The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

# Learning Nullspace Policies

Chris Towell, Matthew Howard and Sethu Vijayakumar

*Abstract*— **Many everyday tasks performed by people, such as reaching, pointing or drawing, resolve redundant degrees of freedom in the arm in a similar way. In this paper we present a novel method for learning the strategy used to resolve redundancy by exploiting the variability in multiple observations of different tasks. We demonstrate the effectiveness of this method on three simulated plants: a toy example, a three link planar arm, and the KUKA lightweight arm.**

## I. Introduction

Humans arms are often redundant with respect to a particular task since the freedom in joint space is usually greater than that required for the task. For example, keeping the hand at a fixed location on a desk still allows the elbow to move through a range of motions. Humans often employ a single strategy to resolve joint redundancy for a range of tasks, for example, the position of the elbow is usually low down, close to the body in a variety of tasks such as pointing, pouring and wiping, as shown in Fig. 1. In robotics, control of redundant manipulators is often decomposed into two orthogonal components using the well known pseudo-inverse solution [10], [9], [15], [11]. A task space component determines the control of joint angles required to achieve a task and a nullspace component determines how any redundancy with respect to the task is resolved. The latter is used to accomplish a secondary, lower priority task to complement the first, for example, for avoidance of joint limits [2], singularities [17] or obstacles [9]. In principle, humans must also solve these problems in task-oriented behaviour, motivating research into methods that can do this decomposition from data. An important benefit to finding this decomposition is as follows.

If a robot has a similar morphology to a demonstrator, it is desirable to learn the nullspace component for transfer to the robot. For example, a humanoid robot has roughly the same degrees of freedom as a human. To facilitate interaction between it and humans, it should move in ways similar to humans with corresponding patterns of joints. This allows humans to predict the robot's movements more easily, making them more comfortable with the robot. In this case we wish to learn the redundancy resolution in such a way as to be able to transfer to the robot, and to generalise to a range of novel tasks.

Udwadia [16] describes the pseudo-inverse solution in terms of constraints. The task space component of a motion in the pseudo-inverse solution can be thought of as a constraint on the nullspace component. Much work has been done to exploit statistical regularities in constrained demonstrations in order to extract features relevant to the task (for example [4], [5], [1], [6]). Typically such work learns from demonstrations with fixed constraints. Howard [8] has pursued the alternative of learning unconstrained policies that are maximally consistent with observations under different constraints.

In this paper, we make use of the idea that the pseudo-inverse solution to the inverse kinematics problem is a problem of constraints. We extend the approach in [8] to seek inconsistencies in the demonstrations of different tasks in order to learn the nullspace resolution. We demonstrate that this method clearly outperforms the standard form of direct policy learning and that it can then be successfully applied to novel tasks.

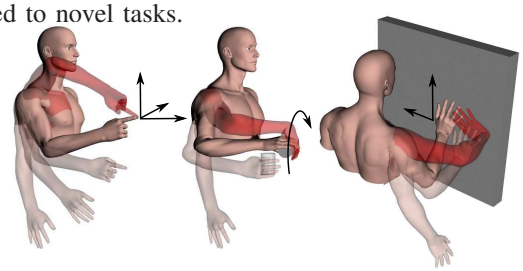


Fig. 1. Three different tasks: moving the finger to an $x, y, z$, position, pouring liquid and wiping a surface. In each case, redundancy is resolved in the same way. The red arms show alternative, less natural ways to resolve redundancy. By observing several examples of each task, we learn the single underlying policy that resolves redundancy.

## II. Problem Definition

In this section, we characterise the approach of direct policy learning (DPL) [14], [12] as applied to the problem of learning from observations under task constraints. The general form of DPL is as follows. If $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$ represent states and actions respectively, we seek to learn the mapping

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

given paired observations of $\mathbf{u}(t)$ and $\mathbf{x}(t)$ in the form of trajectories. For example, in kinematic control the states and actions may be the joint positions and velocities, respectively. Alternatively, in dynamics control, the state may include joint positions and velocities, with torques as actions. Importantly, it is typically assumed that in demonstrations, the actions $\mathbf{u}$ of the policy $\boldsymbol{\pi}$ are directly observed [14], [12].

In this paper, we wish to learn policies that describe how redundancy is resolved with respect to higher priority task constraints. Specifically, we assume that our observations contain different components of motion due to both the nullspace policy, and the task constraints. In such cases, standard approaches to DPL encounter several difficulties.

For example, consider the problem of learning the policy used to resolve redundancy in a pointing task, as shown in Fig. 2. There, the task is to move the finger tip to a specific position (red target). The nullspace policy resolves the redundancy by attempting to move the joints to the most comfortable posture (here, a posture with joint angles near

C. Towell, M. Howard and S. Vijayakumar are with the Institute of Perception Action and Behaviour, University of Edinburgh, Scotland, UK. c.c.towell@sms.ed.ac.uk

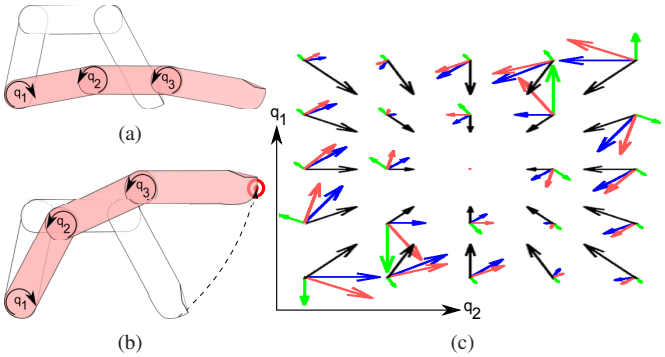



Fig. 2. (a) Movement due to the nullspace policy, with no task constraint. The finger approaches a nearly straight, 'comfortable' posture. (b) Movement under a 'pointing' task. The task constraints drive the finger tip to the Cartesian position indicated by the target. The nullspace policy acts to resolve the redundancy in the remaining one degree of freedom. (c) Vector field representation of the movement. The two axes correspond to the first two joint angles. Arrows indicate joint velocities observed (red), velocities due to task constraints (green), velocities due to the nullspace component (blue) and velocities due to the nullspace policy (black)

zero where the finger is slightly bent). Fig. 2(a) shows the movement in the absence of the task, and the corresponding vector field representation of this is shown in black in Fig. 2(c). As can be observed, in the absence of the task constraints, each of the vectors point to the zero (central) position. On the other hand, Fig. 2(b) shows the the finger's movement to the target under the task constraints. The red arrows in Fig. 2(c) show the corresponding *observed* joint velocities, with the nullspace component shown in blue and the task space component shown in green. Clearly, directly applying DPL on the observed movements (red arrows) will give a poor approximation of the underlying policy (black) or the nullspace component (blue). Instead, we must consider the structure of the data in terms of the task constraints in order to inform learning in this setting.

