Technical Report

IRI-TR-09-05

Robot learning of container-emptying skills through haptic demonstration

Leonel Rozo
Pablo Jimenez
Carme Torras

Institut de Robòtica i Informàtica Industrial

Abstract

Locally weighted learning algorithms are suitable strategies for trajectory learning and skill acquisition, in the context of programming by demonstration. Input streams other than visual information, as used in most applications up to date, reveal themselves as quite useful in trajectory learning experiments where visual sources are not available. In this work we have used force/torque feedback through a haptic device for teaching a teleoperated robot to empty a rigid container. Structure vibrations and container inertia appeared to considerably disrupt the sensing process, so a filtering algorithm had to be devised. Then, the memory-based LWPLS and the non-memory-based LWPR algorithms [8, 13, 10] were implemented, their comparison leading to very similar results, with the same pattern as regards to both the involved robot joints and the different initial experimental conditions. Tests where the teacher was instructed to follow a strategy compared to others where he was not lead to useful conclusions that permit devising the new research stages, where the taught motion will be refined by autonomous robot rehearsal through reinforcement learning.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Tel (fax): +34 93 401 5750 (5751)
http://www.iri.upc.edu

Corresponding author:
Leonel Rozo
tel: +34 93 401 5784
lrozo@iri.upc.edu
http://www.iri.upc.edu/people/lrozo

1 Introduction

Personal, domestic or service robots are intended to be able to perform everyday tasks. Such tasks, like household chores, have to be executed under highly varying conditions and thus it is very complex, if not impossible, to base the performance of the robot entirely on a formal mock-up of reality. Hence, within the European project "Perception, Action & Cognition through Learning of Object-Action Complexes (PACO-PLUS)" (CogSys Integrated Project IST-4-27657), we are relying on *learning* to endow robots with the necessary skills. As training takes place in a real-world scenario, the possible arising contingencies are implicitly contemplated in the acquisition process.

The goal in skill acquisition is to learn policies, that is, to establish the appropriate correspondence between perceived states and actions [9]. Adequate policies could be discovered after exhaustive trial-and-error search (where nonetheless some kind of measure of success should be externally provided), but the natural way of learning skills is by observing how they are performed by a *teacher*. This is known as learning (or programming) by demonstration [3], or imitation learning, which suits well a domestic setting as the teacher does not need to be a programmer, but just know how to execute the task.

Due to differences between human and robot morphologies, learning is not aimed at reproducing exactly the teacher's motions, but at identifying the relevant execution traits, so that the robot can afterwards refine its motion autonomously through rehearsal.

Locally weighted learning (LWL) fits well these demands, and especially that of copying with changing distributions, which may easily lead to catastrophic interference within many neural network paradigms. LWL methods have been successfully used in a variety of applications, like learning the 50-dimensional inverse dynamics of a robot arm [14], devil-sticking and pole-balancing [8], or air hockey playing [2]. Thus, we have adapted two such methods, namely Locally Weighted Partial Least Squares (LWPLS) and Locally Weighted Projection Regression (LWPR), to our particular setting and task.

Unlike most existing contributions to skill learning by demonstration, our training algorithms do no rely exclusively on positional information, but mainly on force/torque feedback. This is a distinctive feature, whose relevance becomes evident when visual information is insufficient to determine the state of the system. In particular, we address applications that involve emptying a container through a hole. For an opaque container, empty or full states are visually indistinguishable. In our experimental setup, described in Section 2, the content is assumed heavy enough to be detected by a force/torque sensor mounted on the robot's wrist (not only its presence/absence, but also approximately where the load lies inside the container).

The remaining of the paper is structured as follows. Section 3.1 provides a brief description of LWL methods, and Section 3.2 explains how we have adapted them to the present context. The obtained results and their interpretation are described in Section 4. Finally, some conclusions are drawn in Section 5 and future work is indicated.

