Learning Force-Based Robot Skills from Haptic Demonstration

Leonel ROZO [1] , Pablo JIMÉNEZ and Carme TORRAS
Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028 Barcelona, Spain.

Abstract. Locally weighted as well as Gaussian mixtures 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. For the first time, force/torque feedback through a haptic device has been used for teaching a teleoperated robot to empty a rigid container. The memory-based **LWPLS** and the non-memory-based **LWPR** algorithms [1,2,3], as well as both the batch and the incremental versions of GMM/GMR [4,5] 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.

Keywords. Robot learning, haptic device, teleoperation, LWL, GMM/GMR

Introduction

Personal, domestic and 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 PACO-PLUS project, we are relying on *learning* to endow robots with the necessary skills [6]. 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 [7]. The natural way of learning skills is by observing how they are performed by a *teacher*. This is known as programming by demonstration or imitation learning [8], 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.

[1]Corresponding author: Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028 Barcelona, Spain.

Learning paradigms based on local characterization like Locally weighted learning (LWL) or Gaussian Mixture Models and Regression (GMM/GMR) fit well these demands, and especially that of coping 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 devil-sticking and pole-balancing [1], or air hockey playing [9], among others. We have adapted two such methods, namely Locally Weighted Partial Least Squares and Locally Weighted Projection Regression, to our particular setting and task. GMM/GMR algorithms have recently been used with great success in human gesture imitation [5] and we have also tested them in our setting.

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 1, the content is assumed heavy enough to be detected by a force/torque sensor mounted on the robot's wrist.

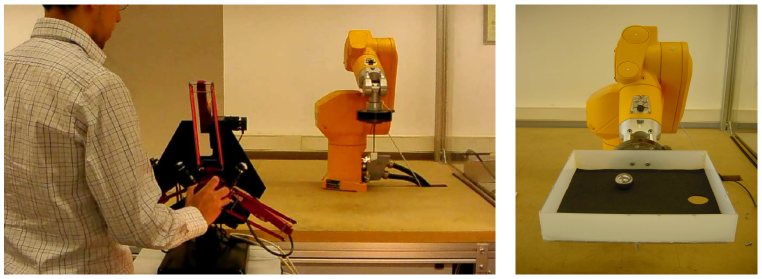
The remaining of the paper is structured as follows. Section 2.1 provides a brief description of LWL and GMM/GMR methods, and Section 2.2 explains how we have adapted them to the present context. The obtained results and their interpretation are described in Section 3. Finally, some conclusions are drawn in Section 4 and future work is indicated.

1. Experimental Setting

In our experimental setting a STAUBLI RX-60 robotic arm with a force/torque (F/T) 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(a)). 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 external source like the robot's F/T sensor. In sum, this setting enables the user to command remotely the robot arm while feeling the interaction forces and torques produced on the robot arm's wrist.

In our experimental setup, the robot arm has a rigid rectangular container with a hole attached at its wrist, as shown in Figure 1(b). Inside the box, a ball is free to roll around. The goal 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



(a) Teleoperation setting (b) Robot arm and container

Figure 1. Learning by demonstration setting using a haptic device

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 would like to feed back to the haptic device only those signals which are generated by the ball's dynamics, thus it is necessary to eliminate both noise and forces/torques due to 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 [10], 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. 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 [11,12]. For more details about filtering and dynamic compensation processes applied to this setting, refer to [16].

The setting described has an evident academic flavour. The container and the ball have been dimensioned so as to provide a suitable collection of measurements. Further experiments will include more realistic settings. Nonetheless 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 and GMM/GMR techniques.

2. Learning the Manipulation Task

2.1. *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. In what follows we briefly describe the specific algorithms used in this work, grouped in two families: *Locally weighted learning* (**LWL**) based methods and algorithms based on *Gaussian mixture models and regression* (**GMM/GMR**).

2.1.1. Locally Weighted Learning

LWL aims at nonlinear function approximation by using piecewise linear functions [1]. Under this key concept, several algorithms have been developed, grouped into two families: memory-based LWL and non-memory-based LWL. In what follows, brief descriptions of the two implemented algorithms belonging to these families, are given.

