Learning Constrained Null Space Policies

Jorren Bosga

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2016

Abstract

Many complex movements of robotic systems with a high level of redundancy can be thought of as consisting of a task subjected by constraints. It is often desirable to retrieve this unconstrained task, or policy, in order to generalize it to new situations. Standard methods such as direct policy learning are unable to achieve this. A recently developed null space policy learning method, Constraint Consistent Learning (CCL), is able to learn unconstrained policies from constrained motion data and is able to generalize to unseen constraints. However, CCL requires strict conditions on the data in order to perform well, and its range of learnable policies is limited. In this thesis, we show that CCL can be combined with a related method that learns the constraints to perform Constrained Policy Learning (CPL). We show that this novel method has fewer data restrictions, performs well in complex scenarios and generalizes better than standard CCL. This is demonstrated by learning joint limit avoidance and non-linear surface wiping policies for a simulated kinematic 7-DOF robotic arm.

Acknowledgements

My sincere gratitude goes out towards Dr. Vladimir Ivan for the support and guidance he provided in his role as supervisor. I would also like to thank Dr. Leopoldo Armesto Angel, without whom this thesis would certainly have been of lesser quality. Both Vlad and Leo have played a major part in bridging the gap between the mathematical nature of the subject and the somewhat less mathematical nature of the author.

I would also like to thank Dr. Sethu Vijayakumar, for his confidence in me and for placing me with the SLMC group, which has made this project such an enriching and enjoyable experience.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jorren Bosga*)

Table of Contents

Chapter 1

Introduction

Movement in complex systems, such as humans, often contains a high level of redundancy since the degrees of freedom available to perform a task are usually higher than what is actually necessary to execute that task. In high degree of freedom (DOF) systems, this problem of executing a task has many solutions due to the high-dimensional state space. This allows for a certain flexibility in finding an optimal solution, so that this redundancy may be resolved according to some strategy that achieves a secondary objective. For instance, in a reaching movement with an arm, this might entail that the hand reaches its target with a minimal expenditure of energy, or does so in a way that ensures a comfortable position for the rest of the arm. Such approaches to redundancy resolution are employed by humans [1] as well as other redundant systems such as (humanoid) robots [2]. Performing a task while effectively handling redundancy is known as the optimal control problem (OCP). The optimal control problem is relevant to most areas of robotics, and has in the past been tackled by means of dedicated heuristics and solvers specific to each area. However, the problem remains open as no general solution has been found to date, and different situations and robots often call for a wide variety of approaches to redundancy resolution. Besides the variation in solutions, there are also several different methods of *finding* these solutions.

One broad approach to the optimal control problem is viewing redundancy resolution from a hierarchical perspective, 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 joint angles necessary to accomplish a certain task, and the null space component determines how the remaining redundant angles are used. This means that the null space may be used to achieve some secondary goal of lower priority. Once this secondary goal is encoded, some part of the null space will

be devoted to that task, leaving a smaller portion of null space. In this space, another tertiary task may be encoded, and so on. There are two ways of viewing this hierarchy: one might consider the primary task to be the task you wish to perform, such as reaching, and a lower-priority task to be a secondary goal such as avoiding joint limits [3], self-collisions [4], or kinematic singularities [5]. Alternatively, one can view the reaching task as the lowest priority, only to be performed after all the other top-level tasks are satisfied, perhaps because some of these tasks may be essential to the safe functioning of the system. So in a humanoid robot, the primary task might consist of joint limit avoidance. The second task may be balancing, and the final task with the lowest priority may be a reaching movement. Counter-intuitively, the main task the robots needs to perform (the reaching movement) takes place in the null space of the other tasks. It is therefore easier to think of the higher-priority tasks as constraints that need to be satisfied before the desired movement task can be executed.

This perspective becomes more intuitive when we also include external, or environmental constraints, that complement the class of task constraints outlined above. These constraints also restrict the space in which the desired task may be performed, such as the side of a saucepan restricting the stirring motion of the spoon to its radius or uneven ground restricting the cyclic motion of a runner's legs. Environmental constraints are often of a more static nature compared to their task constraint counterparts, hard conditions imposed on the desired task rather than another task of which the movements must be reconciled with the desired task. This makes them somewhat simpler to compute with, and they can be modeled hierarchically as well.

This hierarchical approach to redundancy has been widely used in robotics [6] in several variations, such as task sequencing [7], priority strategy [8] and hierarchical quadratic programming [9][10].

These methods all use explicitly formulated constraints and attempt to solve these constraints in the quickest way possible. However, the curse of dimensionality remains an issue, especially in applications such as model predictive control, where solutions to the whole hierarchy of constraints have to be recalculated very frequently [11]. As the amount of dimensions of the data increases, the volume of the space expands so quickly that the available data become sparse. This sparsity makes it difficult to find a globally optimal solution to the OCP, and a reliable solution requires the amount of data to grow exponentially with the number of dimensions. This makes finding it computationally expensive and time-consuming [12].

To circumvent this problem, one might attempt to learn a movement policy, a mapping from states to actions, that encodes behavior that is consistent with the set of constraints, instead of continuously calculating constraint-consistent solutions. An important advantage of using a policy is that after learning, it is computationally cheap. However, the policy has to be learned offline because it requires training examples to build the policy. These examples can be generated by means of demonstrations that either consist of human movement or movements made by the robot itself (or another robot) that is guided by a human. This approach is known as imitation learning, and one straightforward way to learn behaviors from this is through direct policy learning (DPL) [13]. This method uses the demonstrations to learn the control policy directly by supervised learning, evaluating the accuracy by taking the squared error between the result of the policy and the observed action. However, DPL has some limitations, such as the lack of a guarantee of stable behavior under variable start configurations. Additionally, one might argue that an offline approach simply shifts the problems with dimensionality to a different stage of the process. This is true, to the extent that there is still an equally large amount of data necessary to accurately learn a policy. However, the potential gain is in the possibility of learning a policy from data that is generalizable. Such a policy would need to be learned once and could be applied to many different situations, and this would relatively decrease the computational cost. Unfortunately, direct regression methods such as DPL generalize poorly because they only learn the commands adequate in a specific situation, so they mimic behavior rather than learn it [14].

As we have seen in other approaches to the OCP, the behavior can actually be represented as a combination of constraints and some action or policy in the null space, and in order to generalize to tasks that differ to some extent from the demonstrated tasks, it is helpful to learn these two components. In fact, by learning both the constraints and the policy, 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).

Recent research focusing on null space policy learning has led to the development of a method for learning movement policies from constrained motion data [15], as well as a method for learning the constraints imposed on these null space policies [16] in a way that allows the separate recovery of the unconstrained null space policy and the imposed constraints. Although this approach significantly outperforms direct policy learning methods, 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 movement policies.

This thesis extends this direction of research by providing an extensive analysis of the conditions this method requires to perform accurately. Additionally, we present a method that uses estimation of the constraints to learn the null space policy more accurately. We also formulate a policy that satisfies the assumptions made by the method on the nature of the data but is of higher complexity than earlier policies. Furthermore, we show the effectiveness of the combined approach by learning this complex policy in a high-dimensional system.

Section 2 contains a discussion of related work, in which we review the current state of the art concerning null space learning. We provide a largely chronological description of the development of the method, and treat the literature accordingly. The first half of the section is devoted to learning null space policies, and in the second half null space projection learning is presented. In Section 3 we elaborate on our methods and experiments, starting with a discussion of the nature of the data required to accurately learn null space policies. We then present our formulation of a complex surface wiping policy. The next subsections contain our experimental results and analysis, starting with learning a single step of the policy in random configurations, followed by learning the full policy in a fixed configuration and full policy learning in random configurations. Lastly, we discuss our results and provide directions for future research in Section 4.

Chapter 2

Background

In this section we discuss the state of the art in learning movement policies from constrained motion data to provide a theoretical framework for the research done in this thesis and to emphasize the novel contributions made. To do this, we divide the chapter in two parts.

In the first part, we focus on Constraint Consistent Learning (CCL) and its advantages over direct policy learning. We then present the core ideas of CCL and discuss its development from being a method limited to potential-based policies to a more general approach capable of learning generic state-dependent policies.

In the second part we look at null space projection learning, which is a method complementary to null space learning: instead of learning the policy, it is able to learn the constraints. We discuss its original use in analyzing walking gaits and its broader applicability to constrained motion data. We will also highlight the limitations of the method in its current form and we note that even though this approach has a number of attractive characteristics, its applicability to real-world scenarios is still somewhat limited.

2.1 Constraint Consistent Learning

Many movements can be thought of as some task being performed subject to environmental constraints or task constraints, or both. When opening a door for instance, the movement of the hand is restricted to the opening arc of the door because the hand is holding the doorknob. Another environmental constraint might be the surface of a table that restricts the movement of the hand as it is wiping the table. Task constraints are also common, as they encode conditions required to successfully perform a task.

Consider the action of pouring water from a bottle to a cup. The task constraint that ensures success encodes that the bottle must be oriented so that the water actually falls into the cup. Or, when cleaning a window, it is essential that the hand is constrained to actually being in contact with the surface of the window. Commonsensical as though these examples may seem, it is a very challenging task to discern and separate the null space task from the two types of constraints. In this section, we will discuss how constraint consistent learning addresses this challenge.

2.1.1 Learning Potential-Based Policies

When using direct learning methods on constrained motion data where the constraints may vary across the data, there are three main problems: (i) a non-convexity in the problem arises from the fact that constraints may change between observations. This means that using DPL will lead to an inaccurate solution that averages the policy over the observations of the policy under various constraints, (ii) there is degeneracy in the set of policies that could have produced the constrained movement [17], which means that there exist multiple policies that could have produced the movement. (iii) The resulting policy will not perform well in situations with previously unseen constraints because it has learned only how to behave in a specific constrained setting.

Constraint consistent learning begins with addressing these problems by considering control policies that can be separated into a task space and a null space component. This takes the form

$$\mathbf{u}(\mathbf{x}) = {}^{ts}\mathbf{u}(\mathbf{x}) + {}^{ns}\mathbf{u}(\mathbf{x}) \tag{2.1}$$

where ${}^{ts}\mathbf{u}(\mathbf{x})$ is the contribution to the observed action by the task space component and ${}^{ns}\mathbf{u}(\mathbf{x})$ is the part caused by the null space component. These two can in turn be represented differently to give

$$\mathbf{u}(\mathbf{x}) = \mathbf{A}(\mathbf{x})^{\dagger}\mathbf{b}(\mathbf{x}) + \mathbf{N}(\mathbf{x})\pi(\mathbf{x}) \tag{2.2}$$

where $\mathbf{A}^{\dagger}(\mathbf{x})$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}(\mathbf{x})$. $\mathbf{x} \in \mathbb{R}^{\mathcal{P}}$ represents the observed state, $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^{\mathcal{Q}}$ represents the observed action. The *task space policy* $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^{\mathcal{S}}(\mathcal{S} < Q)$ describes a task-dependent policy. The *constraint matrix* $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{\mathcal{S}\times\mathcal{Q}}$ projects the task-space policy onto the relevant part of the control space. The second term follows from the redundancy in the system (e.g. $\mathcal{S} < Q$), where $\pi(\mathbf{x}) \in \mathbb{R}^{\mathcal{Q}}$ is the *null space policy*, which resolves redundancy by allowing secondary objectives. The

null space projection matrix $\mathbf{N}(\mathbf{x}) \in \mathbb{R}^{Q \times Q}$ projects $\pi(\mathbf{x})$ onto the null space of $\mathbf{A}(\mathbf{x})$ and is defined as

$$\mathbf{N}(\mathbf{x}) = (\mathbf{I} - \mathbf{A}(\mathbf{x})^{\dagger}\mathbf{A}(\mathbf{x})) \tag{2.3}$$

For a data set of N data points, we assume that $\mathbf{A}(\mathbf{x}_n)$, $\mathbf{N}(\mathbf{x}_n)$, $\mathbf{b}(\mathbf{x}_n)$, and $\pi(\mathbf{x}_n)$ are not explicitly known. From this, we wish to learn the underlying unconstrained null space policy $\pi(\mathbf{x}_n)$.