2 Experimental setting

In our experimental setting a STAUBLI RX-60 robotic arm with a force/torque sensor placed on its wrist (the Shunk's FTC-050 sensor) receives its motion commands through a Force Dimension's 6-DOF Delta haptic device (see Figure 1). The user handles the end-effector of the haptic device, and these displacements and orientation changes are transformed into motion commands by the controller of the device and sent to the controller of the robot. Unlike conventional teleoperation interfaces, the haptic device allows the user to feel forces and torques on its end-effector, which may be provided by an internal computer model, or, like in this case, by an

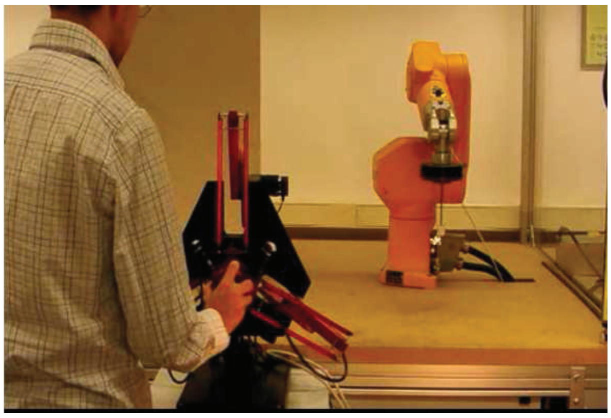

Figure 1: Teleoperation setting

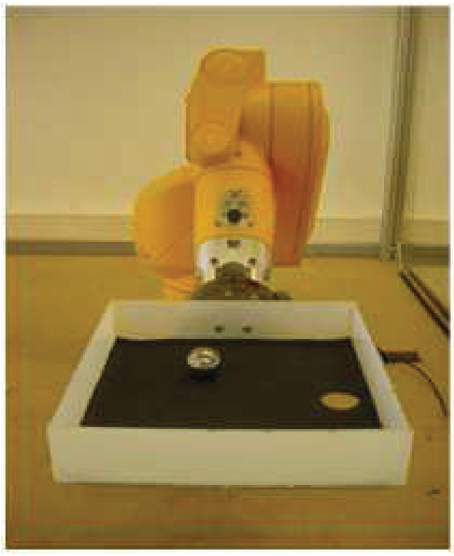

Figure 2: Robot arm and container

external source like the robot's force sensor. In sum, this setting enables the user to command remotely the robot arm (i.e., to teleoperate the robot) while feeling the interaction forces and torques produced on the robot arm's wrist. In other words, it provides a sensorial (specifically, a haptic) channel for experiencing what is happening in the remote scene.

In our experimental setup, the robot arm has a rigid rectangular container with a hole attached at its wrist, as shown in Figure 2 (alternatively, the robot could hold the box with its gripper). Inside the box, a ball is free to roll around. The objective is to teach the robot to extract the ball out from the box by reorienting the box until the ball falls through the hole. The relevant aspect here is that only forces/torques on the robot's wrist (which are implicitly related to the forces/torques generated on the container by the ball) are being sensed while the teleoperator performs a demonstration of the task. The teacher has both visual (a direct visual perception of the scene) and haptic feedback, whereas the robot receives exclusively haptic information.

Formally speaking, each position/orientation $\mathbf{x}$ generated by the teleoperator at the end-effector of the haptic device is transformed to the robot's frame and sent to the robot's controller as desired configuration in the operational space. This controller sends the command for moving the robot to such position. At the same time, all forces/torques sensed on the robot's wrist from the F/T sensor are filtered, transformed to the haptic device's frame and reproduced on the teleoperator's hand through the haptic interface. There is a key issue related to the filtering process: the forces/torques signals $\mathbf{F/T}_s$ correspond to forces/torques generated by the ball $\mathbf{F/T}_b$, those generated by the container's mass $\mathbf{F/T}_m$ and noise ε:

$$\mathbf{F/T}_s = \mathbf{F/T}_b + \mathbf{F/T}_m + \varepsilon \tag{1}$$

We want to reproduce on the local place only those signals which are generated by the ball's dynamics, thus it is necessary to eliminate both noise and forces/torques of the container's mass. As the container is not a perfectly rigid structure, it vibrates when the robot moves, and the reproduction of these vibrations on the teleoperator site is an undesired effect. It can be avoided by implementing a digital filter that cuts out all vibration signals on the force/torque sensor, in a similar way as in [4, 5], where a method for suppressing residual vibrations in flexible payloads, carried by robot manipulators, is developed by preconditioning the robot joint trajectories using FIR digital filters. To this end, the signal's fundamental frequency was determined by subjecting the structure to vibrations (considering together the container and the ball inside it, with the aim of obtaining a lower fundamental frequency than if the container was empty, in this way it is possible to guarantee that vibrations will be removed, independently of the presence or not of the ball). Such vibrations are generated by applying, in a stable configuration of the robot (Figure 2), a perpendicular force to the container's base. The application point of this force is centered at the opposite edge where the container is hold by the robot wrist. This force corresponds to an impact manually applied on the application point; of course, this force was applied several times with the aim of recording a data set where we computed the fundamental frequency from. The generated data on the sensor were recorded and its frequency spectrum was analyzed. This fundamental frequency was the cutoff frequency for our low pass filter which was designed by using the "*Constrained Least Squares*" technique and the MATLAB's FDAtool. The filter order was 75 and the cutoff frequency equals to 7.5 Hz.

