Learning Constrained Generalizable Policies by Demonstration

Leopoldo Armesto*, João Moura†, Vladimir Ivan‡, Antonio Sala§, Sethu Vijayakumar‡
*Instituto de Diseño y Fabricación, Universitat Politècnica de València, C/Camino de Vera s/n, 46019, Valencia, Spain
†Institute of Sensors, Signals and Systems, Heriot-Watt University, Mountbatten Building, EH14 4AS, Edinburgh, UK
‡Institute of Action, Perception, and Behaviour, University of Edinburgh, Crichton Street 10, EH8 9AB, Edinburgh, UK
§Instituto U. de Automática e Inf. Industrial, Universitat Politècnica de València, C/Camino de Vera s/n, 46019, Valencia, Spain

***Abstract*—Many practical tasks in robotic systems, such as cleaning windows, writing or grasping, are inherently constrained. Learning policies subject to constraints is a challenging problem. We propose a *locally weighted constrained projection learning* method (LWCPL) that first estimates the constraint and then exploits this estimate across multiple observations of the constrained motion to learn an unconstrained policy. The generalization is achieved by projecting the unconstrained policy onto a new, previously unseen, constraint. We do not require any prior knowledge about the task or the policy, so we can use generic regressors to model the task and the policy. However, any prior beliefs about the structure of the motion can be expressed by choosing task-specific regressors. In particular, we can use robot kinematics and motion priors to improve the accuracy. Our evaluation results show that LWCPL outperform the state of the art method in accuracy of learning the constraints as well as the unconstrained policy, even in noisy conditions. We have validated our method by learning a wiping task from human demonstration on flat surfaces and reproducing it on an unknown curved surface using a force/torque based controller to achieve tool alignment. We show that, despite of the differences between the training and validation scenarios, we learn a policy that still provides the desired wiping motion.**

I. INTRODUCTION

When performing a given task in an unfamiliar environment, humans easily adapt the skills or previously learned motions to novel situations and environments. For instance, the operator in Fig. 1 wipes the front panels of the train by employing a small set of motions and skills that generalize to different train geometries and positioning [16]. However, current robotic systems often require computationally expensive replanning and precise scans of the new environment to reproduce a learnt task. In addition to this, movement in complex, high degree of freedom manipulation systems often contains a high level of redundancy. The degrees of freedom available to perform a task are usually higher than what is necessary to execute that task. This allows certain flexibility in finding an appropriate solution, so that this redundancy may be resolved according to some strategy that achieves a secondary objective, while the primary task is not affected. Such approaches to redundancy resolution are employed by humans [4] as well as other redundant systems such as (humanoid) robots [6].

The redundancy resolution may be also interpreted as a form of hierarchical task decomposition, in which the complete space of available movement is split up into a task space component and a null space component. For instance, one might consider a primary task, such as reaching or trajectory tracking, and a lower-priority task to be a secondary goal such as avoiding joint limits [8], self-collisions [20], or kinematic singularities [22]. This notion is particularly evident when considering motions modulated by external or environmental constraints. For instance, in the wiping task of Fig. 1, the tool is constrained by the window surface; the primary task is to keep the tool aligned and in contact and the secondary task is to provide surface coverage while maintaining comfortable arm position. Several variants of this hierarchical approach to redundancy have been used in robotics [13]. This core concept has been applied to task sequencing [15], task prioritisation [3] and hierarchical quadratic programming [7][9]. These methods attempt to minimize a cost function subject to explicitly formulated constraints. However, they suffer from the curse of dimensionality and are typically unsuitable for real time applications in high dimensions.

Fig. 1. Manual cleaning of an electric train. Willesden depot, London (2016)

To circumvent this problem, one might attempt to learn a movement policy, a mapping from states to actions, that encodes behaviour consistent with the set of constraints, instead of continuously calculating constraint-consistent actions. This mapping can be learned from data captured during demonstrations, consisting of human or robot motions. This

approach falls under the category of imitation learning [1], and one straightforward way to learn behaviours from this is through direct policy learning (DPL) [18]. However, DPL suffers from poor generalisation [1] under varying constraints.

A recently developed method for learning both the null space policy [21] and the constraints [14] significantly outperforms direct policy learning methods, but it imposes strict assumptions on the nature of the data used for learning. So far, its effectiveness has been demonstrated mostly in low-dimensional scenarios with relatively simple (linear) movement policies.

A. Contribution

One of the main difficulties of learning from demonstration arises because raw observations contain policies acting on multiple levels of unknown task hierarchy. To decompose this hierarchy into subtasks, we present a closed-form solution to [14], which outperforms the state of the art techniques in both computation speed and accuracy. In addition to this, we split the raw observation into task and null-space components in a more efficient way than the method proposed in [21]. This improvement allows us to estimate null-space projection matrices from data of different tasks which can be used for learning an unconstrained policy by observing multiple projections of such policy at the same configuration [10].

In our experiments, we provide demonstrations of a wiping motion on flat surfaces and test the learnt policy on curved surfaces, achieving generalization to previously unseen environments. Additionally, we define a surface alignment task using a force sensor which allows us to perform wiping on curved surfaces. This resulting constraint is different from the alignment constraint used for training on the flat surfaces, showing generalization to an entirely different task with a different null-space.

II. Constraint matrix and task learning

Let's assume that there's an unconstrained *ideal* controller $\mathbf{u}^*(\mathbf{x}) := \boldsymbol{\pi}(\mathbf{x})$ and that a dynamic system must be operated under some task that imposes a set of constraints:

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

where $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{s\times q}$ ($s < q$) is a Pfaffian constraint matrix and $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^s$ is known as the task policy. Unfortunately, $\mathbf{u}(\mathbf{x}) \neq \mathbf{u}^*(\mathbf{x})$ because, by assumption, the system is constrained.

This implies that the actions can be decomposed into a task-space component, ${}^{ts}\mathbf{u}(\mathbf{x}) \in \mathbb{R}^q$ which satisfies the constraint, and a null-space component, ${}^{ns}\mathbf{u}(\mathbf{x}) \in \mathbb{R}^q$:

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

$${}^{ts}\mathbf{u}(\mathbf{x}) := \mathbf{A}(\mathbf{x})^\dagger\mathbf{b}(\mathbf{x}) \tag{3}$$

$${}^{ns}\mathbf{u}(\mathbf{x}) := \mathbf{N}(\mathbf{x})\boldsymbol{\pi}(\mathbf{x}). \tag{4}$$

where $\mathbf{N}(\mathbf{x}) \in \mathbb{R}^{q\times q}$ is a null-space projection matrix computed as:

$$\mathbf{N}(\mathbf{x}) := \mathbf{I} - \mathbf{A}(\mathbf{x})^\dagger\mathbf{A}(\mathbf{x}), \tag{5}$$

where $\dagger$ denotes Moore-Penrose pseudo-inverse. Note that by definition ${}^{ts}\mathbf{u}(\mathbf{x}) \perp {}^{ns}\mathbf{u}(\mathbf{x})$.