The initial mathematical basis for the solution to this problem was presented in [17] in the context of Resolved Motion Rate Control (RMRC) and it uses Euclid's Theorem to recover the null space policy. To simplify the problem, the method assumes that ${}^{ns}\mathbf{u}_{null}(\mathbf{x})$ is known. It defines null space policy $\pi(\mathbf{x})$ as an arbitrary vector field of the form

$$\pi(\mathbf{x}) = \nabla_{\mathbf{x}} \times \Phi(\mathbf{x}) + \nabla_{\mathbf{x}}\varphi(\mathbf{x}) \tag{2.4}$$

where Φ and φ are vector and scalar potentials. These potential policies can be seen as attractor landscapes where minima of the potential function are attractors and gradient descent is greedily applied at every step. This means that these potential policies are only dependent on state, and not on time, which greatly reduces their complexity. Potential-based policies are therefore frequently used to represent null space policies in control for redundant manipulators [18] [3]. In the context of RMRC, (2.2) is represented as

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q},t)^{\dagger}\dot{\mathbf{r}} + \mathbf{N}(\mathbf{q},t)\pi \tag{2.5}$$

To illustrate the process of learning the policy, consider a system with a two-dimensional joint space, $\mathbf{q} \equiv (q_1, q_2)^T$ and a one-dimensional task space $r^i, i = 1, \ldots, n$. The Jacobian, defined as

$$\mathbf{J}^i(\mathbf{q}) = (\alpha_1, \alpha_2)^i = \alpha^i \tag{2.6}$$

is locally linear in the region of $\mathbf{q}$. Because in this example the joint space has two dimensions and the constraint has one, the null space policy is projected to a line in joint space with intersection r^i, as shown in the left pane of Figure 2.1. Note that because the matrix $\mathbf{N}$ projects the policy to the null space of the task, the task space and null space component are orthogonal by definition. When the constraint changes,

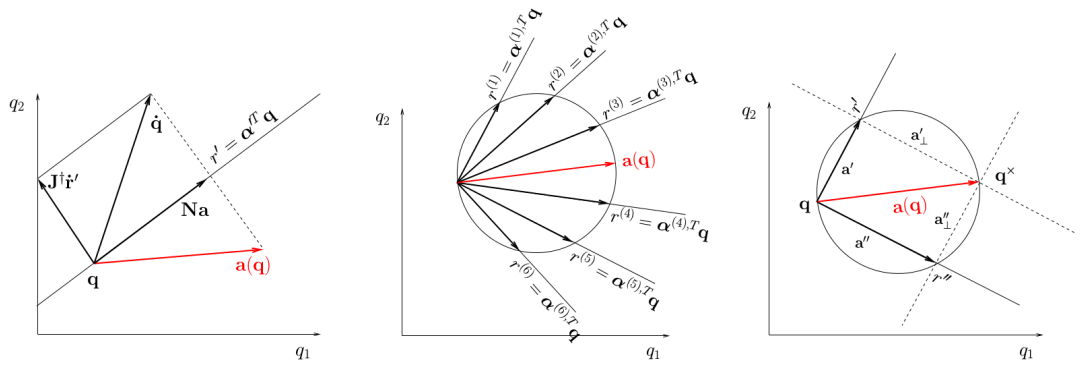


Figure 2.1: Under the task constraints in (2.6), the null-space policy is projected onto a manifold $r = \alpha^{(t)T}\mathbf{q}$ (left), orthogonal to the task space motion. Under multiple constraints the projected policy vectors lie inscribed in a hypersphere in state-space (centre). Euclid's Theorem can be used to reconstruct $\pi(\mathbf{q})$ given observations under different constraints (right) [17]. Note that in this image, $\mathbf{a}(\mathbf{q}) = \pi(\mathbf{q})$

so does the rotation of this line. This means that a number of differently constrained observations for one particular state will lie on a circle (or a hypersphere in higher dimensions) with diameter $||\pi(\mathbf{q})||$, shown in the middle pane of Figure 2.1. Because Euclid's Theorem states that any triangle inscribed in a semi-circle is a right-angle triangle, the null space policy $\pi(\mathbf{q})$ is given by the intersection of the lines orthogonal to *any* two projections **a**' and **a**", as shown in the right pane of Figure 2.1. In higher dimensions, we can construct planes normal to the projections if the observations form a basis set of the space, and solve for the intersection point.

This first attempt at a solution for learning policies from constrained motion data has two major shortcomings: the first is the assumption of having access to only the null space component of the observations, ${}^{ns}\mathbf{u}_{null}(\mathbf{x})$. This assumption is unlikely to hold in more realistic scenarios where there is little control over the input data. The second limitation is the requirement for a spanning set of projections to exactly reconstruct the policy. In simple two-dimensional systems with few possible states, having a basis set of observations for every state is still feasible, but for higher dimensions this quickly becomes unrealistic. This issue is dealt with in the literature by restricting the class of policies to conservative potential-based policies (e.g. policies without a rotational component) so that the policy can be learned through inverse optimal control.

The unrealistic assumption of having access to only the null space component is

sidestepped in [19] where only scenarios with 'hard' constraints are considered, where task space policy $\mathbf{b}(\mathbf{x}) = 0$ so that (2.2) simplifies to

$$\mathbf{u}(\mathbf{x}) = {}^{ns}\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\pi(\mathbf{x}) \tag{2.7}$$

This is a small change in perspective rather than a structural solution but it does make the method more realistic by restricting the scenarios it can consider. Also in [19], the method described above is extended by using local models and Euler integration to learn the potentials. These local potentials (which are effectively partial policies) are not necessarily aligned in a way that the global potential can be estimated accurately, so this paper borrows an alignment technique from the field of non-linear dimensionality reduction [20] that learns the optimal alignments offsets. Because of the conservative nature of the potential policies, there is a convex analytical solution to the optimal values of each local model. With these locally optimal models, it is then possible to estimate the global potential by means of regression. This is done by locally weighted projection regression (LWPR) [21], a locally weighted regression method that exploits lower-dimensional patterns in high-dimensional data. This approach leads to reasonably accurate prediction results in toy examples and in a simulation of the ASIMO humanoid robot.

In [22] this same approach described more extensively and applied to learning a quadratic potential function resulting in a reaching motion in the physical ASIMO platform, again with good results. However, even though it is reported that the policy was accurately learned in the 22-DOF ASIMO robot, the actual null space policy is defined in the considerably lower-dimensional endeffector space, which has 6 degrees of freedom. Additionally, the study highlights that constraints that prevent movement parallel to the motion of the unconstrained policy lead to inaccurate learning of the unconstrained policy because there are no observations that intersect on the plane of the policy, which highlights the importance of the nature of the learning data. These limitations notwithstanding, the method's preliminary success on a physical humanoid robot indicates its initial promise. It was therefore a logical next step to extend the class of 'learnable' policies from potential-based to generic, state-dependent policies as well.

2.2 Learning Generic State-Dependent Policies

The next study on the topic of null space policy learning extends the range of policies that can be learned with this method to include any policy that can be represented as a vector function of state, while still preserving the analytical solution to the local slopes of the policy [23]. Even though the policies that are learned in this contribution are still potential-based, the formulation is changed so that the null space policies do not *have* to be potential-based. In effect, what this means is that the extended formulation is capable of handling constrained policies that exhibit a non-zero *curl*, e.g. policies that encode periodic or rotational behavior.

The scenarios treated in this research also assume that the constraints are static (e.g. there is no active task space policy) so that again $\mathbf{u}(\mathbf{x}) = {}^{ns}\mathbf{u}(\mathbf{x}) = \mathbf{N}(\mathbf{x})\pi(\mathbf{x})$, which means that it is not necessary to first determine which aspects of the motion take place in the task space, and which in the null space. Practically, this may for instance describe a situation in which a robotic arm performs a wiping motion as its null space policy whilst being constrained to a planar surface by the projection matrix.

Instead of optimizing local potential offsets, this improved approach minimizes a risk function. In order to clearly present the logic behind this approach, we first discuss a number of risk functions that are considered in [23] as possible candidates. To begin with the simplest case, the *standard risk* function takes the distance between the observations and the approximated policy, which would obtain the best fit for the unconstrained policy if there were no constraints, or in data with the same constraint it would at least learn the best fit for the constrained policy under that particular set of constraints.

$$E_{direct}[\tilde{\pi}] = \sum_{n=1}^{N} ||\mathbf{u}_n - \tilde{\pi}(\mathbf{x}_n)||^2 \tag{2.8}$$

where $\tilde{\pi}$ denotes the estimated null space policy, and we will use the symbol ˜ to denote estimations throughout the rest of this thesis.

The issue with using this standard risk function is that the data are not uniform with regard to the constraints. If the data consisted of a movement policy executed under only a single constraint, it would be impossible to separate the two components, but it would also be reasonable to assume that it suffices to learn only the *constrained* policy. This would be achieved by minimizing this standard risk function. However, our data contains observations of the policy under an unknown number of different constraints, so this objective function would lead to the policy averaging associated with DPL. It is

therefore not considered a satisfactory risk function. Another option that is presented is to minimize the *unconstrained policy error* (UPE), which minimizes the distance between the true policy and the learnt policy:

$$E_{upe}[\tilde{\pi}] = \sum_{n=1}^{N} ||\pi(\mathbf{x}_n) - \tilde{\pi}(\mathbf{x}_n)||^2 \tag{2.9}$$

However, the true unconstrained policy is unknown by assumption, so it is not accessible for learning purposes. Even though this risk function is not useful for learning, if the ground truth policy is known (which is usually the case, in simulated scenarios at least), it can be used as an error measure for the accuracy of the learnt policy.

A different approach would be to optimize for the *constrained policy error* (CPE), which minimizes the distance between the observations and the learnt policy projected onto the null space of the task, which would give the fit that is most consistent with the constrained observations.

$$E_{cpe}[\tilde{\pi}] = \sum_{n=1}^{N} ||\mathbf{u}_n - \mathbf{N}(\mathbf{x}_n)\tilde{\pi}(\mathbf{x}_n)||^2 \tag{2.10}$$

This approach is also problematic, because the constraints $\mathbf{N}(\mathbf{x})$ are not explicitly known and approximating them may encounter problems of ambiguity, where it is unclear whether constraints are external and environmental, or encoded in the policy. However, when $\mathbf{N}$ is known for evaluation purposes, the CPE may be used to measure how well the learnt policy fits with the constrained observations. The CPE is related to the UPE as follows: the term inside the sum of the CPE can be written as

$$||\mathbf{u} - \mathbf{N}\tilde{\pi}(\mathbf{x})||^2 = ||\mathbf{N}\pi - \mathbf{N}\tilde{\pi}(\mathbf{x})||^2 = ||\mathbf{N}(\pi - \tilde{\pi}(\mathbf{x}))||^2 \tag{2.11}$$

and because $||\mathbf{N}|| \leq 1$ (since it projects to a subspace) and the norm property $||\mathbf{A} \cdot \mathbf{B}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{B}||$, the CPE is always less than the UPE.

So $\mathbf{N}$ is not known but the aim is to learn a policy that is consistent with the observations, e.g. a policy that can be projected to the null space in such a way that the observed commands are recovered. This means that we require a projection that projects onto the space of the actual projection matrix, or a subspace of it. One projection that does this is the 1-D projection onto the observed command itself, so, omitting subscripts for clarity: $\mathbf{P} = \hat{\mathbf{u}}\hat{\mathbf{u}}^T$, with $\hat{\mathbf{u}} = \frac{\mathbf{u}}{||\mathbf{u}||}$. Note that we are still working with the assumption that we have access to the null space component of the observations, so that also here $\mathbf{u}(\mathbf{x}) = {}^{ns}\mathbf{u}(\mathbf{x})$. The main benefit of this solution is thus that $\mathbf{u}$ is given so it is

no longer necessary to explicitly consider the constraint matrix [23][24]. Minimizing the inconsistency then becomes

$$E_i[\tilde{\pi}] = \sum_{n=1}^{N} ||\mathbf{u}_n - \hat{\mathbf{u}}_n \hat{\mathbf{u}}_n^T \tilde{\pi}(\mathbf{x}_n)||^2 \tag{2.12}$$

or, represented differently:

$$E_i[\tilde{\pi}] = \sum_{n=1}^{N} ||\mathbf{u}_n|| - \hat{\mathbf{u}}_n^T \tilde{\pi}(\mathbf{x}_n)^2 \tag{2.13}$$

Now we have a risk function to minimize, but we still need a form in which to represent the policy so that it can be learned with regression. The main approach advocated in literature concerning CCL is *locally linear policy models*, which we will discuss in the following section.

2.2.1 Locally linear policy models

The null space policy is likely to be non-linear, but it is often the case that the policy function we try to approximate is relatively linear in subspaces of the state and action spaces, similar to the slope of a potential-based policy on a small interval. It is therefore possible to fit a set of spatially localized linear models that individually minimize the inconsistency in (2.12). Each does so with respect to the data covered by each model's receptive field. These models are of the form $\tilde{\pi}_m(\mathbf{x}) = \mathbf{B}_m \bar{\mathbf{x}} = \mathbf{B}_m(\mathbf{x}^T, 1)^T$ where $\mathbf{B}$ is a matrix containing the weights of the linear model, models are indexed by m and a bias term is added to the state vector $\mathbf{x}$. We learn each model independently: for a linear model centred at $\mathbf{c}_m$ with a Gaussian kernel with variance σ^2, we would minimize

$$E_i(\mathbf{B}_m) = \sum_{n=1}^{N} w_{nm}(||\mathbf{u}_n|| - \hat{\mathbf{u}}_n^T \mathbf{B}_m \bar{\mathbf{x}}_n)^2 \tag{2.14}$$

which, when vectorizing so that $\mathbf{b}_m = vec(\mathbf{B}_m)$ and $\mathbf{v}_n = vec(\hat{\mathbf{u}}_n \bar{\mathbf{x}}_n^T)$, can be rewritten as

$$E_i(\mathbf{b}_m) = \sum_{n=1}^{N} w_{nm}(||\mathbf{u}_n|| - \mathbf{v}_n^T \mathbf{b}_m)^2 \tag{2.15}$$

The weight factors

$$w_{nm} = \exp(-\frac{1}{2\sigma^2}||\mathbf{x}_n - \mathbf{c}_m||^2) \tag{2.16}$$

weigh the importance of each observation $(\mathbf{x}_n, \mathbf{u}_n)$, giving more weight to nearby samples. The optimal slopes $\mathbf{B}_m$ in vector form are then retrieved by

$$\mathbf{b}_m^{opt} = \arg\min E_i(\mathbf{b}_m) = \mathbf{H}_m^{-1}\mathbf{g}_m \tag{2.17}$$

where $\mathbf{H}_m = \sum_n w_{nm}\mathbf{v}_n\mathbf{v}_n^T$ is the Hessian, and $\mathbf{g}_m = \sum_n w_{nm}||\mathbf{u}_n||\mathbf{v}_n^T$.

