Learning to serve: an experimental study for a new learning from demonstrations framework

Okan Koç[1], Jan Peters[1,2]

Abstract— **Learning from demonstrations is an easy and intuitive way to show examples of successful behavior to a robot. However, the fact that humans optimize or take advantage of their body and not of the robot, usually called the *embodiment problem* in robotics, often prevents industrial robots from executing the task in a straightforward way. The shown movements often do not or cannot utilize the degrees of freedom of the robot efficiently, and moreover suffer from excessive execution errors. In this paper, we explore a variety of solutions that address these shortcomings. In particular, we learn sparse movement primitive parameters from several demonstrations of a successful table tennis serve. The number of parameters learned using our procedure is independent of the degrees of freedom of the robot. Moreover, they can be ranked according to their importance in the regression task. Learning few parameters that are ranked is a desirable feature to combat the curse of dimensionality in Reinforcement Learning. Preliminary real robot experiments on the Barrett WAM for a table tennis serve using the learned movement primitives show that the representation can capture successfully the style of the movement with few parameters.**

I. Introduction

Humans are good at using their bodies to great effect, taking advantage of their muscular structure and soft but flexible actuation. Much of dexterous manipulation, or dynamic movement generation reflects this awareness of the human body. When teaching the robots to achieve similar tasks autonomously, however, we inevitably impose and transfer our biases to the robot. This problem of *embodiment* can cripple the execution and possibly prevent the robots from taking advantage of their kinematics structure and actuation mechanisms.

In dynamic games like table tennis, we can easily observe humans taking utmost advantage of their bodies and pushing it to its maximum, i.e., optimizing their output bearing in mind their kinematic and dynamic limits. Table tennis serves, for instance, incorporate flicks (very fast accelerations of the wrist) that are designed to give an unsuspected spin and motion profile to the ball. Teaching such movements to the robots, in a learning from demonstration framework, suffers in particular from two drawbacks. Firstly, during the shown movement, as discussed above, the human is unable to move the shoulder joints of the robot adequately, which can potentially be used by the robot to great effect. Secondly, the fast movements of the wrists may not be tracked accurately

by the robot, which is the case for the cable-driven seven degree of freedom (DoF) Barrett WAM arm, see Figure 1.

In this paper, we explore different learning from demonstrations (LfD) approaches to compensate for the execution and transfer deficiencies resulting from the demonstrated serves. The initial policy or the movement template, extracted as a set of movement primitives, can be thought of as a good initialization for a reinforcement learning (RL) agent. By capturing the essence of the shown demonstrations in as few parameters as possible, we simplify and increase the effectiveness of the skill transfer to the robot. Sparsity is achieved in our framework by using a new iterative optimization approach, where a multi-task Elastic Net regression is alternated with a nonlinear optimization. The Elastic Net projects the solutions to a sparse set of features, and during the nonlinear optimization these features (the basis functions) are adapted to the data in a secondary optimization. Moreover these features are shared across multiple demonstrations, increasing the effectiveness of the feature learning strategy.

The fewer number of learned parameters using our iterative optimization procedure, compared to more traditional approaches, is independent of the robot DoF. This is a desirable property for Reinforcement Learning to adapt the learned parameters online. Moreover, by using the Elastic Net path, we can rank the parameters in terms of importance, or effectiveness in explaining the demonstration data. We perform preliminary experiments on the Barrett WAM on a table tennis serve to validate the effectiveness of our new movement primitives.

Robot table tennis has captivated the attention of the robot control and learning community as a challenging and dynamic task, and research in this task has been ongoing ever since the nineties. After the pioneering work of Anderson's analytical player [1], there have been various approaches focusing on certain parts of the game, such as simplifications in trajectory generation using a virtual hitting plane [19], [16] or learning striking trajectories from demonstrations [12]. Learning approaches to generate better strikes include [15], [5]. Recently, [13] has introduced a new trajectory generation framework in table tennis, where they solve a free final-time optimal control problem to generate minimum acceleration striking trajectories. This kinematic optimization approach was extended and evaluated in the real robot table tennis setup in [14].

Learning from demonstrations (LfD) is a promising framework for learning various robotic tasks efficiently without using hard-coded approaches specific for each task. It has also been used to initialize policy-search RL approaches

[1]Max Planck Institute for Intelligent Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany `okan.koc@tuebingen.mpg.de`
[2]Technische Universitaet Darmstadt, FG Intelligente Autonome Systeme Hochschulstr. 10, 64289 Darmstadt, Germany `peters@ias.tu-darmstadt.de`

in robot learning [11], to great effect. There are, by now, many different frameworks for LfD, including dynamical systems approaches such as the Dynamical Movement Primitives (DMP) [6], learning control Lyapunov-functions [9], and various probabilistic approaches, including probabilistic movement primitives [18] and Gaussian mixture models [8].

