In: Proceedings IEEE Intl. Conf. on Robotics and Automation (ICRA), Singapore (2017)

Efficient Learning of Constraints and Generic Null Space Policies

Leopoldo Armesto[1] and Jorren Bosga[2] and Vladimir Ivan[2] and Sethu Vijayakumar[2]

***Abstract*— A large class of motions can be decomposed into a movement task and null-space policy subject to a set of constraints. When learning such motions from demonstrations, we aim to achieve generalisation across different unseen constraints and to increase the robustness to noise while keeping the computational cost low. There exists a variety of methods for learning the movement policy and the constraints. The effectiveness of these techniques has been demonstrated in low-dimensional scenarios and simple motions. In this paper, we present a fast and accurate approach to learning constraints from observations. This novel formulation of the problem allows the constraint learning method to be coupled with the policy learning method to improve policy learning accuracy, which enables us to learn more complex motions. We demonstrate our approach by learning a complex surface wiping policy in a 7-DOF robotic arm.**

I. INTRODUCTION

Movement in complex, multi joint systems often contains a high level of redundancy. The degrees of freedom available to perform a task are usually higher than what is actually necessary to execute for that task. This allows certain flexibility in finding an appropriate solution. This redundancy may be resolved according to some strategy that achieves a secondary objective. Such approaches to redundancy resolution are employed by humans [5] as well as other redundant systems such as (humanoid) robots [6].

The redundancy 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. The task space component represents the degrees of freedom (DoF) necessary to accomplish the primary task, while the null space component determines how the redundancy gets resolved. This means that the null space may be used to achieve some secondary goal of lower priority. For instance, one might consider the primary task to consist of the desired activity, such as reaching, and a lower-priority task to be a secondary goal such as avoiding joint limits [9], self-collisions [23], or kinematic singularities [25]. This notion becomes more intuitive when we also consider external or environmental constraints. These static constraints also restrict the space in which the desired task may be performed. For instance, motion might be constrained by a surface for a wiping application (see Fig. 1). Several variants of this hierarchical approach to redundancy have been used in robotics [15]. This core concept has been applied to task sequencing [19], task prioritisation [3] and hierarchical quadratic programming [8][11], but they use explicit constraint formulations to compute the required motion.

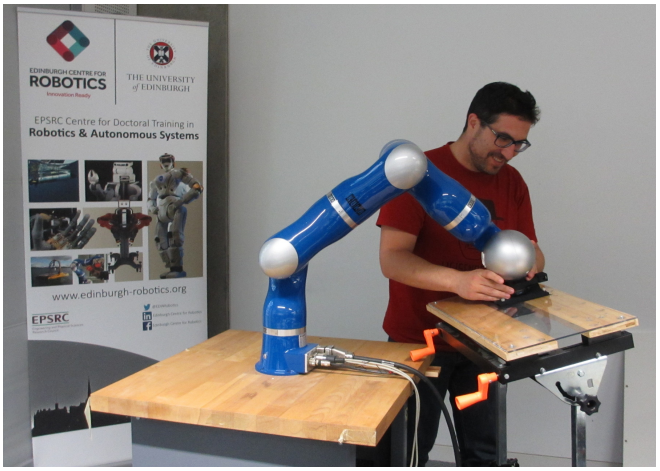


Fig. 1. Task demonstration carried out with the Kuka LWR Robot.

To avoid that, motion can be learned from data captured from demonstrations. These demonstrations either consist of human movements or movements made by the robot itself (or another robot) that is guided by a human. This approach falls under the category of imitation learning [2], and one straightforward way to learn behaviours from this is through direct policy learning (DPL) [22]. This method uses demonstrations to learn control policies through supervised learning, however, DPL suffers from poor generalisation [2].

Therefore, in order to generalize to tasks that differ from the demonstrations, it is helpful to learn the two components: the unconstrained policy and the constraints. In fact, by learning both separately, generalisation could be achieved across constraints (e.g. utilizing the learnt policy under different constraints) as well as within constraints (e.g. utilizing a new policy under the learnt constraints). A recently developed method for learning both the null space policy [12] and the constraints [16] significantly outperforms direct policy learning methods, but it imposes strict assumptions on the nature of the data used for learning and so far its effectiveness has been demonstrated mostly in low-dimensional scenarios with relatively simple (linear) movement policies. In [4] authors propose to use virtual-damper systems, combined with Gaussian Mixture regression to learn different tasks

This work is supported by the Spanish Goverment (Research Projects DPI2013-42302-R and DPI2016-81002-R and José Castillejo Research Grant Ref. JC2015-00304) and Universitat Politècnica de València (PAID-00-15).

[1]L. Armesto is with Control Systems and Engineering, Universitat Politècnica de València, C/Camino de Vera s/n, 46019, Valencia Spain larmesto@idf.upv.es

[2]J. Bosga, V. Ivan and S. Vijayakumar are with the Institute of Action, Perception, and Behaviour, University of Edinburgh, Crichton Street 10, EH8 9AB, Edinburgh, UK. jorrenbosga@gmail.com, v.ivan@ed.ac.uk, sethu.vijayakumar@ed.ac.uk

parametrization by taking the advantage of the variability in the demonstrations to extract task-space invariant features and achieve generalization to new situations. In [10], their aim is to select the task for a given set of demonstrations, which is solved iteratively.