A. Problem Statement

We assume that $\mathbf{A}(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$ are unknown, but they can be expressed as linearly dependent on some regressors. We obtained the motion data through a set of demonstrations. In this case, the direct policy learning (DPL) approach would be to learn a static policy (mapping) from states $\mathbf{x}$ to actions $\mathbf{u}$. However, since the actions of the system are subject to some constrained motion, the DPL is not valid.

Learning the policy $\boldsymbol{\pi}(\mathbf{x})$ from (2) can be a difficult task when the constraint defined by $\mathbf{A}(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$ is unknown. However, if $\mathbf{A}(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$ are first estimated, it is a relatively straight-forward problem, particularly if the policy can be described as linearly dependent of some parameter vector. In that case the problem could be solved using least-squares. Thus, it is reasonable to divide the overall problem into two sequential steps: 1) estimate $\tilde{\mathbf{A}}(\mathbf{x})$ and $\tilde{\mathbf{b}}(\mathbf{x})$ for each task; 2) estimate $\tilde{\boldsymbol{\pi}}(\mathbf{x})$, combining all data and estimated constrains and tasks. Note that symbol $\tilde{}$ denotes estimation.

A given dataset contains a set of ν sub-datasets $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_\nu\}$. Each sub-set contains a different task/constraint with samples of pairs of raw observations $\mathcal{X}_i = \{(\mathbf{x}_{1,i}, \mathbf{u}_{1,i}), \ldots, (\mathbf{x}_{N,i}, \mathbf{u}_{N,i})\}$, where N is the number of observations for each task/constraint. However, the unconstrained policy $\boldsymbol{\pi}(\mathbf{x})$ is the same for all provided demonstrations by assumption, so that we will be able to observe enough projections of such policy to reconstruct it [10, Theorem 3.1].

In [14], the authors presented a method for estimating the null-space projection matrix. The main drawback of that approach is that the estimation is performed by solving a non-convex optimization problem using a spherical representation. This often leads to long computation times and decreased performance. In this paper we present a closed-form solution of this problem.

B. Closed-form constraint estimation

In this section, we define a new method for estimating the constraint matrix, allowing us to estimate the null-space projection matrix, which will be used to split the action observations into the task-space and null-space components.

If $\mathbf{A}(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$ can be expressed as a linear combination of regressors, they could be defined as:

$$\mathbf{A}(\mathbf{x}) := \mathbf{W}_A\boldsymbol{\Phi}_A(\mathbf{x}) \tag{6}$$

$$\mathbf{b}(\mathbf{x}) := \mathbf{W}_b\boldsymbol{\Phi}_b(\mathbf{x}), \tag{7}$$

where $\mathbf{W}_A \in \mathbb{R}^{s\times w_A}$ and $\mathbf{W}_b \in \mathbb{R}^{s\times w_b}$ are the constant matrices related to a set of parameters to be learned and $\boldsymbol{\Phi}_A(\mathbf{x}) \in \mathbb{R}^{w_A\times p}$ and $\boldsymbol{\Phi}_b(\mathbf{x}) \in \mathbb{R}^{w_b}$ are some regressors assumed to be known. [1]

From (6) and (7), we can express (1) in compact form:

$$\mathbf{W}\mathbf{H}(\mathbf{x}, \mathbf{u}) = \mathbf{0}, \tag{8}$$

[1] s is the number of constraints (rank(A)), w_A and w_b are the dimensionality of each regressor function respectively, and p is the dimensionality of the control space $u \in \mathbb{R}^p$.

with,

$$\mathbf{W} := \begin{bmatrix}\mathbf{W}_A & \mathbf{W}_b\end{bmatrix} \in \mathbb{R}^{s\times w} \tag{9}$$

$$\mathbf{H}(\mathbf{x},\mathbf{u}) := \begin{bmatrix}\boldsymbol{\Phi}_A(\mathbf{x})\mathbf{u} \\ -\boldsymbol{\Phi}_b(\mathbf{x})\end{bmatrix} \in \mathbb{R}^{w}, \tag{10}$$

being $w = w_A + w_b$.

Expression (10) can be evaluated for any data-point in the sub-dataset. Thus, by substituting $\mathbf{x}$ and $\mathbf{u}$ with the values from the dataset into (10) we can construct a matrix specific for each task. The columns of this matrix will then correspond to each data-point as follows:

$$\mathcal{H} := \begin{bmatrix}\mathbf{H}(\mathbf{x}_1,\mathbf{u}_1) & \mathbf{H}(\mathbf{x}_2,\mathbf{u}_2) & \dots & \mathbf{H}(\mathbf{x}_N,\mathbf{u}_N)\end{bmatrix}, \tag{11}$$

where $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N$ are the raw observations of the action (all belonging to the same sub-dataset).

The singular value decomposition [5] of $\mathcal{H} \in \mathbb{R}^{w\times N}$ will provide the estimation $\tilde{\mathbf{W}}$. The estimated parameters can be derived from the SVD decomposition:

$$\mathcal{H} = \mathcal{U}\mathcal{S}\mathcal{V}^\top, \tag{12}$$

where $\mathcal{S} \in \mathbb{R}^{w\times N}$ is a diagonal matrix with non-negative elements corresponding to the singular values of $\mathcal{H}$ in decreasing order. $\mathcal{U} \in \mathbb{R}^{w\times w}$ and $\mathcal{V} \in \mathbb{R}^{N\times N}$ are unitary matrices containing the left and right singular vectors of $\mathcal{H}$. The rank of $\mathcal{H}$ can be inferred from the analysis of the singular values by setting a threshold. If $\mathrm{rank}(\mathcal{H}) > w - s$, then it might be related with an under-parametrization of $\mathbf{A}(\mathbf{x})$ and/or $\mathbf{b}(\mathbf{x})$, or incorrect selection of regressors. On the other hand, if $\mathrm{rank}(\mathcal{H}) < w - s$, then it might be caused by over-parametrization or data which is not rich enough. The ideal scenario is when there are s singular values close to zero while the reminder of singular values are large. In that case, the last s rows of $\mathcal{U}^\top$, denoted as $\underset{\mathbf{w}_s}{\mathrm{rows}}\, \mathcal{U}^\top$ with $\mathbf{w}_s = \{w-s+1, \dots, w\}$, correspond to the singular vectors associated with smallest singular values and are, therefore, the closed-form estimate of (8):

$$\tilde{\mathbf{W}} = \begin{bmatrix}\tilde{\mathbf{W}}_A & \tilde{\mathbf{W}}_b\end{bmatrix} := \underset{\mathbf{w}_s}{\mathrm{rows}}\, \mathcal{U}^\top. \tag{13}$$

When we evaluate equation (13) using each sub-dataset, it is straight forward to obtain estimated values for the constraint matrix $\tilde{\mathbf{A}}(\mathbf{x})$, the task $\tilde{\mathbf{b}}(\mathbf{x})$ and null-space projection matrix $\tilde{\mathbf{N}}(\mathbf{x})$, by using equations (5), (6), and (7).