With this filter all forces/torques produced by the structure vibrations were eliminated, so in a second stage it was necessary to dynamically compensate the forces/torques generated by the container's mass in the sensor's frame. Here, the main idea is to model the container force/torques generated by its dynamics, and to use this model for removing them from the sensor readings which have been a relevant problem for other settings [7, 12]. For achieving this aim, let us denote the position of the center of gravity of the container as $\mathbf{p}$, its mass as m, $\mathbf{I}$ as its moment of inertia, $\mathbf{Fs/Ns}$ and $\mathbf{Fe/Ne}$ as the sensor and external forces/torques respectively, $\mathbf{r}_s$ and $\mathbf{r}_e$ the vectors from the center of gravity of the container to the sensor and external forces frame, and using the Newton-Euler equations, we obtain:

$$\Sigma\mathbf{F} = m\ddot{\mathbf{p}} = m\mathbf{g} + \mathbf{F}_e + \mathbf{F}_s \tag{2}$$
$$\Sigma\mathbf{T} = \mathbf{I}\ddot{\mathbf{r}} + \dot{\mathbf{r}} \times \mathbf{I}\dot{\mathbf{r}} = \mathbf{N}_s + \mathbf{r}_s \times \mathbf{F}_s + \mathbf{N}_e + \mathbf{r}_e \times \mathbf{F}_e \tag{3}$$

Assuming very low linear and angular accelerations, as well as a low angular velocity for simplicity - which empirically did not seem to have any negative impact for the dynamical compensation - we obtain:

$$\mathbf{F}_s = m\mathbf{g} + \mathbf{F}_e \tag{4}$$
$$\mathbf{N}_s + \mathbf{r}_s \times \mathbf{F}_s = \mathbf{N}_e + \mathbf{r}_e \times \mathbf{F}_e \tag{5}$$

Solving these equations the forces/torques produced by the container dynamics are obtained, and they can be removed from the measured forces and torques in the subsequent experiments.

In this way, the remaining forces/torques will be those generated by the ball in the container. These signals will be transformed to the haptic's frame, scaled and reproduced on the haptic interface.

The setting described in this section has an evident academic flavour. The container and the ball have been dimensioned so as to provide a suitable collection of measurements. Further experiments (as described in the last section) will include more realistic settings. Nontheless and despite their simplicity, these experiments are very appropriate to show how force-feedback-based learning by demonstration can be carried out and how a simple motor strategy can be successfully taught to the robot, while constituting a valuable test bed for the implementation and performance evaluation of LWL techniques.

3 Learning the manipulation task

3.1 Learning algorithms

Trajectory-level skill learning involves in general the acquisition of a quite complex function: complex due to the high dimensionality (spatial position and orientation, velocities, dynamics) and to the fact that it does not have usually a compact analytical representation. *Locally weighted learning* aims at nonlinear function approximation by using piecewise linear functions [8]. Under this key concept, several algorithms have been developed, grouped into two families: memory-based LWL and non-memory-based LWL. The choice of the type of algorithm is a matter of flexibility in data interpretation and storage and lookup requirements. In what follows, brief descriptions of the two implemented algorithms, which belong to each one of the two families, are given.

Locally Weighted Partial Least Squares (**LWPLS**) is a suitable method to reduce the computational complexity of *Locally Weighted Regression* (**LWR**) [1] and to avoid its numerical problems [10]. The idea behind PLS is to fit linear models by using a hierarchy of univariate regressions along selected projections on the input space. These projections are chosen in accordance with input/output correlation, moreover PLS assures the subsequent projections will be orthogonal in the input space. The approach followed by Schaal et al. [11, 10] is based on the fact that: being globally high dimensional does not imply that data remains high dimensional if viewed locally. Thus, they set out to carry out a PLS regression in a local fashion by weighting the data around the query point. In this way, the dimensionality reduction process is developed in the query point's neighborhood. The following **LWPLS** method will be shown assuming a univariate output, however, it is possible to carry out a univariate **LWPLS** for each variable in the output data set for multivariate cases, in other words, this means to learn separate models for all outputs.