A detailed introduction and analysis of l_1-regularized l_2-norm regression (from hereon referred to as *Lasso*) can be found in [4]. Interest in Lasso lies in the fact that Lasso can perform feature selection automatically. Lasso was extended to the *multi-task* case (i.e., multi-output case with shared features) in [17]. The *Elastic Net* imposing additional $l2$-regularization to Lasso was introduced in [21], where it was noted that a basic transformation converts the problem to a standard Lasso regression, and this is also valid in the multi-task setting. A new incremental procedure to solve ordinary least squares regression as well as Lasso problems was proposed in [2]. This algorithm, called *Least Angle Regression* or *LARS* for short, yields piecewise linear homotopy paths of the regression problem as a function of the $l1$-regularization term. Elastic Net paths can also be generated with Lars, since the Elastic Net can be converted to a standard Lasso problem.

To the best of our knowledge, the multi-task Elastic Net was not combined before with Radial Basis Functions in a nonlinear feature selection and optimization framework. We also think that ranking the learned parameters in terms of importance is a new approach that was not explored in the RL community.

II. NOTATION

We start by formulating the constraints that need to be satisfied for a successful table tennis serve. The notation that
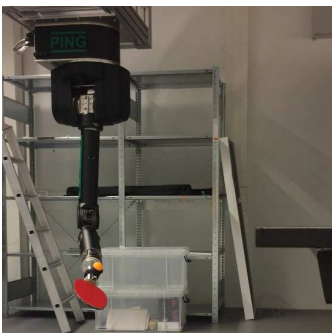

Fig. 1: Our robot table tennis setup with a seven DOF Barrett WAM robot, ready to serve a table tennis ball. Two cameras on the opposite side of the ceiling track the ball continuously at a rate of 180 Hz. A metal piece is attached to the endeffector of the Barrett WAM, which connects to a standard sized table tennis racket. An egg-holder on the metal piece holds the ball before the serve. We compare and evaluate throughout the paper different learning from demonstration approaches to achieve a successful table tennis serve. We propose a new iterative optimization approach to learn sparse parameters and to adapt the features of our movement primitives on multiple demonstration data.

we use throughout the paper is standard: for a robot arm with n degrees of freedom (DoF), the joint configurations are $\mathbf{q} \in \mathbb{Q} = \{\mathbf{q} \in \mathbb{R}^n \mid \mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max}\}$. The recorded joint positions over a movement is represented as a matrix $\mathbf{q}(t) \in \mathbb{R}^{N \times n}$ of N rows, with columns storing the joint position at the corresponding time step. If multiple demonstrations are needed for learning, i.e., $\mathbf{q}_{ij}(t)$ is recorded for $i = 1, \ldots, n$ DoF and $j = 1, \ldots, d$ demonstrations, these recordings are stacked to form the $\mathbf{Q}$ matrix. The degrees of freedom are concatenated horizontally in this case for a single demonstration, while the columns store the different demonstration data, i.e., $\mathbf{Q}_{i,j} = \mathbf{q}_{ij}(t) \in \mathbb{R}^N$ for a recording of N time points.

The Frobenius norm of a matrix $\|\mathbf{M}\|_F^2 = \sum_i \sum_j m_{ij}^2$, whereas the $\|\cdot\|_{21}$ norm used in the Elastic Net is defined as $\|\mathbf{M}\|_{21} = \sum_i \sqrt{\sum_j m_{ij}^2}$, i.e., $l2$-norm along the columns (degrees of freedom in our setting) and $l1$-norm along the rows (time steps).

III. METHOD

In this section, we discuss learning an initial sparse movement pattern from human demonstrations. We present first an algorithm that requires only a single human demonstration, and then present a suitable variant that can be employed for multiple demonstrations. This variant of the algorithm decouples the number of learned parameters from the degrees of freedom of the robot.

A. Learning a sparse representation from a single demonstration

Given a single demonstration $\mathbf{q}(\mathbf{t})$ at the (observed) time points $\mathbf{t}$, we'd like to extract a movement primitive that can be easily refined later via (policy search) RL and optimization. Throughout the parametric optimization, we'd like to impose a good fit with as few basis functions as possible, while keeping the accelerations low during the trained movement pattern. Having low accelerations is beneficial both for robot safety as well as improving the tracking (execution) accuracy of the trajectories [14]. Mathematically, the criterion that we optimize can be written as

$$\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \|\mathbf{q}(\mathbf{t}) - \boldsymbol{\Psi}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}\|_F^2 + \lambda_1 \|\boldsymbol{\theta}\|_{21} + \lambda_2 \|\ddot{\boldsymbol{\Psi}}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}\|_F^2, \ (1)$$