### A. Constraint Model

One way of thinking about the combination of task and nullspace policy in the above example is to think of the task as a constraint on the nullspace policy. This, unlike standard DPL, allows us to account for the fact that part of the policy is obscured by the constraint, and the remaining part will have some task component added to it. The dimension in which the policy is obscured is the same as that in which the task space component is added. Consider the set of consistent $k$-dimensional constraints

$$\mathbf{A}(\mathbf{x})\mathbf{u}(\mathbf{x},t) = \mathbf{b}(\mathbf{x},t) \tag{1}$$

with $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$. The general solution to this set of equations is

$$\mathbf{u}(\mathbf{x},t) = \mathbf{A}(\mathbf{x})^\dagger \mathbf{b}(\mathbf{x},t) + \mathbf{N}(\mathbf{x})\boldsymbol{\pi}(\mathbf{x}) \tag{2}$$

where $\mathbf{A}^\dagger$ denotes the unique Moore-Penrose pseudo-inverse of the matrix $\mathbf{A}$ and $\mathbf{N}(\mathbf{x}) = (\mathbf{I} - \mathbf{A}(\mathbf{x})^\dagger \mathbf{A}(\mathbf{x})) \in \mathbb{R}^{d\times d}$, $\boldsymbol{\pi}(\mathbf{x}) \in \mathbb{R}^d$ is an arbitrary vector. For clarity, we will now drop the dependence on $\mathbf{x}$ and $t$.

In the case of kinematic control of the end-effector of a manipulator we can identify (1)–(2) with the well known pseudo-inverse solution to inverse kinematics control. The states are the joint angles $\mathbf{x} = \mathbf{q}$ and the actions are the joint velocities $\mathbf{u} = \dot{\mathbf{q}}$. $\mathbf{b}$ is a policy which outputs end-effector velocities and $\mathbf{A}$ is the Jacobian $\mathbf{J}$ relating joint velocities to end-effector velocities. For redundant manipulators, the nullspace of $\mathbf{J}$ is not empty and $\boldsymbol{\pi}$ can be used to control motion in the joint space without affecting the task-space motion. Policies $\boldsymbol{\pi}$ indexed by joint angles can be used to drive the joint configuration towards a comfortable position and are compatible with the human cost functions proposed in [3].

If our observations of $\mathbf{u}$ and $\mathbf{x}$ are generated by (2) with the same policy $\boldsymbol{\pi}$, then the general problem of constrained DPL is to recover this policy. In [8], constraints of the form given by (1) are considered where $\mathbf{b} = \mathbf{0}$. Here we consider the more complex case of non-zero $\mathbf{b}$.

We will term the two parts of (2) the task space component ${}^{ts}\mathbf{u}$ and the nullspace component ${}^{ns}\mathbf{u}$

$$\mathbf{u} = \mathbf{A}^\dagger \mathbf{b} + \mathbf{N}\boldsymbol{\pi} = {}^{ts}\mathbf{u} + {}^{ns}\mathbf{u}. \tag{3}$$

As noted earlier, it would be useful to obtain this decomposition into the two components and even to obtain the nullspace policy $\boldsymbol{\pi}$. This would allow us to model the redundancy resolution observed in a variety of tasks (such as in Fig. 1) or to apply the same strategy to a new task, defined in a different space.

For learning $\boldsymbol{\pi}$, we assume that multiple training examples are available across a variety of tasks. The difficulty here is that, for any given observation, we do not know the exact form of the task, i.e., we may not know $\mathbf{A}$, $\mathbf{b}$ or $\mathbf{N}$. This is especially apparent in learning from human demonstrations where, for example, the exact end-effector Jacobian is unknown, and, even if it were, it is often not clear exactly which end-effector degrees of freedom are controlled as part of the task. For example, if you point at a far away target, the orientation of the hand is controlled such that it points towards the target. It is less clear whether the $x, y, z$ position of the hand is part of the task or whether a comfortable position is chosen as part of the redundancy resolution.

In addition to this, the problem of learning $\boldsymbol{\pi}$ is also non-convex in two ways. The observed action $\mathbf{u}$ can appear differently under different tasks due to variations in $\mathbf{b}$ for the same $\boldsymbol{\pi}$. Also, two nullspace components can appear differently under two different task spaces due to variation in the constraint matrix $\mathbf{A}$ for the same $\boldsymbol{\pi}$. For DPL, this means that we cannot expect the mean of observations to give us the nullspace policy.

The problem is also degenerate in two ways. There may be multiple policies $\boldsymbol{\pi}$ that are projected by $\mathbf{N}$ to the same nullspace component and there may be multiple ways to decompose $\mathbf{u}$ into two orthogonal components depending on what the true task space consists of.

Despite these difficulties, we consider a class of problems where we are able to group observations as having been generated in a specific task space (having the same constraint matrix $\mathbf{A}$). Such tasks may be those which require the $x, y, z$ Cartesian position of the end-effector (for example drawing) or those which require control over orientation (for example pouring liquid from a cup) and, in a real world scenario, would be straightforward to label. If we make this assumption, although we may not know the nature of the constraint, and given sufficient variation in tasks, then we

will show that a model of the nullspace policy $\boldsymbol{\pi}$ can still be learnt.

## 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 using the same underlying policy $\boldsymbol{\pi}(\mathbf{x})$ to resolve redundancy, which for a particular observation might have been constrained by task constraints, that is $\mathbf{u}_n = \mathbf{A}_n^\dagger \mathbf{b}_n + \mathbf{N}_n \boldsymbol{\pi}(\mathbf{x}_n)$ for task space movement $\mathbf{b}_n$ and constraint $\mathbf{A}_n$. We assume that the latter ($\mathbf{A}_n$ and $\mathbf{b}_n$) are *not explicitly known* for any given observation, but that observations may be grouped into $K$ subsets of $N$ data points[1], each recorded under a different constraint (i.e., the $k$th data set contains observations under the $k$th task constraint $\mathbf{A}_k(\mathbf{x})$). Our goal is to reconstruct the nullspace policy $\boldsymbol{\pi}(\mathbf{x})$.

Given only $\mathbf{x}_n$ and $\mathbf{u}_n$, one may be tempted to simply minimise the standard risk

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

which would correspond to the standard DPL approach. However, this would ignore the constraints and task space movements, and correspond to a naive averaging of commands from different circumstances.

Since we know that our data contains constraints, a second tempting possibility is to directly use constraint consistent learning (CCL) [7]. This estimates a policy $\tilde{\boldsymbol{\pi}}(\mathbf{x})$ by minimising the inconsistency error [7]

$$E_i[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{K \times N} \|\mathbf{u}_n - \mathbf{P}_n \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2; \quad \mathbf{P}_n = \frac{\mathbf{u}_n \mathbf{u}_n^T}{\|\mathbf{u}_n\|^2}. \tag{5}$$

However, as discussed in Sec. II, constraint consistent learning relies on the assumption that $\mathbf{b}(\mathbf{x}, t) = \mathbf{0}$, i.e. that the task constraints are stationary. In our setting, the non-zero task space movement ${}^{ts}\mathbf{u}(\mathbf{x}, t)$, interferes with learning, resulting in poor performance.