Given a query point $\mathbf{x}_q$ and p training points $\mathbf{x}_i, y_i$:

$$\mathbf{W} = \begin{pmatrix} w_{11} & 0 & \dots & 0 \\ 0 & w_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & w_{nn} \end{pmatrix}$$

where $w_{ii} = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_q)^T \mathbf{D}(\mathbf{x}_i - \mathbf{x}_q)\right)$.

Note that the weights corresponding to the training points decrease with the distance to the query point and can be considered null if they are far enough. The optimization of the distance metric is done by "*leave-one-out cross validation* (**LOOCV**)". To avoid too many open parameters (**D** is a square matrix in the number of input dimensions), **D** is assumed to be a diagonal matrix depending on only one scale parameter h: $\mathbf{D} = h \cdot diag[n_1, \ldots, n_n]$, where the n_i normalize the range of each input dimension [8].

A key issue in this method is to know how many projections r provide enough information for describing the initial data set in the case that input data are not locally statistically independent and approximately locally linear (if they are, just a single projection is necessary for obtaining an optimal linear approximation [10]). Every projection in **LWPLS** reduces the residual error by a certain amount. This means that, after i projections, there is a residual error MSE_i left in fitting the training data. When another projection is added, the question is whether this new projection is capable of reducing the residual error even more. Thus, comparing the residual error at i with the previous one, $i-1$, gives a measure of how much progress has made in improving the fit. Thus a simple heuristic to stop adding projections is to require that, for every new projection, the squared error be reduced at least by a certain ratio ϕ:

$$\frac{MSE_i}{MSE_{i-1}} < \phi \tag{6}$$

where $\phi \in [0, 1]$.

In the multivariate case (instead of a scalar y_i we have a vector $\mathbf{y}_i$), every output should have its own distance metric.

A quite determinant point of concern remains open with LWPLS, as a typical memory-based system. Namely, if the learning system receives a large, possibly never ending stream of input data, as it is typical in online robot learning, both memory requirements to store all data as well as the computational cost to evaluate the algorithms become too large. Under these circumstances, a non-memory based version of LWL is desirable such that each new data point is incrementally incorporated in the learning system and lookup speed becomes accelerated. The corresponding online version of the LWPLS technique is *Locally Weighted Projection Regression* (**LWPR**), which employs nonparametric regression with locally linear models [14]. In order to stay computationally efficient and numerically robust, each local model performs the regression analysis with a small number of univariate regressions in selected directions in input space in the spirit of partial least squares regression. The properties of LWPR are that it i) learns rapidly with second-order learning methods based on incremental training, ii) uses statistically sound stochastic LOOCV for learning without the need to memorize training data, iii) adjusts its weighting kernels based only on local information in order to minimize the danger of negative interference of incremental learning, iv) has a computational complexity that is linear in the number of inputs, and v) can deal with a large number of - possibly redundant - inputs [13]. The interested reader is addressed to these references for a detailed description of the different steps of this algorithm. Here we point some key differences with respect to **LWPLS**, one of them is that lies in separating the process of local model updating from computing the prediction at the query point. While in **LWPLS** the weight matrix is computed with respect to the query point, **LWPR** uses a number K of receptive fields centered on fixed locations $\mathbf{c}_k$ (in order to minimize negative interference due to changing input distributions), computed again as Gaussian kernels with the parameter of the distance metric:

$$w_k = \exp\left(-\tfrac{1}{2}(\mathbf{x} - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x} - \mathbf{c}_k)\right)$$

In addition, other relevant differences between these learning methods are: i) **LPWR** implements a incremental version of PLS for computing the regression for each local model, and ii) **LPWR** allows to update the distance metric in an incremental way for finding a good **D** value for each local model, while **LWPLS** has a unique and fixed distance metric for all local models

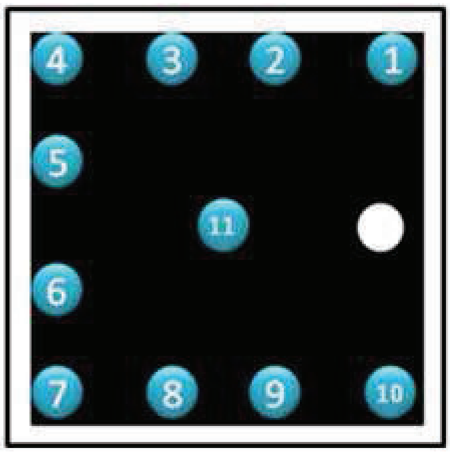

Figure 3: Initial positions of the ball

(of course this can previously optimized by means of LOOCV). On the other hand, in **LWPR** local models are created on an 'as-needed' basis.

