cost J in the $(n-k)$-dimensional task null-space;

Find The null-space optimisation criterion J.

In general, the complexity of this problem will depend on what information is available a priori. This includes knowing what variables, $\boldsymbol{\theta}$, are relevant to the calculation of the cost and whether observation data is *complete* with respect to those variables. For the purposes of this paper, we will assume that all the information required to reconstruct J is contained in our observation data. A second important consideration is knowledge about the *optimisation mechanism* used in conjunction with the cost function. This is crucial when we wish to directly derive the cost from observations since, for the same cost function, different optimisation schemes may produce different behaviours. Here, we make the assumption that the optimisation mechanism is known and takes the form of online gradient ascent in the task null-space. This allows us to infer values for J from observed trajectories by formulating the optimisation as a differential equation (where α is a rate constant)

$$\dot{\boldsymbol{\theta}} = -\alpha \nabla J \tag{1}$$

and then, solving that equation for J. In the following we motivate this assumption in the context of common kinematic and dynamic control schemes.

A. Resolved Velocity Kinematic Control

For velocity-based kinematic control one of the most popular control schemes is that of resolved motion rate control (RMRC) [18] which assumes a linearised forward model of the mechanical system

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$

where $\mathbf{x} \in \Re^k$ and $\mathbf{q} \in \Re^n$ are the task- and joint-space coordinates respectively, $\dot{\mathbf{x}}$ and $\dot{\mathbf{q}}$ denote the task- and joint-space velocities and $\mathbf{J}(\mathbf{q})$ is the Jacobian relating the two. Typically, the task is to track a desired trajectory $\mathbf{x}_d(t)$, i.e. the constraint can be formulated as $\mathbf{h} \equiv \mathbf{x}_d(t) - \mathbf{x}(t) = 0$. Also it is usually the case that $k < n$ so that the joint-space trajectory $\mathbf{q}(t)$ is redundant with respect to the task.

To exploit this redundancy a common strategy is to make use of the well-known Liégeois inverse kinematic model [19]

$$\dot{\mathbf{q}} = \mathbf{G}_1\dot{\mathbf{x}} - (\mathbf{I} - \mathbf{G}_2\mathbf{J})\mathbf{a} \tag{3}$$

where $\mathbf{I}$ denotes the identity matrix, $\mathbf{a} \in \Re^n$ is some arbitrary vector and $\mathbf{G}_1$ and $\mathbf{G}_2$ are generalised inverses of $\mathbf{J}$. It was recently shown that the Liégeois model can be used to represent most velocity-control methods (where an exact prescribed $\dot{\mathbf{x}}_d$ is given) [20]. Choosing $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{G}$ where

$$\mathbf{G} = \mathbf{W}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T)^{-1} \equiv \mathbf{W}^{-1/2}(\mathbf{J}\mathbf{W}^{-1/2})^{\dagger} \tag{4}$$

i.e. the $\mathbf{W}$-weighted pseudo-inverse[1] of $\mathbf{J}$, and $\mathbf{a}$ as

$$\mathbf{a} = -\alpha\mathbf{W}^{-1}\nabla J_q \tag{5}$$

where $J_q \equiv f(\mathbf{q})$, many established velocity-control techniques can be cast in this form [20]. It has been long established (see e.g. [2]) that in the case that $\mathbf{a} = 0$, if there exists a solution for $\dot{\mathbf{q}}$ in (2), the control law defined by (3)–(5) will choose that which minimises

$$J_{LS} = \dot{\mathbf{q}}^T\mathbf{W}\dot{\mathbf{q}} \tag{6}$$

and obeys the constraints $\mathbf{h}$ (or minimises violation of these constraints in terms of the least-squared transformation error $||\dot{\mathbf{x}} - \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}||^2$). One of the nice properties of this control scheme is that it gives a clear decomposition of the task- and null-space parts of the motion. While the first term in (3) handles the constraints $\mathbf{h}$, the second term determines how