In this paper, we present a new method for learning the constraints under learning by demonstration scenario. Compared to [16], we provide a new metric that minimizes the error in the task-space, rather than in the null-space. We show that our approach outperforms state of the art techniques in both computation speed and accuracy. This improvement allows us to use the estimated constraints for learning the null space policy, which leads to more accurate policy estimations, which in turn enables us to efficiently learn more complex policies.

The paper is organized as follows: in section II, we first provide some preliminaries to our work. In describe section III a method for learning constraints is proposed and used in section IV for estimating the unconstrained policy. Section V describes the wiping policy used in simulated data, provides an analysis performance using such data and also provides some experimental results. In summary, we demonstrate that our approach is effective in learning policies in complex scenarios such as learning a wiping policy with a simulated and a real 7-DOF robotic arm.

II. Preliminaries

In our problem setting, we assume little knowledge about the system: we have access to constrained motion data consisting only of state-action tuples $(\mathbf{x}, \mathbf{u})$, with $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^q$. We assume that the observed actions can be decomposed into a task space and a null space component, which are orthogonal with respect to each other:

$$\mathbf{u}(\mathbf{x}) = \mathbf{A}(\mathbf{x})^\dagger \mathbf{b}(\mathbf{x}) + \mathbf{N}(\mathbf{x})\boldsymbol{\pi}(\mathbf{x}) \tag{1}$$

where $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{s \times q}$ $(s < q)$ is a Pfaffian constraint matrix and $\mathbf{A}^\dagger(\mathbf{x})$ its Moore-Penrose pseudo-inverse, $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^s$ is the task space policy and the *null space projection matrix* $\mathbf{N}(\mathbf{x}) \in \mathbb{R}^{q \times q}$ projects the *null space policy* $\boldsymbol{\pi}(\mathbf{x}) \in \mathbb{R}^q$ onto the null space of $\mathbf{A}(\mathbf{x})$:

$$\mathbf{N}(\mathbf{x}) = (\mathbf{I} - \mathbf{A}(\mathbf{x})^\dagger \mathbf{A}(\mathbf{x})) \tag{2}$$

If the task-space component of the observations, ${}^{ts}\mathbf{u}(\mathbf{x}) = \mathbf{A}(\mathbf{x})^\dagger \mathbf{b}(\mathbf{x})$, is known or could be neglected $\mathbf{b}(\mathbf{x}) \approx \mathbf{0}$, then (1) simplifies to:

$${}^{ns}\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\boldsymbol{\pi}(\mathbf{x}) \tag{3}$$

where ${}^{ns}\mathbf{u}(\mathbf{x})$ is the null space component of $\mathbf{u}(\mathbf{x})$, that is ${}^{ns}\mathbf{u}(\mathbf{x}) = \mathbf{u}(\mathbf{x}) - {}^{ts}\mathbf{u}(\mathbf{x})$. Here, both $\mathbf{N}(\mathbf{x})$ and $\boldsymbol{\pi}(\mathbf{x})$

Different metrics have been described in the literature for measuring performance when learning the null space policy from observed actions [12]. There are also two similar metrics that are used to evaluate the accuracy of the learnt projection matrix [18].

Normalised Unconstrained Policy Error (nUPE): error between the ground-truth and estimated policies:

$$E_{nUPE}[\tilde{\boldsymbol{\pi}}] = \frac{1}{N\|\sigma_\pi^2\|} \sum_{n=1}^{N} \left\|\boldsymbol{\pi}(\mathbf{x}_n) - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\right\|^2, \tag{4}$$

where symbol $\tilde{}$ denotes estimation and $\|\sigma_\pi^2\|$ is the variance of the ground-truth policy.

Normalised Constrained Policy Error (nCPE): error in the null-space. Here, we assume that $\mathbf{N}(\mathbf{x})$ is available as well as the null-space component of the action:

$$E_{nCPE}[\tilde{\boldsymbol{\pi}}] = \frac{1}{N\|\sigma_\pi^2\|} \sum_{n=1}^{N} \left\|{}^{ns}\mathbf{u}_n - \mathbf{N}(\mathbf{x}_n)\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\right\|^2. \tag{5}$$

Normalised Constraint Consistent Policy Error (nCCE): projects the policy onto a 1D sub-space which is consistent with the constraint using $\mathbf{P} = {}^{ns}\hat{\mathbf{u}}\,{}^{ns}\hat{\mathbf{u}}^T$, with ${}^{ns}\hat{\mathbf{u}} = \frac{{}^{ns}\mathbf{u}}{||{}^{ns}\mathbf{u}||}$ [12]. Minimizing the inconsistency then becomes:

$$E_{nCCE}[\tilde{\boldsymbol{\pi}}] = \frac{1}{N\|\sigma_\pi^2\|} \sum_{n=1}^{N} \left\|{}^{ns}\mathbf{u}_n - {}^{ns}\hat{\mathbf{u}}_n^{ns}\hat{\mathbf{u}}_n^T \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\right\|^2. \tag{6}$$

It is well known, [12], that $nCCE$ is a lower bound, meaning that $nUPE > nCPE > nCCE$. Therefore, having low figures in $nCCE$ might not necessarily imply that the estimated policy matches the ground-truth one.