Summarizing, **LWPR** consists of the following steps:

```
1. Initialize the LWPR with no receptive field (RF)
2. for every new training sample (x,y):
    (a) for $k = 1$ to $K$ (Number of receptive fields):
        i. Calculate the activation by using the Gaussian kernel
        ii. Update projections, regression and Distance Metric
        iii. Check if number of projections needs to be increased
    (b) if no RF was activated by more than $w_{gen}$ (which is a threshold for generating new RFs)
        i. Create a new RF with $R = 2$, $\mathbf{c} = \mathbf{x}$ and $\mathbf{D} = \mathbf{D}_{def}$
```

Loop 2(a) performs essentially the same actions as in **LWPLS**, with slight differences (see [13]). Again the parameter to be tuned is the distance metric, which can be learned for each local model individually by stochastic gradient descent in a penalized *leave-one-out cross validation*, as described in [13].

3.2 Implementation issues

Regarding the learning stage, a set of demonstrations was carried out by teleoperating the robot arm until taking the ball out of the box in each example. Figure 3 displays the initial positions where the ball was released. Starting at each predefined initial position, twenty demonstrations were developed, from which ten were carried out using a particular motion strategy: take the ball to the wall adjacent to the hole, then take the ball along this wall to the hole. The other ten examples were demonstrated using a random strategy; it means the most important aspect was taking the ball out, regardless of the movements.

The software application samples each demonstration at 100 Hz, recording the robot joints positions and the filtered and compensated forces/torques in the robot's frame. These recorded data were normalized by 0-1 scaling them for obtaining a suitable data set. Test samples for evaluating the learning technique performance were obtained by simply selecting (and removing) one out of each ten executions of the training sets, corresponding to both the random and strategy experiments. In order to evaluate the appropriateness of LWL techniques to our manipulation task, the **LWPLS** algorithm [10] was implemented using the "strategy" data set. As this method works in batch mode, each demonstration was reduced by taking just the tenth part of it (i.e. each demonstration was sampled at 10 Hz), lowering the computational cost of the training stage. Several trainings were carried out with the aim of tuning the distance metric parameter for obtaining better results at the prediction stage. So, the training data set was used for finding the "best" values for h, by giving different values to h and evaluating the mean square errors for the predictions obtained with each value. Those h values with the least mean square error for each output were chosen for obtaining the predictions for the queries in a subsequent stage. After the optimal h values were found, each remaining experiment was used as query for the LWPLS algorithm and the mean square errors were computed for each output. It is important to highlight that the inputs for the LWPLS method are forces and torques in the robot's frame $(F_x, F_y, F_z, T_x, T_y, T_z)$ and the outputs are the six robot joints $(q_1, \ldots, q_6)$ for this case. Here, we desired to test the LWL techniques robustness in front of irrelevant inputs as it is the case of the first three robot joints that control the end-effector position, which seem less relevant for our application, since the orientation of the container is what the teacher significatively modifies for achieving the goal of the task.

The features of **LWPR** as a non-memory-based learning system, stressed in the previous section, fit ideally our problem, and thus we have tested it on our experimental setting. We used the same training "strategy" dataset as for the **LWPLS** algorithm and a "random" dataset, with the same input and output sets. Again, for both datasets, we needed to perform a distance metric tuning process obtaining the one that provides the best predictions of our training data. In this process, first of all, we disabled the incremental updating of the distance metric with the aim of analyzing which initial values for $\mathbf{D}$ have a good performance by keeping the distance metric fixed in the training phase. Along trainings with different values, we checked the prediction performance for the training dataset and retrained the model with an increased distance metric until a satisfying accuracy was achieved. Afterwards we enabled the distance metric adaptation stage and using the "best" initial $\mathbf{D}$ values, we carried out the training stage again, now with the aim of finding the optimal $\mathbf{D}$ values through its incremental updating for each local model in our LWPR model. It is important to highlight that the smaller the kernels (with high initial values for $\mathbf{D}$), the better the fitting error on the training data, but this means nothing for generalization. After we ran the prediction phase using the same test experiments used for LWPLS and the corresponding remaining experiments in the "random" dataset, with the aim of computing the MSEs and evaluating if the system learns the demonstrated task.