One of the key differences between our approach and the one presented in [14] is that in this paper we propose the minimization of the error in the task-space, while in [14] the error was defined in the null-space. Secondly, in [14] the assumption of having access to the null-space was imposed, while here we can deal with data containing both task and null-space components. We do not present the complete comparison between these two approaches here due to the lack of space, but our preliminary results have shown an improvement over $\approx 10^5$ in computational time and $\approx 10^{14}$ in accuracy when using the closed-form method.

C. Component split

In [21], a method for learning from raw observations containing task and null-space components was proposed. The authors proposed the estimation of a null-space policy (i.e.: the policy projected over the null-space of a task, ${}^{ns}\mathbf{u}(\mathbf{x})$), however this leads to an inaccurate non-convex optimization procedure. In this paper, we use the estimation of $\tilde{\mathbf{N}}(\mathbf{x})$ to compute ${}^{ns}\mathbf{u}(\mathbf{x})$.

Corollary 2.1: For any arbitrary action we can split its task-space and null-space components by using the null-space projection matrix.

For any observed action $\mathbf{u}$ projected over the null-space of the task, the null-space component of such observation is:

$$\mathbf{N}(\mathbf{x})\mathbf{u} = \mathbf{N}(\mathbf{x})({}^{ts}\mathbf{u} + {}^{ns}\mathbf{u}) = \mathbf{N}(\mathbf{x}){}^{ns}\mathbf{u} = {}^{ns}\mathbf{u}, \tag{14}$$

since $\mathbf{N}(\mathbf{x}){}^{ts}\mathbf{u} = \mathbf{0}$ by definition and $\mathbf{N}(\mathbf{x})$ is a projector over the null-space.

Therefore, each data-point in the dataset, can be split into its null-space and task-space components as:

$${}^{ns}\tilde{\mathbf{u}}_n = \tilde{\mathbf{N}}(\mathbf{x}_n)\mathbf{u}_n \tag{15}$$

$${}^{ts}\tilde{\mathbf{u}}_n = \mathbf{u}_n - {}^{ns}\tilde{\mathbf{u}}_n. \tag{16}$$

III. Learning the Unconstrained Policy

In this section, we propose a *constrained policy learning* (CPL) method that combines the projection matrix learning method proposed in the previous section with unconstrained policy learning method. We first individually estimate a projection matrix $\tilde{\mathbf{N}}_i(\mathbf{x})$ for each constraint separately using just the sub-dataset $\mathcal{X}_i$ as proposed in section II-A. Secondly, we estimate the unconstrained policy, using all estimated null-space projection matrices and the whole dataset:

$$E_{cpl}[\tilde{\boldsymbol{\pi}}] = \sum_{i=1}^{\nu}\sum_{n=1}^{N}\left\| {}^{ns}\tilde{\mathbf{u}}_{n,i} - \tilde{\mathbf{N}}_i(\mathbf{x}_{n,i})\tilde{\boldsymbol{\pi}}(\mathbf{x}_{n,i})\right\|_2^2, \tag{17}$$

The main difference with respect to the *constraint consistent learning* (CCL) [11]) is that this minimization produces policies that are consistent with the full set of constraints, instead of with a subspace of the constraints defined by projections of the action observations where $\tilde{\mathbf{N}}(\mathbf{x}) \approx \tilde{\mathbf{N}}(\mathbf{u}) := \frac{\mathbf{u}\mathbf{u}^\top}{\|\mathbf{u}\|_2^2}$. This leads to increased accuracy because the 1D projection error is only a lower bound of the error computed using the full constraint set.

In order to model the policy, we use weighted linear models, each of which is locally approximating the policy:

$$\tilde{\boldsymbol{\pi}}_m(\mathbf{x}) := \boldsymbol{\Psi}(\mathbf{x})\mathbf{b}_m, \tag{18}$$

where $\mathbf{b}_m$ is a vector of parameters to learn and $\boldsymbol{\Psi}(\mathbf{x})$ is a vector of appropriate regressors. The global policy is then a weighted combination of M partial models:

$$\tilde{\boldsymbol{\pi}}(\mathbf{x}) := \frac{\sum_{m=1}^{M} w_m(\mathbf{x})\tilde{\boldsymbol{\pi}}_m(\mathbf{x})}{\sum_{m=1}^{M} w_m(\mathbf{x})} \tag{19}$$

where $w_m(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{c}_m)^\top \mathbf{D}_m^{-1}(\mathbf{x}-\mathbf{c}_m)}$ is the importance weight of each observation according to the distance to the

local model's Gaussian receptive field, with centre $\mathbf{c}_m$ and variance $\mathbf{D}_m$ (a diagonal matrix). Centres of receptive fields can be obtained from data by running the K-means algorithm [12] or they can be computed using any other method.

Each model can be learned independently [17] and therefore minimization of (17) can be rewritten as:

$$E_{cpl}[\hat{\mathbf{b}}_m] = \sum_{k=1}^{K} w_m(\mathbf{x}_k) \left\| {}^{ns}\tilde{\mathbf{u}}_k - \tilde{\mathbf{N}}_k(\mathbf{x}_k)\mathbf{\Psi}(\mathbf{x}_k)\hat{\mathbf{b}}_m \right\|_2^2, \tag{20}$$

where we slightly abuse of notation by denoting ${}^{ns}\tilde{\mathbf{u}}_k \equiv {}^{ns}\tilde{\mathbf{u}}_{n,i}$, $\mathbf{x}_k \equiv \mathbf{x}_{n,i}$ and $\tilde{\mathbf{N}}_k(\mathbf{x}_k) \equiv \tilde{\mathbf{N}}_i(\mathbf{x}_{n,i})$ the combined dataset and K the total number of points.

Equation (20) can be solved via Weighted Least Squares (WLS):

$$\hat{\mathbf{b}}_m = \left(\mathcal{R}^\top \mathcal{W}_m \mathcal{R}\right)^{-1} \mathcal{R}^\top \mathcal{W}_m \mathcal{Y}, \tag{21}$$

with:

$$\mathcal{Y} = \begin{bmatrix} {}^{ns}\tilde{\mathbf{u}}_1^\top & \dots & {}^{ns}\tilde{\mathbf{u}}_K^\top \end{bmatrix}^\top, \tag{22}$$

$$\mathcal{R} = \begin{bmatrix} \tilde{\mathbf{N}}_1(\mathbf{x}_1)\mathbf{\Psi}(\mathbf{x}_1) \\ \vdots \\ \tilde{\mathbf{N}}_K(\mathbf{x}_K)\mathbf{\Psi}(\mathbf{x}_K) \end{bmatrix}, \tag{23}$$

$$\mathcal{W}_m = \operatorname{diag}\left(w_m(\mathbf{x}_1) \otimes \mathbf{I}_{q\times q}, \dots, w_m(\mathbf{x}_K) \otimes \mathbf{I}_{q\times q}\right), \tag{24}$$

where $\otimes$ denotes the Kronecker product operator.