Normalised Projected Policy Error (nPPE): measures the same error as (5), but in this case measures the accuracy of estimating the projection matrix and thus the ground-truth policy is assumed to be known:

$$E_{nPPE}[\tilde{\mathbf{N}}] = \frac{1}{N\|\sigma_u^2\|} \sum_{n=1}^{N} \left\|{}^{ns}\mathbf{u}_n - \tilde{\mathbf{N}}(\mathbf{x}_n)\boldsymbol{\pi}(\mathbf{x}_n)\right\|^2 \tag{7}$$

where σ_u^2 is the variance of the observed action.

Normalised Projected Observation Error (nPOE): measures the error of the projection matrix using the null-space component of the action:

$$E_{nPOE}[\tilde{\mathbf{N}}] = \frac{1}{N\|\sigma_u^2\|} \sum_{n=1}^{N} \left\|{}^{ns}\mathbf{u}_n - \tilde{\mathbf{N}}(\mathbf{x}_n)\,{}^{ns}\mathbf{u}_n\right\|^2 \tag{8}$$

Again, we can see that $nPOE$ is a lower bound and therefore $nPPE > nPOE$ and thus we seek for approaches providing low figures in $nPPE$ rather than $nPOE$.

III. Learning the Constraint Matrix

In this section, we define a new method for estimating the null space projection matrix. Our method formulates the problem so that we can minimize a quadratic function with quadratic constraints, which is a suitable representation for constrained optimization problems. This implies that we will be able to find very good solutions in few iterations, even in high-dimensional spaces. The original method [18] represents the problem in spherical coordinates, so the function to minimize becomes highly non-convex, leading to many local minima in high-dimensional spaces and making it much harder to find a good solution even with fast optimization procedures such as the Levenberg-Marquart algorithm [17].

Also the computational time is slow due to the spherical representation. Our assumption is that $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{s\times q}$ can be decomposed as:

$$\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}\mathbf{J}(\mathbf{x}) \tag{9}$$

where $\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{t\times q}$ is the Jacobian of the task and $\mathbf{\Lambda}$ is the constraint to be learned. We assume the Jacobian is known, which is a reasonable assumption since it can be calculated using the kinematic model of the robot when one is provided. Let us also assume that $\mathbf{\Lambda}$ consists of a set of s row-independent unit vectors:

$$\mathbf{\Lambda} = \begin{bmatrix}\boldsymbol{\lambda}_1 & \dots & \boldsymbol{\lambda}_s\end{bmatrix}^T = \begin{bmatrix}\lambda_{1,1} & \lambda_{1,2} & \dots & \lambda_{1,t}\\ \vdots & \vdots & \vdots & \vdots \\ \lambda_{s,1} & \lambda_{s,2} & \dots & \lambda_{s,t}\end{bmatrix} \in \mathbb{R}^{s\times t} \tag{10}$$

where $\boldsymbol{\lambda}_i^T\boldsymbol{\lambda}_j + q_{i,j} = 0 \quad \forall j \geq i,\ i = 1,\dots,s$, with $q_{i,j} = -1$ if $i = j$ (unit vector) and $q_{i,j} = 0$ otherwise (row-independent vectors). At this point we have the same problem statement as in [18], [17]. The differences come in the metric used to estimate $\mathbf{\Lambda}$. In [18], [17], authors used a metric minimizing the error in the null-space, using the property that ${}^{ns}\mathbf{u}_n = \mathbf{N}(\mathbf{x}_n){}^{ns}\mathbf{u}_n$, and thus a good estimation of $\mathbf{N}(\mathbf{x}_n)$ must verify this. On the contrary, we focus our attention to the property $\mathbf{A}(\mathbf{x}){}^{ns}\mathbf{u}(\mathbf{x}) = \mathbf{0}$. It turns out that the benefit of using this new approach is that we can represent the problem in a quadratic setting. As a consequence, we define a new metric to learn the constraint in the *constraint space*:

$$E_c[\tilde{\mathbf{A}}] = \frac{1}{2}\sum_{n=1}^{N}\left\|\tilde{\mathbf{A}}(\mathbf{x}_n){}^{ns}\mathbf{u}_n\right\|^2 \tag{11}$$

We can substitute for $\tilde{\mathbf{A}}(\mathbf{x})$ its representation in (9) so that we can derive an expression in quadratic form. Note that we can now directly minimize $\tilde{\mathbf{\Lambda}}$:

$$E_c[\tilde{\mathbf{\Lambda}}] = \frac{1}{2}\sum_{n=1}^{N}{}^{ns}\mathbf{u}_n^T\mathbf{J}(\mathbf{x}_n)^T\tilde{\mathbf{\Lambda}}^T\tilde{\mathbf{\Lambda}}\mathbf{J}(\mathbf{x}_n){}^{ns}\mathbf{u}_n = \frac{1}{2}\tilde{\boldsymbol{\lambda}}^T\mathcal{D}\tilde{\boldsymbol{\lambda}} \tag{12}$$