We now have a formulation of the optimal solution for fitting *one* local linear policy model to a small section of the trajectory that attempts to minimize inconsistency. For approximating the global policy, we combine the local linear models using the convex combination:

$$\tilde{\pi}(\mathbf{x}) = \frac{\sum_{m=1}^{M} w_m \mathbf{B}_m \bar{\mathbf{x}}}{\sum_{m=1}^{M} w_m}; \quad w_m = \exp(-\frac{1}{2\sigma^2}||\mathbf{x}_n - \mathbf{c}_m||^2) \tag{2.18}$$

Interestingly, this locally linear method is based on Receptive Field Weighted Regression (RFWR) [25] in the sense that each local model optimizes the inconsistency individually. However, the research in which this method was used to learn null space policies does not make use of other useful properties of RFWR, such as the individual receptive field size and shape adjustments. Instead, local models are initialized with static and uniform variance of their receptive fields, and are placed manually or using a K-means algorithm. This may lead to fitting issues, depending on the spread of the data and the dimensionality of the problem, as we will see in later sections.

This concludes a full description of the method presented in [23]. For an extensive discussion of both potential-based and generic null space policy learning see [26].

2.2.2 Extensions of Constraint Consistent Learning

Even though the procedures described above are in principle sufficient to learn certain null space policies, the solutions provided by this method might suffer from degeneracy in the set of models because the algorithm will fit a wrong policy if the data has too little variability in the constraints (the fewer constraints are available, the more possible policies there are). One solution to this is to add a secondary objective function to the inconsistency risk function that ensures that the learnt policies are at least consistent with the constrained observations [24]. Formally, we would minimize

$$E_2[\tilde{\pi}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \tilde{\pi}(\mathbf{x}_n)\|^2 \tag{2.19}$$

under the constraint that

$$\tilde{\pi} \in \arg\min_{\pi'} \left\{ E_i[\pi'] \right\} \tag{2.20}$$

where E_i is the inconsistency cost function (2.12) and π' refers to the policy that minimizes that function. The constraint ensures that the policies used to mimize the secondary objective function come from the set of candidate policies identified by the primary objective function.

Additionally, this method has also been extended in [27] with a procedure that is able to separate the null space component from the task space component in cases where there is a non-zero task space policy, so in cases where $\mathbf{u} = {}^{ts}\mathbf{u} + {}^{ns}\mathbf{u}$. They employ a method that is similar to the one used to separate the null space policy from the projection matrix, although it lacks an analytical solution. Instead, they estimate the null space component by means of iterative non-linear optimization. The method minimizes a cost function that is similar to (2.12), but it does not use the observed actions to project the null space policy because the observed actions now also contain a task space component. Instead, it uses, ${}^{ns}\tilde{\pi}_k$, the current estimate of the null space component for each subset of data points generated under a single constraint to project the observations and minimize the inconsistency. Formally, the cost function becomes

$$E_1[{}^{ns}\tilde{\pi}_k] = \sum_{n=1}^{N} ||\tilde{\mathbf{P}}_{k,n}\mathbf{u}_{k,n} - {}^{ns}\tilde{\pi}_k(\mathbf{x}_n)||^2 \tag{2.21}$$

where $\tilde{\mathbf{P}}_{k,n} = {}^{ns}\tilde{\pi}^{ns}_{k,n}\tilde{\pi}^{T}_{k,n}/||{}^{ns}\tilde{\pi}_{k,n}||^2$. $\mathbf{u}_{k,n}$ is the nth data point in the kth subset and we defined ${}^{ns}\tilde{\pi}_{k,n} = {}^{ns}\tilde{\pi}_k(\mathbf{x}_n)$. This procedure is essentially the same as earlier, where the null space component is approximated by projecting onto a 1-D space inside the null space. Because now the projection estimate depends on an *estimate* of the null space policy, its accuracy depends on the accuracy of the null space policy estimate so initializing multiple times will give different results. This procedure, together with CCL, has been applied successfully to analyzing the components of human walking gaits [28].

So there have been several additions to the method that extend the class of policies that can be learned, and that increase the accuracy of the learnt policy when it is subjected to the same constraints as in learning, effectively ensuring that performance is good in cases where the demonstrator was successful. In the next section we discuss an approach that is a fully complementary method rather than an extension of the policy learning method discussed in previous sections, namely null space projection learning, which learns the constraints.

2.3 Projection Matrix Learning

Recall that we assume that constrained movements can be decomposed into a task space and a null space component (e.g. (2.2)). For simplicity, we maintain the assumption that there is a static task space component which results in the relation $\mathbf{u} = \mathbf{N}(\mathbf{x})\pi(\mathbf{x})$. The aim of the approach outlined in previous sections is to learn null space policy $\pi(\mathbf{x})$. The approach discussed in this section achieves a complementary goal: it learns the projection matrix $\mathbf{N}(\mathbf{x})$. Where learning the unconstrained policy allows the policy to be generalized over unseen constraints, learning the constraints allows the application of the constraints to previously unseen policies. This can for instance be used in the scenario where one null space policy is known and the goal is to compare it with another, unknown null space policy. We assume that although this last policy is not known, the null space component $\mathbf{N}(\mathbf{x})\pi(\mathbf{x})$ is available. Estimating $\mathbf{N}(\mathbf{x})$ enables us to project the known null space policy to the same null space as the unknown policy, which allows us to compare $\tilde{\mathbf{N}}(\mathbf{x})\pi_k(\mathbf{x})$ with $\mathbf{N}(\mathbf{x})\pi_u(\mathbf{x})$ where $\tilde{\mathbf{N}}$ is the estimated projection matrix and π_k and π_u are the known and unknown null space policies. The first use of this method employs this strategy to compare unconstrained walking gaits and measure their (ab)normality [29]. We will first discuss the core ideas behind this approach as they were first introduced, after which we will review how the method has been expanded in subsequent research. We highlight the potential of combined constraint and policy learning, and note issues in high dimensional scenarios with estimating $\mathbf{N}(\mathbf{x})$ through the standard approaches.

2.3.1 Estimating the Projection Matrix

The core ideas about estimating the projection matrix are presented in [29] and [28], and we will discuss them here. As we defined in (2.7), $\mathbf{N}(\mathbf{x})$ is a projection matrix that projects a vector onto the image space of $\mathbf{A}(\mathbf{x})$. Since $\mathbf{u}$ is in the image space of $\mathbf{N}(\mathbf{x})$ by definition, we can state that $\mathbf{u} = \mathbf{N}(\mathbf{x})\mathbf{u}$. This equation gives a risk function over which $\mathbf{N}(\mathbf{x})$ can be optimized, which is written as

$$E[\tilde{\mathbf{N}}(\mathbf{x})] = \sum_{n}^{N} \left\| \tilde{\mathbf{N}}(\mathbf{x})\mathbf{u}_n(\mathbf{x}) - \mathbf{u}_n(\mathbf{x}) \right\| \tag{2.22}$$

However, it is unclear at this point how to perform the optimisation of the projection matrix, e.g. the search over the range of possible projections. The literature solves this problem by noting the relation between the projection matrix and the constraint matrix

given in (2.3), which shows that it suffices to find an $\mathbf{A}(\mathbf{x})$ that matches the direction of the constraint as closely as possible. Because only the direction of the constraints is important, the state independent part of a single constraint can be modeled as a unit vector $\tilde{\mathbf{A}} = [cos(\theta)sin(\theta)]$, where $\theta \in [0, \pi]$ covers all possible cases of $\mathbf{N}(\mathbf{x})$ [28]. It is important to note that π refers to the mathematical constant in this case, and that this definition of the constraint $\tilde{\mathbf{A}}$ is not state dependent. The literature does not clearly make this distinction, but there are in fact two constraint matrices: one that is state dependent and one that is not. We will return to this notion in Section 2.3.2.

These preliminary formulations are expanded to constraints of higher dimensions, and a more extensive description of the constraint matrix is provided in subsequent work [30], which maintains the same interpretation of the constraint matrix. First, the risk function in (2.22) is reformulated in terms of the constraint matrix using (2.3), so that

$$\begin{aligned} ||\mathbf{u}_n - \tilde{\mathbf{N}}\mathbf{u}_n||^2 &= ||\mathbf{u}_n - (\mathbf{I} - \tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}})\mathbf{u}_n||^2 \\ &= ||\mathbf{u}_n - \mathbf{u}_n + \tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\mathbf{u}_n||^2 \\ &= ||\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\mathbf{u}_n||^2 \end{aligned} \tag{2.23}$$

where $\tilde{\mathbf{A}} \in \mathcal{R}^{S\times Q}$ is an estimate of the constraint matrix $\mathbf{A}$. We can expand the norm: $||\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\mathbf{u}_n||^2 = \mathbf{u}_n^T(\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}})^T\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\mathbf{u}_n$, and using the identities of the pseudo-inverse $(\mathbf{A}^\dagger\mathbf{A})^T = \mathbf{A}^\dagger\mathbf{A}$ and $\mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger$, the objective function (2.22) can be expressed in a simplified form:

$$E[\tilde{\mathbf{N}}] = \sum_{n=1}^{N} \mathbf{u}_n^T\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\mathbf{u}_n \tag{2.24}$$

This formulation has the advantage that no knowledge of the null space policy $\pi(\mathbf{x})$ or the true projection matrix $\mathbf{N}$ is required. To elucidate the process of finding $\mathbf{A}$ by investigating possible directions of the constraint represented by θ, we will review the structure of the constraint matrix as presented in the literature in the next section.

2.3.2 Representation of the Constraint Matrix

In the previous sections, we have established that the estimate of the constraint matrix $\mathbf{A}$ can be represented by a system of orthogonal unit vectors and that the optimal $\mathbf{A}$ can be found by searching the range of angles $\theta \in [0, \pi]$. In this section we will further elaborate on this notion, as described in [30].

It should be noted at this point that the preliminary literature on projection matrix estimation published before [31] introduces the constraint and projection matrices as being dependent on state, but this dependence is dropped. The constraint matrix is then in fact treated as if it is not state-dependent, as demonstrated by its decomposition into orthogonal constraint vectors dependent on θ.

The most likely explanation for this apparent inconsistency is that there are in fact two scenarios: one in which the Jacobian relating the joint space to the task space is known, and one in which it is not. This is implied in the experiments section of [30] and is further described in [31]. In the first case, the constraint matrix can be decomposed into

$$\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}\mathbf{J}(\mathbf{x}) \tag{2.25}$$

where $\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{\mathcal{T}\times Q}$ is the Jacobian of the task, which is assumed known and $\mathbf{\Lambda}$ is the constraint to be learned. Then, the estimate $\tilde{\mathbf{\Lambda}}$ is assumed to consist of a set of $\mathcal{S}$ orthonormal vectors

$$\tilde{\mathbf{\Lambda}} = \begin{bmatrix} \hat{\boldsymbol{\lambda}}_1^T \\ \hat{\boldsymbol{\lambda}}_2^T \\ \vdots \\ \hat{\boldsymbol{\lambda}}_{\mathcal{S}}^T \end{bmatrix} \tag{2.26}$$

where $\hat{\boldsymbol{\lambda}}_s^T = (\lambda_{s,1}, \lambda_{s,2}, ..., \lambda_{s,Q})$ corresponds to the sth constraint of the observations and all constraints are orthogonal, e.g. $\hat{\boldsymbol{\lambda}}_i \perp \hat{\boldsymbol{\lambda}}_j$ for all $i \neq j$. Also note that $\hat{\boldsymbol{\lambda}}_i = \frac{\boldsymbol{\lambda}_i}{\|\boldsymbol{\lambda}_i\|}$ is the unit vector of the constraint $\boldsymbol{\lambda}_i$. Since only unit vectors appear in the equation, the projection matrix is independent of the magnitude of each of the constraints so we only need to approximate the direction of the constraint vectors.

Using this representation and (2.25) and substituting in (2.24), we can write the form of the projection matrix estimate as

$$E[\tilde{\mathbf{N}}] = \sum_{n=1}^{N} \mathbf{u}_n^T (\tilde{\mathbf{\Lambda}}\mathbf{J}_n(\mathbf{x}))^{\dagger} (\tilde{\mathbf{\Lambda}}\mathbf{J}_n(\mathbf{x}))\mathbf{u}_n \tag{2.27}$$

From here, the optimal $\mathbf{\Lambda}$ is found by iteratively searching the space of $\boldsymbol{\lambda}_s$ that minimizes (2.27). Each $\boldsymbol{\lambda}_s$ in turn can again be modeled in terms of angles θ, over which it is possible to optimize. However, even though the optimization is not state dependent, it is still non-linear and thus has to be performed using numerical optimization.