[1] $\mathbf{A}^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{A}$

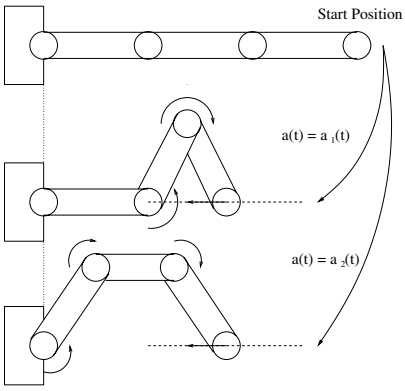

Fig. 2. Two ways to utilise the null-space under the constraint that the end-effector tracks a desired trajectory $\mathbf{h} \equiv \mathbf{x}(t) - \mathbf{x_d}(t) = 0$ (dashed line) with a three link arm. The choice $\mathbf{a}(t) = \mathbf{a}_1(t)$ uses the second and third joints more, whereas the choice $\mathbf{a}(t) = \mathbf{a_2}(t)$ uses the three joints equally.

the null-space is utilised. By appropriate choice of J_q, joint-space motion can be controlled with no effect on task-space motion (see Fig. 2). This scheme has been used for a variety of purposes such as joint-limit avoidance, obstacle avoidance and singularity avoidance.

B. Dynamic Control

For control at the level of dynamics and resolved acceleration kinematics, a similar approach can be taken. In particular, we can make use of a general methodology that was recently proposed for the design of optimal control laws for mechanical systems [21]–[23]. While kinematic control as described in the previous section assumes the existence of some 'dynamics compensator' allowing control to take place in kinematic position and velocity space [19], this methodology is based directly on physical principles of rigid-body mechanics.

The methodology assumes a robot model based on the Lagrangian equations of motion

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{F_c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F_g}(\mathbf{q}) \tag{7}$$

where $\mathbf{u} \in \Re^n$ is the commanded force, $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \Re^n$ are joint-space positions, velocities and accelerations $\mathbf{M}(\mathbf{q}) \in \Re^{n \times n}$ is a mass or inertia matrix, $\mathbf{F_c}(\mathbf{q}, \dot{\mathbf{q}}) \in \Re^n$ represents centrifugal and Coriolis forces and $\mathbf{F_g}(\mathbf{q}) \in \Re^n$ represents gravity. If the task description is defined in terms of a set of constraints of the form $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$ where $\mathbf{h} \in \Re^k$ and if these constraints can be reformulated as

$$\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}, t) \tag{8}$$

it has been shown [21]–[23] that the class of controllers given by

$$\mathbf{u} = \mathbf{W}^{-1/2}(\mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2})^{\dagger}(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) \tag{9}$$

where $\mathbf{F} = -\mathbf{F_c} - \mathbf{F_g}$, both satisfies the task constraint (8) and minimises the quantity

$$J_{DLS}(t) = \mathbf{u}^T\mathbf{W}\mathbf{u} \tag{10}$$

where $\mathbf{W}$ is some pre-defined metric. The choice of the metric $\mathbf{W}$ can be used to classify several control paradigms [21], such as resolved acceleration kinematic control ($\mathbf{W} = \mathbf{M}^{-2}$) or the Operational Space Formulation [24] ($\mathbf{W} = \mathbf{M}^{-1}$).

Since here our interest is in redundant systems (i.e. those for which $k < n$), use of the pseudo-inverse in (9) means we can again decompose the applied control force into the task- and null-spaces of the system [21]. By analogy with (3) and (4), we can modify (9) so that

$$\mathbf{u} = \mathbf{W}^{-1/2}\mathbf{T}^{\dagger}(\mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{F}) - (\mathbf{I} - \mathbf{W}^{-1/2}\mathbf{T}^{\dagger}\mathbf{T})\mathbf{a} \tag{11}$$

where $\mathbf{T} \equiv \mathbf{A}\mathbf{M}^{-1}\mathbf{W}^{-1/2}$. If we define

$$\mathbf{a} = -\alpha\mathbf{W}^{-1}\nabla J_u \tag{12}$$

we again have a potential, J_u, in the null-space of $\mathbf{T}$ which will be minimised with every time-step. Fig. 3 gives an example of a task where different choices of $\mathbf{a}$ are possible for the task of applying a force to a mass. The null-space term (12) could be used for joint-stabilisation as in [21] or impedance control for over-actuated arms.