where $\mathcal{D} = \sum_{n=1}^{N}\mathbf{D}_n^T\mathbf{D}_n$ and $\tilde{\boldsymbol{\lambda}} = [\tilde{\boldsymbol{\lambda}}_1^T,\dots,\tilde{\boldsymbol{\lambda}}_s^T]^T \equiv \mathrm{vec}(\tilde{\mathbf{\Lambda}}^T) \in \mathbb{R}^{st\times 1}$ and

$$\mathbf{D}_n = \begin{bmatrix}{}^{ns}\mathbf{u}_n^T\mathbf{J}(\mathbf{x}_n)^T & \mathbf{0} & \dots & \mathbf{0}\\ \mathbf{0} & {}^{ns}\mathbf{u}_n^T\mathbf{J}(\mathbf{x}_n)^T & \dots & \mathbf{0}\\ \vdots & \ddots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \dots & {}^{ns}\mathbf{u}_n^T\mathbf{J}(\mathbf{x}_n)^T\end{bmatrix} \in \mathbb{R}^{s\times st} \tag{13}$$

It is interesting to remark that the Hessian of the quadratic expression in (12) can be computed in advance for a given data set, which speeds up computations.

The orthonormality constraint between vectors can also be expressed in a quadratic form. We define the constrained problem as:

$$\begin{aligned}\underset{\tilde{\boldsymbol{\lambda}}}{\text{argmin}} \quad & \tfrac{1}{2}\tilde{\boldsymbol{\lambda}}^T\mathcal{D}\tilde{\boldsymbol{\lambda}}\\ \text{s.t.} \quad & \tfrac{1}{2}\tilde{\boldsymbol{\lambda}}^T\mathcal{H}_{i,j}\tilde{\boldsymbol{\lambda}} + q_{i,j} = 0\end{aligned} \tag{14}$$

where $\mathcal{H}_{i,j}$ is a zero matrix filled with block-diagonal identities on the i-th and j-th positions.

$$\mathcal{H}_{i,j} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_{i,j} & \dots & \mathbf{0}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \mathbf{0} & \mathbf{I}_{j,i} & \dots & \mathbf{0} & \dots & \mathbf{0}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0}\end{bmatrix} \in \mathbb{R}^{st\times st} \tag{15}$$

This problem can be solved using standard quadratic optimization tools. In particular, we use *fmincon* function in Matlab 2015b release using the interior-point algorithm.

IV. Constrained Policy Learning

In this section, we propose a method that combines the projection matrix learning method proposed in the previous section with policy learning as described in the literature. The result is a *constrained policy learning* (CPL) technique, where for a set with data from different constraints, we first estimate the projection matrix for each data point individually and then estimate the policy using learnt projection matrices. This amounts to the minimization of two error functions:

$$E_c[\tilde{\mathbf{A}}_i] = \frac{1}{2}\sum_{n=1}^{N}\left\|\tilde{\mathbf{A}}_i(\mathbf{x}_{n,i}){}^{ns}\mathbf{u}_{n,i}\right\|^2 \tag{16}$$

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

where $\mathbf{A}_i$ refers to the i-th constraint and N_c is the number of different constraints provided in the dataset.

The main difference with respect to *constraint consistent learning* (CCL), the method used for policy learning in the literature [12], 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 observations. This leads to increased accuracy because the 1D projection error is only the lower bound of the error computed using the full constraint set.

In order to model the policy, we use weighted linear models, where each model can be learned independently [21]. This approach has many partial models that all contain a linear approximation of the policy in their local space:

$$\boldsymbol{\pi}_m(\mathbf{x}) = \mathbf{B}_m\begin{bmatrix}\mathbf{x}^T & 1\end{bmatrix}^T \tag{18}$$

where $\mathbf{B}_m$ is a matrix containing the weights of locally linear model m to be learned. The global policy is then a weighted combination of these partial models:

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

where $w_m(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{c}_m)^T\mathbf{D}^{-1}(\mathbf{x}-\mathbf{c}_m)}$ is the importance weight of each observation according to the distance to the local model's Gaussian receptive field with center $\mathbf{c}_m \in \mathbb{R}^{p\times 1}$ and variance $\mathbf{D} \in \mathbb{R}^{p\times p}$ a diagonal matrix. Centers of receptive fields are obtained by running the K-means

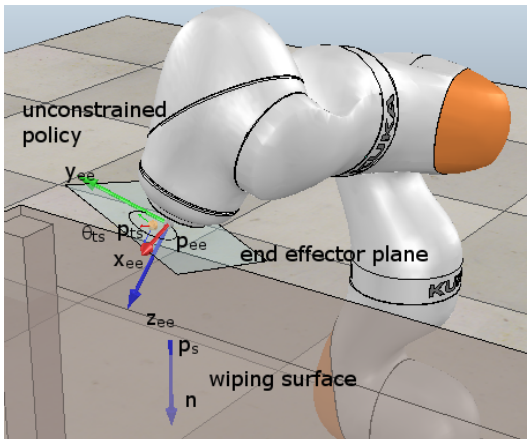


Fig. 2. **Simulated experimental setup using an Antropomorphic 7-DOF KUKA LBR IIWA R800 Robot** in V-REP [7] to obtained artifitial data.

algorithm [14] on the provided data. Both CPL and CCL can be implemented with local learning, resulting in Locally Weighted CPL (LWCPL) and Locally Weighted CCL (LWCCL).