The other scenario is when the Jacobian is not known. In this case, it is possible to define state dependent constraint matrix $\mathbf{A}(\mathbf{x})$ as a set of $\mathcal{S}$ unit vectors in the same way as we defined $\mathbf{\Lambda}$:

$$\tilde{\mathbf{A}}(\mathbf{x}) = \begin{bmatrix} \hat{\mathbf{a}}_1(\mathbf{x})^T \\ \hat{\mathbf{a}}_2(\mathbf{x})^T \\ \vdots \\ \hat{\mathbf{a}}_{\mathcal{S}}(\mathbf{x})^T \end{bmatrix} \tag{2.28}$$

In the same way as above, the optimum can be found using angles, but in this case they are state dependent and in the literature, they are represented by a set of radial basis functions (RBFs), e.g. $\boldsymbol{\theta}_s(\mathbf{x}_n) = \mathbf{W}_s \beta(\mathbf{x}_n)$ where $\mathbf{W}_s \in \mathbb{R}^{(\mathcal{S}-1)\times\mathcal{M}}$ is a matrix of weights, and $\beta(\mathbf{x}_n) \in \mathbb{R}^{\mathcal{M}}$ is a vector of $\mathcal{M}$ fixed basis functions.

It is clear that this approach is computationally much more expensive than the scenario in which the Jacobian is known. In high-dimensional problems, many RBFs will be necessary and iterating over the search space of $\boldsymbol{\theta}_s(\mathbf{x}_n)$ will be expensive as well. Even though both scenarios can be computationally expensive, there have been attempts at combining projection matrix estimation with constraint consistent learning.

2.3.3 Combined Constraint and Policy Learning

Projection matrix learning as described in Section 2.3 has been used in [30] in combination with CCL as presented in Section 2.2 to show that generalisation can be achieved across constraints (e.g. applying the learnt policy to novel constraints) as well as within constraints (e.g. applying novel policies to a learnt constraint). To demonstrate this, a linear attractor policy and a joint limit avoidance policy were learned in a simulated 7-DOF robotic arm from motion data generated under four different constraints using CCL. The linear policy was accurately estimated, but the joint limit avoidance policy reproduction contained significant errors. Then, a new constraint was introduced and learned using projection matrix learning. Accuracy was measured against the ground truth observations, $\mathbf{N}\pi$, both for the accuracy of the estimated projection matrix by itself, $\tilde{\mathbf{N}}\pi$, and the two estimations together, $\tilde{\mathbf{N}}\tilde{\pi}$. The projection matrix was estimated with high accuracy, and the observations under the linear policy were recovered accurately, but the errors in the estimated joint limit avoidance policy were also visible using the estimated projection matrix.

These preliminary results show that it is possible to accurately learn the null space

policy as well as the constraints from constrained motion data for simple policies. This is promising, especially considering the little information necessary and the potential applications of generalizable policies and constraints. However, the method still has a number of aspects that can be improved, such as the policy complexity and the description of the method and the data it requires. In the next section, we address some of these remaining challenges.

Chapter 3

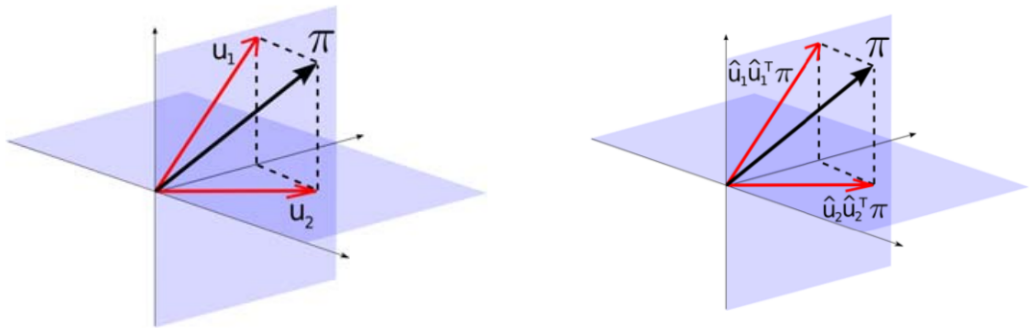
Methods and Experiments

In this thesis, we will focus only on the core approach to policy learning described in Section 2.2. The state-dependent approach is more generic than the potential-based method and performs as well or better in terms of NUPE and NCPE (see results in [22] and [23]), so it seems a logical decision to add to the former. Since there is still much complexity to be explored in policies that contain a static task space component, we will restrict ourselves to this class of policies. Lastly, the additional cost term for improved robustness described in (2.19) is not directly in line with our research objectives and its use may obscure the core findings presented in this thesis, so therefore we will not use it.

Our main aim is to assess the performance of null space policy learning in more realistic, complex scenarios to further document the bounds of the class of learnable policies. To do this we first provide an explicit description of the nature of the data required for accurate learning, as the use of suitable data is crucial to good performance of the method. Additionally, we provide a detailed description of the policy we aim to learn and discuss how it compares to previously learnt policies. Then, we present some improvements to the projection matrix learning procedure and we show how the projection matrix can be used to learn policies more accurately. We validate these contributions in data sets with varying characteristics and complexity and present our experimental results.

3.1 Characterizing the Requirements for Good Data

Constraint consistent learning relies heavily on assumptions of orthogonality in the data. First of all, it assumes that the observations can be decomposed into orthogonal



(a) (b)

Figure 3.1: **Geometric interpretation of CCL.** (a) shows two differently constrained observations for one state. (b) shows how the null space policy should be consistent with the observations when projected to their subspace [23].

task space and null space components, e.g. $\mathbf{u} = {}^{ts}\mathbf{u} + {}^{ns}\mathbf{u}$ where ${}^{ts}\mathbf{u} \perp {}^{ns}\mathbf{u}$. Then, we assume that constraint matrix $\mathbf{A}(\mathbf{x})$ consists of a set of orthonormal unit vectors so that it is full row rank. Not satisfying these assumptions will lead to large errors in estimation of both the unconstrained policy and the projection matrix. In this thesis, we generated the data so that it was mathematically guaranteed that these assumptions were met.

Depending on the purpose of the research, having multiple constrained observations is important. If the goal is to learn the null space policy under an invariant constraint, it suffices to have only one constrained observation per state. The result would be a policy that is consistent with the constrained observations, which with only one constraint would amount to direct policy learning. Of course, because the unconstrained policy is indistinguishable from the projected policy this would lead to a poor estimation of the true unconstrained policy, illustrated by a high normalised unconstrained policy error.

As shown in Figure 3.1, it is necessary to have at least two constrained observations per state in order to make an attempt at estimating the unconstrained policy. Although this suffices in a two-dimensional system, this does not imply that the correct null space policy will also be learned in higher dimensions. This is due to the issue of degeneracy in the set of policies that are consistent with the observations, as illustrated by Figure 3.2. For any set of constrained observations, there are multiple candidate policies that could have produced the observed movement, illustrated by π and π' in the figure. This is because the projection eliminates components of the unconstrained policy that are

orthogonal to the image of the projection matrix so that the component of the policy is undetermined in that direction [23]. This remains problematic throughout, as we are never able to obtain sufficient information about the unconstrained policy to guarantee that it is fully reconstructed. However, as the number of constrained observations for any given state increases, the set of candidate policies becomes smaller. This means that generating the motion data under as many different constraints as possible is crucial for ensuring low unconstrained policy errors.

In direct policy learning and in imitation learning in general, motion data is often generated through demonstrations, which are either performed by a teacher or the robot itself. These methods are not appropriate for the purposes of this thesis, because they do not provide access to the ground truth policy (necessary to calculate the NUPE), the ground truth projection matrix, and the ground truth null space component (required for calculating the NCPE). Since the work in this thesis is fairly preliminary, it requires proper validation measures. It is therefore convenient, if not necessary, to use simulated data for our experiments. This has the added benefit that generating a large number of samples is fast and can be done in a controlled and exact fashion. This feature is especially important in our case because the performance of standard CCL is highly dependent on the quality of the data. Any credible improvements on the method should therefore be tested on highly controlled data first.

However, even in simulation it is still challenging to obtain good data. Especially achieving a high number of constrained observations becomes an issue when using realistic data consisting of trajectories generated by the same null space policy under a number of different constraints. It is common to generate these trajectories from random start poses in order to obtain a data set that is rich with respect to the state space. However, if the state space is large, it is likely that there will be many configurations that are paired with only few constrained actions. Even if the simulated robot would always start in the same pose, it is likely that under different constraints, the trajectories will diverge so much that the resultant configurations will lack sufficient constrained observations. Consider a robotic arm: suppose trajectories are generated according to a simple linear point attractor policy and the endeffector is constrained in one of the x, y, or z planes. If for the same starting pose the policy is executed under each of these three different constraints, the resulting respective configurations will likely be very different over the course of a trajectory. So even though there is variation in the observed actions for the initial configuration, this variation quickly leads to many

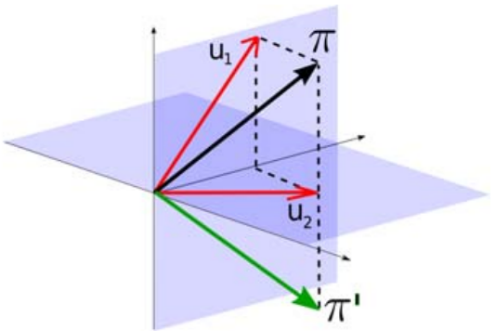


Figure 3.2: **Degeneracy in the set of policies.** If only $\mathbf{u}_2$ is observed, both π and π' would be valid unconstrained policy estimates [23].

different recorded states that have only a single observed action.

This issue may arise in any data set containing full trajectories, and it seems that this is a problem difficult to avoid when considering high-dimensional scenarios. In our experiments, we attempted to account for this problem by generating a data set consisting of only the first step of each trajectory in Section 3.5.3, which allows for variation in the observations even with random initializations of the plant. Additionally, we generated another data set consisting of full trajectories performed from the same start position under a large number of constraints in order to achieve enough variation, which is described in Section 3.5.4.

We expect do not expect these scenarios to completely remove problems with degeneracy. However, not being able to guarantee a good approximation of the unconstrained policy does not mean that the learnt policy cannot generalize over constraints. In Figure 3.2, both policies are consistent with constrained observation $\mathbf{u}_2$ and may actually generalize well to another constraint. This is also the case in higher-dimensional cases, which we will demonstrate in our experiments.

3.2 Description of the Policy and Constraints

In the literature, most of the policies that have been learned with this method in low dimensions (≤ 6 DOF) have been linear, sinusoidal, limit cycle, or inverted Gaussian potential policies [15]. In higher-dimensions (> 7 DOF), the range of policies thus far is restricted to (nearly) linear joint limit avoidance policies [24]. In this thesis, we aim to extend the current body of learnable policies to more complex policies in high dimensions. To do this, we take inspiration from the car washing [26] and surface

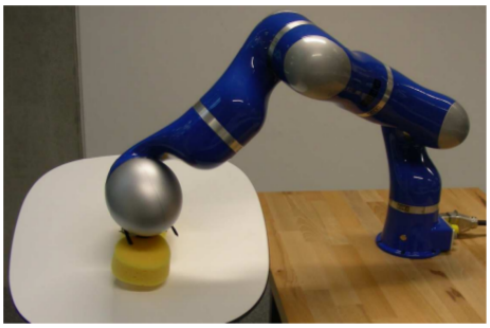


Figure 3.3: **Antropomorphic 7-DOF DLR Light Weight Robot**. In our experiments we use a kinematic simulation of this platform [24]. The robot is seen here holding a sponge to the surface of a table, which is fairly similar to the policy we are proposing.

wiping [24] examples provided in the literature and define a policy that consists of a circular wiping motion. In its unconstrained form, this policy only encodes the circular wiping, and it is through the constraints that this motion is applied to surfaces of different orientations. Our goal is to construct this policy in a way that is as realistic as possible. This means that besides its circular wiping motion it also contains a number of additional terms, making it highly non-linear and more complex than the joint limit avoidance policy used in the literature [23][24]. It is also important to note that even though the rotational component of the policy is defined in two dimensions, the final policy is embedded in 7-dimensional space. This is higher than in other cases where a circular motion was learned (e.g. [26]). The platform that we use to execute the policy is a kinematic simulation of the 7-DOF DLR Light Weight Robot-III (Figure 3.3), which has also been used extensively in the literature [23][24].