Instead, our proposal is to use a new two-step approach to learning. In the first step, we use the $K$ data subsets to learn a set of intermediate policies ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$, $k = 1, \ldots, K$. The latter should capture the nullspace component of motion ${}^{ns}\mathbf{u}(\mathbf{x}, t)$ under each of the $K$ task constraints $\mathbf{A}_k(\mathbf{x})$, while eliminating as far as possible the task space component ${}^{ts}\mathbf{u}(\mathbf{x}, t)$. Having learnt these intermediate models, we can then combine our observations into a single model that captures the policy used for redundancy resolution across tasks. For learning the latter, we propose to bootstrap CCL on the predictions from the intermediate policies, in order to estimate the true underlying policy $\boldsymbol{\pi}(\mathbf{x})$. A schematic of the approach is illustrated in Fig. 3.

### *Step 1: Learning the Nullspace Component*

In this step, we process each of the $K$ data subsets to learn a set of intermediate policies ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$, $k = 1, \ldots, K$, that capture the nullspace component of motion ${}^{ns}\mathbf{u}(\mathbf{x})$. In other words, for the $k$th data subset, we seek a policy that minimises

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

where ${}^{ns}\mathbf{u}_{k,n}$ is the true nullspace component of the $n$th data point in the $k$th data subset. Note that, by assumption, we do not have access to samples ${}^{ns}\mathbf{u}_{k,n}$, so we cannot directly optimise (6).

Instead, we seek to eliminate the components of motion that are due to the task constraints, and learn a model that is *consistent* with the observations. The key to our approach, is to use a projection to do that elimination, that is, we seek a projection $\mathbf{P}$ for which

$$\mathbf{P}\,\mathbf{u} = \mathbf{P}\,({}^{ts}\mathbf{u} + {}^{ns}\mathbf{u}) = {}^{ns}\mathbf{u}. \tag{7}$$

One such projection is the matrix $\mathbf{N}(\mathbf{x})$ since, by definition, its image space (or any subspace of this) is orthogonal to the task (ref. Fig. 3(a)). However, this is also not possible since $\mathbf{N}(\mathbf{x})$ is also unavailable by assumption.

A second possibility would be to replace $\mathbf{P}$, with a projection onto the true nullspace component ${}^{ns}\mathbf{u}$, i.e., defining $\mathbf{P} \equiv {}^{ns}\mathbf{u}\,{}^{ns}\mathbf{u}^T / \|{}^{ns}\mathbf{u}\|^2 = {}^{ns}\mathbf{P}$. Since ${}^{ns}\mathbf{u}$ is, by definition, orthogonal to ${}^{ts}\mathbf{u}$, this would effectively eliminate any task space components in the observed data (as can be seen, for example, in the projections of $\mathbf{u}_1$, $\mathbf{u}_2$ onto ${}^{ns}\mathbf{u}_1$ in Fig. 3(b)). However, since samples of ${}^{ns}\mathbf{u}$ are also not directly available, such an approach is also not possible with the data assumed given.

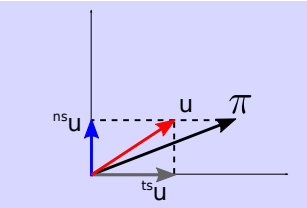
However, motivated by this observation, we can instead make an approximation of the required projection. Our proposal is to replace $\mathbf{P}$, with a projection based on an *estimate of the nullspace component*, and proceed to iteratively refine that estimate in order to optimise consistency with the observations. For this, we propose to minimise the error function

$$E_1[{}^{ns}\tilde{\boldsymbol{\pi}}_k] = \sum_{n=1}^{N} \|\tilde{\mathbf{P}}_{k,n}\mathbf{u}_{k,n} - {}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x}_n)\|^2 \tag{8}$$

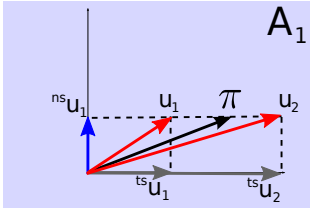
with $\tilde{\mathbf{P}}_{k,n} = {}^{ns}\tilde{\boldsymbol{\pi}}_{k,n}\,{}^{ns}\tilde{\boldsymbol{\pi}}_{k,n}^T / \|{}^{ns}\tilde{\boldsymbol{\pi}}_{k,n}\|^2$. Here, $\mathbf{u}_{k,n}$ is the $n$th data point in the $k$th data subset, and we defined ${}^{ns}\tilde{\boldsymbol{\pi}}_{k,n} = {}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x}_n)$. Minimising (8) corresponds to minimising the difference between the current model of the nullspace movement, ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$, and the observations *projected onto that model*. An illustration is shown in Fig. 3(b)-(c).

Effectively, in (8) we approximate $\mathbf{N}(\mathbf{x})$ with a projection onto a 1-D space in which, if our current estimate of ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$ is accurate, the true nullspace component ${}^{ns}\mathbf{u}_n$ lies. At this point we note that the quality of this approximation will, in general, depend on our on how well the current estimate ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$ captures the true underlying policy. Since we pursue an iterative approach, this means that the initialisation of model parameters has a significant effect on the accuracy of our final estimate. In practice, in the absence of any prior information about the policy, we can draw several random sets of parameters for initialisation, run the optimisation, and select the model that best minimises (8).

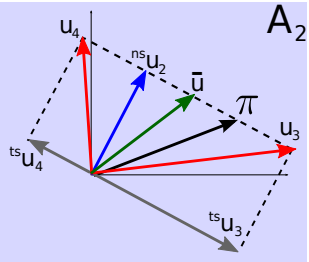
[1]For clarity, here we will assume that the subsets are of equal size, but in general the sizes may differ.



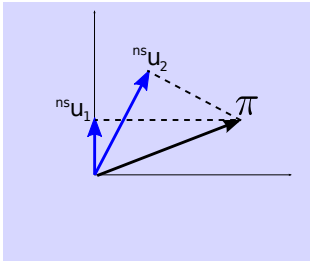

(a) Raw observation $\mathbf{u}$ consists of two orthogonal components ${}^{ts}\mathbf{u}$ and ${}^{ns}\mathbf{u}$ (not explicitly observable).




(b) Two observations $\mathbf{u}_1$, $\mathbf{u}_2$ of different tasks performed in the task space defined by $\mathbf{A}_1$.




(c) Two observations $\mathbf{u}_3$, $\mathbf{u}_4$ of tasks performed in a second task space defined by $\mathbf{A}_2$. Naive regression results in prediction $\bar{\mathbf{u}}$.




(d) Reconstruction of the underlying policy $\boldsymbol{\pi}$ by CCL on the model predictions ${}^{ns}\tilde{\boldsymbol{\pi}}_1$, ${}^{ns}\tilde{\boldsymbol{\pi}}_2$.