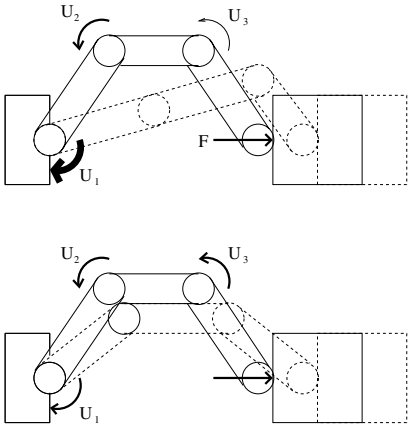

Fig. 3. Two ways to utilise the null-space in dynamics control when applying a force F to mass (box) with a fixed-base three link arm. The upper scheme applies a large torque to the base joint, a medium torque to the second joint and a small torque to the third joint. The lower scheme uses equal torques for each joint. The choice of $\mathbf{a}$ in (11) determines the scheme used.

III. Learning Cost Functions

In order to set up a robot controller that utilises the null space optimisation schemes discussed in the previous section, it is frequently necessary to rely on some hand-designed cost functions. Here we suggest a mechanism to reverse engineer the cost function used by a system assumed to obey the same control laws as the plant to be controlled. Our approach has some parallels in the empirical studies of human subjects such as [1], [15] in that our aim is to directly model cost functions from observations of behaviour. However, the setting explored here differs in that we do not receive explicit information about preferred (i.e. low cost) behaviour as compared to the forced-choice paradigm used in [15]. Instead, we directly observe the optimisation

process from the null-space movement and the preferences in behaviour are implicit in the path taken through the cost function parameter space during that movement.

Parallels can also be found in recent work on inverse reinforcement learning (IRL) such as [16], [17]. In both approaches the goal is to infer the reward (i.e. cost) function optimised without explicit information about that reward. However, in the IRL approach the reward function is inferred from the actions of a behaviour *policy* assumed optimal in the sense of Bellman optimality (i.e. maximum expected reward over future states and actions). In contrast, our approach deals with actions generated from online optimisation of the cost as more commonly found in robot control schemes.

The method developed below is therefore tailored to inferring the form of the cost function from passive observation of trajectories, and the incremental building of a model of the function from these observations. Next, we outline the approach taken to model the cost function and to extract training data for the modelling process.

A. Modelling Approach

Our approach is to try to directly learn the mapping:

$$\boldsymbol{\theta} \longrightarrow J(\boldsymbol{\theta}) \tag{13}$$

where $\boldsymbol{\theta}$ is the vector of parameters of J which may contain control variables[2] $\mathbf{c} = (c_1, \ldots, c_n)^T$ as well as independent (i.e. uncontrolled) state-variables $\mathbf{x} = (x_1, \ldots, x_m)^T$ which may be significant to the optimisation. Assuming the existence of an explicit signal for J that is measurable as the system moves through $\boldsymbol{\theta}$, (13) can be modelled with a supervised learning approach.

Our algorithm of choice is locally weighted projection regression (LWPR) [25] a fast non-parametric regression tool. LWPR has several features that make it well-suited to this task. Its Gaussian-kernel formulation gives smooth approximations of functions ensuring stability of gradient-based optimisation techniques. It is capable of dealing efficiently with high input dimensionality as found in high-redundancy movement systems and its partial least squares based dimensionality reduction feature effectively prunes out irrelevant or redundant input information. Furthermore, it learns incrementally; useful for learning from trajectories as and when they arrive. Finally, since LWPR is a non-parametric technique we need not assume any prior knowledge of the form of the cost function other than that it is a smooth, static function of the input parameters $\boldsymbol{\theta}$.