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. For a given wiping center position $p_{ts} \in \mathbb{R}^2$ with respect to the end-effector frame, we formulate a candidate target position to perform the wiping movement: p^{ee}. We define this as

$$p^{ee} = R(\theta_{ts}) \begin{pmatrix} r(1-\cos\omega\delta t) - K_r(r - \|p_{ts}\|) \\ r\sin\omega\delta t \end{pmatrix} \tag{3.1}$$

where δt is the sampling time, K_r is a gain ensuring that the wiping policy has the center of rotation at a distance of 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:

$$R(\theta_{ts}) = \begin{pmatrix} \frac{p_{ts,x}}{\|p_{ts}\|} & \frac{p_{ts,y}}{\|p_{ts}\|} \\ \frac{p_{ts,y}}{\|p_{ts}\|} & -\frac{p_{ts,x}}{\|p_{ts}\|} \end{pmatrix} \tag{3.2}$$

The velocity vector to reach p^{ee} for one step is:

$$\dot{p}^{ee} = \frac{p^{ee}}{\delta t} \tag{3.3}$$

The joint velocities are computed using the Jacobian with respect to the end-effector frame for the x and y coordinates, which we define as $J_{x,y}(q)$:

$$\dot{q} = J_{x,y}^{-1}(q)\dot{p}^{ee} \tag{3.4}$$

In addition to this, the wiping policy also includes a joint limit avoidance policy for the 7^{th} and last joint. Since the orientation of this joint does not affect to the wiping, we want to keep it as close as possible to 0 to avoid joint limits. In addition to this, we also would like to keep joints 1 and 3 as close as possible to each other because these joints are complementary. This ensures that the overall angle is distributed among those joints. In summary, we include these aspects in the policy so that the resulting joint velocities are:

$$\dot{q}_7 = \dot{q}_7 - K_7 q_7 \tag{3.5}$$

$$\dot{q}_1 = \dot{q}_1 - K_3(q_1 - q_3) \tag{3.6}$$

$$\dot{q}_3 = \dot{q}_3 - K_1(q_3 - q_1) \tag{3.7}$$

An important feature of this policy is that it is entirely state-dependent as opposed to also being dependent on time. In this sense it is similar to the potential based policies in the literature, because the policy can be seen as a gradient over the state space. In the literature generic null space policies are generally introduced as being time-dependent, although it is unclear whether this notion is actually adhered to in experiments. Either way, a time dependent policy would require the alignment of the time indices of the true and learnt policies. This massively increases the state space: in the state-dependent case, we have state-observation pairs $(\mathbf{x}_1, \mathbf{u}_1)$ and $(\mathbf{x}_1, \mathbf{u}_2)$, which both contribute to the variation in observations for state $\mathbf{x}_1$. In the time-dependent case we may have the pairs $(\mathbf{x}_{1,t-1}, \mathbf{u}_1)$ and $(\mathbf{x}_{1,t+1}, \mathbf{u}_2)$ where t stands for the time step. Now, the two observations $\mathbf{u}_1$ and $\mathbf{u}_2$ no longer contribute to the richness of same state $\mathbf{x}_1$ because the two states $\mathbf{x}_{1,t-1}$ and $\mathbf{x}_{1,t+1}$ are considered different because they occur at a different time step. This leads to an increased richness in state space, but to a decreased level of variation in the constrained observations per state, which is important for accurate learning.

This time-dependence may be necessary in cases where the trajectory has to cross the same point in multiple instances at specific points in time. However, our policy does not require this feature, so in our case it suffices to learn only a mapping from states to actions, without the dependence on time.

So now we have a policy that performs a small amount of joint limit avoidance and primarily executes a circular wiping motion in the *xy* plane of the end-effector, regardless of the pose of the robot. Of course, to keep the complete setting as realistic as possible, we would like the policy to wipe a number of different surfaces. These are represented by the constraints and consist of planar surfaces generated at various orientations. This problem is more complex than the wiping scenario described in for instance [24] where (i) no null space policy is defined at all because the data there is generated by human demonstrations, and (ii) in the training data the end-effector (e.g. the human hand) is always already in contact with the wiping surface. In our case, we have a policy that executes the wiping motion in a natural way, and a plane on which the wiping should take place. For realistic wiping, the end-effector should maintain contact with the surface [32], so we want the distance between the end-effector and the planar constraint to be zero. We also want the end-effector to be orthogonal to the surface to simplify the execution of the policy. However, there is no term that actually ensures that these constraints are satisfied. It is therefore necessary to introduce an active task space component that is capable of enforcing these conditions. We will briefly define it here.

Consider a transformation matrix $\mathbf{T}$, given by

$$\mathbf{T} = \begin{bmatrix} \boldsymbol{\theta}_x & \boldsymbol{\theta}_y & \boldsymbol{\theta}_z & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

where $\boldsymbol{\theta}_x$, $\boldsymbol{\theta}_y$, and $\boldsymbol{\theta}_z$ are vectors that encode the orientation of the end-effector, and $\mathbf{t}$ is a vector that represents the position of the end-effector. We define $\mathbf{ps}$ to be the position of a target point on the planar surface to wipe. The purpose of task space policy $\mathbf{b}(\mathbf{x})$ is to ensure that the end-effector is in contact with the surface, and that the orientation of the end-effector is orthogonal to the wiping surface. We can thus define $\mathbf{b}(\mathbf{x})$ as

$$\mathbf{b}(\mathbf{x}) = -K \cdot \begin{bmatrix} \mathbf{n}^T(\mathbf{t} - \mathbf{ps}) \\ \mathbf{n}^T \boldsymbol{\theta}_x \\ \mathbf{n}^T \boldsymbol{\theta}_y \end{bmatrix} \tag{3.9}$$

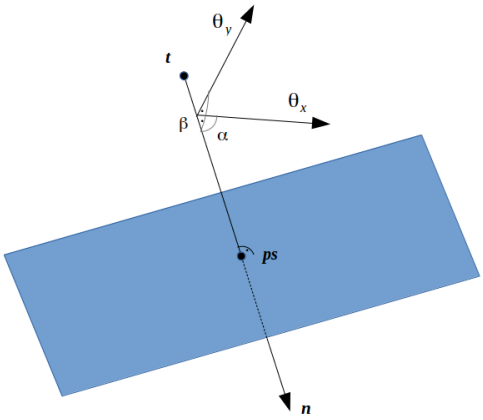


Figure 3.4: **Visualization of task space policy components.** The task space policy is responsible for ensuring that the distance between **t** and **ps** is minimized, and that θ_x and θ_y remain orthogonal to **n**.

where K is some constant and $\mathbf{n}$ is the vector normal to **ps**. All terms in the vector component of $\mathbf{b}$ should be 0. The third orientation term, $\boldsymbol{\theta}_z$, can be omitted because only two axes are required to ensure orthogonality. As shown in Figure 3.4, the angles α and β between the normal vector $\mathbf{n}$ and the two orientation vectors should be 90° so that $\cos\alpha = \mathbf{n}^T\boldsymbol{\theta}_x = 0$ and $\cos\beta = \mathbf{n}^T\boldsymbol{\theta}_y = 0$.

In a simulation, consider a setup where neither of the two constraints (zero distance and orthogonality) are satisfied. The effect $\mathbf{b}(\mathbf{x})$ will have is that it will force the end-effector towards the surface while also adjusting its orientation. At the same time, the wiping policy will already be executed so visually this will give the impression of the robot arm spiralling down to its designated wiping surface. Obviously, this formulation of the task space and null space is problematic for the scope of this thesis, since we only consider cases where the task component is static. However, we consider a policy that encodes natural behavior to be more valuable than one that purely demonstrates the feasibility of our approach in an idealized, unrealistic setting. This means that in these simulated experiments, we assume that we have access to the null space component. It is important to note that this is a reasonable assumption because there are methods available for the recovery of the null space component (see Sections 2.2.2 and 4.2). Unfortunately, estimating it is beyond the scope of this thesis.

3.3 Using the Projection Matrix for Policy Learning in High Dimensions

As described in Section 2.2, standard constraint consistent learning minimizes the inconsistency error, which consists of the policy estimate projected to a 1-D subspace of the null space and the observations. Having access to (an estimate of) the projection matrix allows the minimization of the constrained policy error, which measures the distance between the observations and the learnt policy projected onto the (estimated) null space of the task. By definition, the inconsistency error is a lower bound to the constrained policy error because it projects the policy to a subspace of the null space. Using the CPE for estimation instead should therefore lead to more accurate policy learning. Formally, CCL minimizes

$$E_i[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} ||{}^{ns}\mathbf{u}_n - {}^{ns}\hat{\mathbf{u}}_n {}^{ns}\hat{\mathbf{u}}_n^T \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)||^2 \tag{3.10}$$

whereas with an estimate of the projection matrix, we minimize

$$E_{cpe}(\hat{\boldsymbol{\pi}}) = \sum_{n=1}^{N} ||{}^{ns}\mathbf{u}_n - \tilde{\mathbf{N}}(\mathbf{x}_n)\hat{\boldsymbol{\pi}}(\mathbf{x}_n)||^2 \tag{3.11}$$

where we still assume that $\mathbf{u} = {}^{ns}\mathbf{u}$. In most earlier work on null space policy learning, an estimate of the projection matrix was not available, which explains the necessity of using the lower bound inconsistency error. When estimating $\mathbf{N}$ is a possibility, it can be used in combination with null space policy learning. However, in the literature this has so far only taken the form of estimating the projection matrix and the policy separately (with standard CCL), and combining both estimates. Here, we are directly using the projection matrix to improve the accuracy of constraint consistent learning. Since we are now minimizing the Constrained Policy Error, we will refer to this approach as Constrained Policy Learning (CPL). Even though this improvement is conceptually straightforward, we will see in the next section that CPL could potentially be too computationally expensive, so we propose a faster method for projection matrix estimation.

3.3.1 Learning the Constraint Matrix in High-dimensional Spaces

Concatenating projection matrix learning and CCL could potentially lead to a computationally expensive approach. CCL requires an analytical solution for each local

model, so the total computation time of CCL depends on the number of local models. Depending on the strategy for placing the local models, it may become more expensive, although the model placement only happens once per data set. Even so, K-means is used often [15][23][26][29], in which case the distance weights need to be recalculated at every placement attempt for every data point, for every centre. If the variance is set low (e.g. when thin receptive fields are desired) and K-means is initialized with too few centres, this iterative process can be time-consuming.

Moreover, when estimating $\mathbf{N}$, the exhaustive search over θ proposed in [28] quickly becomes unreasonably expensive as the dimensionality of the data increases. The approach for learning the state-dependent constraint vector presented in [31] suffers from the same problems with local model placement as CCL, and it does not have an analytical solution. Even when the Jacobian is known, estimating $\mathbf{N}$ is a least squares problem so it is necessary to employ numerical optimization. The problem is non-convex, which presents the risk of local minima, so multiple initializations are used to avoid those, further increasing computation time. So, it is clear that if the projection matrix is to be used effectively in policy learning, we require a fast estimation method.

In this section, we define a very fast computational method for estimating the projection matrix. We first highlight the key features of our proposed approach:

- By minimizing (2.24), one minimizes the error of the null-space component. In our case, we are interested in minimizing the null-space component projected over the task space (using the Jacobian).
- Our method minimizes a quadratic function with a quadratic constraint, which is a suitable representation for interior-point constrained optimization problems. This implies that we will be able to find very good solutions even in high-dimensional spaces, unlike the original method[1] discussed in Sections 2.3.1 and 2.3.2.

Below we present our approach for learning the constraint matrix, which also provides us with the projection matrix. It should be noted that in this formulation, we assume that the Jacobian is known. This is a realistic assumption as it can often be obtained from both simulated and physical platforms.

[1] Because the standard representation uses spherical coordinates, the function to minimize becomes highly non-convex which leads to many local minima, making it much harder to find a solution even using fast optimization procedures such as the Levenberg-Marquart algorithm as suggested in [31]

First we assume that the constraint matrix $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{\mathcal{S}\times\mathcal{Q}}$ can be decomposed as:

$$\mathbf{A}(\mathbf{x}) = \mathbf{\Lambda}\mathbf{J}(\mathbf{x}) \tag{3.12}$$

where $\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{\mathcal{T}\times\mathcal{Q}}$ is the known Jacobian of the task, assumed to be known and $\mathbf{\Lambda}$ is the constraint matrix to be learned. We also assume that $\mathbf{\Lambda}$ is formed of a set of $\mathcal{S}$ row-independent unit vectors as follows:

$$\mathbf{\Lambda} = \begin{bmatrix} \boldsymbol{\lambda}_1^T \\ \vdots \\ \boldsymbol{\lambda}_{\mathcal{S}}^T \end{bmatrix} = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \dots & \lambda_{1,\mathcal{T}} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_{\mathcal{S},1} & \lambda_{\mathcal{S},2} & \dots & \lambda_{\mathcal{S},\mathcal{T}} \end{bmatrix} \tag{3.13}$$

where $\boldsymbol{\lambda}_i^T \boldsymbol{\lambda}_j + q_{i,j} = 0 \quad \forall j \geq i,\ i = 1,\dots,\mathcal{S}$, with $q_{i,j} = -1$ if $i = j$ and $q_{i,j} = 0$ otherwise. The $\mathbf{\Lambda}$ matrix does not need to consist of unit length row vectors, but it is convenient for the optimization procedure explained below because it avoids the trivial solution $\mathbf{\Lambda} = \mathbf{0}$.

In this formulation we assume that the null space component observations are available to us and we do not consider the task space observations, so we define $\mathbf{u} = {}^{ns}\mathbf{u}$. By considering that by definition $\mathbf{A}(\mathbf{x})\mathbf{u} = \mathbf{0}$ because the null space and task space components are orthogonal, we define a new metric to learn the constraint in the task space:

$$E_{ts}[\tilde{\mathbf{A}}] = \frac{1}{2}\sum_{n=1}^{N} \left\|\tilde{\mathbf{A}}(\mathbf{x}_n)\mathbf{u}_n\right\|^2 \tag{3.14}$$

where $\tilde{\mathbf{A}}$ denotes the estimate of the constraint matrix. If $\mathbf{A}(\mathbf{x})$ would be comprised of a set of unit length independent row vectors, we can easily see that its pseudo-inverse would be:

$$\mathbf{A}(\mathbf{x})^{\dagger} = \mathbf{A}(\mathbf{x})^T = \begin{bmatrix} \mathbf{a}_1(\mathbf{x}) & \dots & \mathbf{a}_{\mathcal{S}}(\mathbf{x}) \end{bmatrix} \tag{3.15}$$

However, we cannot assume that this is true for the constraint matrix per se, but with (3.12) we can recognize that $\tilde{\mathbf{A}}$ directly depends on $\tilde{\mathbf{\Lambda}}$ because the Jacobian is known, so we can substitute (3.12) in (3.14) and expand its norm so that we can derive an expression in quadratic form:

$$E_{ts}[\tilde{\mathbf{\Lambda}}] = \frac{1}{2}\sum_{n=1}^{N} \mathbf{u}_n^T \mathbf{J}(\mathbf{x}_n)^T \tilde{\mathbf{\Lambda}}^T \tilde{\mathbf{\Lambda}}\mathbf{J}(\mathbf{x}_n)\mathbf{u}_n = \frac{1}{2}\tilde{\boldsymbol{\lambda}}^T \mathcal{D}_{u,x}\tilde{\boldsymbol{\lambda}} \tag{3.16}$$

where $\mathcal{D}_{u,x} = \sum_{n=1}^{N} \mathbf{J}(\mathbf{x}_n)\mathbf{u}_n\mathbf{u}_n^T\mathbf{J}^T$ and $\tilde{\boldsymbol{\lambda}} = [\tilde{\boldsymbol{\lambda}}_1^T, , \dots, \tilde{\boldsymbol{\lambda}}_{\mathcal{S}}^T]^T \equiv \text{vec}(\tilde{\mathbf{\Lambda}}^T)$.

