Sharpening haptic inputs for teaching a manipulation skill to a robot

Leonel Rozo, Pablo Jiménez and Carme Torras

***Abstract*— Gaussian mixtures-based learning algorithms are suitable strategies for trajectory learning and skill acquisition, in the context of programming by demonstration (PbD). 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. Moreover, some input variables seemed much more relevant to the particular task to be learned than others, which lead us to analyze the training data in order to select those relevant features through principal component analysis and a mutual information criterion. Then, a batch version of GMM/GMR [1], [2] was implemented using different training datasets (original, pre-processed data through PCA and MI). 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.**

I. INTRODUCTION

If robots are to collaborate with humans at home, at work, and in other human-centered environments where manipulation skills are required, approaches where a layman could teach a robot such skills by just demonstrating them become essential. These approaches are generally named learning (or programming) by demonstration [3], [4], or imitation learning. Key features in this context are: the teacher does not need to have expertise in robot programming but just in the manipulation skill, no predefined setup is required, and the skill is not ameanable to being taught by only symbolic/logic means (e.g., through verbal instruction). 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.

Learning paradigms based on local function approximation like Locally weighted learning (LWL) or Gaussian Mixture Models and Regression (GMM/GMR) fit well the aforementioned demands. LWL methods have been successfully used in a variety of applications, like devil-sticking and pole-balancing [5] or air hockey playing [6], among others. GMM/GMR algorithms have recently been used with great success in human gesture imitation [2] and teaching physical collaborative tasks [7], [8].

Within the PACO-PLUS project, we have devised a decision-making framework where skill learning serves to encode basic robot actions, which constitute the action repertoire of a symbolic rule system [9] that learns state-action rules, which in turn are the basic operators of a logic planner. In a previous work [10], we adapted both LWL and GMM/GMR methods to our setting and reported results for a few variants of them, assessing the importance of several parameters and factors in their performance. Something that became clear was the crucial role of a good conditioning of inputs for both the speed and quality of learning. Thus, we devote attention to this issue in this paper.

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 II, 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). Thus, the main goal of this paper is to analyze the demonstration data with the aim of obtaining a suitably conditioned and reduced data set that permits learning tasks based on complex force/torque signals in a fast and reliable way.

The remaining of the paper is structured as follows. Section II describes the container-emptying task addressed and the experimental setup. The next section is devoted to the necessary preprocessing of the haptic signals. Then, a brief description of the GMM/GMR method and its implementation, together with the analysis of the inputs relevance to the addressed manipulation task is covered in Section IV. The obtained results and their interpretation are described in Section V. Finally, some conclusions are drawn in Section VI and future work is indicated.

II. DESCRIPTION OF THE TASK AND EXPERIMENTAL SET-UP.

We aim at teaching a robot to extract a metallic ball (250 g) from inside a box-like container (30x30x6 cm^3), which has a hole on its base. The robot has to orient the box in such a way that the ball is forced to roll towards the hole and fall through it. The key concept is that the task is exclusively

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

The authors are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens Artigas 4-6, 08028 Barcelona, Spain. {lrozo, jimenez, torras}@iri.upc.edu

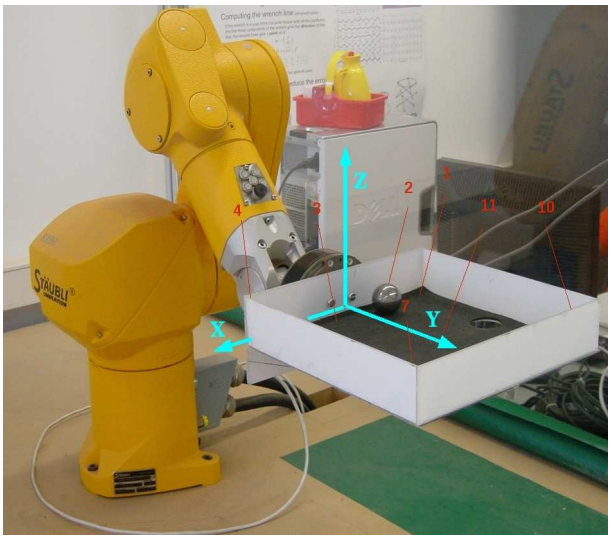

Fig. 1. Robot arm with F/N sensor and box. Observe the hole on the right side of the box. The ball is in initial position 2. Initial positions 1-10 are arranged along the the border of the box, in counterclockwise order, position 11 is just in the middle. The frame axes to which forces and torques are referred to are also displayed.