IV. Learning Surface-Constrained Policies

In [14, 11, 21], among others, the learnt policies were quasi-linear potential-based policies such as joint limit avoidance, limit cycle, etc. As a consequence, the regressors used for the unconstrained policy were also simplistic, such as linear regressors of the state. No previous knowledge of the policy structure was exploited. However, in this paper we cover more complex scenarios, by performing arbitrary motions over flat surfaces. We take the advantage of defining task-specific regressors. Similarly to the previously cited papers, we have access to the robot Jacobian.

Our approach can be used in applications such as wiping, dusting, sweeping, scratching, writing, etc. where the robot is constrained by a surface on which the task is being performed. In all these examples, a task could be defined in terms of minimizing the distance to the surface and the misalignment between the surface and a plane relative to the robot's tool (see Figure 4). The null-space of this task would be any joint motion resulting in the robot's tool moving on top of the surface, i.e. movements tangential to the surface. We assume that this motion is a result of projecting an unconstrained policy into the null-space of the task. By doing this, we will be able to generalize such policy to different surfaces (tasks).

A. Learning the task and the constraint

In flat-surface scenarios as shown in Figure 2, the task error can be defined using the distance of the tool to the surface and its misalignment:

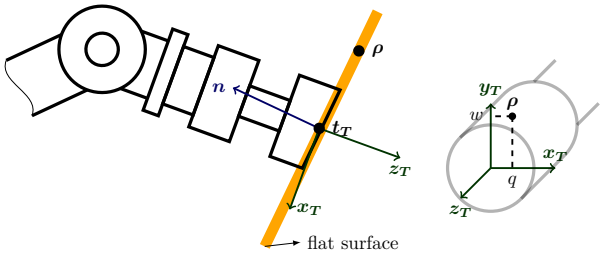


Fig. 2. A two dimensional illustration of a robot performing the demonstrated motion on a flat surface. $\boldsymbol{\rho}$ is a point on the $\boldsymbol{x_T}$-$\boldsymbol{y_T}$ plane used as a centre of the wiping motion performed in the null-space of the surface alignment task.

$$\mathbf{e}(\mathbf{x}) := \begin{bmatrix} \mathbf{n}\cdot(\mathbf{t}_T(\mathbf{x}) - \mathbf{p}) \\ \mathbf{n}\cdot\mathbf{x}_T(\mathbf{x}) \\ \mathbf{n}\cdot\mathbf{y}_T(\mathbf{x}) \end{bmatrix}, \tag{25}$$

where $\mathbf{n}$ is the surface normal vector and $\mathbf{p}$ is a point on the surface. In differential kinematics [19] the state of a robot can be described by the joint velocities, $\dot{\mathbf{x}}$, and its relation with respect to the velocity vector (error) of a task, $\dot{\mathbf{e}}(\mathbf{x})$:

$$err\dot{\mathbf{e}}(\mathbf{x}) = \mathbf{J}(\mathbf{x})\dot{\mathbf{x}}, \tag{26}$$

where $\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}$ is the analytical Jacobian of the task. We substitute $\mathbf{A}(\mathbf{x}) \equiv \mathbf{J}(\mathbf{x})$ and $\mathbf{u} = \dot{\mathbf{x}}$ in (1) while $\mathbf{b}(\mathbf{x})$ becomes the policy ensuring that the error converges to zero.

From (26) and (25) we can select the following regressors:

$$\mathbf{\Phi}_A(\mathbf{x}) := \mathbf{J}_T(\mathbf{x}) \equiv \begin{bmatrix} \frac{\partial \mathbf{t}_T(\mathbf{x})}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{x}_T(\mathbf{x})}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{y}_T(\mathbf{x})}{\partial \mathbf{x}} \end{bmatrix}, \tag{27}$$

$$\mathbf{\Phi}_b(\mathbf{x}) := \begin{bmatrix} \mathbf{t}_T(\mathbf{x})^\top & \mathbf{x}_T(\mathbf{x})^\top & \mathbf{y}_T(\mathbf{x})^\top & 1 \end{bmatrix}, \tag{28}$$

where the task could be defined as a linear policy minimizing the error. Estimation in real-data scenarios coming from human demonstrations can be challenging since, as discussed in Section II-B, all assumptions about linearity on the parameters might not hold for the proposed regressors and therefore we might need to increase the number of parameters to accommodate the complexity of data. In theory, the regressors for $\mathbf{\Phi}_A(\mathbf{x})$ should be correct as long as the demonstrations are always constrained to the surfaces and the task is to minimize misalignment error. However, the regressors $\mathbf{\Phi}_b(\mathbf{x})$ might be insufficient due to an unknown task policy introduced by the human operator. From the analysis of the singular values, we can clearly identify situations where extra parametrization is needed.

B. Learning the unconstrained policy

The primary task is to align the tool with the surface (2DoF) and maintain the contact (1DoF). This implies a task with 3 DoF. We can reasonably assume that any motion along the surface will be part of the null-space of the primary task. Therefore, a secondary task, with 2 additional DoF could be a motion along the tool's plane $\boldsymbol{\rho}(\mathbf{x})$:

$$\boldsymbol{\rho}(\mathbf{x}, q, w) := \mathbf{t}_T(\mathbf{x}) + q\mathbf{x}_T(\mathbf{x}) + w\mathbf{y}_T(\mathbf{x}). \tag{29}$$

This policy is independent of the surface orientation and it can be application-specific, i.e.: circles for wiping, lines for sweeping, letters for writing, etc. The secondary task is projected over the null-space of the primary task, which implies that in order to generate an appropriate motion on a surface, the tool must be aligned and in contact with it.

Motion along $\boldsymbol{\rho}(\mathbf{x})$ imposes an additional virtual constraint with it's own null-space and therefore there might exist a third task which is projected over that null-space, if the robot has additional DoF. This third task is typically a task trying to reach a comfortable configuration or a configuration that tries to avoid joint limits, obstacles, etc.