Fig. 3. Illustration of our approach. Raw observations $\mathbf{u}_1$, $\mathbf{u}_2$ in space $\mathbf{A}_1$ project onto the nullspace component ${}^{ns}\mathbf{u}_1$. Similarly, $\mathbf{u}_3$ and $\mathbf{u}_4$ in space $\mathbf{A}_2$ project onto nullspace component ${}^{ns}\mathbf{u}_2$. Naive regression on the data causes model averaging and a poor prediction (e.g., $\bar{\mathbf{u}}$). We therefore first seek the nullspace components ${}^{ns}\mathbf{u}_1$, ${}^{ns}\mathbf{u}_2$ then apply CCL to find the underlying policy $\boldsymbol{\pi}$.

A second point to note is 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 regression techniques. However, for the experiments in this paper, we restrict ourselves to two classes of function approximator for learning the nullspace component of the observations. These are (i) simple parametric models with fixed basis functions (Sec. III-A), and (ii) locally linear models (Sec. III-B). In the following we briefly outline how these models can be trained using the proposed approach.

### A. Parametric Policy Models

A convenient model for capturing the nullspace component of the $k$th data subset is given by ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x}) = \mathbf{W}_k\mathbf{b}_k(\mathbf{x})$, where $\mathbf{W}_k \in \mathbb{R}^{d\times M}$ is a matrix of weights, and $\mathbf{b}_k(\mathbf{x}) \in \mathbb{R}^M$ is a vector of fixed basis functions. This notably includes the case of (globally) linear models where we set $\mathbf{b}_k(\mathbf{x}) = \bar{\mathbf{x}} = (\mathbf{x}^T, 1)^T$, or the case of normalised radial basis functions (RBFs) $b_{k,i}(\mathbf{x}) = \frac{K(\mathbf{x}-\mathbf{c}_i)}{\sum_{j=1}^{M} K(\mathbf{x}-\mathbf{c}_j)}$ calculated from Gaussian kernels $K(\cdot)$ around $M$ pre-determined centres $\mathbf{c}_i$, $i = 1\ldots M$.

For the $k$th data subset, with this model, the error (8) becomes

$$E_1(\mathbf{W}_k) = \sum_{n=1}^{N} \left\| \frac{\mathbf{W}_k\mathbf{b}_{k,n}(\mathbf{W}_k\mathbf{b}_{k,n})^T\mathbf{u}_{k,n}}{\|\mathbf{W}_k\mathbf{b}_{k,n}\|^2} - \mathbf{W}_k\mathbf{b}_{k,n} \right\|^2 \tag{9}$$

where we defined $\mathbf{b}_{k,n} = \mathbf{b}_k(\mathbf{x}_n)$. Due to the 4th-order dependence on $\mathbf{W}_k$, this is a non-linear least squares problem which cannot easily be solved for $\mathbf{W}_k$ in closed form. However, in order to find the optimal weights

$$\mathbf{W}_k^{opt} = \arg\min E_1(\mathbf{W}_k) \tag{10}$$

we can apply fast numerical optimization techniques suited to solving such problems. In our experiments, we use the efficient Levenberg-Marquardt (LM) algorithm to optimise the parameters based on (9).

### B. Locally Linear Policy Models

The parametric models of the previous section quickly encounter difficulties as the dimensionality of the input space increases. As an alternative, we can use local learning techniques. For this, we fit multiple locally weighted linear models ${}^{ns}\tilde{\boldsymbol{\pi}}_{k,m}(\mathbf{x}) = \mathbf{B}_{k,m}\bar{\mathbf{x}} = \mathbf{B}_{k,m}(\mathbf{x}^T, 1)^T$ to the $k$th data subset, learning each local model independently [13].

For a linear model centred at $\mathbf{c}_m$ with an isotropic Gaussian receptive field with variance $\sigma^2$, we would minimise

$$E_1(\mathbf{B}_{k,m}) = \sum_{n=1}^{N} w_{n,m} \left\| \mathbf{P}_{k,n,m}\mathbf{u}_{k,n} - \mathbf{B}_{k,m}\bar{\mathbf{x}}_n \right\|^2 \tag{11}$$

where $\mathbf{P}_{k,n,m} = \mathbf{B}_{k,m}\bar{\mathbf{x}}_n(\mathbf{B}_{k,m}\bar{\mathbf{x}}_n)^T / \|\mathbf{B}_{k,m}\bar{\mathbf{x}}_n\|^2$. 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

$$\mathbf{B}_{k,m}^{opt} = \arg\min E_1(\mathbf{B}_{k,m}) \tag{12}$$

are retrieved with a non-linear least squares optimiser. Similar to the parametric approach, in our experiments we use the LM algorithm for this.

Finally, for the global prediction of the nullspace policy, we combine the local linear models using the convex combination

$${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x}) = \frac{\sum_{m=1}^{M} w_m \mathbf{B}_{k,m}\bar{\mathbf{x}}}{\sum_{m=1}^{M} w_m}$$

where $w_m = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{c}_m\|^2\right)$.

### *Step 2: Learning the Underlying Policy*

Applying the approach described in the previous sections to each of the $K$ data subsets, we are then left with a set of intermediate models ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$ that approximate the nullspace component of motion for tasks performed in the $K$ different task spaces. Our task now is to combine these intermediate models ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$ to find a single consistent approximation of the underlying policy $\boldsymbol{\pi}(\mathbf{x})$.

This is relatively straightforward using CCL [7]. Specifically, we make predictions from the intermediate models to form a concatenated data set $\{\mathbf{x}_n, {}^{ns}\tilde{\boldsymbol{\pi}}_{k,n}\}_{n=1}^{N}$ for $k = 1,\ldots,K$, where ${}^{ns}\tilde{\boldsymbol{\pi}}_{k,n} \equiv {}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x}_n)$. We then directly apply CCL on this data by seeking a policy estimate that minimises the objective function

$$E_2[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N\times K} \|{}^{ns}\tilde{\boldsymbol{\pi}}_n - \tilde{\mathbf{P}}_n\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2; \quad \tilde{\mathbf{P}}_n = \frac{{}^{ns}\tilde{\boldsymbol{\pi}}_n\,{}^{ns}\tilde{\boldsymbol{\pi}}_n^T}{\|{}^{ns}\tilde{\boldsymbol{\pi}}_n\|^2}. \tag{13}$$

As described in [7], a closed-form solution to this optimisation exists for both parametric and local linear policy models, making this final step highly efficient. The outcome is a policy model that is consistent with each of the intermediate

**Algorithm 1**

1: Split demonstrations into K data subsets, one for each type of task constraint.
2: **for** $k = 1$ **to** $K$ **do**
3: Learn intermediate policy ${}^{ns}\tilde{\boldsymbol{\pi}}_k(\mathbf{x})$ by minimising $E_1$ (8) using numerical optimization.
4: Output predictions for ${}^{ns}\mathbf{u}_{k,n}$ using learnt policy.
5: **end for**
6: Combine predictions into a single dataset.
7: Use CCL [7] to learn the underlying nullspace policy $\boldsymbol{\pi}$.

models (ref. Fig. 3(d)), and can be used to make predictions about redundancy resolution, even under task constraints that are previously unseen in the data. The whole process is summarised in Algorithm 1.