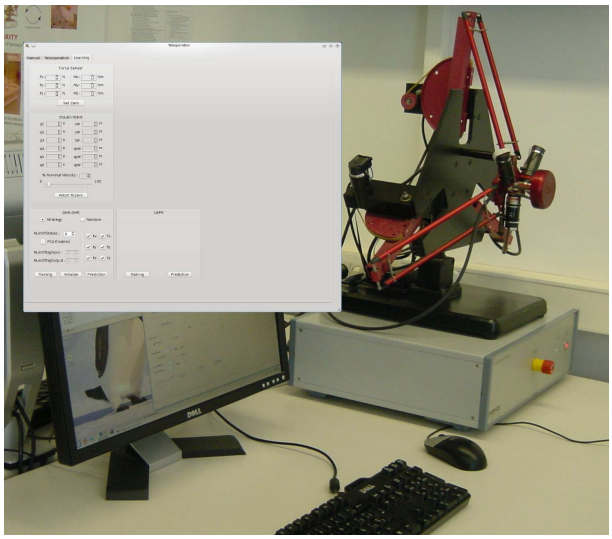

Fig. 2. Delta Haptic Device, whose end-effector is the knob in the middle. The superimposed image shows a snapshot of the GUI.

based on force/torque (F/N) feedback, i.e., the robot does only sense the forces and torques exerted on its wrist by the box and the ball, which change with the orientation of the box and the position of the ball. The teacher, instead, has an additional source of information by watching the scene. That is, besides feeling the weight distribution when performing the task, the user has visual feedback. To this end, the robot arm (a 6-DOF STAUBLI RX-60, in our case) is equipped with an F/N sensor (Shunk FTC-050) attached to its wrist. In order to simplify the experiments and to avoid having to consider the dynamic effects of a robotic hand while moving, the box is directly fixed to the F/N sensor (Figure 1).

On the user's side of our telemanipulation system, the teacher holds the end-effector of a Force Dimension 6-DOF Delta Haptic Device (Figure 2), and moves it around. The displacements and orientation changes produced at the end-effector are transformed by the controller to motion commands for the robot. Furthermore, the device allows the human teacher to feel on the same end-effector the forces and torques sensed at the robotic wrist. A graphical user interface running on a PC allows to test the correct operation of the system as well as to initiate the data gathering process, setting some learning parameters, and to execute the training or the prediction phase of the learning system, as explained later. The current F/N readings and the joint angles of the robot are also displayed. In sum, the teacher performs the necessary motions to extract the ball from the box while feeling how the ball rolls around, and the user's motion are reproduced at the robot arm carrying the box.

More formally, each position and orientation provided by the teacher at the end-effector of the haptic device (the six-component vector $\mathbf{x}$) is transformed to the robot's frame, thus becoming the desired configuration in the operational space. This instruction is sent to the robot's controller, whose duty is to apply the inverse transformation in order to supply the corresponding signal to each link of the arm, and move it to this pose. Simultaneously, the six-component $\mathbf{F/N}_s$ vectors, as read by the F/N sensor at the robot's wrist, are preprocessed (as described in the next section), transformed to the frame of the haptic device, and reproduced at its end-effector (and thus felt by the user or teleoperator).

III. Preprocessing the generated signals.

The F/N signals originating at the sensor have to be preprocessed before becoming suitable stimuli to be felt by the teacher, reflecting exclusively the dynamics of the rolling ball ($\mathbf{F/N}_b$). The original F/N signals can be considered to be composed by the following components:

$$\mathbf{F/N}_s = \mathbf{F/N}_b + \mathbf{F/N}_m + \varepsilon \quad (1)$$

with $\mathbf{F/N}_m$ corresponding to the container's mass and the noise ε due mainly to the vibration of the box. Next we describe how to eliminate the noise and the dynamic effects of the box.

A. Filtering the noise due to vibrations.

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 and a source of instability for our telemanipulation system. 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 [11], 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 manually a repeated impact perpendicularly to the container's base, at the front edge in Figure 1.

The frequency spectrum of the generated data was analyzed, obtaining the fundamental frequency as the cutoff frequency of our low-pass filter. This filter 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.

B. Compensating the dynamics of the box.

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 [12], [13]. To achieve this aim, let us denote the position of the center of gravity of the container as **p**, its mass as *m*, **I** as its moment of inertia, $\mathbf{F}_s/\mathbf{N}_s$ and $\mathbf{F}_e/\mathbf{N}_e$ 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. Then, using the Newton-Euler equations, we obtain:

$$\Sigma\mathbf{F} = m\ddot{\mathbf{p}} = m\mathbf{g} + \mathbf{F}_e + \mathbf{F}_s \quad (2)$$

$$\Sigma\mathbf{N} = \mathbf{I}\ddot{\mathbf{r}} + \dot{\mathbf{r}} \times \mathbf{I}\dot{\mathbf{r}} = \mathbf{N}_s + \mathbf{r}_s \times \mathbf{F}_s + \mathbf{N}_e + \mathbf{r}_e \times \mathbf{F}_e \quad (3)$$

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:

$$\mathbf{F}_s = -m\mathbf{g} - \mathbf{F}_e \quad (4)$$

$$\mathbf{N}_s + \mathbf{r}_s \times \mathbf{F}_s = -\mathbf{N}_e - \mathbf{r}_e \times \mathbf{F}_e \quad (5)$$

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.