The orthogonality and unit length conditions imposed on $\tilde{\boldsymbol{\lambda}}$ can also be expressed in quadratic form:

$$\frac{1}{2}\tilde{\boldsymbol{\lambda}}^T \mathcal{H}_{i,j}\tilde{\boldsymbol{\lambda}} + q_{i,j} = 0 \tag{3.17}$$

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

$$\mathcal{H}_{i,j} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_{i,j} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{I}_{j,i} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{3.18}$$

Though it has a rather unusual shape, its function is in fact fairly simple. The purpose of (3.17) is to take from the complete vectorized constraints $\tilde{\boldsymbol{\lambda}}$ each single constraint $\tilde{\boldsymbol{\lambda}}_i \in \mathcal{S}$ and ensure that the all conditions are satisfied with respect to all the other constraints $\tilde{\boldsymbol{\lambda}}_j \neq i \in \mathcal{S}$. The function of $\mathcal{H}$ is thus to provide a structure through which we can retrieve the two constraints we wish to investigate. To do so, this structure contains identity matrices $\mathbf{I} \in \mathbb{R}^{Q \times Q}$ at positions ij and ji.

It should be noted that (3.17) represents a different metric compared to the one defined in (2.27). It too can be adapted for the particular case where the Jacobian is known:

$$E_{ns}[\tilde{\boldsymbol{\Lambda}}] = \frac{1}{2} \sum_{n=1}^{N} \mathbf{u}_n^T \left(\tilde{\boldsymbol{\Lambda}} \mathbf{J}(\mathbf{x}_n) \right)^{\dagger} \left(\tilde{\boldsymbol{\Lambda}} \mathbf{J}(\mathbf{x}_n) \right) \mathbf{u}_n \tag{3.19}$$

but, as it is defined in the literature, it can not be expressed in quadratic form because of the pseudo-inverse operator. However, if we maintain our definition of $\mathbf{A}(\mathbf{x})$ in (3.15), the task-space metric (3.16) and null-space metric (3.19) are the same:

$$E_{ns}[\tilde{\mathbf{A}}] = \frac{1}{2} \sum_{n=1}^{N} \mathbf{u}_n^T \tilde{\mathbf{A}}(\mathbf{x}_n)^{\dagger} \tilde{\mathbf{A}}(\mathbf{x}_n) \mathbf{u}_n = \frac{1}{2} \sum_{n=1}^{N} \mathbf{u}_n^T \tilde{\mathbf{A}}(\mathbf{x}_n)^T \tilde{\mathbf{A}}(\mathbf{x}_n) \mathbf{u}_n = E_{ts}[\tilde{\mathbf{A}}(\mathbf{x})] \tag{3.20}$$

Indeed, learning a unit length row independent matrix $\mathbf{A}(\mathbf{x})$ implies that we will be able to estimate the ground-truth constraint up-to a some unknown scaling factor. Even in the case where the task-space component is not zero, we can learn the appropriate projection matrix because the scaling factors cancel out due to the pseudo-inverse of $\mathbf{A}(\mathbf{x})$.

3.4 Evaluation Metrics

As discussed in Section 2, the literature employs two main performance metrics. The first is the unconstrained policy error (2.9) normalised by the variance of the policy:

$$E_{nupe}[\tilde{\pi}] = \frac{1}{N\sigma_{\pi}^2} \sum_{n=1}^{N} ||\pi(\mathbf{x}_n) - \tilde{\pi}(\mathbf{x}_n)||^2 \tag{3.21}$$

where N denotes the number of data points and σ_{π}^2 is the variance of the policy. The second is the constrained policy error (2.10), also normalised:

$$E_{ncpe}[\tilde{\pi}] = \frac{1}{N\sigma_{\pi}^2} \sum_{n=1}^{N} ||\mathbf{u}_n - \mathbf{N}(\mathbf{x}_n)\tilde{\pi}(\mathbf{x}_n)||^2 \tag{3.22}$$

We propose to add the inconsistency error (2.12) as a metric as well in the form of a normalised constraint consistent error (NCCE), even though it is a lower bound to the NCPE and the NUPE. Its normalised form is

$$E_{ncce}[\tilde{\pi}] = \frac{1}{N\sigma_{\pi}^2} \sum_{n=1}^{N} ||\mathbf{u}_n - \hat{\mathbf{u}}_n\hat{\mathbf{u}}_n^T\tilde{\pi}(\mathbf{x}_n)||^2 \tag{3.23}$$

In constraint consistent learning it is used for learning and because of CCL's analytical solution it is generally very low. In constrained policy learning we use an estimate of the projection matrix for learning, which amounts to using the normalised constrained policy error. We therefore expect constraint consistent learning to have a low normalised constraint consistent error, and CPL to have low normalised constrained policy error. Constrained policy learning will also have a low constraint consistent error (lower than its NCPE), but it is difficult to predict whether it will be lower than the constraint consistent error of CCL because it is not directly minimized.

Of our three metrics, the normalised constrained policy error is the most important indicator of the performance of constrained policy learning, because it measures how well the policy is reproduced under constrained circumstances. It can be used as an indicator of both training accuracy (when measuring the NCPE under seen constraints) and generalisation performance (when measuring the NCPE under new constraints). We will use it only as a generalisation metric since CPL minimizes it directly, resulting in a very low training error.

The accuracy of the normalised unconstrained policy error is less important because of degeneracy in the set of candidate policies in scenarios with high dimensionality, as discussed in Section 3.1. Because of the degeneracy it is likely that even if a policy is learned, in a way that is consistent with the constraint, it may be very different from the true unconstrained policy. This is due to the fact that we cannot cover the whole space of possible policies in any system with a relatively high dimensionality. However, the learnt policy may still generalize to unseen constraints. When applying

the learnt policy, it will almost always be in a setting in which it is constrained (in the case of our wiping policy there has to be a surface to wipe), so we argue that estimating the true unconstrained policy is not especially crucial, and correctly estimating the constrained policy is.

We also require performance measures to evaluate our proposed method for estimating the projection matrix. For this, we use the normalised projected policy error (NPPE) and the normalised projected observation error (NPOE) [30]. The NPPE is given by

$$E_{nppe}[\tilde{\mathbf{N}}] = \frac{1}{N\sigma_{\pi}^2} \sum_{n=1}^{N} \left\| \mathbf{N}\pi_n - \tilde{\mathbf{N}}\pi_n \right\|^2 \tag{3.24}$$

The NPPE corresponds directly to the normalised constrained policy error (NCPE) defined in Section 2 since we are considering the case where $\mathbf{u} = \mathbf{N}\pi$. However, the difference here is that instead of measuring the accuracy of the estimated policy $\tilde{\pi}$, we measure the accuracy of the estimated projection matrix $\tilde{\mathbf{N}}$. As with the other error measures discussed above, it is unrealistic to assume that we have access to the ground truth policy for learning. However, in our case it is appropriate to use it as purely an indicator of performance. Additionally, we have the normalised projected observation error (NPOE), which indicates how well the estimated projection matrix projects to the null space:

$$E_{npoe}[\tilde{\mathbf{N}}] = \frac{1}{N\sigma_{\pi}^2} \sum_{n=1}^{N} \left\| \mathbf{u}_n - \tilde{\mathbf{N}}\mathbf{u}_n \right\|^2 \tag{3.25}$$

We will use these error metrics to assess performance of our proposed methods in the experiments in the next section.

3.5 Experiments

This section contains the experiments that we have performed to test the performance of the proposed learning strategy on a complex surface wiping policy executed by a kinematic simulation of the 7-DOF DLR LWR-III robotic arm. In this thesis we use trajectories generated by the (simulated) robot itself as demonstrations from which to learn the underlying policy. The demonstrations consist of the robot performing a constrained movement, so that the data consist only of kinematic information and no prior knowledge of the policy or the constraints. The trajectories generated in these experiments are kinematic, so the states are joint positions and actions are joint velocities.

The advantage of using the same robot for deomonstrating and learning is that there is a perfect match in kinematic structure of the teacher and student as they are the same. This relieves us from the need to consider movement recognition, pose tracking, coordinate transformations, and other matters relating to perception [33], leaving us to focus on purely the learning aspect.

We first present experiments that show the accuracy and robustness of our proposed projection learning method. We then do the same for constrained policy learning. Then, we apply constrained policy learning to several data sets containing variations of the wiping policy described in Section 3.2 in order to test performance of the learning method on complex data with various characteristics.

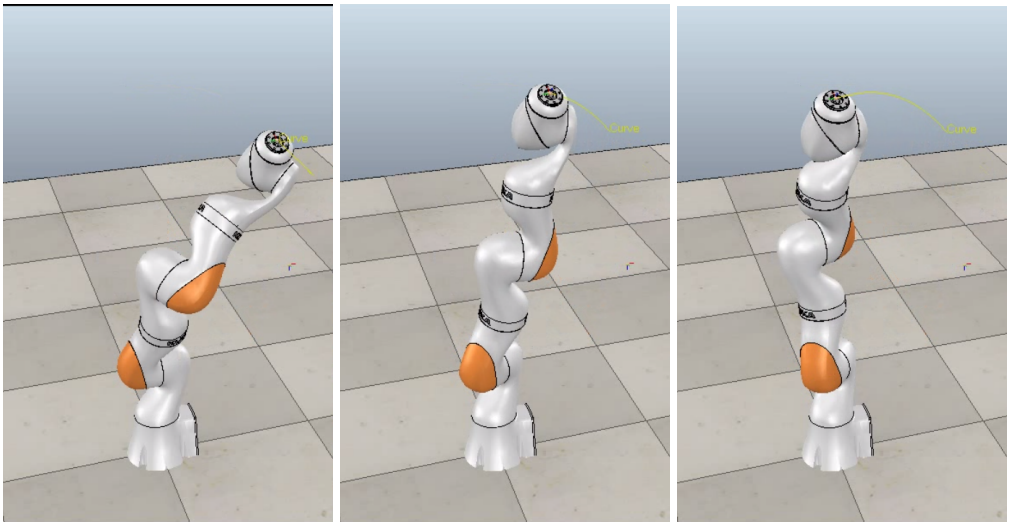
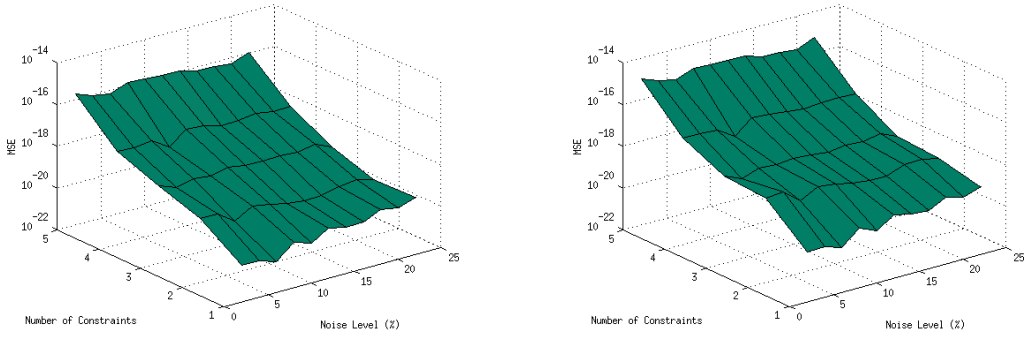


Figure 3.5: **Simulated joint limit avoidance.** In this experiment, the orientation of the end-effector was constrained. The arm is randomly initialized in a pose where the 4^{th} and 6^{th} joint are near their limits, and it executes a policy that results in a trajectory towards a comfortable position.

3.5.1 Projection Matrix Learning Performance

In this section, we demonstrate the accuracy of our proposed method for projection matrix learning. To do this, we use a joint limit avoidance scenario, where positional axes and orientations are constrained in several different combinations. This policy is also used in the literature [23] [24], and a typical example is shown in Figure 3.5. It is simpler than the wiping policy used later in this thesis, yet still non-linear to some extent. It is simple enough to be the first policy this method is tested on, and complex enough to show robustness of the method.



(a) Projected Observation Error (b) Projected Policy Error

Figure 3.6: **Error plots of projection matrix accuracy** (a) shows the normalised POE and (b) shows the normalised PPE. The values shown are means over 50 trials. Errors are very low in both POE and PPE, and the method is very robust to noise.