## IV. Experiments

To demonstrate the performance of the algorithm, we applied it to scenarios on three simulated plants. Firstly an artificial toy two-dimensional system, then a planar three link arm and finally a higher dimensional 7-DOF Kuka lightweight arm. Data was generated by numerically integrating (2).

### A. Toy Example

The goal of the first set of experiments was to demonstrate the principles involved in our approach, and characterise its performance for learning polices of varying complexity, under different noise conditions and with varying amounts of data.

For this, we set up a simple toy system consisting of a linear attractor policy

$$\boldsymbol{\pi}(\mathbf{x}) = \beta(\mathbf{x}_0 - \mathbf{x}) \tag{14}$$

with states $\mathbf{x} \in \mathbb{R}^2$ and actions $\mathbf{u} \in \mathbb{R}^2$ representing position and velocities, respectively. The policy has a single attractor point which we set to $\mathbf{x}_0 = \mathbf{0}$, and the scaling factor was set to $\beta = 0.1$. Policies such as (14) are commonly used for joint limit avoidance in many inverse kinematics control schemes [2].

The policy (14) was subject to 1-D task constraints of the form

$$\mathbf{A} = \hat{\boldsymbol{\alpha}}^T \; ; \quad \boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T \tag{15}$$

so that for any given choice of $\mathbf{A}$, the task space is defined as the direction parallel to the normalised vector $\hat{\boldsymbol{\alpha}}$ (for example, if $\hat{\boldsymbol{\alpha}} = (1, 0)^T$, then the task space consists of the first dimension of the state-space).

To simulate the effect of observing multiple tasks in different spaces, we collected data in which, for each demonstration, a randomly generated task space policy acted in a random subspace of the system. Specifically, for each trajectory, the elements of $\boldsymbol{\alpha}$ were chosen from the uniform distribution $\alpha_i \sim U[0, 1]$. Using this as the task space, movements were then generated with a linear attractor policy

$$\mathbf{b}(\mathbf{x}) = \beta_{ts}(\mathbf{r}^* - \mathbf{r}). \tag{16}$$

Here, $\mathbf{r}$ denotes the current position in task space, $\mathbf{r}^*$ denotes the task space target and we chose $\beta_{ts} = 0.1$. For each trajectory the task space target was drawn uniform randomly, i.e., $\mathbf{r}^* \sim U[-2, 2]$. The task, therefore, is to move with fixed velocity to the target point $\mathbf{r}^*$ along the direction given by $\hat{\boldsymbol{\alpha}}$.

Under this set up, we collected data under $K = 2$ different task constraints, where, for each constraint, we collected $40$ trajectories from random start states, each of length $40$ steps (in total $N = 1600$ data points per task constraint), reserving $10\%$ of the total data set as unseen test data.

For the learning, we used a parametric policy representation (see Sec. III-A) consisting of $6 \times 6$ grid of Gaussian radial basis functions arranged around the maximum extents of the data. The widths $\sigma^2$ were fixed to give suitable overlap between basis functions. For comparison, we also tried learning with the direct regression approach, whereby we directly trained on the raw observations $\mathbf{u}$ by minimising (4), using the same parametric policy model. We repeated this experiment for $50$ data sets and evaluated (i) the normalised mean-squared error (nMSE) in the estimation of the nullspace component of the data $E_{ns}$, (ii) the normalised error according to the proposed objective function $E_1$, (iii) the normalised constrained policy error (nCPE), and (iv) the normalised unconstrained policy error (nUPE) [7] on the test data. The latter two measure the difference between the estimated policy $\tilde{\boldsymbol{\pi}}$ and that of the true underlying policy $\boldsymbol{\pi}$ either when subject to the same constraints as in the data, or when fully unconstrained [7], and as such, give an estimate as to how well the policy will generalise to new, unseen constraints. We also repeated the experiment for two additional nullspace policies with differing functional forms, namely,

1) a sinusoidal policy: $\boldsymbol{\pi}(\mathbf{x}) = \nabla_{\mathbf{x}}\phi(\mathbf{x})$ where $\phi(\mathbf{x}) = -\beta \sin(x_1)\cos(x_2)$ and $\beta = 0.1$;
2) a limit cycle policy: $\dot{r} = r(\rho^2 - r^2)$, $\dot{\theta} = \omega$ with radius $\rho^2 = 2$, angular velocity $\omega = -2\ rad\ s^{-1}$, where $x_1 = \beta r \cos\theta$, $x_2 = \beta r \sin\theta$ and $\beta = 0.01$.

Tables I & II show the results averaged over 50 trials for each experiment.

The scores of $E_{ns}$ in Table I tell us that the estimation of the nullspace component ${}^{ns}\mathbf{u}$ using the proposed approach is orders of magnitude better than using the naive method, a fact confirmed the corresponding low scores for $E_1$. Looking at Table II we see that this also translates to low error in estimating the underlying underlying policy $\boldsymbol{\pi}$, as evinced by the very low values for the nUPE and nCPE. (again, orders of magnitude lower in error compared to the direct regression approach).

Comparing the figures for the three different policy types, we also see that increasing complexity of the policy results in a harder learning problem: compare the error figures for the linear policy to those of the of the limit cycle and sinusoidal policies.

Finally, we note that in all cases the nCPE was at least one order of magnitude better for the policies learnt with the proposed method as compared to those learnt with direct regression. This supports the view that, even if we cannot exactly reconstruct the original nullspace policy, we can at least obtain a single policy which matches the nullspace component under the observed constraints.

To further characterise the performance, we also looked at the effect of varying levels of noise and amounts of training

| Policy | Method | $E_{ns}$ | $E_1$ |
|---|---|---|---|
| Linear | Direct | $0.40617 \pm 0.28809$ | $0.43799 \pm 0.26530$ |
| | Novel | $0.00042 \pm 0.00188$ | $0.00031 \pm 0.00233$ |
| Sinusoidal | Direct | $0.60510 \pm 0.82434$ | $0.72154 \pm 0.40734$ |
| | Novel | $0.00822 \pm 0.02430$ | $0.00343 \pm 0.01839$ |
| Limit cycle | Direct | $1.31894 \pm 1.02806$ | $3.85736 \pm 1.76578$ |
| | Novel | $0.01590 \pm 0.04186$ | $0.01290 \pm 0.05013$ |

TABLE I

NORMALISED ERROR IN PREDICTING THE NULLSPACE COMPONENT OF MOTION (STEP 1). RESULTS ARE (MEAN±S.D.) OVER 100 TRIALS (50 TRIALS × 2 CONSTRAINTS).

| Policy | Method | nUPE | nCPE |
|---|---|---|---|
| Linear | Direct | $0.82792 \pm 0.05979$ | $0.02212 \pm 0.01746$ |
| | Novel | $0.00384 \pm 0.01499$ | $0.00003 \pm 0.00006$ |
| Sinusoidal | Direct | $0.84798 \pm 0.16709$ | $0.04465 \pm 0.04334$ |
| | Novel | $0.13302 \pm 0.15719$ | $0.00287 \pm 0.00266$ |
| Limit cycle | Direct | $0.78840 \pm 0.25528$ | $0.04080 \pm 0.04512$ |
| | Novel | $0.14135 \pm 0.23641$ | $0.00386 \pm 0.00606$ |

TABLE II

NORMALISED ERROR IN PREDICTING THE UNDERLYING POLICY (STEPS 1 AND 2). RESULTS ARE (MEAN±S.D.) OVER 50 TRIALS.