where $\boldsymbol{\Psi}(\mathbf{t},\boldsymbol{\beta}) \in \mathbb{R}^{N \times p}$ are the evaluations of the basis functions at $\mathbf{t}$, $\boldsymbol{\theta} \in \mathbb{R}^{p \times n}$ are the (sparse) regression parameters, and $\mathbf{q}(\mathbf{t})$ are the joint observations during the shown movement. The nonlinear radial basis functions (RBF) are parameterized by $\boldsymbol{\beta}$. A $l2$-penalty is put on the accelerations $\ddot{\boldsymbol{\Psi}}(\mathbf{t},\boldsymbol{\beta})\boldsymbol{\theta}$ of the extracted movement pattern, while a penalty with the $l1$-norm on the (rows of the) regression parameters $\boldsymbol{\theta}$ encourages sparsity of the found solutions.

This regression problem, for fixed $\boldsymbol{\beta}$, is known as the multi-task *Elastic Net* in the literature, where the features are shared among the sparse parameters along each degree of freedom. As opposed to the standard (multi-task) Lasso, the 2-norm penalty in the optimization (III-A) penalizing the

accelerations throughout the motion, also adds stability to the Lasso solutions [21].

The solution to the weighted Elastic Net problem (III-A) for fixed $\boldsymbol{\beta}$ can be obtained by transforming the problem to an equivalent (unweighted) Lasso problem, solving it via a convex optimizer (e.g., *coordinate descent* is very effective for Lasso problems), and then transforming the solutions back to Elastic Net parameters.

We can solve the original problem (III-A) iteratively (as in Expectation-Maximization type of algorithms) by first starting the iteration with a Lasso solution of an overly-parameterized radial basis function regression. At each iteration, the RBF parameters $\boldsymbol{\beta}_i$ corresponding to the basis functions with nonzero Lasso regression parameters $\boldsymbol{\theta}_{ij} > 0$, $j = 1, \ldots, n$ are updated for each $i = 1, \ldots, p$ via nonlinear optimization. These two alternating steps can be continued till convergence, or rather terminated in a fixed number of steps. The iterations converge when the change in function value of the total cost in (III-A) is below a certain tolerance ϵ. Depending on the initial solution parameters $\boldsymbol{\beta}_0$ and $\boldsymbol{\theta}_0$, the iteration converges to a local minimum.

The full procedure is shown in Algorithm 1 in detail. We call the resulting algorithm *Learning Sparse Demonstration Parameters* or *LSDP* for short. The algorithm alternates between the multi-task Elastic Net (lines 3 and 9) and the nonlinear optimizer (BFGS, in line 7). In between, the zero entries of the regression parameters $\boldsymbol{\theta}$ and the corresponding columns of $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ are removed in the Prune step (lines 4 and 12). The pruning operation simplifies the optimization in the upcoming iterations, as the pruned RBF cannot then be re-elected later. We use the squared exponential kernel to construct our basis functions, i.e., for every i, j we use

$$\boldsymbol{\Psi}_{ij}(t_i) = \exp((-t_i - \boldsymbol{\mu}_j)^2/(2\boldsymbol{\sigma}_j^2)), \qquad (2)$$

to form the (i, j)'th element of the $\boldsymbol{\Psi}$ matrix. The data is initially centered in line 2, i.e., the mean of each joint recording is subtracted from the signal, and the means $\mathbf{q}_0$ are stored as the intercepts for the particular demonstration.

For a good performance of the algorithm, i.e., obtaining low residuals with a sparse set and low accelerations, choosing the regularizer constants λ_1 and λ_2 suitably is crucial. These parameters can be set using cross-validation either before Algorithm 1 or in line 3 together with the regression. The regularizers should be scaled down accordingly with the decreasing residual norms (see line 11), otherwise the algorithm can converge to the empty set for the parameters $\boldsymbol{\theta}$. See the Experiments section for more discussion on the implementation details.