IV. LEARNING THE MANIPULATION TASK

A. GMM and GMR

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 algorithm used in this work: *Gaussian mixture models and regression* (**GMM/GMR**). For more details, please refer to the cited works.

The main idea behind **GMM/GMR** 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. These Gaussians are defined by a probability density function:

$$p(Z_j) = \sum_{k=1}^{K} p(k)p(Z_j|k) \quad (6)$$

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 [1]. The parameters in (6) are:

$$p(k) = \pi_k \quad (7)$$

$$p(Z_j|k) = N(Z_j; \mu_k, \Sigma_k) \quad (8)$$

where π_k, μ_k and Σ_k correspond respectively to the prior, mean and covariance matrix of the *k*th Gaussian. The *k-means* clustering technique is used to compute the GMM's initial values and afterwards the GMM are trained by using the standard *Expectation-Maximization* algorithm in order to determine the best representation of the data from the Gaussian components [14].

Once the trained GMM are obtained, a general form of the data can be recovered by applying GMR. For a set of query points, their corresponding predictions can be estimated through regression. So, for each Gaussian component *k*, both input and output data are separated by expressing the mean and covariance matrix as:

$$\mu_k = \{\mu_{i,k}, \mu_{o,k}\} \quad , \quad \Sigma_k = \begin{pmatrix} \Sigma_{ii,k} & \Sigma_{io,k} \\ \Sigma_{oi,k} & \Sigma_{oo,k} \end{pmatrix}$$

Then, the conditional expectation of output data $Z_{o,k}$ given the query Z_i, and the estimated conditional covariance matrix of $Z_{o,k}$ given Z_i are:

$$\hat{Z}_{o,k} = \mu_{o,k} + \Sigma_{oi,k}(\Sigma_{i,k})^{-1}(Z_i - \mu_{i,k}) \quad (9)$$

$$\hat{\Sigma}_{o,k} = \Sigma_{o,k} - \Sigma_{oi,k}(\Sigma_{i,k})^{-1}\Sigma_{io,k} \quad (10)$$

Thus, the conditional expectation and the conditional covariance of $\hat{Z}_o$ given Z_i, for a mixture of *K* Gaussians are:

$$\hat{Z}_o = \textstyle\sum_{k=1}^{K} \beta_k \hat{Z}_{o,k} \quad , \quad \hat{\Sigma}_o = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{o,k} \quad (11)$$

where $\beta_k = \frac{p(Z_i|k)}{\sum_{l=1}^{K} p(Z_i|l)}$.

In this way, it is possible to compute a prediction for a given query from $\hat{Z}_o$. In batch mode, the GMM are computed from the input and output data saved in memory, by solving the regression with the resulting GMM parameters. Nonetheless, also incremental versions for GMM/GMR based learning exist, as the two proposed by Calinon and Billard [2], the *direct update* and *generative update* methods, where the first one showed a better performance confirmed by our implementation and test [10]. In the direct method, the EM algorithm is modified by separating those parts dedicated to the data already used to train the model from those devoted to the newly available data, based on the assumption that the posterior probabilities will not change as new data are introduced to update the model (by temporal coherence).

That is, first the model is created with *N* datapoints Z_j and updated in an iterative way during *T EM-steps*, until the parameters $\{\pi_k^{(T)}, \mu_k^{(T)}, \Sigma_k^{(T)}, E_k^{(T)}\}$ converge. Then, as soon as there are new data available corresponding to new trajectories, $\tilde{T}$ *EM-steps* are again carried out to adjust the current model to the new $\tilde{N}$ datapoints, taking as initial values of parameters those obtained from the previous stage (see [2] for details).

B. Assessing input relevance to the task

In the context of supervised learning, with an output **Y** which is a function of a set of inputs $\{\mathbf{X}_0, ..., \mathbf{X}_n\}$ a well-known result concerns the dependency of the output with respect to each of its inputs. Knowing the inputs relevance with respect to the output allows to reduce the input space for a learning algorithm by removing low-relevance variables as well as eliminating noise which can make harder to learn a specific task. Mutual information-based feature selection (**MIFS**) is a very suitable tool for achieving these objectives.