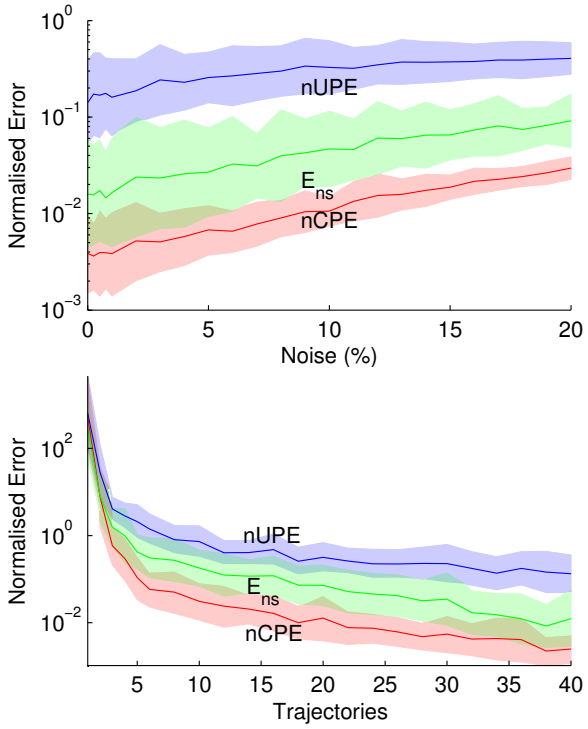



Fig. 4. Top: Normalised UPE, CPE and $E_{ns}$ for increasing noise levels in the observed $\mathbf{x}_n$ and $\mathbf{u}_n$ for the limit cycle nullspace policy. Bottom: Normalised UPE, CPE and $E_{ns}$ versus data set size as the number of training trajectories increases.

data for the limit cycle policy. Fig. 4(top) shows how the nUPE, nCPE and normalised nullspace error $E_{ns}$ error vary with increasing levels of noise in the observed state $\mathbf{x}_n$ and commands $\mathbf{u}_n$, up to 20% of the scale of the data. Fig. 4(bottom) shows how these errors vary with different numbers of input trajectories. As can be seen, our method shows a gradual decrease is error as the amount of data increases and an increase in the error as noise increases.

### *B. Three Link Arm*

The goal of our next set of experiments was to evaluate the efficacy of the method on a more realistic problem, and test our ability to generalise across different tasks defined in different spaces. For this, we chose to investigate a scenario in which we wish to learn the redundancy resolution for a kinematically controlled planar three link arm.

The set up was as follows: The state and action spaces of the arm were described by the joint angles $\mathbf{x} = \mathbf{q} \in \mathbb{R}^3$ and velocities $\mathbf{u} = \dot{\mathbf{q}} \in \mathbb{R}^3$, respectively. For ease of comparison with the toy example, we used the same linear policy (14) for redundancy resolution, this time with $\beta = 1$ and with attractor point $\mathbf{x}_0 = (10°, -10°, 10°)^T$. This point was chosen as a safe, default posture, away from singularities and joint limits.

Under this redundancy resolution regime, we collected data from the arm as it performed different tasks in several task spaces. Specifically, tasks were defined in (i) the 2-D space describing the Cartesian position of the end-effector (i.e., $\mathbf{r} = (x, y)^T$); (ii) the space defined by the $x$-coordinate and orientation of the end-effector (i.e., $\mathbf{r} = (x, \theta)^T$), and; (iii) the space defined by the $y$-coordinate and end-effector orientation (i.e., $\mathbf{r} = (y, \theta)^T$).

Within each of these spaces the arm was controlled to randomly perform different tasks. Specifically, in each space the arm followed a linear policy (16) (this time with $\beta = 1$) to track to randomly selected targets. The latter were drawn uniformly for each trajectory from $x^* \sim U[-1, 1]$, $y^* \sim U[0, 2]$ and $\theta^* \sim U[0°, 180°]$. These values were chosen to limit the arm to approximately the top half of the workspace with $y > 0$. Only targets with a valid inverse kinematics solution were considered.

For each task space, 40 trajectories each of length 40 steps were generated at a sampling rate of 50Hz. The start states for each trajectory were drawn from the uniform distribution $q_1 \sim U[0°, 10°]$, $q_2 \sim U[90°, 100°]$, $q_3 \sim U[0°, 10°]$. Of the total data, 10% was reserved as unseen test data.

For the learning, we used parametric models (see Sec. III-A) consisting of 100 Gaussian RBFs with centres chosen according to k-means and with widths taken as the mean of the distances between centres. We trained the same parametric model (i) with the proposed approach, and, for comparison; (ii) with direct regression on the raw observations $\mathbf{u}$. We repeated this experiment for 50 trials, and evaluated the normalised error in terms of the four metrics: (6), (8), the UPE, and the CPE.

Table III shows the results for step 1 and Table IV shows the results for the whole process. Table III shows that we can learn a very good approximation of the nullspace component ${}^{ns}\mathbf{u}$ and that minimizing $E_1$ again tends to minimize $E_{ns}$. Table IV shows we can learn the nullspace policy far better than the direct method and obtain a reasonable estimate of $\boldsymbol{\pi}$. The very low nCPE again demonstrates that if the same constraints as those observed are applied to our single learnt policy, then the output closely matches the constrained true nullspace policy.

Fig. 5(a) and 5(b) show an example of using the learnt nullspace policy with known $\mathbf{A}$ and $\mathbf{b}$ in (2) to generate a new trajectory. The task space matches one of the observed task spaces and was to control the $x, y$ position of the end effector and move to the point $(1.5, 1)^T$. A novel start position of $(90°, 45°, -20°)$ was used. The experiment was run for 200 time steps to demonstrate convergence to the

| Cstr. | Method | $E_{ns}$ | $E_1$ |
|---|---|---|---|
| $x, y$ | Direct | $33.93435 \pm 4.70679$ | $2.0504 \quad \pm 0.5077$ |
| | Novel | $0.00037 \pm 0.00136$ | $7.3647\times10^{-12} \quad \pm 1.2412 \times10^{-11}$ |
| $x, \theta$ | Direct | $17.84229 \pm 3.36997$ | $0.4301 \quad \pm 0.1635$ |
| | Novel | $0.00010 \pm 0.00020$ | $2.1258 \times10^{-10} \quad \pm 5.7595 \times10^{-10}$ |
| $y, \theta$ | Direct | $28.69063 \pm 3.44916$ | $0.7642 \quad \pm 0.2130$ |
| | Novel | $0.00118 \pm 0.00133$ | $3.0107 \times10^{-9} \quad \pm 3.1828 \times10^{-9}$ |