B. Collecting Data

The supervised learning of (13) requires that our training data consists of tuples of the form $(\boldsymbol{\theta}, J)$ for training the model. However, since in general, the cost J is not directly observable from movements, we require some way of inferring its value in a given state. Here we outline how this can be done purely from observations of the movement system.

[2]For kinematic control $\mathbf{c} = \dot{\mathbf{q}}$, for dynamics control $\mathbf{c} = \mathbf{u}$.

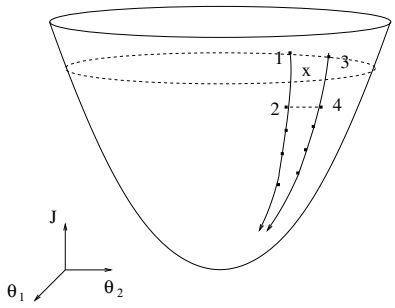

Fig. 4. Equalising trajectories to build a global model of the cost function.

1) Reconstructing J along a Single Null-space Trajectory: The first step in building the model of the cost function is to reconstruct the form of the function along each trajectory that the system performs. We assume for the moment that this trajectory is purely due to optimisation in the null-space and that the task constraints do not cause motion in $\boldsymbol{\theta}$. This is the case when, for example, $\dot{\mathbf{x}} = 0$ in the kinematic control scheme outlined in II-A.

The simplest method requires that we take samples of $\boldsymbol{\theta}$ during each trajectory at sampling rate r. This results in a set of via-points $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_{r\tau})^T$ for a trajectory of duration τ and the aim is to infer a value for J at each of these points. In order to do this, we take note of two observations. The first observation is that if we set r sufficiently high, the optimisation dynamics can be modelled as locally linear. This means that the trajectory generation mechanism (1) can be approximated by

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \nabla J(\boldsymbol{\theta}_n) \tag{14}$$

where the differentiation is with respect to $\boldsymbol{\theta}$. This gives us an estimate of the gradient at each point along the trajectory in terms of the distance between consecutive sampling points in $\boldsymbol{\theta}$. The second observation is that the absolute value of J is unimportant for the optimisation. This means that for any given trajectory, we can select one of the points along the trajectory and assign it some arbitrary value J_{ref}. Then, if we use the Euler method

$$J(\boldsymbol{\theta}_{n+1}) = J(\boldsymbol{\theta}_n) + (\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n)^T \nabla J(\boldsymbol{\theta}_n) \tag{15}$$

starting at that point and using J_{ref} as the initial value, we can iterate to find the value of J (measured relative to J_{ref}) for each remaining point on that trajectory.

2) Building a Global Model: The result of the above process is a set of data-points $\left((\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{r\tau})^T, (J(\boldsymbol{\theta}_1), \ldots, J(\boldsymbol{\theta}_{r\tau}))^T\right)$ for each trajectory recorded. The problem, however, is that there is no guarantee that the values for J from any two trajectories are measured with respect to the same reference point. In other words, we do not know the translation of the single-trajectory data sets relative to each other; this is analogous to obtaining 'indifference curves' [15] where utility has equal value without explicit knowledge of the relative utility between

them. However, assuming the cost surface to be a single-valued static function, the cost at the point of intersection between two trajectories is equal. This means that novel trajectories that intersect those previously seen can be translated to ensure the costs are equal at that point.

If necessary, this principle can also be extended to novel trajectories that approach (but do not intersect) seen ones to within some small distance by making local linear extrapolations of the modelled surface at the point of closest approach. The principle is shown in Fig. 4. Having assigned values for J for the two trajectories, the cost at the new point x can be estimated from a local linear regression of the plane defined by points 1–4. In this example, since the surface is two-dimensional, the learning process can be seeded with just two intersecting trajectories. It should be noted that in general, for an N-dimensional surface, at least N trajectories will be needed to define a surface from which we can extrapolate and grow our global model.

IV. Experiments

We performed simulations of kinematic controllers of the form described in Section II-A that has been implemented on the ASIMO humanoid robot [8]. However, in order to better illustrate the issues involved in the methodology, we report experiments performed on a reduced version of the system consisting of a planar three-link arm of unit link length and with revolute joints.