To generate the data, we first generate a random starting posture by drawing 7 joint angles uniformly from the half of the range of each joint, so $\mathbf{q}_i \sim U[-0.5\mathbf{q}_i^{max}; 0.5\mathbf{q}_i^{max}]$ where $\mathbf{q}^{max} = (170^\circ, 120^\circ, 170^\circ, 120^\circ, 170^\circ, 120^\circ, 170^\circ)^T$. We then define a joint limit avoidance policy as

$$\pi(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \qquad \phi(\mathbf{x}) = \sum_{i=1}^{7} |x_i|^{1.8} \tag{3.26}$$

The policy is subjected to 5 different constraint sets. The first constraint set only constrains the x position of the end-effector, the second constrains both the x and y positions, and so on until there is only a single degree of freedom: the z orientation, or θ_z. The 5^{th} constraint thus constrains $(x, y, z, \theta_x, \theta_y)$.

For all constraint sets we generate 20 experiments that execute the policy from different start poses, and each experiment consists of 20 trajectories that start from the same pose (but are different due to the random noise). Each trajectory consists of a 60 points. We perform this set of experiments 11 times, where each set is performed with a different level of noise contamination of the policy, ranging from 2% to 20% in increments of 2.

We then learn the projection matrix with the method proposed in Section 2.3 and calculate the normalised projected policy error (NPPE) and the normalised projected observation error (NPOE) over all the data used for learning. The results are shown in Figure 3.6. Both error measures have very low values overall, and there is a clear loss in accuracy as the number of constraints that need to be estimated increases. This is

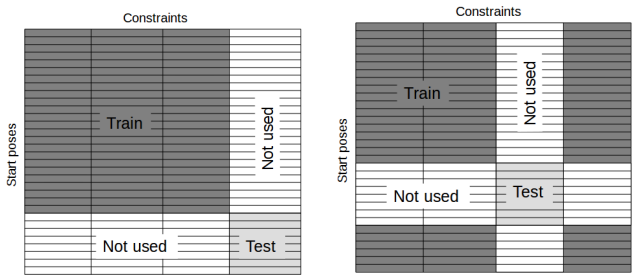


Figure 3.7: **Two random train/test splits.** Only the parts of the data are used for testing (light gray) that do not have any overlap at all with the training data (dark gray). Any data that has some overlap with the training set in either start poses or constraints is discarded (white).

due to the combination of an increased number of orthogonality conditions that need to be satisfied, and small inaccuracies in satisfying them, which leads to a cumulative error as illustrated by the largely linear increase on the log scale in the figure.

Interestingly, there is only a very small difference in error for sets with 2% and 20% noise. This is likely due to the fact that our approach does not directly use the policy to learn the constraints, and thus is less affect by its noisiness. We consider these preliminary results to be remarkably good, especially the method's robustness to noise.

3.5.2 Constrained Policy Learning Performance

Here, we repeat the experiment from the previous section, but we use the learnt projection matrix to then estimate the null space policy. The goal of this experiment is to show that constrained policy learning is effective in simple, slightly non-linear scenarios, and that it is reasonably robust to noise. The most important difference with the previous experiment is that we use three constraints (but different compositions), which we consider a reasonable amount for a joint limit avoidance task. The policy is subjected to 4 different sets of constraints, given as (x, y, z), $(\theta_x, \theta_y, \theta_z)$, (x, z, θ_y), and (y, θ_x, θ_z). Here, the letters denote which end-effector coordinates are controlled in the task space, where (x, y, z) means the task was defined in end-effector position coordinates, but ignored the orientation. Similarly, (x, z, θ_y) denotes that the task was defined in the x- and z-coordinates and the orientation around the y-axis, while leaving the y-position and orientation around the x- and z-axes unconstrained.

Similar to the previous section we generate 10 experiments for all constraints sets,

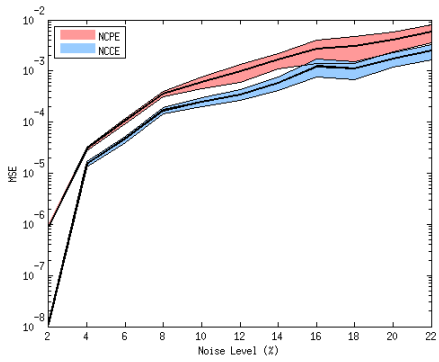


Figure 3.8: **CPL robustness to noise.** This figure shows the robustness of the policy tested on unseen data with increasing levels of noise contamination of the policy. The plot shows the mean $\pm$ standard deviation over 10 trials, where the colored areas indicate one standard deviation from the mean.

which execute the policy from different start poses. We have 10 experiments with each 10 trajectories, generated for 4 different constraint sets, for 11 different noise levels.

From this set, we select a train and test set according to Figure 3.7. In particular, we selected the data from the last constraint, the last trial, and last experiment as a test set in all noise levels. Even though this results in a rather small test set (60 points), the points are completely novel and errors are normalized. Another option would have been to randomly select a test set at every run in order to cover more of the space, but since we are also testing for robustness to noise it is crucial that we use the same set for testing in all cases.

The results are shown in Figure 3.8 for both NCPE and NCCE. The unconstrained policy error has been omitted because it was consistently high. The figure shows that our method is very accurate at low noise levels in this data set. Both error measures roughly follow the noise level, and even with over 20% noise in the data set the errors are $< 10^{-2}$. For a comparison see Figure 5 in [23], where $10^{-3} < NCPE < 10^{-1.5}$ for $1\% - 20\%$ noise in a 2-dimensional toy example. This method is significantly more accurate than standard CCL, especially with low noise levels.

We acknowledge that this highly accurate performance may be due to the specific constraint we used for testing. On the other hand, none of the four different constraint sets are qualitatively different, so we expect similar results on test sets with any of the other constraint sets. Overall, we consider these results to be very good, and in the next section we present the method's performance in more complex scenarios.

3.5.3 Randomly Initialized Single-Step Policy Learning

In this experiment we aim to establish the basis for our approach to learning the complex wiping policy. The main priority in this first experiment is to ensure that there is a sufficient number of constrained observations of the policy for every state of the system. To achieve this, each trajectory is a random initialization of the full plant and the trajectories consist of single time steps. Additionally, each initialization has this single step of the policy executed under 8 different constraints. We have chosen this particular number of constraints because as a general rule in CCL, performance increases with the number of constraints. We choose a number that is only slightly higher than the dimensionality of the policy, because in order to demonstrate the efficacy of our method we feel it is good practice to show its performance on a realistic number of constraints. These constraints are generated at the start of the experiment and each consists of a surface to which the end-effector must be orthogonal, and the orientation of the surface is uniform randomly selected for both the x and y axes according to $\theta_x, \theta_y \sim U[-20^\circ, +20^\circ]$. Additionally, the position around which the wiping motion should take place is drawn uniform randomly according to $x, y \sim U[-0.3, 0.3]$ where x and y are planar coordinates in the frame of the surface and the values are in meters. The variation in the constrained observations comes from the fact that in each trial, the robot's position is exactly the same but the constraints are different. This means the end-effector will not start directly on, or orthogonal to the wiping surface for every trial (as shown in Figure 3.9) so the task space component will attempt to direct the end-effector to a position that satisfies the constraints, which will lead to variation in the observations. With this setup, the data is very rich in constraints which largely eliminates the degeneracy issues, which will in turn lead to accurate unconstrained policy estimations. In total, this data set consists of 7500 random initializations where the policy is executed for one step under 8 different constraints. This results in a total of 60000 data points, each of which with $\mathbf{x} \in \mathbb{R}^9$ and $\mathbf{u} \in \mathbb{R}^7$. The state space is 9-dimensional because in addition to the 7 joint angles, there are 2 additional parameters that specify the target wiping location on the surface. In each trial, the training and test sets are randomly split such that there is no overlap in random poses and constraints as illustrated in Figure 3.7.

For both CCL and CPL we then set up the local models using K-means so that each data point has a weight of ≥ 0.7. The weight was determined by the distance of each point to a local model, according to

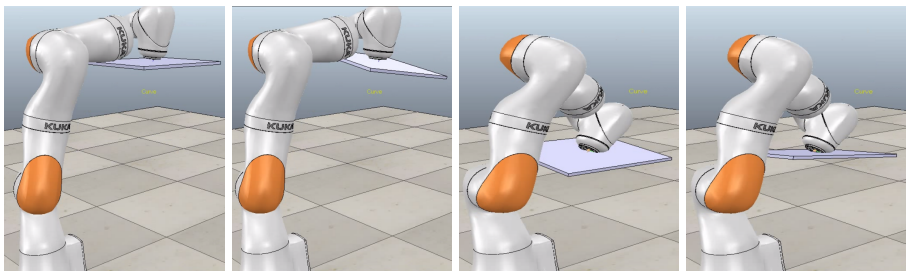


Figure 3.9: **Single step policy under different constraints and start poses.** Shown are two different surface slopes for two different randomly initialized poses.

$$w_{nm} = \exp(-\frac{1}{2\sigma^2}||\mathbf{x}_n - \mathbf{c}_m||^2) \tag{3.27}$$

where w_{nm} is the weight of the n^{th} data point in the m^{th} model. We used the standard deviation of the input data to set the width of the local models, and we added a scaling factor to manually adjust this:

$$var = \left(\frac{s}{\sigma(X)}\right)^{-1} \tag{3.28}$$

where s is the scaling factor. It is clear from the equation that with a scaling factor of 1, the variance equals the standard deviation. Note that because this variance is not strictly the square of the standard deviation, we denote it with var instead of σ^2. For clarity, we reformulate (3.27) with this slight modification so that

$$w_{nm} = \exp(-\frac{1}{2var}||\mathbf{x}_n - \mathbf{c}_m||^2) \tag{3.29}$$

In this case, we used a scaling factor of 1, which resulted in around 2000 models. We then estimated the projection matrix and trained CPL as well as CCL. We repeated this process for five trials, each with a randomly selected train / test split, and the results are shown in Table 3.1.

Table 3.1: **Results for LWCCL and LWCPL on randomly initialized, single steps of the policy.** Errors are given in mean $\pm$ standard deviation over 5 trials with randomly chosen train / test sets. Importantly, results are 10^{-3} and in radians.

Method	NUPE	NCPE	NCCE
LWCCL (10^{-3})	1.5513 ± 0.0750	0.1792 ± 0.0077	0.0048 ± 0.0003
LWCPL (10^{-3})	10.023 ± 1.0805	5.0427 ± 0.2174	1.2512 ± 0.2326

Both methods perform well (errors are 10^{-3}), and the hierarchy in errors is as expected: the NUPE is the highest, followed by the NCPE and the NCCE. Interestingly, constraint consistent learning outperforms the proposed method in this experiment by at least an order of magnitude on all metrics. The explanation for this is likely to be found in the fact that CCL is dependent on both actions and states in the sense that it is most accurate when there are many constrained observations for every state. CPL does not rely on this feature of the data, but it does require a rich sampling of the state space in general (to make optimal use of the Jacobian) in order to be able to generalize well. This data set lacks exploration in the state space because of the single-step execution of the policy. Still, CPL is able to generalize the policy well to a new configuration and a new constraint with an NCPE that translates into an offset of about 0.3° (the errors in Table 3.1 are in radians). Of course, the main downside to this experiment is that the data is unrealistic in the sense that it can not (or only very inconveniently) be generated by demonstrations or on a real robot. We therefore test our approach on a more realistic data set in the next experiment.

3.5.4 Fixed Initialization Full Policy Learning

In this experiment we demonstrate that a full constrained policy can be learned accurately using CPL. Complementary to the experiment described above, we now use a data set consisting of full wiping motions (instead of single steps) under different constraints, but all starting from the same configuration. By starting each trial from the same initial configuration, we generate motions with a significant amount of overlap in states between the separate trials. This increases the likelihood of having multiple, differently constrained observations per state. Specifically, this data again consists of 8 different, randomly generated, planar constraints under each of which a full wiping motion of 150 steps is generated, resulting in a total of 1200 data points. The constraints are generated in the exact same fashion as in the previous experiment, as well as the wiping locations. Here, the same procedure for splitting the train and test sets was employed as shown in Figure 3.7. Because the spread of the data is rather low due to the fixed initialization, we use thin receptive fields with a variance scaling factor of 25 in (3.28). With these parameters we use both methods to estimate the policy, and results are shown in Table 3.2.

The results are rather varied, in the sense that constrained policy learning outperforms CCL in estimating the constrained policy, and constraint consistent learning

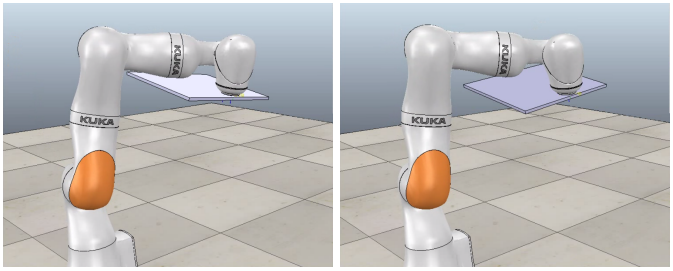


Figure 3.10: **Fixed initialization wiping under different constraints.** Poses shown are from the same starting configuration. The difference with Figure 3.9 is that these simulations contain the full wiping motion instead of only a single step.

outperforms CPL in estimating the unconstrained policy. However, the sizes of both NUPEs indicate that the true unconstrained policy was very difficult to learn, likely because of the degeneracy. This is corroborated by the remarkably large standard deviations for both methods, which suggest that in each trial a different unconstrained policy was learned. Constraint consistent learning has a lower NUPE because it learned a policy that was consistent with the observations and relatively close to the unconstrained policy. CPL does not require consistency with the observations and is therefore more free in the unconstrained policies it can learn, as long as the policy is correct when it is projected.