The second and third tasks can be treated as one single unconstrained policy as defined in our problem statement. Thus, each local model in (18) could be expressed as:

$$\tilde{\boldsymbol{\pi}}_m(\mathbf{x}) := \mathbf{J}_T^{\boldsymbol{\rho}}(\mathbf{x})^\dagger \tilde{\boldsymbol{\kappa}}_m(\mathbf{x}) + \mathbf{N}_T^{\boldsymbol{\rho}}(\mathbf{x}) \tilde{\boldsymbol{\gamma}}_m(\mathbf{x}), \tag{30}$$

where $\boldsymbol{\kappa}_m(\mathbf{x}) \in \mathbb{R}^2$ is the secondary task policy and $\boldsymbol{\gamma}_m(\mathbf{x}) \in \mathbb{R}^q$ is the tertiary task and $\mathbf{J}_T^{\boldsymbol{\rho}}(\mathbf{x}) \in \mathbb{R}^{2\times q}$ is the Jacobian of the X and Y coordinates of the tool expressed in the tool frame (and $\mathbf{N}_T^{\boldsymbol{\rho}}(\mathbf{x})$ its null-space projection matrix):

$$\mathbf{J}_T^{\boldsymbol{\rho}}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_T(\mathbf{x})^\top & \mathbf{0}^\top & \mathbf{0}^\top \\ \mathbf{y}_T(\mathbf{x})^\top & \mathbf{0}^\top & \mathbf{0}^\top \end{bmatrix} \mathbf{J}_T(\mathbf{x}). \tag{31}$$

If the policies $\boldsymbol{\kappa}_m(\mathbf{x})$ and $\boldsymbol{\gamma}_m(\mathbf{x})$ can be expressed as a linear combination of regressors:

$$\boldsymbol{\kappa}_m(\mathbf{x}) := \boldsymbol{\Psi}_\kappa(\mathbf{x}) \mathbf{b}_{m,\kappa} \tag{32}$$

$$\boldsymbol{\gamma}_m(\mathbf{x}) := \boldsymbol{\Psi}_\gamma(\mathbf{x}) \mathbf{b}_{m,\gamma}, \tag{33}$$

the regressors and weights to learn in (18) would be:

$$\boldsymbol{\Psi}(\mathbf{x}) := \begin{bmatrix} \mathbf{J}_T^{\boldsymbol{\rho}}(\mathbf{x})^\dagger \boldsymbol{\Psi}_\kappa(\mathbf{x}) & \mathbf{N}_T^{\boldsymbol{\rho}}(\mathbf{x}) \boldsymbol{\Psi}_\gamma(\mathbf{x}) \end{bmatrix} \tag{34}$$

$$\mathbf{b}_m := \begin{bmatrix} \mathbf{b}_{m,\kappa}^\top & \mathbf{b}_{m,\gamma}^\top \end{bmatrix}^\top. \tag{35}$$

For example, in a circular wiping motion, the regressors could be defined as:

$$\boldsymbol{\Psi}_\kappa(\mathbf{x}) = \left[\mathbf{c}_{\boldsymbol{\rho}}^\perp(\mathbf{x}) \quad \mathbf{c}_{\boldsymbol{\rho}}(\mathbf{x}) \left(1 - \frac{r}{\|\mathbf{c}_{\boldsymbol{\rho}}(\mathbf{x})\|_2} \right) \right], \tag{36}$$

where $\mathbf{c}_{\boldsymbol{\rho}}(\mathbf{x}) \in \mathbb{R}^2$ are the coordinates of the circle centre (expressed in $\boldsymbol{\rho}(\mathbf{x})$, that is relative to the end-effector frame), $\mathbf{c}_{\boldsymbol{\rho}}^\perp(\mathbf{x})$ is perpendicular to $\mathbf{c}_{\boldsymbol{\rho}}(\mathbf{x})$ and r is the radius. When training a policy from a real robot (from a demonstration), these parameters can be easily extracted from the data.

On the other hand, if no prior knowledge about the lower priority policy is give, this could be described with some linear regressors:

$$\boldsymbol{\Psi}_\gamma(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^\top & 1 \end{bmatrix} \otimes \mathbf{I}_{q\times q}. \tag{37}$$

V. Performance Analysis

We are now interested in evaluating the policy learning performance of the method described in Section III (constraint projection learning or CPL) and comparing the results with the method proposed in [11] (constraint consistent learning or CCL). We generated a dataset where a total of 100 trials were carried out for the wiping policy using simulated data, where each trial contains trajectories of 150 points corresponding to 25 random initial configurations.

Method	N_c	Regressors	nUPE	nCPE	nCCE
LWCCL	8	linear	9893.40	2024.00	0.25
		non-linear	853.83	788.54	1.16
	12	linear	12122.30	5420.80	0.17
		non-linear	880.16	849.10	1.09
	20	linear	12947.40	9086.40	0.20
		non-linear	466.45	441.38	0.92
LWCPL	8	linear	30.93	2.71	0.84
		non-linear	15.19	2.39	1.35
	12	linear	30.69	1.37	0.64
		non-linear	14.28	2.55	1.27
	20	linear	30.09	0.85	0.49
		non-linear	11.31	2.15	1.11

TABLE I
WIPING POLICY LEARNING ERRORS (UNITS ARE 10^{-2}).

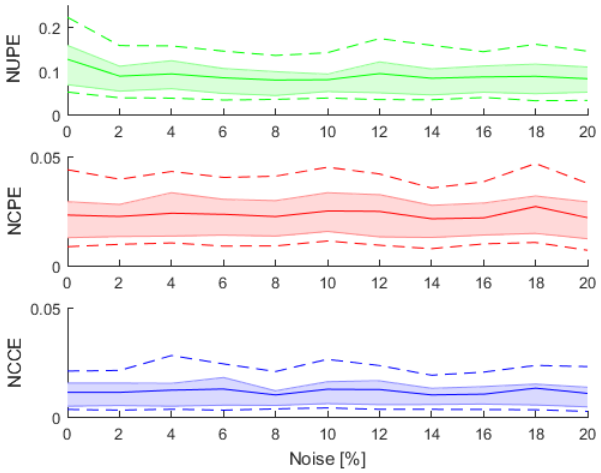


Fig. 3. Policy learning errors of a data set containing 25 different random configurations with 12 different constraint orientations. Noise varies from 0% to 20% with 100 trials for each experiment.

We want to analyse the effect of having different numbers of constraints, N_c coming from different surface orientations, with $N_c = \{8, 12, 20\}$. Considering that only 90% of data is used for training. The case with 8 different orientations uses 7 orientations for training and 1 for testing. This is the minimum number of observations required in order to resolve the policy ambiguity [11] if data were collected from exactly the same configuration with a 7 DoF robot.

The policy uses a set of receptive fields based on RBFs that were randomly initialized using the k-means algorithm [12], requiring at least 25 local models per trial. This value may change for different trials. We keep increasing the number of models incrementally until the whole space is covered with a minimum guaranteed weight of 0.7[2]. We use the same field placement for either locally weighted constraint projection learning (LWCPL) or locally weighted constraint consistent learning (LWCCL) when evaluating their performance.

[2]The minimum weight of 0.7 has been empirically validated across different trials for a 7Dof KUKA LWR3 and IIWA robots.