B. Coupling the parameters across dimensions

The algorithm *LSDP* discussed in the previous subsection uses the multi-task Elastic Net to enforce the same basis functions for each degree of freedom (along the columns of $\mathbf{q}(t)$ and $\boldsymbol{\theta}$), and the parameters are decoupled across the degrees of freedom (DoF) of the robot. In particular, the number of regression parameters grow linearly with the robot

---

Algorithm 1 Learning sparse parameters with regression (*LSDP*) for a single demonstration

---

Require: $\mathbf{q}$, $\mathbf{t}$, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$, λ_1, λ_2, $\epsilon > 0$
1: Initialize $\boldsymbol{\beta}_0 = [\boldsymbol{\mu}, \boldsymbol{\sigma}^2]$
2: Center the data, $\mathbf{q}_0, \mathbf{q} \leftarrow \texttt{Center}(\mathbf{q})$
3: Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_0$ and $\mathbf{t}$
4: $\boldsymbol{\theta}_0 \leftarrow \texttt{MultiTaskElasticNet}(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{q}, \lambda_1, \lambda_2)$
5: $\boldsymbol{\theta}_0, \boldsymbol{\beta}_0 \leftarrow \texttt{Prune}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0)$
6: Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}$ and $\mathbf{t}$
7: **repeat** $k = 1, \ldots,$
8:      $\boldsymbol{\beta}_k \leftarrow \texttt{BFGS}(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \boldsymbol{\beta}_{k-1}, \boldsymbol{\theta}_{k-1}, \mathbf{q}, \lambda_1, \lambda_2)$
9:      Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_k$ and $\mathbf{t}$
10:      $\boldsymbol{\theta}_k \leftarrow \texttt{MultiTaskElasticNet}(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{q}, \lambda_1, \lambda_2)$
11:      Calculate residual norm r_k, total cost f_k using (III-A)
12:      Scale penalties $\lambda_i \leftarrow \lambda_i r_k^2/r_{k-1}^2$, $i = 1, 2$
13:      $\boldsymbol{\theta}_k, \boldsymbol{\beta}_k \leftarrow \texttt{Prune}(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k)$
14:      Form $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}$ and $\mathbf{t}$
15: **until** $\|f_k - f_{k-1}\| < \epsilon$

DoF, which is undesirable for applying policy search RL approaches to high dimensional robotic systems especially.

Furthermore, the algorithm has to be applied for each demonstration separately, i.e., there is no *coupling* or information shared between the demonstrations. In order to enforce rather the features to be shared *across demonstrations* rather than the robot DoFs, we discuss here first a variant of the algorithm *LSDP*, which we call coupled *LSDP*, or *cLSDP* for short.

Algorithm 2 requires only a few changes compared to Algorithm 1. The data is centered for each demonstration to obtain the intercepts $\mathbf{Q}_0$. The algorithm stacks in lines $1 - 3$ the dependent regression variables $\mathbf{q}_i$ and RBF parameters $\boldsymbol{\beta}_i$ for each degree of freedom $i = 1, \ldots, n$ across rows to form the matrices $\mathbf{Q} \in \mathbb{R}^{Nn \times d}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{Nn \times p}$ as well as its second time derivative $\ddot{\boldsymbol{\Psi}}$. Unlike *LSDP*, this procedure n requires n times the RBF parameters $\boldsymbol{\beta}$ to be optimized (line 8), as the features are adapted independently for each DoF. The regression parameters $\boldsymbol{\theta}$ on the other hand, are now reduced n times, and coupled across the DoFs. The parameters for each demonstration are also estimated together, i.e., the columns of the $\boldsymbol{\theta}$ matrix correspond to the regression parameters for different demonstrations.

C. Ranking the demonstration parameters

The regression parameters estimated with *cLSDP* can also be ranked using the Elastic Net regularization path, which traces the evolution of the parameters as the $l1$-norm bound (or equivalently, the regularization term) of the coefficients increases. Using the *LARS* algorithm [2], one can trace the addition of the demonstration parameters in a final Elastic Net path computation after running Algorithm 2. An example path for only eight selected parameters are plotted in Figure 2. These parameters can be ranked according to the evolution, i.e., the coefficients that early on during the path become nonzero are likely to signal more causally effective

Algorithm 2 Learning coupled sparse parameters with regression (*cLSDP*) across multiple demonstrations