4 Results

As it was described above, we have a test experiment for each position with the aim of evaluating the LWL techniques performance. First of all, we used the LWPLS algorithm with the best h values for predicting the output of eleven data sets, each one corresponding to an experiment for a given initial position, using the "strategy dataset". From the prediction errors for each input data point, we computed the mean squared error for each output dimension in each experiment (see Table 1). In this case, as we used the "strategy dataset", it is possible to infer that if the prediction error is low, the predicted action will be similar to that taught, thus if the mean squared error along an experiment for a given position is low, it is possible to conclude the actions were learned in a satisfactory way and thus the movements strategy was learned successfully.

From Table 1, it is possible to see all MSEs for each robot joint are low, with the exception of those belonging to seventh initial position, where their prediction errors for each output were greater than the rest. However, most of the errors are low, which means the predicted values

Table 1: MSEs for each output for each position (using the strategy dataset) - Using the **LWPLS** algorithm

	$MSE(q_1)$	$MSE(q_2)$	$MSE(q_3)$	$MSE(q_4)$	$MSE(q_5)$	$MSE(q_6)$	MSE position
Pos. 1	0.0082	0.0031	0.0020	0.0173	0.0042	0.0137	0.0080
Pos. 2	0.0042	0.0109	0.0051	0.0028	0.0105	0.0047	0.0063
Pos. 3	0.0094	0.0929	0.0869	0.0068	0.0064	0.0064	0.0348
Pos. 4	0.0147	0.0042	0.0123	0.0092	0.0077	0.0071	0.0092
Pos. 5	0.0150	0.0058	0.0095	0.0324	0.0145	0.0106	0.0146
Pos. 6	0.0170	0.0586	0.0598	0.0108	0.0080	0.0070	0.0268
Pos. 7	0.1557	0.1581	0.0951	0.0984	0.0271	0.0244	0.0931
Pos. 8	0.0853	0.0108	0.0284	0.0576	0.0042	0.0120	0.0330
Pos. 9	0.1252	0.0155	0.0775	0.0313	0.0313	0.0153	0.0493
Pos. 10	0.0118	0.0368	0.0138	0.0096	0.0144	0.0074	0.0156
Pos. 11	0.0042	0.0109	0.0051	0.0028	0.0105	0.0047	0.0239
MSE joints	0.0412	0.0416	0.0385	0.0255	0.0142	0.0105	

for each robot joint are close to the real ones, thus the actions based on these predictions will be very similar to the those that were taught by the demonstrator, so the proposed strategy by the teacher was learned successfully.

After testing the LWPLS appropriateness working in batch mode, we wanted to test the LWPR performance with the aim of evaluating if this algorithm was suitable for working on our data set in an incremental way. As with LWPLS, we used the "strategy dataset" for knowing if the robot learning system was able to learn the "hidden" strategy in the training dataset. As above, we computed the mean squared error for each output dimension in each experiment (see Table 2). From these results, it is possible to infer the system could learn the strategy demonstrated by the teacher through the examples in an incremental way. We compared the MSEs obtained in batch and online modes, where it is possible to see that the most difficult initial positions to be learned were 3, 6, 7, 8 and 9 positions in both modes. In contrast, the easiest initial positions to be learned were 1, 2, 4 and 5 positions both in LWPLS and LWPR. Also, it is important to highlight that the largest MSEs correspond to those computed for q_1, q_2 and q_3, that presented also the largest MSEs at the training phase, which seems a normal consequence. This is concordant with the fact that these variables are the least relevant to achieve the task's goal, as mentioned before, since they control the end-effector position and not its orientation, therefore having only a minor effect on the targeted ball motion.

On the other hand, the "random dataset" was also tested with the LWPR method. With the aim of evaluating if the system could learn the examples demonstrated by the teacher, the MSEs for each output dimension in each experiment were computed. Here, the relevant issue is that the robot learning system is able to generalize and create a set of motions (joint positions) for a given input, from a training dataset which apparently does not have a predefined strategy. We reviewed the MSEs in the Table 3, where most of the error are below 0.02 which indicates a good generalization from the "random strategy" training dataset. Thus, it is possible to deduce that the robot learning system was able to learn the task from examples whose movements did not have a predefined set of actions. However, despite the examples provided by the teacher did not have a strategy, it could be possible that there was a common set of motions for each initial position of the ball which the teacher carried out without being aware of this. It means that it is possible that the teacher developed a "*taking the shortest way*" strategy for taking the ball out the container, which has been better learned than that with a predefined set of movements (i.e. "the strategy dataset"). On the other hand, it is interesting to highlight that in this case, when demonstrating unknown robot tasks through teleoperation, a human user may leverage

Table 2: MSEs for each output for each position (using the strategy dataset) - Using the **LWPR** algorithm