V. Policy Learning Performance

A. Wiping policy

In this section, we demonstrate the accuracy of constrained policy learning for a wiping policy (circular motion in the null space). We show that the proposed method can handle complex policies in high dimensions by evaluating its performance using the ground truth wiping policy, where an example of a wiping motion can be shown in Figure 2 (in general, the end-effector plane is colinear with the wiping surface). The platform that we use to execute the policy is a kinematic simulation of the 7-DOF KUKA LBR IIWA R800.

In [12], most of previously learnt policies were restricted either to low-dimensional examples such as linear, sinusoidal, limit cycle, or inverted Gaussian potential policies [12] or joint limit avoidance (roughly close to linear behaviour). In this paper, we aim to extend the car washing and surface wiping examples provided in [13] to a circular wiping motion in the null space. Here the task constraint is to stay in contact with the surface with the end-effector z-axis aligned to the normal of the surface. Thus, the unconstrained policy encodes the circular wiping, and it is through the constraints that this motion is applied to surfaces of different orientations. Our first goal is to estimate the projection matrix of a set of experiments with the same surface orientation. Then, we reconstruct the null space policy from experiments with different surface orientations using the estimated constraints.

The unconstrained wiping policy performs a circular motion of a 2D sub-space with respect to the end-effector frame, on the x and y coordinates. It is important to note that even though the rotational component of the policy is defined in two dimensions (end-effector plane), the final policy is embedded in a 7-dimensional space – much higher than in other scenarios where a circular motion was learned (e.g. [13]). The actions are joint velocities $\mathbf{u} = \dot{\mathbf{q}}$ and the state is the joint configuration $\mathbf{q}$ together with a wiping center position $\mathbf{p}_{ts} \in \mathbb{R}^2$ with respect to the end-effector frame, so we have state $\mathbf{x} \in \mathbb{R}^9$. We formulate a candidate target velocity to perform the wiping movement as follows:

$$\dot{\mathbf{q}} = \mathbf{J}_{xy}^{\dagger}(\mathbf{q})\mathbf{R}(\theta_{ts})\frac{r}{\delta t}\begin{pmatrix} 1-\cos\omega\delta t - K_r\left(1-\frac{\|\mathbf{p}_{ts}\|}{r}\right) \\ \sin\omega\delta t \end{pmatrix} \tag{20}$$

where $\mathbf{J}_{xy}(\mathbf{q})$ is the Jacobian of the end-effector frame for x and y coordinates, δt is the sampling time, K_r is a gain ensuring that the wiping policy has the center of rotation at a distance equal to the given wiping radius parameter r. ω is the angular rate of the circular motion and θ_{ts} is the angle of the wiping center position with respect to end-effector frame and $\mathbf{R}(\theta_{ts})$ is a 2D rotation matrix. The terms to the right of the inverse of the Jacobian in (20) are basically generating a velocity with respect to the end-effector frame which ensures the circular wiping. We have chosen this policy because it is state dependent (no dependency on time) and it has an intuitive interpretation as a circular motion on a 2D surface.

In addition to this, the unconstrained policy also includes a *joint limit avoidance policy* for the 1^{st}, 3^{rd} and 7^{th} joints, see [1] for details. We would like to keep the 7^{th} close to zero, since it has no influence on the wiping and also to keep 1^{st} and 3^{rd} joints as close as possible to each other because these joints are essentially complementary on the "natural" wiping configuration. This ensures that the overall angle is distributed among those joints. In summary, all these aspects are added to the policy so that the resulting joint velocities for those joints include the terms:

$$\dot{q}_7 \leftarrow \dot{q}_7 - K_7 q_7 \tag{21}$$

$$\dot{q}_1 \leftarrow \dot{q}_1 - K_3(q_1 - q_3) \tag{22}$$

$$\dot{q}_3 \leftarrow \dot{q}_3 - K_1(q_3 - q_1) \tag{23}$$

where K_1, K_3 and K_7 are gains.

For realistic wiping simulation, the end-effector should maintain contact with the surface [20], so we want the end-effector plane and the wiping surface to be colinear, which imposes a constraint. In our simulation setting, we have considered a task to avoid long-term divergences, assumed to be known and corresponding task-component is substracted from raw observations, thus the constraint to be satisfied is:

$$\mathbf{A}(\mathbf{x})\mathbf{u}(\mathbf{x}) = \mathbf{b}(\mathbf{x}) \tag{24}$$

Let $\mathbf{T}_{ee}(\mathbf{x})$ be the end-effector transformation matrix:

$$\mathbf{T}_{ee}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_{ee} & \mathbf{y}_{ee} & \mathbf{z}_{ee} & \mathbf{p}_{ee} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

where $\mathbf{x}_{ee}$, $\mathbf{y}_{ee}$, and $\mathbf{z}_{ee}$ are vectors that encode the orientation of the end-effector, and $\mathbf{p}_{ee}$ is a vector that represents the position of the end-effector. We define $\mathbf{p}_s$ to be the closest point on the planar surface to the end-effector, and $\mathbf{n}$ the normal vector to that surface. We can then define the error of the task as:

$$\mathbf{t}_e(\mathbf{x}) = \begin{bmatrix} \mathbf{n}^T(\mathbf{p}_{ee} - \mathbf{p}_s) \\ \mathbf{n}^T\mathbf{x}_{ee} \\ \mathbf{n}^T\mathbf{y}_{ee} \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{26}$$

Method	Time[s]	$\log_{10}$nPPE	$\log_{10}$nPOE
literature [18]	470.89±152.34	-12.86±0.83	-13.52±0.54
proposed	0.039±0.009	-27.47±2.84	-28.31±2.58

TABLE I

LEARNING PROJECTION MATRIX $\mathbf{N}$: COMPUTATIONAL TIME AND ERRORS USING MATLAB R2015B AND A I7-4712MQ CPU @2.3 GHZ.