---

Require: $\mathbf{q}_{ij}$, $\mathbf{t}$, $\boldsymbol{\mu}_i$, $\boldsymbol{\sigma}_i^2$, λ_1, λ_2, $\epsilon > 0$
1: Stack $\mathbf{q}_{ij}$ to form $\mathbf{Q}$, $i \in [1, n]$, $j \in [1, d]$
2: Center the data, $\mathbf{Q}_0, \mathbf{Q} \leftarrow \texttt{CenterStacked}(\mathbf{Q})$
3: Stack $\boldsymbol{\beta}_0 = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_n, \boldsymbol{\sigma}_1^2, \ldots, \boldsymbol{\sigma}_n^2]$
4: Stack $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_0$ and $\mathbf{t}$ across DoFs
5: $\boldsymbol{\theta}_0 \leftarrow \texttt{MultiTaskElasticNet}(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{Q}, \lambda_1, \lambda_2)$
6: $\boldsymbol{\theta}_0, \boldsymbol{\beta}_0 \leftarrow \texttt{PruneStacked}(\boldsymbol{\theta}_0, \boldsymbol{\beta}_0)$
7: Stack $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}$ and $\mathbf{t}$ across DoFs
8: **repeat** $k = 1, \ldots,$
9: $\quad \boldsymbol{\beta}_k \leftarrow \texttt{BFGS}(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \boldsymbol{\beta}_{k-1}, \boldsymbol{\theta}_{k-1}, \mathbf{Q}, \lambda_1, \lambda_2)$
10: $\quad$ Stack $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}_k$ and $\mathbf{t}$ across DoFs
11: $\quad \boldsymbol{\theta}_k \leftarrow \texttt{MultiTaskElasticNet}(\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}, \mathbf{Q}, \lambda_1, \lambda_2)$
12: $\quad$ Calculate residual norm r_k, total cost f_k using (III-A)
13: $\quad$ Scale penalties $\lambda_i \leftarrow \lambda_i r_k^2 / r_{k-1}^2$, $i = 1, 2$
14: $\quad \boldsymbol{\theta}_k, \boldsymbol{\beta}_k \leftarrow \texttt{PruneStacked}(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k)$
15: $\quad$ Stack $\boldsymbol{\Psi}, \ddot{\boldsymbol{\Psi}}$ using $\boldsymbol{\beta}$ and $\mathbf{t}$ across DoFs
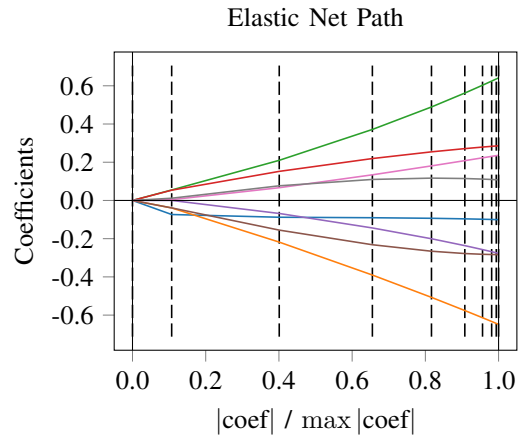16: **until** $\|f_k - f_{k-1}\| < \epsilon$

Fig. 2: An example Elastic Net path for eight parameters is shown for one of the demonstrations. As the $l1$-norm constraint is relaxed (or equivalently, the regularization parameter is reduced), the coefficients converge to their ordinary least squares solution at the right hand side of the plot. Each dashed line signals a change in the regularization term, and the coefficients are updated accordingly. We can use this path to rank the parameters of the movement primitive in terms of importance.

components of the motion. More prominent components of the motion can be adapted earlier with RL strategies to reduce the curse of dimensionality in high dimensional robots.

IV. EXPERIMENTS

In this section, we conduct experiments to learn a sparse set of demonstration parameters using the proposed approaches and compare with various competing movement primitive learning methods. Finally we present a real robot experiment on our table tennis platform where we show that the learned sparse movements nevertheless look natural and can be implemented safely on the robot.

A. Learning from Demonstrations

The algorithms *LSDP* and its coupled variant *cLSDP*, discussed in Section III, is applied here on the demonstrated robot joint movements. From a continuous stream of joint recordings, occurring at 500 Hz, we preprocess a predetermined number of d movements, by detecting the maximum d velocities in joint space and windowing around these points for a fixed duration of one second.

The processed examples using this procedure result in the joint matrix $\mathbf{q} \in \mathbb{R}^{500 \times 7}$ for each example. The initial RBF centers $\boldsymbol{\mu}_0 \in \mathbb{R}^{500}$ are placed at every point uniformly and the RBF widths $\boldsymbol{\sigma}_0^2$ are set to 0.1 for every time point. We implement the preprocessing as well as the Algorithms in Python, using the *scikit-learn* toolbox for the multi-task Lasso and the *scipy* toolbox for the nonlinear optimization (BFGS, see lines 8 and 9 for each algorithm). The algorithm *LSDP* stretches, prunes and expands these basis functions throughout the optimization to produce a very sparse, nonuniform set of basis functions shared across the seven degrees of freedom or the demonstrations, respectively. For *LSDP*, the final regression parameters $\boldsymbol{\theta}_i \in \mathbb{R}^7$ are separate for each basis function $\boldsymbol{\Psi}_i$ kept by the algorithm. *cLSDP* on

the other hand shares the set of basis functions across multiple demonstrations. The learned parameters, along with the intercepts, are saved to a json file offline, to be loaded by the real-time robot controller in C++ during the online experiments.