Locally Weighted Partial Least Squares (**LWPLS**) is a suitable method to reduce the computational complexity of *Locally Weighted Regression* (LWR) [13] and to avoid its numerical problems [3]. The idea behind PLS is to fit linear models by using a hierarchy of univariate regressions along selected projections on the input space which are chosen in accordance with input/output correlation and ensuring that the subsequent projections will be orthogonal in the input space. The approach followed by Schaal et al. [14,3] is based on the fact that global high dimensionality does not imply that the data remain high dimensional if viewed locally. Thus, they started performing 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.

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 of running 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 [15]. 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 ***i)*** it learns rapidly with second-order learning methods based on incremental training, ***ii)*** it uses statistically sound stochastic LOOCV for learning without the need to memorize training data, ***iii)*** it adjusts its weighting kernels based only on local information in order to minimize the danger of negative interference of incremental learning, ***iv)*** it has a computational complexity that is linear in the number of inputs, and ***v)*** it can deal with a large number of - possibly redundant - inputs [2].

2.1.2. GMM and GMR

GMM are a probabilistic representation of data where a set of Gaussian models work as universal approximators of densities. In this approach, as it is described in [4], the main idea is to model data from a mixture of *K* Gaussians – of dimensionality *d*, with $d = n + m$, being *n* and *m* the input and output spaces dimensions, respectively – defined by a probability density function:

$$p(Z_j) = \sum_{k=1}^{K} p(k) p(Z_j|k) \tag{2}$$

where Z_j is a datapoint ($Z = \{Z_i, Z_o\}$, with Z_i and Z_o representing the input and output data, respectively), $p(k)$ is the prior and $p(Z_j|k)$ the conditional probability density function. Thus, the parameters in (2) are: $p(k) = \pi_k$ and $p(Z_j|k) = N(Z_j; \mu_k, \Sigma_k)$, with π_k, μ_k and Σ_k defining respectively the prior, mean and covariance matrix of the *k*th Gaussian. The GMM's initial values are computed through the *k-means* clustering technique and then the GMM are trained by using the standard *Expectation-Maximization* algorithm with the aim of finding the best representation of the data from the Gaussian components. From the trained GMM, it is possible to recover the general form of the data by applying GMR. Assuming that a set of query points is available, their corresponding predictions can be estimated through regression.

This algorithm is implemented in batch mode, computing the GMM from the input and output data saved in memory, and then solving the regression with the resulting GMM parameters. However, Calinon and Billard in [5] proposed two incremental versions for GMM/GMR based learning, namely: *direct update* and *generative update* methods, where the first one showed a better performance confirmed by our implementation and test. The direct method is based on the idea of using temporal coherence properties of data to update GMM. The objective is to adapt the EM algorithm by separating those parts dedicated to the data already used to train the model from those devoted to the newly available data, assuming that the set of posterior probabilities remains without changes when new data are used to update the model. Thus, first of all, the model is created with *N* datapoints Z_j and updated in an iterative way during *T EM-steps*, until convergence to the set of parameters $\{\pi_k^{(T)}, \mu_k^{(T)}, \Sigma_k^{(T)}, E_k^{(T)}\}$. After that, when new data are available, $\tilde{T}$ *EM-steps* are again carried out to adjust the current model to the new $\tilde{N}$ datapoints, taking as initial values of parameters those resulting from the former stage.

2.2. Implementation Issues

Demonstrations were carried out by teleoperating the robot arm until taking the ball out of the box in each example. Figure 2 displays the initial positions where the ball was released. Starting at each predefined initial position, twenty demonstrations were performed, of 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 where the teacher just tried to take out the ball regardless of the movements. Test samples for evaluating the learning technique performance were obtained by simply removing one out of each ten executions of the training sets, corresponding to both the random and strategy experiments.

2.2.1. LWL Implementation

Regarding the LWPLS algorithm, several trainings were carried out with the aim of tuning the distance metric parameter for obtaining better results at the prediction stage. After that, 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 high-

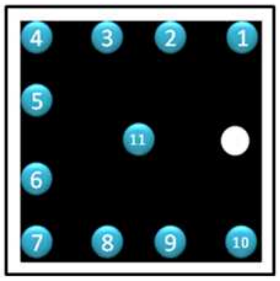


Figure 2. Initial positions of the ball

light that the inputs for the LWPLS method are forces and torques in the robot's frame $(F_x, F_y, F_z, N_x, N_y, N_z)$ and the outputs are the six robot joints $(q_1, \ldots, q_6)$ for this case. Here, we desired to test the robustness of the LWL techniques 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" and "random" datasets as for the LWPLS, 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 this variable 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 values, we carried out the training stage again, now with the aim of finding the optimal distance metric values through its incremental updating for each local model in our LWPR model. 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.