TABLE III

NORMALISED ERROR IN PREDICTING THE NULLSPACE COMPONENT OF MOTION (STEP 1). RESULTS ARE (MEAN±S.D.) OVER 50 TRIALS FOR EACH OF THE 3 CONSTRAINTS.

| Method | nUPE | nCPE |
|---|---|---|
| Direct | $20.85327 \pm 4.81346$ | $0.31210 \pm 0.06641$ |
| Novel | $0.36199 \pm 0.84707$ | $0.00017 \pm 0.00025$ |

TABLE IV

NORMALISED ERROR IN PREDICTING THE UNDERLYING POLICY (STEPS 1 AND 2). RESULTS ARE (MEAN±S.D.) OVER 50 TRIALS.

| Constr. | Direct | Novel |
|---|---|---|
| $x$ | $12.62812 \pm 3.55790$ | $0.13917 \pm 0.39708$ |
| $y$ | $6.87882 \pm 4.22021$ | $0.15620 \pm 0.32314$ |
| $\theta$ | $10.19341 \pm 3.46767$ | $0.12200 \pm 0.33360$ |

TABLE V

NORMALISED ERROR IN PREDICTING THE POLICY IN THE NULLSPACE OF UNSEEN TASK CONSTRAINTS. RESULTS ARE (MEAN±S.D.) OVER 50 TRIALS.

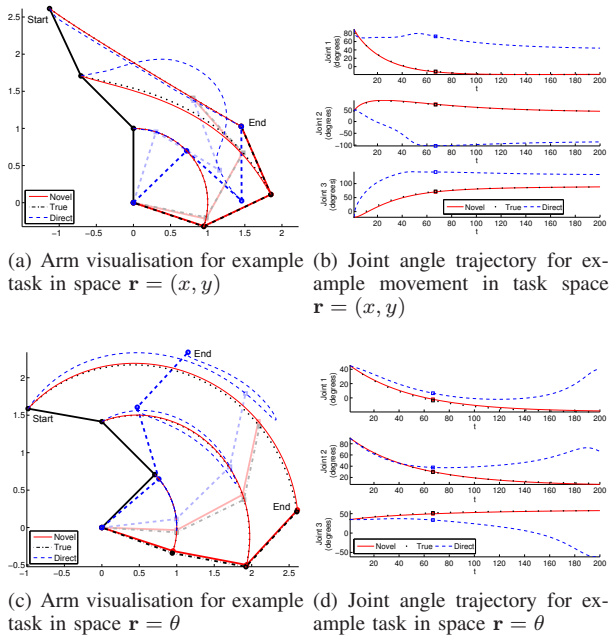
target. The true nullspace policy and the policy learnt using the naive direct method are shown as a comparison. It can be seen that the novel method follows the true joint trajectories extremely well. The direct method, although forced to finish at the correct task space target, ends up with quite different joint angles.

In Fig. 5(c) and 5(d), the task space is one that has not been seen in the training data. It is to control the orientation of the end effector only in order to move to an angle of $45°$. The learnt nullspace policy generalises well, matching the true joint trajectories closely - resolving the redundancy in the correct way. The direct method again arrives at a quite different set of joint angles. Table V shows the nCPE when the learnt policies are constrained by unseen task spaces i.e. how well we can predict ${}^{ns}\mathbf{u}$ under the indicated constraints. The low score shows that as in Fig. 5(c) and 5(d), the learnt nullspace policy can be expected to generalise well to new task spaces.

### C. Kuka Lightweight Arm

The goal of our final set of experiments was to characterise how well the algorithm scales to higher dimensional problems, with more complex, realistic constraints. For this, we used a kinematic simulation of the 7-DOF Kuka lightweight robot (LWR-III) Fig. 6.

The experimental procedure was as follows. We generated a random initial posture by drawing 7 joint angles uniformly around a default start posture $\mathbf{q}_0$ in the range of $\pm 0.4\,rad$, that is $q_i \sim q_{0,i} + U[-0.4; 0.4]\,rad$. We then selected 4 different spaces in which different tasks were performed, denoted here as $(x, y, z)$, $(x, y, \theta_x)$, $(x, z, \theta_x)$, and $(y, z, \theta_x)$. Here, the letters denote which end-effector coordinates were controlled as the task space, that is, $(x, y, z)$ means the task was defined in end-effector position coordinates, but




(a) Arm visualisation for example task in space $\mathbf{r} = (x, y)$

(b) Joint angle trajectory for example movement in task space $\mathbf{r} = (x, y)$




(c) Arm visualisation for example task in space $\mathbf{r} = \theta$

(d) Joint angle trajectory for example task in space $\mathbf{r} = \theta$

Fig. 5. Using the learnt policies to resolve redundancy on the three link arm. Top: Redundancy resolution under a task constraint seen in the data ($\mathbf{r} = (x, y)$). The task is to move the end-effector with linear velocity to the point $\mathbf{r}^* = (1.5, 1)$. Bottom: Redundancy resolution under a task constraint previously unseen in the data ($\mathbf{r} = \theta$). The task is to move to the target orientation $\mathbf{r}^* = 45°$. Movement according to the ground truth policy (black) and the policies learnt with the proposed approach (red) and direct regression (blue) are shown. The opaque arms show midpoints along the trajectories (marked as squares on the joint angle profiles).

ignored the orientation. Similarly, $(x, y, \theta_x)$ means the task was defined in the $x$- and $y$-coordinates and the orientation around the $x$-axis, while leaving the $z$-position and orientation around the $y$- and $z$-axes unconstrained.

Within these different spaces, we then set up a closed loop inverse kinematics policy for tracking to a variety of targets at different speeds. Specifically, we used the linear attractor policy (16), this time with targets chosen according to $x \sim U[.25, .75]$, $y \sim U[-.5, 0]$, $z \sim U[0, .5]$ and $\theta_z \sim \theta_{z,0} + U[0, \frac{\pi}{3}]$ (where $\theta_{z,0}$ is the $z$-angle of the end-effector at joint position $\mathbf{q}_0$). To increase the variation in the task-directed movements we also varied the speed of task space movement by drawing the scaling parameter from $\beta_{ts} \sim U[0, .04]$. Depending on which of the 4 spaces is used, this policy corresponds to qualitatively different task-oriented behaviours. For example, in the $(x, y, z)$ task space (i.e., end-effector positions) the behaviour is similar to reaching to a target. In the $(x, y, \theta_x)$ space, the behaviour is more like a pouring behaviour (tracking to a desired orientation at a point in the horizontal plane).

For resolving redundancy, we used a non-linear joint limit avoidance type policy as $\boldsymbol{\pi}(\mathbf{x}) = -\alpha \nabla \Phi(\mathbf{x})$, with the potential given by $\Phi(\mathbf{x}) = \sum_{i=1}^{7} |x_i|^p$ for $p = 1.5$ and $p = 1.8$. We then generated 250 trajectories with 40 points each, following the combined task and nullspace policies for the 4 different task constraints.