The proposed two algorithms *LSDP* and *cLSDP* are compared against two other baselines below in Table I. The first baseline is the Dynamic Movement Primitives (DMPs) with a fixed number of basis functions. The second baseline is l2-penalized standard regression, with the penalty again on acceleration. During the experiments we used a total of ten basis functions for both the DMPs and the l2-penalized standard regression. The basis functions are spread uniformly around the one second long (preprocessed) demonstrations. DMPs, as a result of the dynamic constraint of reaching a desired goal position, incur very high initial accelerations in joint space. Even if the hyperparameters are optimized accordingly to prevent such high accelerations, slight adjustments of initial joint position will again give rise to high accelerations. The suggestion proposed in [10] to modify the accelerations with the phase can reduce the initial accelerations, but then we have found that the convergence to the goal suggestion can suffer drastically. The fixed basis function regression does not have this problem, but as in DMPs, optimizes a fixed number of parameters, which is typically double the number optimized by *cLSDP* in order to fit the demonstrations well.

Table I summarizes the results of learning movement primitives from five different demonstrations. These five demonstrations are fed together to *cLSDP*, whereas *LSDP* is run separately for each demonstration to obtain the mean and
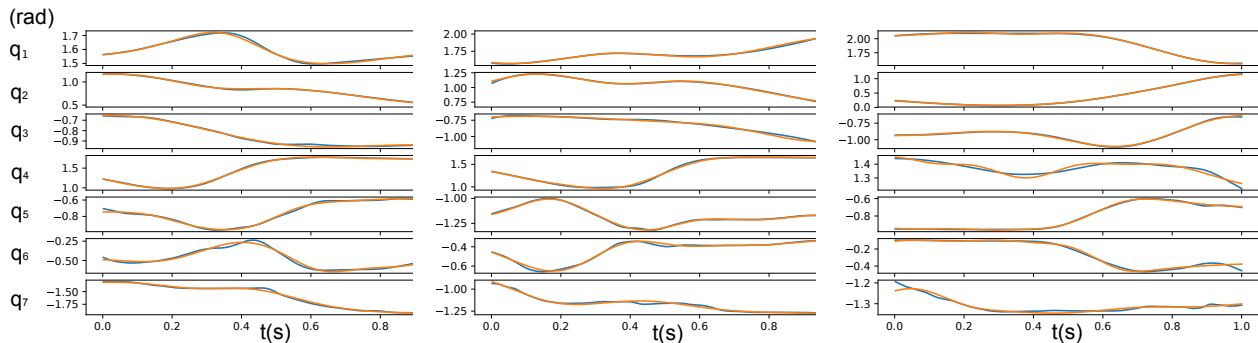
Fig. 3: Three movement primitives learned by *cLSDP*, are plotted in joint space against the recorded movements. The recorded table tennis serve movements, shown in blue, are one second long each. The movement primitives, shown in orange, share the learned parameters across the degrees of freedom of the robot. This is a desirable property for a movement primitive serving as a policy in a policy search based reinforcement learning setup, as the number of rollouts needed to improve the expected reward will then be reduced.

the standard deviations reported in the table. Note that the number of parameters in total used by *cLSDP* (37) is much lower than the on-average 16.8 parameters used by *LSDP* for each robot DoF. The residual is naturally higher, this is a result of the parameters being shared across the dimensions. In particular, we have observed that *cLSDP* does not fit the last three wrist joints as tightly as *LSDP*. This could be because the motion of the wrist is highly varying across the movements and the coupling induced by the algorithm across demonstrations brings these movements closer. See Figure 3 for three example regression results.

Three example demonstrations are plotted in task space in Figure 4 along with the recorded ball positions, detected and triangulated from two cameras opposite to the robot. The initial positions of the racket center and the ball in the egg-holder are marked as 0 in red and blue, respectively. The egg-holder is at a distance of roughly 14 cm to the racket center. During the movement the ball is hit by the human demonstrator moving the robot arm, and as the demonstrator slows down the motion to a halt, the ball is seen flying towards the table.

B. Robot Experiments

Finally, we conduct experiments in our real robot table tennis platform, see Figure 1. Our table tennis playing robot is a seven degree of freedom Barrett WAM arm that is capable of reaching high accelerations and velocities. However it is cable-driven and high accelerations can cause the cables to break easily. A standard size racket is attached

TABLE I: Comparison of different learning from demonstrations approaches, averaged over five different serve demonstrations

	No. par. $(\|\theta\|_0)$	Acc. norm	Res. norm
LSDP	$(16.8 \pm 3.25) \times 7$	59.04 ± 7.0	0.59 ± 0.11
cLSDP	37	55.98 ± 11.78	0.73 ± 0.09
DMPs	11×7	621.73 ± 57.45	0.92 ± 0.06
l2-reg. regr.	11×7	215.45 ± 35.25	2.12 ± 0.47

to the end-effector via a metal bar. The racket has a radius of roughly $r_R = 7.6$ cm. The table and the table tennis balls are standard sized, balls have a radius of 2 cm, and the table geometry is roughly $276 \times 152 \times 76$ cm. Throughout the experiments, the Barrett WAM is placed at a distance of about one meter to the end of the table and its base is located 95 cm above the table. This makes it difficult (but not impossible) for the robot to hit the table.