2.2.2. GMM/GMR Implementation

Both batch and incremental versions of GMM/GMR were tested in a similar fashion as LWL based techniques. Strategy and random datasets were used for evaluating the performance of GMM/GMR-based methods. In a preliminary stage, both input and output sets were subjected to a principal components analysis (PCA) with the aim of reducing the dimensionality of the input and output spaces, by removing irrelevant dimensions and thus solving one of the main learning paradigmatic questions: *What to imitate?* [8]. Analyzing the resulting eigenvectors from PCA, it was possible to infer that just two of them were enough to generate the new reduced space without increasing the prediction error significantly, for both input and output datasets. This reduction is related to the facts that only two inputs are important for carrying out the task (i.e. torques generated about the main axes of the box plane, where the ball rolls on), and that just two robot joints are necessary and sufficient for achieving the given goal (i.e. those robot's wrist joints that control the orientation of the box about its axis on the plane, without taking into account the joint that controls the orientation about the normal axis to the plane which does not generate significant movements of the ball). Thus both PCAs applied to input and output datasets led to select just the two eigenvectors with highest eigenvalues, which transforms the initial data set $Z = \{X, Y\}$ to a new reduced one $\xi = \{\chi, \psi\}$.

Based on ideas proposed by Calinon and Billard [4,5], the GMM/GMR was implemented for obtaining the probability of generating ψ given χ, i.e. $p(\psi|\chi)$. From several tests, only two Gaussian components were considered for training GMM because this configuration showed a very similar performance to those GMM with more models. With GMMs trained, GMR was used for computing the prediction for a set of given queries, corresponding to those experiments extracted from each set of demonstrations for each initial position. Once all predictions were computed from GMR, the MSEs were computed with the aim of evaluating the performance of the GMM/GMR-based algorithms.

3. Results

We have conducted a series of experiments in order to test the performance of the learning algorithms as well as to provide a basis for comparison [16]. Eleven "strategy datasets", each one corresponding to an experiment for a given initial position of the ball in the recipient, were used for testing the algorithms: F/T values along each trajectory constituted the inputs to the prediction modes of these algorithms, the output robot coordinate values were subtracted from the values predicted by the algorithms to obtain the prediction errors. From the prediction errors for each input data point, the mean squared error for each output dimension in each experiment was computed. A low mean squared error along an experiment for a given start position means that the predicted trajectory is similar to the demonstrated one, or, in other words, that the movements strategy was learned successfully. We proceeded similarly with further eleven "random datasets", to evaluate whether the algorithms were 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. Furthermore, as another performance measure, prediction times (i.e., time inverted in computing each prediction) were also measured.

Table 1. Prediction times for tested algorithms

Algorithm	Prediction time(s)
LWPLS	3.0347
LWPR	0.2187
GMM/GMR(Batch)	0.0320
GMM/GMR(Incremental)	0.0302

In sum, the computed performance measures from these experiments allow us to evaluate the following aspects: ***a)*** underlying algorithmic principle, i.e., piecewise linear local approximation versus mixtures of local Gaussian models, ***b)*** batch modes versus incremental versions, ***c)*** incidence of each output dimension of the learned actions on