A. Learning Cost Functions from Trajectories

Our first set of experiments demonstrated the learning of a simple joint-limit avoidance cost function. A controller, C, similar to that described in [8] was used to generate data that optimised the cost function

$$J = (\mathbf{q} - \mathbf{q}_c)^T (\mathbf{q} - \mathbf{q}_c) \tag{16}$$

where $\mathbf{q}_c$ is a constant vector defining the position of the joint centres. The centres were chosen such that the default (minimum cost) position of the arm corresponded to the straight, fully-stretched position. The arm was assigned random initial end-effector (hand) positions and joint configurations from which linear hand trajectories at random speeds and directions were commanded. Samples of the cost and joint configuration $(\mathbf{q}, J)$, at twenty equally-spaced via points along these trajectories were used to incrementally train LWPR.

There was a rapid convergence of the learnt cost function to the true analytical version as the number of trajectories seen increased (refer Fig. 5). After 250 trajectories seen the normalised mean squared cost function prediction error was 0.0624 ± 0.0051. For less than fifty trajectories seen, the nMSE was more variable reflecting the fact that the random trajectories did not necessarily cover the space uniformly.

Since LWPR also has the feature of pruning irrelevant input information it is possible to learn cost functions using our approach without too much care about selecting what inputs to use, provided a subset of the inputs contains the relevant information. To show this we also tried learning the cost

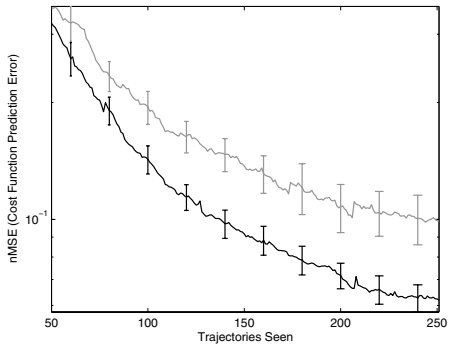

Fig. 5. Cost function prediction error against number of trajectories seen during learning averaged over 50 runs and 250 trajectories when learning $\mathbf{q} \rightarrow J$ (black) and when learning with irrelevant inputs $\mathbf{q}, \dot{\mathbf{q}} \rightarrow J$ (grey).

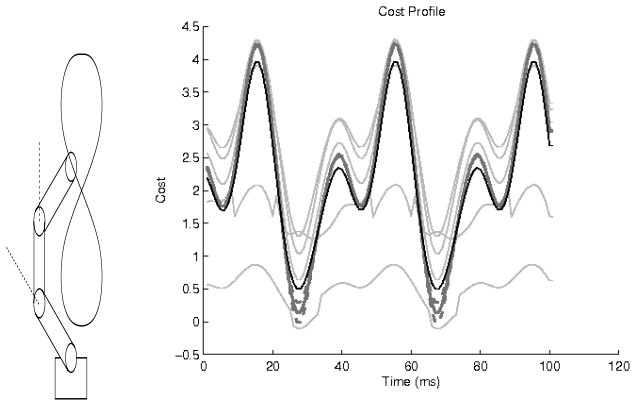

Fig. 6. Cost profile (right) during tracking of the test trajectory recorded from the true cost function (black line) and the learnt cost function after training with 1, 2, 5, 20, 50 (light grey lines) and 250 (thick grey line, error shown as dashed grey lines) trajectories. Since the trajectory was a circuit, the cost profile is periodic with the large peaks corresponding to the end-effector at the base of the eight-figure, forcing the arm into a high-cost 'folded' position (left).

function using the vector of joint positions augmented by the joint velocities, $(\mathbf{q}, \dot{\mathbf{q}})$, as inputs. Since the mapping was now $\Re^6 \rightarrow \Re$ the training took longer (refer Fig. 5), however once complete the null-space behaviour was again quantitatively reproduced.