Therefore, the constraint matrix can be decomposed into a surface dependent term $\mathbf{\Lambda}(\mathbf{n})$ and a state dependent term corresponding to the Jacobian of the task $\mathbf{J}(\mathbf{x})$:

$$\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}(\mathbf{n})\mathbf{J}(\mathbf{x}) \tag{27}$$

with,

$$\mathbf{\Lambda}(\mathbf{n}) = \begin{bmatrix} \mathbf{n}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{n}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{n}^T \end{bmatrix} \in \mathbb{R}^{3\times 9} \tag{28}$$

$$\mathbf{J}(\mathbf{x}) = \nabla_{\mathbf{x}} \begin{bmatrix} \mathbf{p}_{ee}^T & \mathbf{x}_{ee}^T & \mathbf{y}_{ee}^T \end{bmatrix}^T \in \mathbb{R}^{9\times 7} \tag{29}$$

and the task is designed to ensure that task error converges to zero, that is, $\mathbf{b}(\mathbf{x}) = -K\mathbf{t}_e(\mathbf{x})$, being K a gain.

B. Performance Analysis

We analyse the performance of the proposed projection matrix learning in the constraint-space and the one proposed in [18]. We have created a dataset in which robot starts at 100 different random configurations and performs a wiping task under 4 different constraints, each recorded trajectory contains 150 data points and we assume that we have access to the null-space component of the action and that the Jacobian is known. For each experiment the robot is initialized to start at a configuration aligned with the surface (so that the task component is negligible throughout the trajectory) and separated from the wiping position a distance equivalent to the wiping radius.

Table I shows the results of learning the projection matrix with the method proposed in Section III in terms of the normalised projected policy error (nPPE) and the normalised projected observation error (nPOE). The analysis has been performed using $s = 3$ and $q = 7$ and mean and standard deviations have been obtained from 20 random initializations of each method. These results clearly show that our method outperforms the state of the art in both computational time and accuracy.

In addition to this, Figure 3 shows the effect of increasing the number of constraints, i.e.: the number of parameters to optimize by increasing the number of rows s in the constraint. In this particular setting, the constraint simply affects the rows of the robot's end-effector Jacobian, constraining the end-effector to move in specific directions, such as x, $\{x, y\}$, $\{x, y, z\}$, etc. This increases the number of rows of the constraint matrix and consequently the number of parameters to optimize, which is reflected in a steadily increasing nPOE and nPPE.

Now, we are interested in evaluating the policy learning performance of the method described in Section IV (constrained policy learning or CPL) and comparing the results

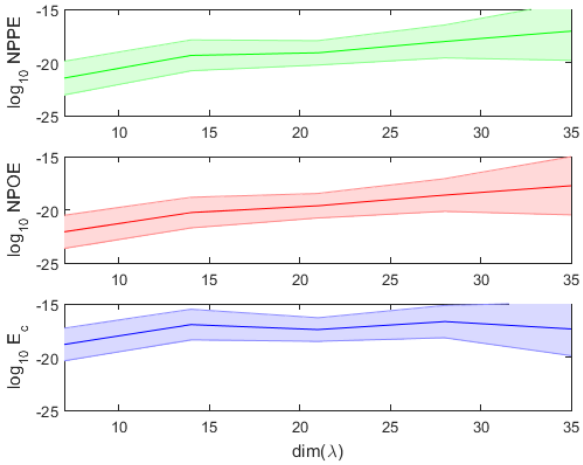
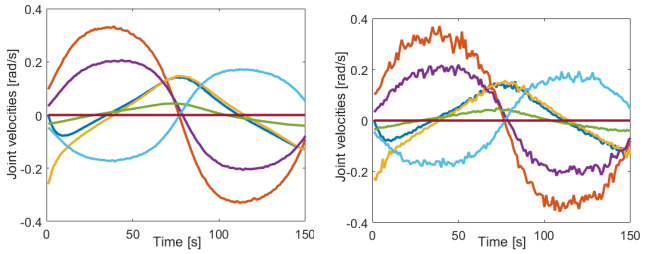


Fig. 3. Error plots of projection matrix accuracy for end-effector constraint: nPPE and nPOE and constraint-space error. Errors are in logarithmic scale.

with the method proposed in [12] (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 and the policy contains different noise levels (an additive Gaussian noise proportional to a percentage of the noise free joint value) varying from 2% to 20%. Figure 4 shows two cases for the same starting configuration with 2% of noise level and 20% of noise level.