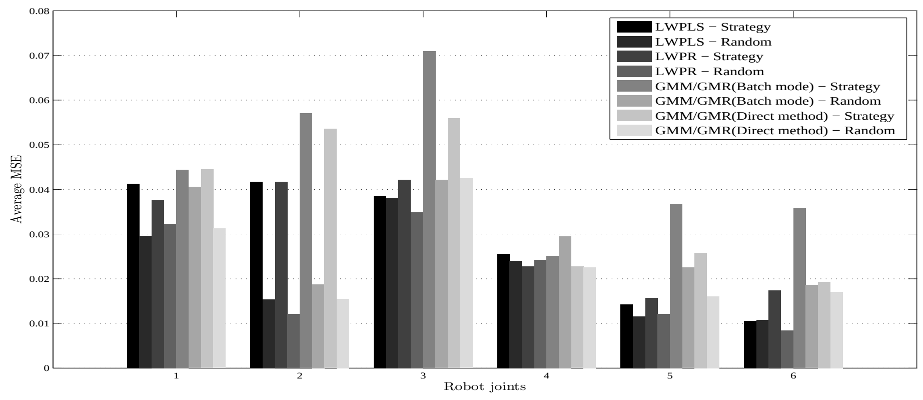


Figure 3. Average MSEs for each robot's joint

performance, ***d)*** influence of the starting position of the ball, ***e)*** learning a specific motion strategy versus learning random trajectories.

The obtained MSEs have been summarized in Figures 3 and 4, whereas prediction times appear in Table 1. The following results can be extracted from analyzing these figures: ***a)*** The performance of the learning algorithms in terms of prediction errors is quite good. Most average MSEs are below 0.04 for any one of the tested learning algorithms and data sets. This means that the predicted values for each robot joint are close to the actual ones, thus the actions based on these predictions are very similar to those taught by the demonstrator, and the "hidden" strategy proposed by the teacher as well as the random trajectories were learned successfully. ***b)*** For this kind of tasks, no major differences exist in the performance of LWPLS/LWPR as compared to GMM/GMR when considering prediction errors, the tendency is of a slightly better performance of the first ones. However, prediction times are much shorter for the Gaussian models. ***c)*** Figure 3 shows that the prediction errors for the three last robot joints are lower than for the first positioning joints. This is concordant with the fact that these variables are the least relevant to achieve the task's goal, since they control the end-effector position and not its orientation, therefore having only a minor effect on the targeted ball motion. ***d)*** The analysis of Figure 4 throws some expected conclusions and some surprising results. That the trajectories starting at positions 1 and 10 belong to the most predictable ones by all the algorithms was to be expected, due to the closeness to the hole and the single degree of freedom involved. For the same reason, it is peculiar that trajectories originating at position 11 are in the group of those most difficult to learn. A possible interpretation of this fact is that the ball does not reach directly the hole, but hits the wall very close to it, and the F/T data corresponding to these positions are only slightly different from those of the exit. The trend is that it is seemingly easier to tilt the platform left and down, that is, to learn the trajectories starting from the edge where the container is hold by the robot, than left and up. Observe the difference between starting positions 5 and 6, which may be magnified by the same reason as starting position 11. ***e)*** It may surprise at first glance that the overall performance of generalizing the "random dataset", without a predefined set of actions, is better than learning the "strategy dataset". This may be ascribed to the existence of a common set of motions for each initial position of the ball which the teacher carried out without being aware of it. The teacher possibly developed a *taking the shortest way* strategy for leading the ball out of the container, which has been

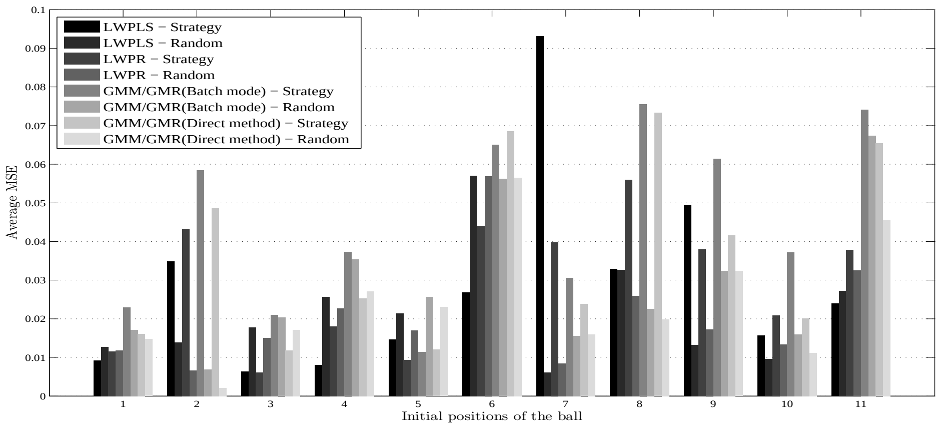


Figure 4. Average MSEs for each initial positions of the ball in the maze

better learned than that with a predefined set of movements. Undeliberate use of information, latent in the user's mind, that is not directly observable by the robot may also be present in demonstrating unknown robot tasks through teleoperation. 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, and such information is not available to the robot).