To test the predictions of the learnt cost function, the arm controller was given an unseen test trajectory (a figure-eight curve) to track (refer Fig. 6, left) while optimising the true cost function (16). In tracking this trajectory, the arm was forced successively into low and high cost positions as the arm had to fold up for the base loop of the eight and unfold for the top loop. This gave a periodic structure to the cost profile that is clearly reproduced by the learnt model (Fig. 6, right).

Our final test was to compare the joint-space trajectories of controller C with those of a second controller, $\hat{C}$, that used the learnt cost function to see how well null-space behaviour was reproduced. Controller $\hat{C}$ was presented with

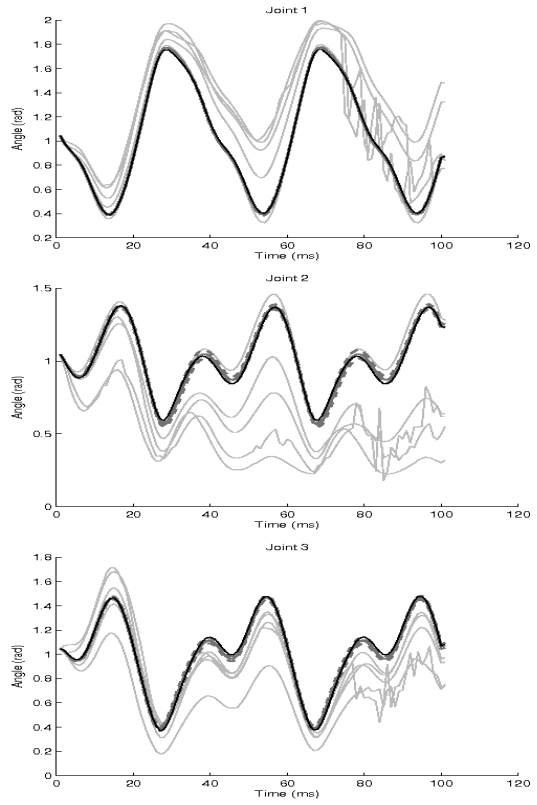

Fig. 7. Joint trajectories using the true cost function (black line) and learnt cost function after training with 1, 2, 5, 20, 50 (light grey lines) and 250 (thick grey line, error shown as dashed grey lines) trajectories.

the same figure-eight test trajectory and optimised movements with respect to the learnt cost function. Fig. 7 shows the trajectories of the three joints during tracking after several intervals of training of the cost function. Fig. 8 summarises the normalised mean squared (joint) tracking error of $\hat{C}$ as compared to the trajectory produced by controller C. The joint trajectories of the controller with the learnt cost function rapidly approached those of the original. Interestingly there was more variability in the joint trajectories of $\hat{C}$ as time progressed, possibly due to slight asymmetries in the learnt cost function being magnified over repeated loops of the figure-eight.

B. Obstacle Avoidance Cost Function

To demonstrate the application of our approach to more complex null-space behaviour the cost function (16) was augmented with a term designed for obstacle avoidance. The new cost function took the analytic form

$$J = (\mathbf{q} - \mathbf{q}_c)^T(\mathbf{q} - \mathbf{q}_c) + \beta \sum_i ||\mathbf{x}_i - \mathbf{x}_{obs}||^{-2} \qquad (17)$$

where $\mathbf{x}_i$ are task-space coordinates of a series of reference points attached to the arm, $\mathbf{x}_{obs}$ is the task-space coordinate of an obstacle and β was a scaling constant. Since the position of the reference points depended on the configuration of the

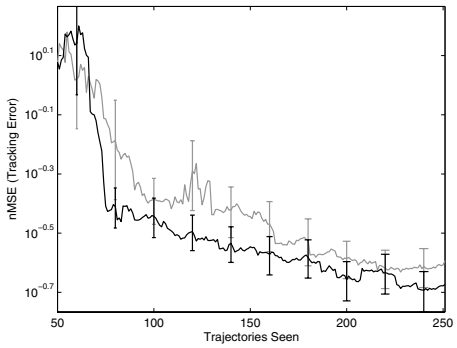

Fig. 8. Normalised mean squared (joint) tracking error against number of trajectories seen during learning, when learning $\mathbf{q} \rightarrow J$ (black) and when learning with irrelevant inputs $\mathbf{q}, \dot{\mathbf{q}} \rightarrow J$ (grey).

arm, there was a highly non-linear relationship between the input parameters, $\boldsymbol{\theta} \equiv (\mathbf{x}, \mathbf{q})$, and the cost.