An egg-holder holds the table tennis ball initially, wrapped around the metal bar connecting the end-effector and the racket, see Figure 1. To get accurate ball positions during the demonstrations and the robot rollouts, we calibrated the two cameras on the ceiling opposite to the robot with respect to the robot base frame, by collecting pairs of image and robot joints correspondances. The ball detection algorithm is run separately online on every image, running at around 180 frames per second. A linear triangulation algorithm provides then ball position estimates and the resulting estimates of the ball center of mass are then triangulated and filtered with an Extended Kalman Filter (EKF).

A successful serve in our robot platform is shown in Figure 5. The ball is initially placed on top of the egg-holder (approximately 14 cm away from the racket center along the racket plane). During the movement, as a result of the robot's accelerating motion, the ball takes off from the robot arm. The ball is then hit by the robot towards the table. The arm then decelerates towards a resting posture as the ball lands on the robot court, passes the net, and lands again on the opposite side. We notice that the initial accelerating motion is critical for a good serve, as without the initial accelerations, the ball has no chance to take-off. The DMPs that we tested immediately start the movements with very large accelerations, these can be dangerous for the robot and the low-level Barrett WAM controllers do not support such movements. We have seen that *cLSDP*, on the contrary, starts with lower accelerations and catches up later.
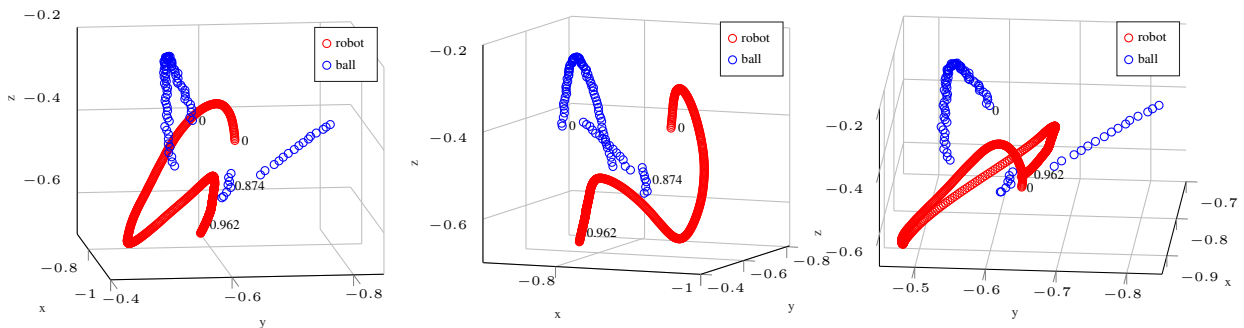
Fig. 4: Three example demonstrations in task space. The initial position of the racket center and the ball in the egg-holder are marked as 0 in red and blue circles, respectively. The egg-holder is located approximately 14 cm away from the racket centre. Before the racket stops moving, the ball is already hit, flying towards the table.
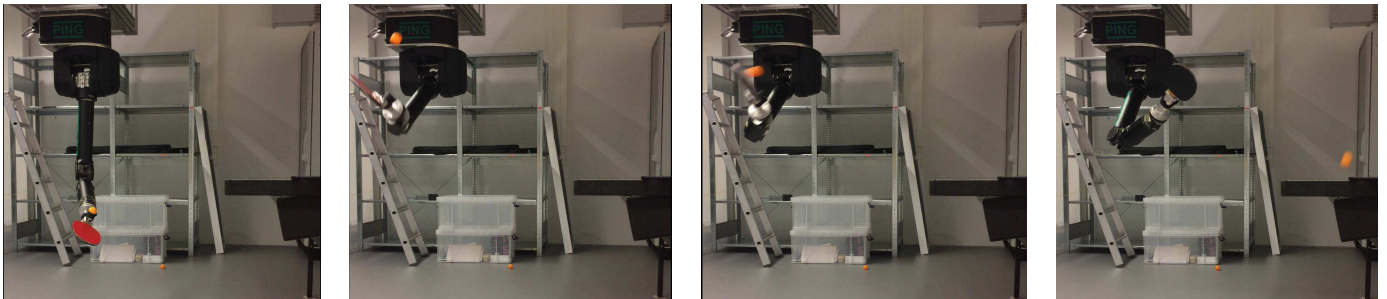

Fig. 5: A successful rollout during real robot experiments. The ball is initially on top of the egg-holder and during the movement, as a result of the acceleration of the arm, it takes-off from the robot, to be later hit by the racket towards the table. The arm then decelerates towards a safe resting posture.