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 for training and 1 for testing. This is the minimum number of observations required in order to resolve the policy ambiguity [12] if data were collected from exactly the same configuration. The policy uses a set of receptive fields based on RBFs that were randomly initialized using the k-means algorithm [14], roughly requiring 2250 local models per trial, although this value may change upon trials, until the whole space is covered with a weight of 0.7. In any case, 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. The results are shown in Table II, and



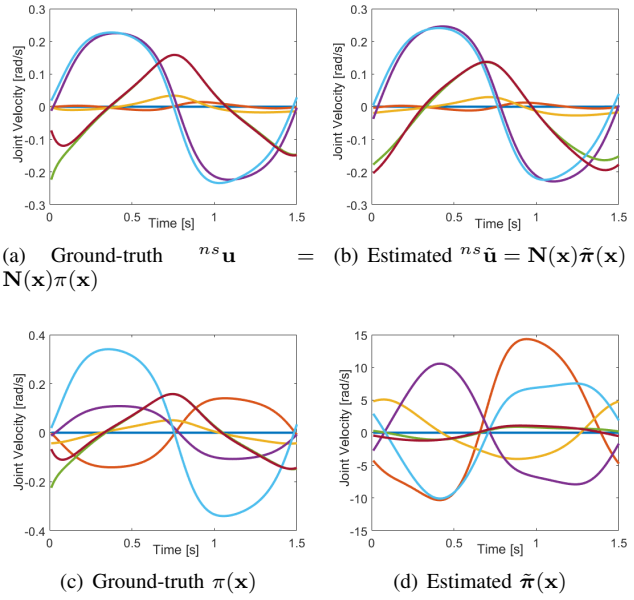
(a) 2% noise (b) 20% noise

Fig. 4. Example of the wiping policy with different noise levels.

Method	N_c	nUPE	nCPE	nCCE
LWCCL	8	98934±129330	20240±26286	2.5±3.8
	12	121223±120067	54208±79345	1.7±1.9
	20	129474±87438	90864±82809	2±1.7
LWCPL	8	309.3±113.2	27.1±22.6	8.4±6.3
	12	306.9±151.1	13.7±8.9	6.4±4.6
	20	300.9±102.9	8.5±4.2	4.9±3.0

TABLE II

WIPING POLICY LEARNING ERRORS (UNITS ARE 10^{-3}).



(a) Ground-truth ${}^{ns}\mathbf{u} = \mathbf{N}(\mathbf{x})\pi(\mathbf{x})$ (b) Estimated ${}^{ns}\tilde{\mathbf{u}} = \mathbf{N}(\mathbf{x})\tilde{\boldsymbol{\pi}}(\mathbf{x})$

(c) Ground-truth $\pi(\mathbf{x})$ (d) Estimated $\tilde{\boldsymbol{\pi}}(\mathbf{x})$

Fig. 5. Projected and unconstrained policy over unseen constraints: (a) Ground-truth projected policy, (b) Estimated projected policy, (c) Ground-truth policy (d) Estimated policy.

LWCPL outperforms LWCCL in terms of accuracy, since LWCCL is directly optimizing the NCCE – which is a lower bound. 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). However, the NUPE is still relatively high, which implies that there are cases in which the policy is ambiguous, in the sense that its projection in the null space is precise, but the underlying policy is very different from the ground truth. This aspect is clearly reflected in Figure 5.

In order to analyse the influence of a noisy policy on LWCPL, we have conducted an additional experiment with 25 different configurations, 12 different constraint orientations and 11 different noise levels (form 2% to 20%). Each experiment has been repeated for 100 trials. The results, shown in Figure 6, suggest that there is no explicit dependence on the policy noise level. This is expected, because the projected noise is Gaussian and the regression of each local model cancels it out by converging to its mean.

C. Experimentation

Now, we are interested in learning by demonstration using the proposed method. The setup consists of a KWR Kuka Robot with 7-DOF (previous version of the same robot used in simulation). We operate the robot in gravity compensation

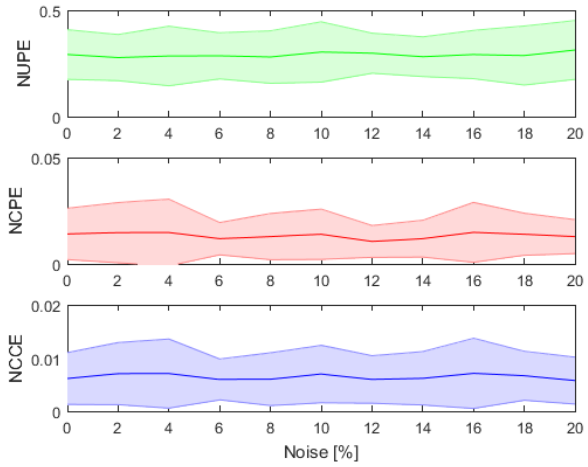


Fig. 6. Policy learning errors of a data set containing 25 different random configurations with 12 different contraint orientations. Noises are from 0% to 20% with 100 trials for each experiment.

mode, see Figure 1. During training, the robot is moved along a surface on which the human demonstrates the wiping motion. We repeat this 15 times at different locations on the surface for 12 different orientations of the surface. We show a subset of this data set in Figure 7. The length of each experiment may vary and trajectories do not describe perfect circles. The robot's end-effector is in contact with the surface and orthogonal to it during the whole motion segment, so that the task-space component is virtually zero.