We performed experiments in learning this cost function for the case where the hand position was constrained to a fixed position and a single obstacle was moving in the vicinity of the arm. This is similar to behaviour shown in humans, for example when carrying a tray of drinks, it is necessary to hold the tray steady to avoid spillage while avoiding obstacles in one's path. Using the learnt cost function the null-space behaviour of moving away from nearby obstacles was reproduced. Fig. 9 shows schematics of how the arm moved as the obstacle came near.

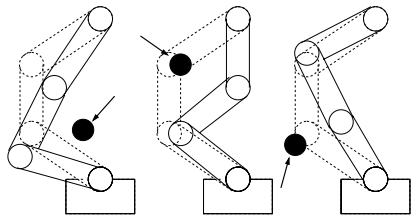

Fig. 9. Schematics of the arm reconfiguring its initial posture (dashed lines) to avoid an approaching obstacle (filled circle) as reproduced by learning the cost function (17).

C. Extracting J for a Constrained Kinematic Manipulator

In our final set of experiments the controller was again given the cost function (16), but this time we attempted to infer the cost function given only the movement data, i.e. given only observations of the joint space velocities $\dot{\mathbf{q}}$. The scheme outlined in Section III-B was used to reconstruct the form of the cost function. The constraint chosen in these experiments was of the kind 'maintain a given end-effector position while minimising the cost'. This kind of constraint can be seen in tasks where a contact point between the manipulator and the environment needs to be maintained (a detailed exploration of contact-constraint tasks can be found in [26]).

Again the manipulator was assigned a random initial hand position and joint configuration for each trajectory. The hand

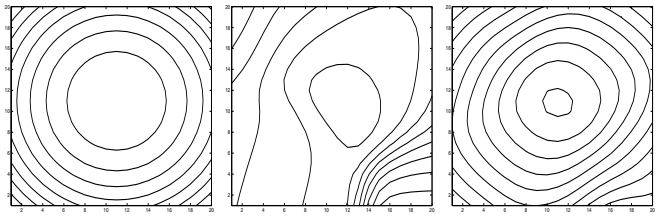

Fig. 10. Reconstruction of the cost function from movement data. (Left) True cost function; Reconstructed cost function after (Centre) 20 trajectories, (Right) 200 trajectories were seen.

was then constrained to the initial position ($\dot{\mathbf{x}} = 0$ in (3)) and the manipulator was allowed to reconfigure itself into the 'most comfortable' position according to the cost function. The trajectory in joint-space was sampled and used to reconstruct the form of the cost function. The resultant data was then used to train LWPR to learn the surface.

Convergence of learning was again fast with profiles of the nMSE similar to those shown in Fig. 5. Fig. 10 shows contour plots of the learnt cost function as a function of two of the joints of the arm after 20 and 200 trajectories seen. The radially symmetric form of the cost function started to emerge after just 20 trajectories were seen.

V. Conclusion

Research into null-space optimisation criteria has, so far, focused on the *design* of cost functions either for explaining human motion or for application to robot control. The disadvantages of this approach are clear in that the design process is highly time-consuming, requires hand tuning of parameters, and the resultant criteria may be applicable only to the system for which they were designed.

We propose a method that automates the process by using statistical machine learning to reverse-engineer the null-space behaviour of existing systems. It has been shown that learnt models of cost functions can be used in place of analytical ones with little loss of performance in terms of joint-space tracking error. Furthermore, it has been shown that cost functions can be reconstructed from trajectory information even if there is no explicit cost signal for supervised learning, given sufficiently dense samples of trajectories.