The results are shown in Table I, where metrics[3] are defined in Table II. LWCCL is directly optimizing the nCCE at the increased cost in the nUPE and nCPE metrics, because nCCE is only a lower bound of the error. On the contrary, LWCPL is able to obtain low errors in the null space projection (nCPE) and the learnt policy is generally much closer to the ground truth policy (nUPE column contains low values). We compare the proposed problem-specific non-linear regressors (36) and (37) for the wiping application, instead of using only linear regressors $\mathbf{\Psi}(\mathbf{x}) = \begin{bmatrix}\mathbf{x}^\top & \mathbf{c}_\rho^\top & 1\end{bmatrix} \otimes \mathbf{I}_{q\times q}$. Therefore, as long as we can provide task-specific regressors, the accuracy will be significantly improved.

We have also analyse the influence of a noisy policy on LWCPL, we have conducted an additional experiment with 25 configurations, 12 constraint orientations and 11 noise levels (form 0% to 20%). Each experiment has been repeated for 100 trials. Noises were added to the task and the null-space components. Our, results, shown in Figure 3, indicate that there is no explicit influence over the estimation accuracy for such noise levels. This could be explained because the noise in the task component is filtered out by the SVD method, while the noise in the null-space component is filtered-out by the WLS method.

VI. Task Generalization using Force Sensor

We show the utility of learning surface-constrained policies through generalization to a novel task. In many scenarios such as in the train cleaning application (Figure 1), it might be hard to obtain a precise model of the surface due to outdoors lighting conditions, different surface materials, and its dimensions. Thus, in practical applications, the constraint surface may not be known. Therefore, we aim to redefine the surface alignment task using a force torque sensor.

A. Contact force alignment task for wiping experiments

Let's consider a soft wiping tool (such as a sponge) which deforms on contact with the surface and exerts a wrench (force and torque) on the solid tool it's mounted on. If we mount a force/torque sensor on this tool, we can measure this contact wrench. Figure 4 shows a 2 dimensional schematic representation of the wiping tool interacting with a surface and the resulting forces.

When the tool aligns with the surface, the contact force is perfectly aligned with the surface normal and we measure no torque around the contact point. Otherwise, the task error is given by

$$\mathbf{e}_F(\mathbf{x}) := \begin{bmatrix} f_c - f_z \\ -m_x \\ -m_y \end{bmatrix}, \tag{38}$$

where f_z is the z component of the contact force and f_c is the desired contact force; and m_x, and m_z are the x and y components of the contact torque relative to the tool axis. In this scenario, the Jacobian of this error with respect to the tool's frame is defined as $\mathbf{J}_F(\mathbf{x}) \in \mathbb{R}^{3\times 7}$:

$$\mathbf{J}_F(\mathbf{x}) = \begin{bmatrix} \mathbf{z}_T^\top & \mathbf{0}^\top \\ \mathbf{0}^\top & \mathbf{x}_T^\top \\ \mathbf{0}^\top & \mathbf{y}_T^\top \end{bmatrix} \bar{\mathbf{J}}_T(\mathbf{x}), \tag{39}$$

where $\bar{\mathbf{J}}_T(\mathbf{x}) \in \mathbb{R}^{6\times 7}$ represents a standard geometric robot jacobian.

The alignment of the tool with the surface is then achieved by substituting $\mathbf{J}_F(\mathbf{x})$ into equation 26. The minimization of this task error ensures the contact and alignment between the wiping tool and the surface. Our task is derived from a PD controller $\mathbf{b}(\mathbf{x}) = K_p\mathbf{e}_F(\mathbf{x}) + K_d\dot{\mathbf{e}}_F(\mathbf{x})$, where K_p and K_d are the PD gains tuned to achieve a stable behaviour.

It is important to remark that the error vector (38) used in wiping a non-flat surface with force feedback is different from the error used during the demonstration on the flat surfaces (25). Actually, they do not have the same dimensions, and the null-space projection matrix derived from the force sensor jacobian is different from the planar surface constraint. Despite of that, the learnt unconstrained policy $\tilde{\boldsymbol{\pi}}(\mathbf{x})$ can be projected using the new projection matrix, without affecting the primary sensor-based task.

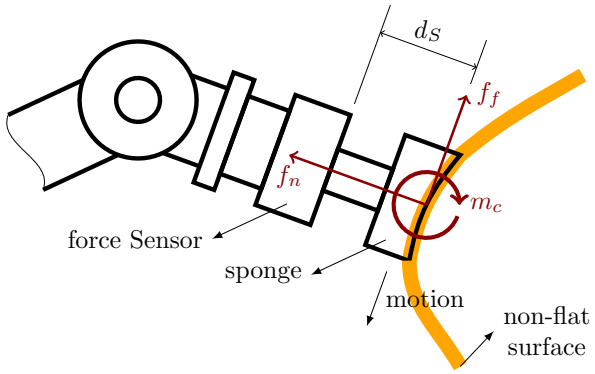


Fig. 4. A two dimensional illustration of a robot performing a constrained task on a curved surface. The robot uses as a tool a force sensor and a soft material (sponge) mounted at the end-effector. The interaction of the wiping tool and the surface causes a friction force f_f, a normal force f_n, and a contact torque m_c, where the arrows indicate the direction in which the values f_x and f_z are measured. The task is to align the tool with the surface normal, by minimizing the contact torque m_c.

B. Experimental results

The experimental apparatus includes the 7 DoF Kuka LWR3 robot with an ATI industrial automation Gamma F/T sensor attached at the end effector, as shown in Figure 7. The sensor has a sponge attached to it to provide a soft interface between the robot and the surface. The force sensor retrieves a 6 dimensional wrench vector expressed in the sensor frame. Therefore, we compute the torque at the contact point by transforming the wrench by distance d_S towards the contact area. We have estimated this distance empirically by pressing the tool against surfaces at different angles. The transformation

[3]nUPE provides a metric to measure the performance of a method for estimating the unconstrained policy with respect to the ground-truth unconstrained policy. nCPE measures the performance of its projection using the null-space projection matrix, while nCCE uses a 1D projection matrix using the observation data itself. It is well known that nUPE>nCPE>nCCE [11]