The NCPE on the other hand indicates that CPL was better able to learn and generalize the estimated policy than CCL. Even though the difference is less than an order of magnitude, variation in errors around this range indicates a large performance difference: CPL is off by about 5° while CCL misses the correct expression of the policy by more than 22° on average. CPL performs better here compared to the previous experiment because the local state space has been explored more, at least with respect to the test set. This means that the states in the test set will not be wildly different from the states encountered during learning, which boosts accuracy. CCL on the other hand has a lower ratio of observed actions to states (compared to the previous experiment) because the trials now contain full wiping motions. So even though its inconsistency error is still low, there is not enough variation between the observations to maintain accuracy.

3.5.5 Randomly Initialized Full Policy Learning

This data set contains both randomly initialized poses and full wiping motions. This experiment is therefore the most important one, because it demonstrates the perfor-

Table 3.2: **Results for LWCCL and LWCPL on fixed initialization, full motion of the policy.** Errors are given in mean $\pm$ standard deviation over 5 trials with randomly chosen train / test sets.

Method	NUPE	NCPE	NCCE
LWCCL	12.327 ± 9.509	0.3888 ± 0.090	0.0052 ± 0.0041
LWCPL	114.87 ± 76.578	0.0871 ± 0.019	0.0276 ± 0.060

mance of constrained policy learning on highly varied and complex data. Additionally, this data set also presents the problem of having few constrained observations per state, which increases the likelihood of degeneracy in the set of possible policies. We thus expect that the NUPE will be high. However, the main goal is to learn a good constrained policy.

The data set consists of 50 randomly initialized start states. Starting from each of these, a full wiping movement of 150 steps is performed under 8 different constraints. The constraints as well as the wiping locations are determined through the procedure used in the previous experiments. In total, this data set contains 60000 data points, and the train and test split is performed as before. Of the 50 experiments, 5 are held back as test set every time, and from those the trajectories following a single constraint are used as the final test set, amounting to 750 data points.

Because the variance in the states is large due to the random initializations, we use a variance scaling factor of 9 to ensure that the receptive fields are not too wide and do not lead to overfitting (large scaling factors lead to thinner models, e.g. (3.28)). Still, this scaling factor is significantly lower than the one used in the previous experiment, while within each initialization the granularity of the data is the same. This means that we are smoothing more in this experiment, but this is necessary to avoid large numbers of local models and excessive computation times. This parameter setting results in 2400 to 2700 receptive fields, depending on the trial. We then perform the same procedures as earlier to estimate the projection matrix and train both methods. This is repeated for five trials, each with a randomly selected train / test split, and the results are shown in Table 3.3.

It is interesting to see that CCL performs badly with respect to the constrained and unconstrained policy errors, while its NCCE is low. This very clearly illustrates that minimizing the inconsistency is in effect minimizing a lower bound, which does not guarantee good policy estimation results. The reason that this becomes apparent in

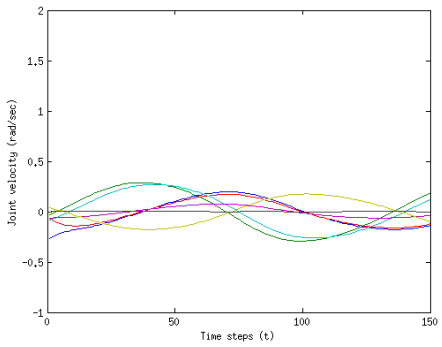
Table 3.3: **Results for LWCCL and LWCPL on randomly initialized, full motion of the policy** Errors are given in mean $\pm$ standard deviation over 5 trials with randomly chosen train / test sets.

Method	NUPE	NCPE	NCCE
LWCCL	155.73 ± 78.047	42.923 ± 42.228	0.0029 ± 0.0023
LWCPL	50.998 ± 24.927	0.0094 ± 0.0006	0.0049 ± 0.0018

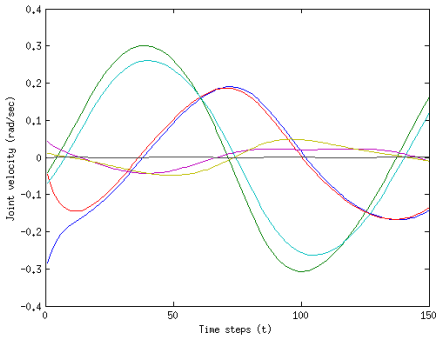
this particular experiment is that CCL is dependent on both states and observed actions for its estimation. It learns a policy that is consistent with the observations, which might have the property that it generalizes well to unseen constraints. However, this policy remains a mapping from states to actions, and it is likely that the generalization property does not extend to previously unseen states. Because in these experiments, we are always testing on a set that has no overlap with the training set, and in this particular experiment the state space is very large, CCL does not perform well on the test set. This is also where the superiority of constrained policy learning becomes apparent. Because its estimation is not state dependent due to the separation of the constraint matrix into the Jacobian and the invariant constraints, CPL is able to generalize very well to previously unseen states as well as unseen constraints, as reflected in its NCPE.

3.5.6 Summary of Results

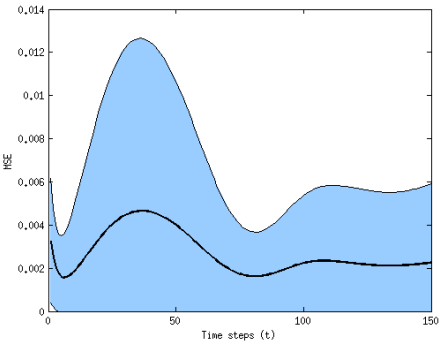
The results presented in this chapter show that constrained policy learning performs worse than constraint consistent learning in simple data sets with many constrained observations per state, although it is still accurate. The unconstrained policy estimations are inaccurate for both methods all experiments, but especially in the two data sets that contain the full wiping motion because of the degeneracy problem. As mentioned before, this metric is not as relevant as the NCPE because policies are unlikely to be expressed in an unconstrained fashion. On the more important NCPE, the proposed method scores higher than the standard approach in the scenario with wiping motions in a fixed position, and its performance even increases by an order of magnitude in the data set that contains randomly initialized wiping motions, while the accuracy of CCL decreases massively here. We consider these results to be rather good, and we will discuss further steps to be taken in the next chapter.



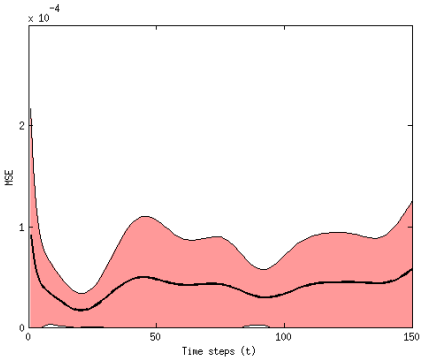
(a) Ground truth unconstrained policy



(b) Ground truth constrained observation



(c) Unconstrained policy error



(d) Constrained policy error

Figure 3.11: **Constrained and unconstrained policy reproduction errors.** From the randomly initialized full policy simulation data set, (a) and (b) show typical unconstrained and constrained wiping motions, where each colored line represents the velocities of a joint over the full motion. (c) and (d) show the mean $\pm$ standard deviation for the reconstruction of the unconstrained and constrained policy for all joints over 400 trials, where the colored area indicates one standard deviation from the mean. (c) is significantly higher than (d) (note the y axis values), and displays more variability.

Chapter 4

Conclusion and Further Research

4.1 Conclusion

In this thesis, we take an off-line approach to learning movements in constrained robotic systems with a high level of redundancy. More specifically, we address the challenge of learning unconstrained policies from constrained motion data with the goal of generalizing the learnt policy to unseen situations and constraints. To do this, we build on constraint consistent learning and null space projection learning, two methods that have been developed to learn null space policies and constraints, respectively.

We describe the development and benefits of these methods, and we investigate their boundaries. In the case of CCL, these boundaries consist of strict data requirements and degrading performance when learning more complex policies. The most significant obstacle encountered in projection learning is the optimization procedure, which consists of either exhaustive search or non-linear optimization. The first can be computationally expensive, and the second can suffer from local minima.

To alleviate these issues we propose a new method: constrained policy learning. It uses an estimate of the projection matrix to learn policies, rather than a projection of the observed actions, providing a state-independent learning method. To prevent this method from becoming too computationally expensive, we also propose a modification of projection matrix learning that uses knowledge of the Jacobian to provide a quadratic objective function, which is very efficient and accurate.

The results presented in the previous chapter show that constrained policy learning can accurately learn complex constrained policies and generalize them to unseen state space and novel constraints. It can cope with complex and varied data sets better than the standard constraint consistent learning because it is no longer state dependent, and

its conditions on the input data are less strict.

However, the proposed method still suffers from the same significant computation expense as CCL due to the local models used for learning. Additionally, we acknowledge that the presented results are fairly preliminary. We therefore consider further validation of the method and exploration of the set of learnable policies to be crucial next steps. We elaborate on these steps for further research in the next section.

4.2 Further Research

More validation. The results presented in this thesis are quite preliminary and are obtained from a 'safe' and relatively ideal simulated environment. It is therefore crucial to validate the proposed approach on platforms of varying complexity, starting with the physical DLR-LWR III platform itself. If the effectiveness of the method is confirmed on this platform, it is a logical next step to increase the complexity, in terms of learnable policies as well as dimensions. One particularly interesting challenge would be to use this method to try to learn a movement policy on a humanoid robot.

Metric for policy learnability. The current body of literature (including this thesis) does not formulate 'hard' boundaries on the set of learnable policies, but rather tests the learning methods on 'simple' and 'complex' policies. These definitions are somewhat vague and arbitrary (although they have been sufficient), so it would be beneficial to investigate the possibility of a formal definition of a metric that assesses how well any given policy can be learned by the proposed method. In fact, the development of such a metric would greatly facilitate progress towards a more complete assessment of the applicability and value of the null space learning method.

Application to dynamics scenarios. Because constrained policy learning seems to handle high dimensionality well, further research could also include an extension of the proposed method to higher-dimensional dynamics scenarios. However, this will likely be computationally intensive due to the increased state space and the accompanying increase in local models used for learning. It might therefore be worthwhile to also investigate the potential benefits of smarter model placement (using methods such as Receptive Field Weighted Regression), or dimensionality reduction, or a combination of these methods (such as Locally Weighted Projection Regression). A dynamics scenario would also include an active task space component due to the physical forces

acting on the system, and it would be unrealistic to assume that these are known. It is therefore important that the existing method for estimating the null space component is improved, since it currently relies on computationally-intensive, non-convex, non-linear optimization. It may be possible to apply the proposed method for projection matrix learning to this estimation problem as well.

Policy parametrisation. Even though null space learning circumvents issues with the curse of dimensionality in on-line approaches such as hierarchical control by learning a movement policy from data, it struggles with high dimensionality in the learning process instead. Motivated by the significant increases in data set size that come with more complex systems, we propose to add specific policy parametrisation to the existing policy learning method. In its current form, its power lies in its generality but this comes at the cost of good or fast performance in high dimensions. Policy parametrisation could analytically outline specific structures of the policy to be learned, thereby bounding the search space in which the policy learning method would operate. Since in many cases at least some information is available about the learning task, this approach would first narrow down the search space to a size in which a data driven method is advantageous, before applying policy learning.

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov. An integrated system for real-time model predictive control of humanoid robots. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pages 292–299. IEEE, 2013.

[12] David W Scott. The curse of dimensionality and dimension reduction. *Multivariate Density Estimation: Theory, Practice, and Visualization*, pages 195–217, 2008.

[13] 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.

[14] 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.

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

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

[17] Matthew Howard and Sethu Vijayakumar. Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In *Proceedings of the Workshop on Robotics and Mathematics*, page 109–115, Coimbra, Portugal, Sep. 17–19, 2007.

[18] François Chaumette and Éric Marchand. A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. *Robotics and Automation, IEEE Transactions on*, 17(5):719–730, 2001.

[19] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Learning potential-based policies from constrained motion. In *IEEE/RSJ International Conference on Humanoid Robots*, 2008.

[20] Jakob J Verbeek, Sam T Roweis, Nikos Vlassis, et al. Non-linear cca and pca by alignment of local models. *Advances in Neural Information Processing Systems*, 16:297–304, 2004.

[21] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1079–1086. Morgan Kaufmann Publishers Inc., 2000.

[22] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Behaviour generation in humanoids by learning potential-based policies from constrained motion. *Applied Bionics and Biomechanics*, 5(4):195–211, 2008.

[23] 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.

[24] Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar. Robust constraint-consistent learning. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4629–4636. IEEE, 2009.

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

[26] 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.

[27] 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.

[28] Hsiu-Chin Lin, Matthew Howard, and Sethu Vijayakumar. A novel approach for representing and generalising periodic gaits. *Robotica*, 32(08):1225–1244, 2014.

[29] Hsiu-Chin Lin, Matthew Howard, and Sethu Vijayakumar. A novel approach for generalising walking gaits across embodiments and behaviours. In *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on*, pages 1009–1015. IEEE, 2014.

[30] 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.

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

[32] 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.

[33] Kerstin Dautenhahn and Chrystopher L Nehaniv. *Imitation in animals and artifacts*. MIT Press Cambridge, MA, 2002.