	$MSE(q_1)$	$MSE(q_2)$	$MSE(q_3)$	$MSE(q_4)$	$MSE(q_5)$	$MSE(q_6)$	MSE position
Pos. 1	0.0124	0.0048	0.0011	0.0304	0.0077	0.0515	0.0179
Pos. 2	0.0029	0.0070	0.0010	0.0007	0.0162	0.0087	0.0060
Pos. 3	0.0056	0.1224	0.1160	0.0054	0.0053	0.0051	0.0433
Pos. 4	0.0211	0.0006	0.0075	0.0186	0.0070	0.0144	0.0115
Pos. 5	0.0035	0.0021	0.0048	0.0215	0.0165	0.0073	0.0092
Pos. 6	0.0136	0.1078	0.1107	0.0114	0.0142	0.0066	0.0440
Pos. 7	0.0771	0.0514	0.0224	0.0482	0.0151	0.0240	0.0397
Pos. 8	0.1430	0.0117	0.0508	0.0909	0.0077	0.0315	0.0599
Pos. 9	0.1032	0.0037	0.0635	0.0152	0.0333	0.0088	0.0379
Pos. 10	0.0146	0.0502	0.0237	0.0051	0.0145	0.0166	0.0207
Pos. 11	0.0162	0.0958	0.0614	0.0025	0.0347	0.0157	0.0377
MSE joints	0.0375	0.0415	0.0420	0.0227	0.0156	0.0244	

information, latent in their mind, that is not observable to the robot. Such information may include user preferences as to how a task should be performed or state information observable to the human but not the robot (e.g. the visual input provides the position of the ball inside the container, such information is not available to the robot). Despite of this, the robot learning system showed a good performance in front these conditions, carrying out good predictions for most of the given queries.

Finally, reviewing Tables 1 and 2, it is possible to infer that it is easier to learn the sequence of movements that take the ball to the hole starting from the edge where the container is hold by the robot than the corresponding movements when starting from the opposite edge, where the fundamental difference lies in the last movement that should be done when the ball is touching the wall adjacent to the hole, since the container orientation for taking the ball along this wall is different. This conclusion is extracted from the fact the lowest MSEs correspond to first five initial positions (with the exception of the third one). This behavior is not present in the results for the "random strategy" (Table 3). Another interesting aspect is that the three cases have the lowest MSEs per position around 0.006-0.0066, however the worst MSE per position corresponding to the **LWPLS** method (0.093) is significatively greater than the worst MSEs corresponding to the **LWPR** algorithm (0.059 for the *strategy* dataset and 0.056 for the *random* dataset). This can be attributed to the fact that **LWPR** provides the updating process of the distance metric for each local model, optimizing the prediction performance in each of them.

5 Conclusions

In the framework of the European project "Perception, Action & Cognition through Learning of Object-Action Complexes (PACO-PLUS)", we aim at devising a system to teach manipulation skills to a robot in a domestic environment. Since no programming expertise should be required from the teacher, we have opted for a programming by demonstration approach where teacher instruction should be followed by autonomous robot rehearsal to adapt the instructed skills to the robot kinematic structure. This paper has described two steps towards this general goal, namely signal conditioning to filter out disruptive sensing components, and haptic teaching comparing the use or not of a strategy.

Force and torque feedback constitute valuable input sources for learning manual skills, especially when no visual information is available. This information, whether provided in batch mode or as a continuous data stream, has been successfully used for feeding, respectively, a LW-

Table 3: MSEs for each output for each position (using the random dataset) - Using the **LWPR** algorithm

	$MSE(q_1)$	$MSE(q_2)$	$MSE(q_3)$	$MSE(q_4)$	$MSE(q_5)$	$MSE(q_6)$	MSE position
Pos. 1	0.0615	0.0196	0.0093	0.0162	0.0132	0.0164	0.0227
Pos. 2	0.0107	0.0140	0.0479	0.0028	0.0047	0.0099	0.0150
Pos. 3	0.0071	0.0156	0.0017	0.0024	0.0084	0.0044	0.0066
Pos. 4	0.0236	0.0102	0.0059	0.0133	0.0115	0.0060	0.0117
Pos. 5	0.0090	0.0034	0.0462	0.0113	0.0184	0.0133	0.0169
Pos. 6	0.0991	0.0152	0.1226	0.0838	0.0108	0.0092	0.0567
Pos. 7	0.0094	0.0025	0.0099	0.0123	0.0025	0.0138	0.0084
Pos. 8	0.0718	0.0191	0.0103	0.0435	0.0052	0.0054	0.0258
Pos. 9	0.0252	0.0141	0.0280	0.0237	0.0095	0.0024	0.0171
Pos. 10	0.0075	0.0105	0.0166	0.0329	0.0104	0.0022	0.0133
Pos. 11	0.0302	0.0081	0.0848	0.0236	0.0386	0.0091	0.0324
MSE joints	0.0322	0.0120	0.0348	0.0241	0.0121	0.0083	