4. Conclusions and Future Works

In the framework of the PACO-PLUS project, 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. 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 LWPLS, LWPR, GMM/GMR and its direct update incremental method. 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, all algorithms behaved similarly on each dataset, producing similar patterns of results as regards to both the involved robot joints and the different initial experimental conditions. These datasets correspond to tests where the teacher was instructed to follow a strategy and others where he was not, and they provided useful expertise that permits 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 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, 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 [17].

Acknowledgements

This work has been partially supported by the European projects “PACO-PLUS” (IST-4-27657) and “GARNICS: Gardening with a Cognitive System” (FP7-247947), the Spanish project “Multimodal Interaction in Pattern Recognition and Computer Vision (MIPRCV)” (Consolider Ingenio 2010 project CSD2007-00018) and the consolidated Robotics group of the Generalitat de Catalunya. L. Rozo was supported by the CSIC under a JAE-PREDOC scholarship.

References

[1] S. Schaal, , C. G. Atkeson, and S. Vijayakumar, “Real-time robot learning with locally weighted statistical learning”, in *IEEE Intl. Conference on Robotics and Automation*, 2000, pp. 288-293.

[2] S. Vijayakumar, A. D’Souza, and S. Schaal, “Incremental online learning in high dimensions”, *Neural Computation*, 2005, vol. 17, pp. 2602-2634.

[3] S. Schaal, C. G. Atkeson, and S. Vijayakumar, “Scalable techniques from nonparametric statistics for real time robot learning”, *Applied Intelligence*, 2002, vol. 17, no. 1, pp. 49-60.

[4] S. Calinon, F. Guenter, and A. Billard, “On learning, representing, and generalizing a task in a humanoid robot”, *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 2007, vol. 37, no. 2, pp. 286-298.

[5] S. Calinon and A. Billard, “Incremental learning of gestures by imitation in a humanoid robot”, in *Intl. Conference on Human-Robot Interaction*, 2007, pp. 255-262.

[6] A. Agostini, E. Celaya, C. Torras, and F. Wörgötter, “Action rule induction from cause-effect pairs learned through robot-teacher interaction”, in *Intl. Conference on Cognitive Systems*, 2008, pp. 213-218.

[7] S. Schaal, “Learning from demonstration”, in *Conference on Advances In Neural Information Processing Systems 9*, 1996, pp. 1040-1046.

[8] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Robot Programming by Demonstration”, *Springer Handbook of Robotics*, 2008, ch. 59, pp. 1371-1394.

[9] D. Bentivegna, C. G. Atkeson, A. Ude, and G. Cheng, “Learning to act from observation and practice,” *Intl. Journal of Humanoid Robots*, 2004, vol. 1, no. 4, pp. 585-611.

[10] 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 *Intl. Symposium on Robotics and Automation*, 2000, pp. 244-249.

[11] J. G. Garcia, A. Robertsson, J. G. Ortega, and R. Johansson, “Generalized contact force estimator for a robot manipulator,” in *IEEE Intl. Conference on Robotics and Automation*, 2006, pp. 4019-4024.

[12] M. Uchiyama and K. Kitagaki, “Dynamic force sensing for high-speed robot manipulation using kalman filtering techniques,” in *28th IEEE Conference on Decision and Control*, 1989, vol.3, pp. 2147-2152.

[13] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning,” *AI Review*, 1997, pp. 11-73.

[14] S. Schaal, S. Vijayakumar, and C. G. Atkeson, “Local dimensionality reduction,” in *Conference On Advances In Neural Information Processing Systems 10*, 1998, pp. 633-639.

[15] S. Vijayakumar and S. Schaal, “Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space,” in *17th Intl. Conference on Machine Learning*, 2000, pp. 1079-1086.

[16] L. Rozo, P. Jimenez, and C. Torras, “Robot learning of container emptying skills through haptic demonstration,” Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Tech. Rep. IRI-TR-09-05, 2009.

[17] 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.