In future work, we intend to extend the current methodology to the case where task constraints perturb the optimisation in the null-space. In such cases, extraction of cost information is more complex since the joint-space movement then consists of a combination of task-space motion and null-space motion, introducing an error in the Euler method dependent on the task-space motion. Furthermore, we intend to demonstrate the use of the proposed methodology in the null-space control of very high-dimensional systems such as the humanoid robot ASIMO.

References

[1] K. Körding and D. Wolpert, "The loss function of sensorimotor learning," in *Proceedings of the National Academy of Sciences*, vol. 101, 2004, pp. 9839–42.

[2] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Reading, MA: AddisonWesley, 1991.

[3] W. L. Nelson, "Physical principles for economies of skilled movements," *Biological Cybernetics*, vol. 46, pp. 135–47, 1983.

[4] F. Chaumette and E. Marchand., "A new redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing." in *IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 1720–1725.

[5] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[6] S. I. Choi and B. K. Kim, "Obstacle avoidance control for redundant manipulators using collidability measure," *Robotica*, vol. 8, pp. 143–151, 2000.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE Intl. Conf. on Robotics and Automation*, vol. 1, 1985, pp. 428–436.

[8] M. U. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Proc. IEEE-RAS International Conference on Humanoid Robots*. IEEE Press, 2005, pp. 238–244.

[9] S. Engelbrecht, "Minimum principles in motor control," *Journal of Mathematical Psychology*, vol. 45, no. 3, pp. 497–542, 2001.

[10] T. Flash and N. Hogan, "The co-ordination of arm movements: An experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, pp. 1688–1703, 1985.

[11] E. Nakano, H. Imamizu, R. Osu, Y. Uno, H. Gomi, T. Yoshioka, and M. Kawato, "Quantitative examinations of internal representations for arm trajectory planning: minimum commanded torque change model," *J Neurophysiol*, vol. 81, pp. 2140–55, 1999.

[12] C. M. Harris and D. M. Wolpert, "Signal-dependent noise determines motor planning," *Nature*, vol. 394, pp. 780–784, 1998.

[13] F. Lacquaniti, C. Terzuolo, and P. Viviani, "The law relating the kinematic and figural aspects of drawing movements," *Acta Psychologica*, vol. 54, pp. 115–130, 1983.

[14] E. Todorov and M. Jordan, "Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements," *Journal of Neurophysiology*, vol. 80, pp. 696–714, 1998.

[15] K. Körding, I. Fukunaga, I. Howard, J. Ingram, and D. Wolpert, "A neuroeconomics approach to inferring utility functions in sensorimotor control," *PLoS Biol*, vol. 2, no. 10, p. 330, 2004.

[16] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 663–670.

[17] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. ICML*, 2004.

[18] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," in *IEEE Trans. Man–Mach. Syst.*, vol. MMS-10, no. 22, 1969, pp. 47–53.

[19] A. Ligeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," in *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, 1977, pp. 868–871.

[20] J. D. English and A. A. Maciejewski, "On the implementation of velocity control for kinematically redundant manipulators," in *IEEE Transactions On Systems, Man, And Cybernetics-Part A: Systems And Humans*, vol. 30, no. 3, 2000, p. 233.

[21] J. Peters, M. Mistry, F. Udwadia, R. Cory, J. Nakanishi, and S. Schaal, "A unifying methodology for the control of robotic systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1575–1582.

[22] F. E. Udwadia, "A new perspective on the tracking control of nonlinear structural and mechanical systems," in *Proc. of the Royal Society of London, Series A*, vol. 459, 2003, pp. 1783–1800.

[23] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," in *Proc. IEEE International Conference on Robotics and Automation*, 2000, pp. 2563–2568.

[24] O. Khatib, *Robot Control: Dynamics, Motion Planning and Analysis*. IEEE Press, 1993, ch. A Unified Approach to Motion and Force Control of Robot Manipulators: The Operational Space Formulation, pp. 277–287.

[25] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[26] J. Park and O. Khatib, "Contact consistent control framework for humanoid robots," in *Proc. 2006 IEEE International Conference on Robotics and Automation*, 2006, pp. 1963–69.