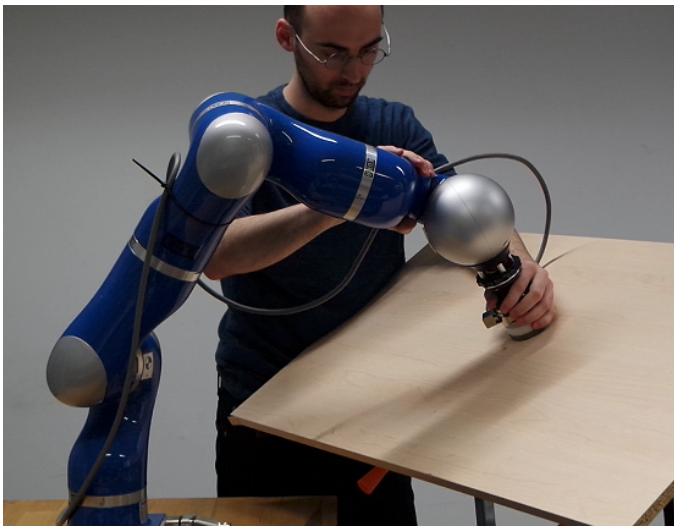
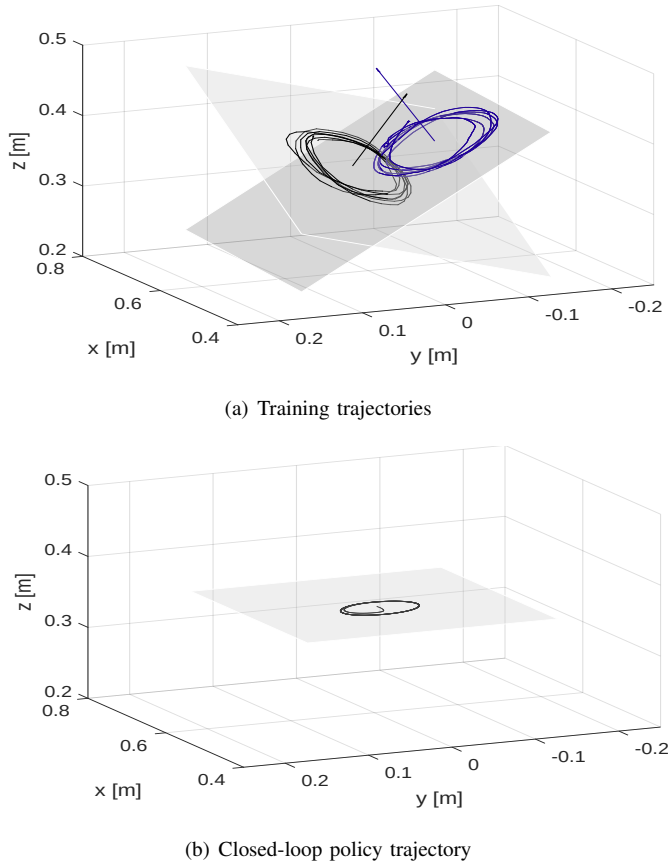


Fig. 5. Demonstration of a circular wiping trajectory on a flat surface. The demonstration was repeated on surfaces of multiple orientations.



(a) Training trajectories

(b) Closed-loop policy trajectory

Fig. 6. Learning by demonstration: a) generated wiping trajectories from human demonstration (twelve in total, but only two shown); b) Closed-loop policy validation using a flat surface of previously unseen orientation (orientation not included in the training set).

uses a constant distance, regardless of its small variations due to using a sponge at the tool tip.

With this setup, we have recorded a dataset of wiping trajectories demonstrated by a human, as shown in Figure 5. We have recorded 12 sets of trajectories, each on a surface of a different orientation (two of them shown in Figure 6(a)). Each demonstration involved several circles with the tool of the robot, containing approximately 2000 data points[4] (using a sampling rate of 100Hz). The demonstrated data was only minimally cropped to ensure the data contained only poses where the tool was in contact with the surface and moving along the demonstrated trajectory. We have preprocessed the data to extract the centre and radius of wiping by taking estimating a 3D circle, details omitted for brevity.

We have used this dataset to learn the constraints (orientations of the surfaces) using the technique presented in Section II. Since, the tool contains a flexible sponge, the task alignment was prone to contain errors introduced by the human demonstrator. The media attachment shows that alignment correction were required to accommodate for these errors. The alignment *speed* varies, occasionally causing more abrupt corrections of the tool orientation. This introduces complexity on the task to learn and thus the regressors in $\mathbf{\Phi}_b(\mathbf{x})$ were extended by additionally including second-order binomials of $\mathbf{x}$ to improve the estimation of the surface orientation. We have subsequently learnt the surface-constrained wiping policy as described in Section IV using the constraint matrices estimated in the previous step. We have used the regressors defined in equation 36. The resulting policy was then stored and used, in closed-loop, together with the force-based surface alignment task described in Section VI-A. In real-data scenarios, metrics nUPE and nCPE defined in Table II can not be computed due to the absence of a ground-truth policy or null-space observations. Thus, in this section we show how a policy, trained from human demonstrations in flat surfaces, generalization to flat and non-flat surfaces. The results in this section are qualitative, while all quantitative results have been already shown in Section V.

The resulting motion on a flat surface matched the demonstration, even in the case where the new surface orientation was not included in the training set, see Figure 6(b). In addition to this, we have also validated the learnt policy on a non-flat surface as shown in Figure 7. The results, providing a circular wiping motion, are depicted in Figure 8. Note that we have demonstrated the wiping motion exclusively on flat surfaces, therefore this shows two aspects of generalization: (I) from a surface alignment task to a force alignment task, and (II) from flat surfaces to a curved surface. See [2] for video recordings of the policy generalization to a curved surface.

VII. Conclusion

This paper presents a new method for learning, from demonstration for motion where the policies lie in the null-space of a task. As it has been shown, the main advantage of this approach, compared to classic direct policy learning, is its

[4]The data included the joint state x and the joint commands u, obtained by differentiating the joint states.

Name	Acronym	Expression	Comments
Normalized Unconstrained Policy Error	nUPE	$\frac{1}{K\|\sigma_\pi^2\|_2}\sum_{k=1}^{K}\|\boldsymbol{\pi}(\mathbf{x}_k)-\tilde{\boldsymbol{\pi}}(\mathbf{x}_k)\|_2^2$ $\boldsymbol{\sigma}_{\boldsymbol{\pi}}^2=\mathrm{var}(\boldsymbol{\pi}(\mathbf{x}_k))$	Measures the difference between the ground-truth unconstrained policy and the estimated one.
Normalized Constrained Policy Error	nCPE	$\frac{1}{K\|\sigma_\pi^2\|}\sum_{k=1}^{K}\|{}^{ns}\mathbf{u}_k-\mathbf{N}(\mathbf{x}_k)\tilde{\boldsymbol{\pi}}(\mathbf{x}_k)\|_2^2$ $\boldsymbol{\sigma}_{\boldsymbol{\pi}}^2=\mathrm{var}(\boldsymbol{\pi}(\mathbf{x}_k))$	Measures the difference between the ground-truth null-space component and the null-space estimated policy.
Constraint Consistent Policy Error	nCCE	$\frac{1}{K\|\sigma_u^2\|}\sum_{k=1}^{K}\left\|{}^{ns}\mathbf{u}_k-\frac{{}^{ns}\mathbf{u}_k{}^{ns}\mathbf{u}_k^\top}{\|{}^{ns}\mathbf{u}_k\|_2^2}\tilde{\boldsymbol{\pi}}(\mathbf{x}_k)\right\|_2^2$ $\boldsymbol{\sigma}_{\boldsymbol{\pi}}^2=\mathrm{var}(\boldsymbol{\pi}(\mathbf{x}_k))$	Measures the difference between the ground-truth null-space component and the 1D projection of the observed actions.