V. CONCLUSION

In this paper we presented a new learning from demonstrations (LfD) approach to represent and learn table tennis serve movements. The proposed algorithms *LSDP* and *cLSDP* learn sparse parameters of the radial basis functions (RBF) from single and multiple demonstrations, respectively. The algorithms employ iterative optimization, alternating between a weighted multi-task Elastic Net regression step that learns sparse parameters given the features and a nonlinear optimization step that adapts the features (more specifically, the widths and centers of the RBFs corresponding to the nonzero regression parameters). The algorithm *cLSDP*, unlike *LSDP*, learns (sparse) parameters that are independent of the robot DoF. This desirable property is achieved by having different basis functions that are adapted across each DoF separately. The multi-task Elastic Net, in this case, forces the joint-dependent features to be shared across multiple demonstrations.

The cost function chosen for the optimization includes the residual of the fit, as well as $l2$-regularization terms on the accelerations and $l1$-regularization on the regression coefficients. We compared the performance of the proposed algorithms with Dynamic Movement Primitives (DMPs) and the standard $l2$-regularized regression, and we evaluated the performance of each on the different components of the chosen cost function. Finally, we presented a successful table tennis serve rollout using our framework on the real robot table tennis setup. The results are preliminary, however, one can see that the style of the movement is preserved while maintaining low accelerations throughout the motion, which is important for the safety of the robot.

The sparsity of the parameters, as well as their decoupling from the robot DoF, is a desirable property for policy-search RL approaches, that could adapt the regression parameters online based on a suitable reward structure. We have presented a way to rank these policy parameters based on how well the parameters explain the (multiple) demonstration recordings. We think that this is a promising research direction to combat the curse of dimensionality in high dimensional robotics learning tasks, and we will focus on it more in future experiments.

One of the things that immediately comes to mind when we think of a good table tennis serve is *spin*. Including the effects of angular velocity in our ball prediction models and adjusting the policies accordingly is one solution to vary the style of the serves. However spin is difficult to model (and to learn) and giving a spin to the ball involves cutting the path of the ball with the racket at an incline. This means that there is an inherent robustness vs. style trade-off in table tennis: trying to impart spin to the balls necessarily reduces the robustness of the learned policies to execution and ball estimation/prediction errors. Future work will focus on such stylistic improvements to our learned policies.

References

[1] Russell L. Anderson. *A Robot Ping-pong Player: Experiment in Real-time Intelligent Control*. MIT Press, Cambridge, MA, USA, 1988.

[2] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[3] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25(3):335 – 348, 1989.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[5] Y. Huang, B. Schölkopf, and J. Peters. Learning optimal striking points for a ping-pong playing robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4587–4592, 2015.

[6] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Comput.*, 25(2):328–373, February 2013.

[7] Steven G. Johnson. The nlopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`.

[8] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, Oct 2011.

[9] Seyed Mohammad Khansari-Zadeh and Aude Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62:752–765, 2014.

[10] J. Kober, K. Muelling, O. Kroemer, C.H. Lampert, B. Schoelkopf, and J. Peters. Movement templates for learning of hitting and batting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[11] J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems 22 (NIPS 2008), Cambridge, MA: MIT Press*, 2009.

[12] O. Koc, G. Maeda, G. Neumann, and J. Peters. Optimizing robot striking movement primitives with iterative learning control. In *15th IEEE-RAS International Conference on Humanoid Robots*, 2015.

[13] O. Koc, G. Maeda, and J. Peters. A new trajectory generation framework in robotic table tennis. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, in press.

[14] Okan Koc, Guilherme Maeda, and Jan Peters. Online optimal trajectory generation for robot table tennis. *Robotics and Autonomous Systems*, 105:121 – 137, 2018.

[15] K. Muelling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research*, 3:263–279, 2013.

[16] K. Muelling, J. Kober, and J. Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior Journal*, (5), 2011.

[17] Guillaume Obozinski and Ben Taskar. Multi-task feature selection. In *In the workshop of structural Knowledge Transfer for Machine Learning in the 23rd International Conference on Machine Learning (ICML 2006). Citeseer*, 2006.

[18] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2616–2624. Curran Associates, Inc., 2013.

[19] M. Ramanantsoa and A. Durey. Towards a stroke construction model. *Int. Journal of Table Tennis Science*, 2:97–114, 1994.

[20] S. Schaal. The SL simulation and real-time control software package. Technical report, 2006.

[21] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.