For learning in the 7-D state space, we selected locally linear models as described in Sec. III-B, where we used receptive fields of fixed width ($\sigma^2 = .25$) 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

| Policy | Constr. | Novel | Direct |
|---|---|---|---|
| $p=2.0$ | $x$ - $y$ - $z$ | $0.175 \pm 0.021$ | $0.400 \pm 0.083$ |
| | $x$ - $y$ - $\theta_x$ | $0.313 \pm 0.022$ | $0.457 \pm 0.048$ |
| | $x$ - $z$ - $\theta_x$ | $0.318 \pm 0.037$ | $0.467 \pm 0.059$ |
| | $y$ - $z$ - $\theta_x$ | $0.133 \pm 0.023$ | $0.263 \pm 0.048$ |
| $p=1.8$ | $x$ - $y$ - $z$ | $0.200 \pm 0.020$ | $0.361 \pm 0.069$ |
| | $x$ - $y$ - $\theta_x$ | $0.317 \pm 0.021$ | $0.426 \pm 0.040$ |
| | $x$ - $z$ - $\theta_x$ | $0.322 \pm 0.030$ | $0.422 \pm 0.049$ |
| | $y$ - $z$ - $\theta_x$ | $0.161 \pm 0.020$ | $0.250 \pm 0.040$ |
| $p=1.5$ | $x$ - $y$ - $z$ | $0.294 \pm 0.013$ | $0.381 \pm 0.048$ |
| | $x$ - $y$ - $\theta_x$ | $0.393 \pm 0.022$ | $0.452 \pm 0.033$ |
| | $x$ - $z$ - $\theta_x$ | $0.422 \pm 0.030$ | $0.448 \pm 0.035$ |
| | $y$ - $z$ - $\theta_x$ | $0.318 \pm 0.021$ | $0.352 \pm 0.031$ |

TABLE VI

NORMALISED ERROR IN PREDICTING THE NULLSPACE PART OF MOTION $E_{ns}$, UNDER TASK CONSTRAINTS IN DIFFERENT SPACES FOR JOINT LIMIT AVOIDANCE POLICIES ON THE KUKA LWR-III. RESULTS ARE MEAN±S.D. OVER 20 TRIALS.

| Policy | Method | nUPE | nCPE |
|---|---|---|---|
| $p=2.0$ | Novel | $0.732 \pm 0.049$ | $0.097 \pm 0.024$ |
| | Direct | $1.008 \pm 0.027$ | $0.090 \pm 0.007$ |
| $p=1.8$ | Novel | $0.755 \pm 0.038$ | $0.091 \pm 0.018$ |
| | Direct | $1.001 \pm 0.024$ | $0.087 \pm 0.005$ |
| $p=1.5$ | Novel | $0.870 \pm 0.043$ | $0.097 \pm 0.009$ |
| | Direct | $1.006 \pm 0.019$ | $0.093 \pm 0.003$ |

TABLE VII

NORMALISED ERROR IN PREDICTING THE UNDERLYING POLICY (STEPS 1 AND 2) FOR JOINT LIMIT AVOIDANCE POLICIES ON THE KUKA LWR-III. RESULTS ARE MEAN±S.D. OVER 20 TRIALS.

Fig. 6. The redundant Kuka Lightweight Arm. A simulated version was used for the experiments in section IV-C.

field with $w_m(\mathbf{x}_n) \geq 0.7$. On average, this yielded between 30-60 local models. For all constraint types (task spaces), we estimated the policy from a training subset, and evaluated it on test data from the same constraint. The results are enumerated in Tables VI & VII.

Looking at Table VI we see that in all cases the proposed approach performed better than direct regression for learning the nullspace component of motion ${}^{ns}\mathbf{u}$. Comparing errors for the different policies, we see that the learning problem became increasingly harder for the more non-linear policies ($p = 1.8$ and $p = 1.5$). We also note that the performance was also affected by the different constraint types (with the $(x, y, \theta_x)$ and $(x, z, \theta_x)$ task spaces presenting the harder learning problems). These trends are reflected in terms of the performance of Step 2 (ref. Table VII), however, we note that in all cases the proposed approach achieved lower nUPE that the direct approach, indicating that policies learnt with this approach have better generalisation across different task constraints.

## V. CONCLUSION

In this work, we introduced a novel technique for learning redundancy resolution (nullspace) policies from demonstrations. We assume that demonstrations are generated with a consistent redundancy resolution strategy and that this acts within the nullspace of the task. Importantly, no knowledge of the task space or task policy is required. In experiments with three simulated plants, we demonstrated that our method learns better nullspace policy estimates compared to standard DPL on the raw observations. A key benefit is that the single learnt nullspace policy can be used to resolve redundancy with any tasks in the observed task spaces, and will often generalise to new task spaces.

In future work we aim to demonstrate the method on human data where no ground truth nullspace policy is known. Currently we learn velocity based nullspace policies. Exactly the same framework can be used for acceleration and force based policies. We aim to examine performance in this case. We also aim to investigate an alternative benefit of learning the nullspace decomposition whereby the nullspace component is removed from observations in order to learn policies for the task space component.

## REFERENCES


[1] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Trans. Systems Man and Cybernetics Part B*, 37(2):286, 2007.

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

[3] H. Cruse, E. Wischmeyer, M. Bruwer, P. Brockfeld, and A. Dress. On the cost functions for the control of the human arm movement. *Biological Cybernetics*, 62(6):519–528, 1990.

[4] N. Delson and H. West. Robot programming by human demonstration: adaptation and inconsistency in constrained motion. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1996.

[5] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *Proc. 15th IEEE Int. Symp. Robot and Human Interactive Communication ROMAN 2006*, pages 358–363, Sept. 2006.

[6] M. Hersch, F. Guenter, S. Calinon, and A. G. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Trans. Robotics*, 24(6):1463–1467, 2008.

[7] M. Howard, S. Klanke, M. Gienger, C. Goerick, and S. Vijayakumar. A novel method for learning policies from variable constraint data. *Autonomous Robots*, 27(2):105–121, 2009.

[8] M. Howard, S. Klanke, M. Gienger, C. Goerick, and S. Vijayakumar. A novel method for learning policies from constrained motion. In *IEEE Int. Conf. Robotics and Automation*, 2009.

[9] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robotics Research*, 5(1):90, 1986.

[10] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Systems Man and Cybernetics*, 7:868–871, 1977.

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

[12] J. Peters and S. Schaal. Learning to control in operational space. *Int. J. Robotics Research*, 27:197–212, 2008.

[13] S. Schaal and C. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084, 1998.

[14] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philos Trans R Soc Lond B Biol Sci*, 358(1431):537–547, Mar 2003.

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

[16] F. Udwadia and R. Kalaba. *Analytical Dynamics: A New Approach*. Cambridge University Press, 2008.

[17] T. Yoshikawa. Manipulability of robotic mechanisms. *Int. J. Robotics Research*, 4:3–9, 1985.