TABLE II
METRICS FOR PERFORMANCE EVALUATION ASSUMING ACCESS TO GROUND-TRUTH DATA.

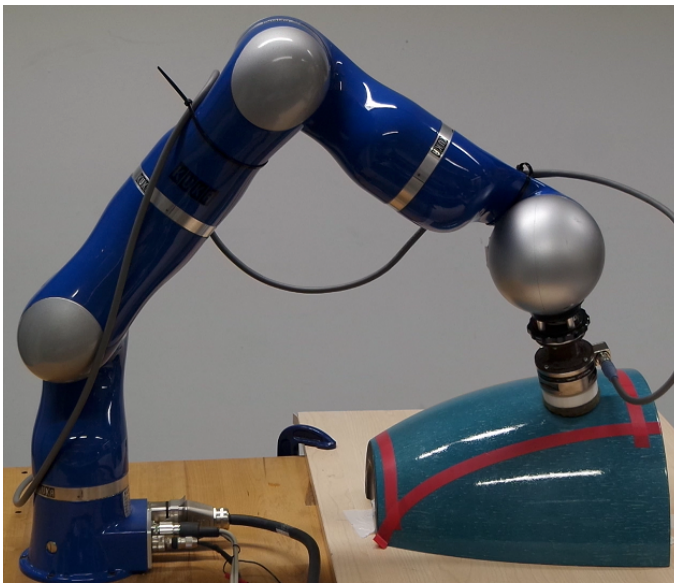


Fig. 7. Kuka lightweight robotic arm equipped with a force/torque sensor wiping a curved surface.

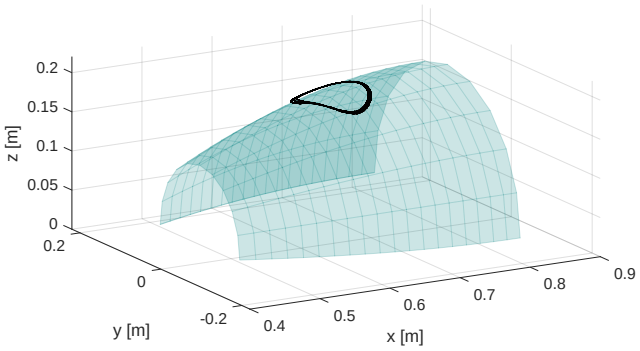


Fig. 8. A wiping policy has been trained from human demonstrations on flat surfaces (without using the force sensor); the policy generalizes to non-flat surfaces using a force-sensor based task to align the tool dynamically.

ability to generalize to unseen tasks. Actually, the task used for the training was defined by a human operator, while in the validation we use a force-based task to adapt and aligned the tool to an unknown surface.

Following our approach, one could add additional constrains such as ensuring that the robot end-effector is inside a given region, while performing a constrained wiping motion or to relax the imposed constrains, such as surface alignment, i.e.: in some circumstances one might be interested in performing a wiping motion without being aligned. Particularly at regions close to the robot workspace limits.

The amount of generalization has its limits though. We learn the wiping policy from demonstrations. These have to cover the space sufficiently to allow the receptive field model to be trained properly (models are trained locally based on data). The true power of this approach is, however, that it easily extends to higher dimensions. In our future work we intend to exploit more challenging application domains as well as learning constrained tasks for dynamic systems.

APPENDIX

Different metrics have been described in the literature for measuring performance when learning the unconstrained policy from constrained motion [11]. We provide an overview of these metrics in Table II.

ACKNOWLEDGEMENTS

This work is supported by the Spanish Goverment (José Castillejo Research Grant Ref. JC2015-00304) and Universitat Politècnica de València (PAID-00-15) and the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh, under the grants EP/L016834/1, EP/J015040/1. The authors are grateful to the financial support of Spanish Ministry of Economy and European Union, grant DPI2016-81002-R (AEI/FEDER, UE). The authors would like to acknowledge Dr. Mustafa Suphi Erden for the discussions about the control strategy for the wiping task using force feedback, in the context of the Cap Front Cleaning Robot Project supported by the Railway Safety and Standards Board (RSSB), UK.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57 (5):469–483, 2009.

[2] Leopoldo Armesto, Vlad Ivan, Joao Moura, Sethu Vijayakumar, and Antonio Sala. Efficient learning of constraints and generic null space policies. URL https://www.youtube.com/watch?v=2n0yW1yI524.

[3] Paolo Baerlocher and Ronan Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The visual computer*, 20(6):402–417, 2004.

[4] Holk Cruse and M Brüwer. The human arm as a redundant manipulator: the control of path and joint angles. *Biological cybernetics*, 57(1-2):137–144, 1987.

[5] James Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11:873–912, 1990.

[6] Aaron D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In *Proc. of International Conference on Intelligent Robots and Systems IEEE/RSJ*, volume 1, pages 298–303, 2001.

[7] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014.

[8] Michael Gienger, Herbert Janssen, and Christian Goerick. Task-oriented whole body motion for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, pages 238–244, 2005.

[9] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, pages 1–19, 2015.

[10] Matthew Howard and Sethu Vijayakumar. Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In *In: Workshop on Robotics and Mathematics (RoboMat)*, 2007.

[11] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. A novel method for learning policies from variable constraint data. *Autonomous Robots*, 27(2):105–121, 2009.

[12] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017616.

[13] Oussama Khatib, Luis Sentis, and Jae-Heung Park. A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *European Robotics Symposium 2008*, pages 303–312, 2008.

[14] Hsiu-Chin Lin, Matthew Howard, and Sethu Vijayakumar. Learning null space projections. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2613–2619, 2015.

[15] Nicolas Mansard and François Chaumette. Task sequencing for high-level sensor-based control. *IEEE Transactions on Robotics*, 23(1):60–72, 2007.

[16] Joao Moura and Mustafa Suphi Erden. Formulation of a Control and Path Planning Approach for a Cab front Cleaning Robot. In *Procedia CIRP*, volume 59, pages 67–71, 2017. doi: 10.1016/j.procir.2016.09.024.

[17] Stefan Schaal and Christopher G Atkeson. Constructive incremental learning from only local information. *Neural computation*, 10(8):2047–2084, 1998.

[18] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1431):537–547, 2003.

[19] B. Siciliano, L. Sciavicco, L. Villani, and G Oriolo. *Differential Kinematics and Statics*, pages 105–160. Springer London, London, 2009. ISBN 978-1-84628-642-1. doi: 10.1007/978-1-84628-642-1_3.

[20] Hisashi Sugiura, Michael Gienger, Herbert Janssen, and Christian Goerick. Real-time self collision avoidance for humanoids by means of nullspace criteria and task intervals. In *IEEE-RAS International Conference on Humanoid Robots*, pages 575–580, 2006.

[21] Chris Towell, Matthew Howard, and Sethu Vijayakumar. Learning nullspace policies. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 241–248, 2010.

[22] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.