PLS and a LWPR algorithm, representatives for memory-based and non-memory-based LWL methods. These algorithms have been able to learn simple rigid-container emptying skills, as shown by the reduced obtained mean square errors, as a measure of the discrepancy between real and predicted (as output of the learning process) trajectories. Coherently, both algorithms produced the same pattern of results as regards to both the involved robot joints and the different initial experimental conditions. Tests where the teacher was instructed to follow a strategy compared to others where he was not provided useful expertise that permit devising the new research stages, where the taught motion will be refined using reinforcement learning.

In future work, more involved strategies may arise by including obstacles inside the container, like the walls of a maze. We think of the described experimental setting as a first step in the consideration of other sensorial input, emptying a pill box, for example, where the weight of the last pills may not be significant enough, as compared to the box, finer touch/impact sensors or even sound (together with a sensing directed action as shaking) could be taken into account instead. Another setting that includes the need to resort to non-visual information and which can be regarded as the natural extension of the present work consists in emptying deformable containers like bags, which may adopt shapes that make it difficult to visually distinguish whether there is still something inside. Related work, bag-emptying learning based on a virtual reality telerobotic interface and using a Q-learning algorithm, can be found in [6].

6 Acknowledgments

This work has been partially supported by the European project "Perception, Action & Cognition through Learning of Object-Action Complexes (PACO-PLUS)" (CogSys Integrated Project IST-4-27657), and the Spanish project "Multimodal Interaction in Pattern Recognition and Computer Vision (MIPRCV)" (Consolider Ingenio 2010 project CSD2007-00018).

References

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *AI Review*, pages 11–73, 1997.

[2] D. Bentivegna, C. G. Atkeson, A. Ude, and G. Cheng. Learning to act from observation and practice. *International Journal of Humanoid Robots*, 1(4):585–611, 2004.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Springer Handbook of Robotics*, chapter 59. Robot Programming by Demonstration, pages 1371–1394. Springer, Berlin, Heidelberg, 2008.

[4] D. Economou, C. Lee, C. Mavroidis, and I. Antoniadis. Robust vibration suppression in flexible payloads carried by robot manipulators using digital filtering of joint trajectories. In *International Symposium on Robotics and Automation*, pages 244–249, November 2000.

[5] D. Economou, C. Mavroidis, and I. Antoniadis. Robust residual vibration suppression using iir digital filters. In P. Horacek, editor, *System, Structure and Control 2001*, pages 687–692, Kidlington, Oxford, 2001. Elsevier Science Ltd.

[6] Y. Edan, , U. Kartoun, and H. Stern. Cooperative human-robot learning system using a virtual reality telerobotic interface. In *Conference on Advances in Internet Technologies and Applications*, 2004.

[7] J. G. Garcia, A. Robertsson, J. G. Ortega, and R. Johansson. Generalized contact force estimator for a robot manipulator. In *Robotics and Automation, 2006. ICRA 2006*, pages 4019–4024, 2006.

[8] S. Schaal, , C. G. Atkeson, and S. Vijayakumar. Real-time robot learning with locally weighted statistical learning. In *Robotics and Automation, 2000. ICRA 2000*, pages 288–293, 2000.

[9] S. Schaal. Learning from demonstration. In *NIPS '96: Proceedings Of The 1997 Conference On Advances In Neural Information Processing Systems 9*, pages 1040–1046, Cambridge, MA, USA, 1996. MIT Press.

[10] S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[11] S. Schaal, S. Vijayakumar, and C. G. Atkeson. Local dimensionality reduction. In *NIPS '97: Proceedings Of The 1997 Conference On Advances In Neural Information Processing Systems 10*, pages 633–639, Cambridge, MA, USA, 1998. MIT Press.

[12] M. Uchiyama and K. Kitagaki. Dynamic force sensing for high-speed robot manipulation using kalman filtering techniques. In *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*, pages 2147–2152 vol.3, 1989.

[13] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

[14] S. Vijayakumar and S. Schaal. Locally weighted projection regression : An o(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 1079–1086, 2000.