Figure 8 shows the results of the learnt constraint errors (for training). As expected, the accuracy is less than in the simulated experiments. This is because the demonstrator introduced some residual task-space component, which is affecting the constraint estimation. However, despite the inconsistencies introduced by the demonstrator, the error metrics still show good results. Figure 9 validates that the projection of the policy lies on a surface. The nCPE and nCCPE errors for LWCPL were 0.2425 and 0.1106 respectively, using the variance of the observed input as normalization factor. On the other hand, the nCPE and nCCPE errors for the LWCCL method were 2.02 and 0.115, respectively. Figure 9 shows the learnt trajectories, in closed-loop, starting at the same configurations as the testing data. We would like to point out that the trajectories are not circular (some of them converge, other diverge, which is expected as the policy is not designed for infinite-horizon control), but in general they show a wiping motion pattern. This behaviour is due to limitations of the potential (state dependent) policy representation, rather than inaccuracies in the learning method.

VI. CONCLUSION

In this paper, we have addressed the problem of learning generalizable null space policies from constrained motion data. We have proposed a method for learning the constraints, which allows us to combine null space projection learning with null space policy learning. The main advantage of the method is its quadratic formulation, with quadratic constraints, which makes it suitable for specialized optimization

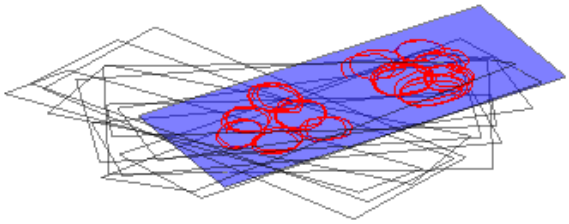


Fig. 7. Surface orientations used for training (transparent with border) and surface orientation used for testing together with testing trajectories.

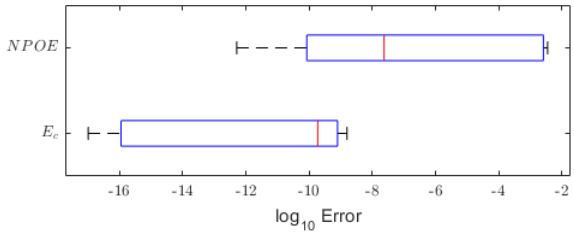


Fig. 8. Normalized errors of projection matrix estimation of experimental data using a LWR KUKA robot.

solvers. Since we can compute Hessians in advance from data, the convergence of the method is very fast and with low computational cost.

We have showed that this method can be used for learning null space policies, obtaining more accurate estimations than previous methods. The paper also addresses the problem of learning from demonstrations, in which we have been able to reproduce the basis of a wiping pattern using a real robot arm experiment.

It should be noted, though, that we assume that task space component is known or zero, which may introduce some biases to our method if this is not the case. To address this, we will investigate, in further research, how to split the null space component and the task-space component similar to [24] without using complex learning models.

References

[1] KUKA AG.

[2] 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.

[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] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 323–329, Nov 2012.

[5] 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.

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

[7] M. Freese E. Rohmer, S. P. N. Singh. V-rep: a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[8] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, page 0278364914521306, 2014.

[9] Michael Gienger, Herbert Janssen, and Christian Goerick. Task-oriented whole body motion for humanoid robots. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 238–244. IEEE, 2005.

[10] S. Hak, N. Mansard, O. Stasse, and J. P. Laumond. Reverse control for humanoid robot task recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(6):1524–1537, Dec 2012.

[11] 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.

[12] Matthew Howard. *Learning Control Policies from Constrained Motion*. PhD thesis, University of Edinburgh, 2009.

[13] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Methods for learning control policies from variable-constraint demonstrations. In *From Motor Learning to Interaction Learning in Robots*, pages 253–291. Springer, 2010.

[14] 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 Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, July 2002.

[15] 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. Springer, 2008.

[16] Hsiu-Chin Lin. *A novel approach for representing, generalising, and quantifying periodic gaits*. PhD thesis, University of Edinburgh, 2015.

[17] Hsiu-Chin Lin and Matthew Howard. Learning null space projections in operational space formulation. *arXiv preprint arXiv:1607.07611*, 2016.

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

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

[20] Jaeheung Park and Oussama Khatib. Contact consistent control framework for humanoid robots. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1963–1969. IEEE, 2006.

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

[22] 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.

[23] 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 *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 575–580. IEEE, 2006.

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

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

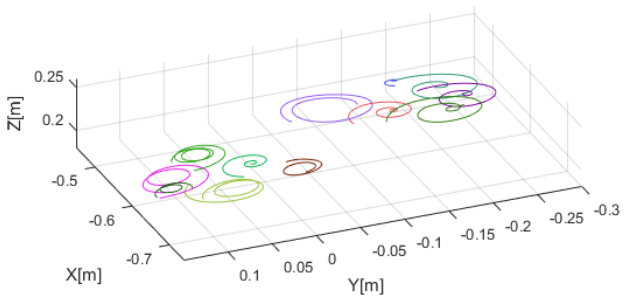


Fig. 9. Closed-loop trajectories with the learnt policy using LWCPL.