Mutual information (**MI**) is one of the most fundamental information measures in information theory. Initially, it was mostly used at engineering of noisy communication channels, but other fields of application have arisen as well. The concept behind mutual information applied to feature selection [15] is the reduction of the output data uncertainty, considering each input variable. Depending on how the uncertainty of the output data is reduced, an input gives more or less information about the output, or in other words, it is highly or lowly correlated to the output. In order to know the grade of importance of an input **X** with respect to the output **Y** it is necessary to compute the MI $I(\mathbf{X}, \mathbf{Y})$ between them (for more details see [16]):

$$I(\mathbf{X}, \mathbf{Y}) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \qquad (12)$$

With the aim of reducing the input space dimensionality (speeding up learning and prediction stages) and removing noise (simplifying the task to learn), the training data were subjected to MI analysis. At a first phase, mutual information was computed for each pair input/output – considering as inputs forces/torques sensed and transformed to the robot's frame, and as outputs each robot joint – using entire trajectories from the training data. From all MI values obtained for each variable with respect to each output, a simple average MI value was calculated which shows a good estimation on how relevant an input variable is in comparison with all outputs, as Figure 3 shows. In general terms, the input variables F_y and N_z show less relevance whereas N_x and N_y are the most correlated variables with outputs which does make sense as they are the variables that give the most useful information for knowing where the ball is inside the box (see Figure 4). Therefore, it is possible to carry out the training and prediction phases removing F_y and N_z for both learning approaches without influencing their performance significantly. Figure 3 shows MI values for different number of intervals *k* used for computing the conditional and marginal probabilities for equation 12.

C. Implementation issues

In the learning stage, demonstrations consisted in teleoperating the robot arm, tilting the container until the ball left the box through the hole. Starting at each predefined initial position (see Figure 1), twenty demonstrations were performed, in ten of which a particular motion strategy was used: 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 without caring about performing specific motions.

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. 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. Moreover, when testing the batch version, each demonstration was reduced by taking just the tenth part of it (i.e. each demonstration was sampled at 10 Hz), so as to lower the computational cost of the training stage.

At a first stage, our initial training data consisted of an inputs set corresponding to forces and torques in the robot's frame $(F_x, F_y, F_z, N_x, N_y, N_z)$ and an outputs set made up of the six robot joints $(q_1, \ldots, q_6)$. Strategy and random datasets were used as described above. In a preliminary stage, a principal components analysis (PCA) was applied to the input and the output sets, so as to reduce the dimensionality of the input and output spaces: the data are projected to a *latent space* and irrelevant dimensions are removed. This addresses one of the main learning paradigmatic questions, *What to imitate?* [3]. The eigenvectors obtained from PCA were analyzed, and we concluded that two of them were enough to generate the new reduced space without increasing the prediction error significantly, for both the input and output datasets. The fact 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, N_x and N_y), 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 axes on the plane, q_5 and q_6) seem to confirm that such reduction is possible. Following these criteria, 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 $\xi = \{\chi, \psi\}$.

The GMM/GMR was implemented for obtaining the probability of generating ψ given χ, i.e. $p(\psi|\chi)$, following [1], [2]. After some experiments, it could be concluded that only two Gaussian components had to be considered for training GMM, obtaining performance which was close to those GMM trained with more models. After training GMMs, we used GMR to compute the prediction for a set of given queries. These queries were extracted from each set of

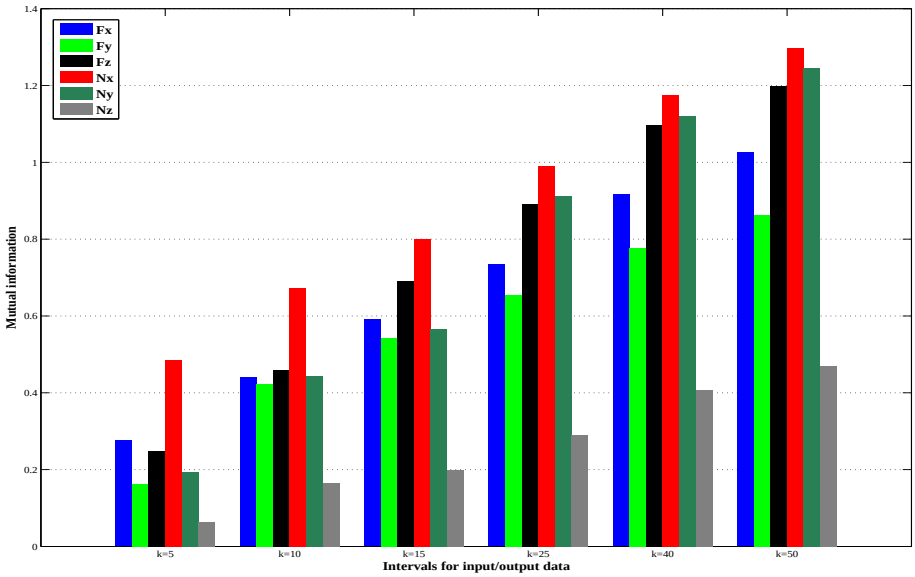

Fig. 3. Mutual information values

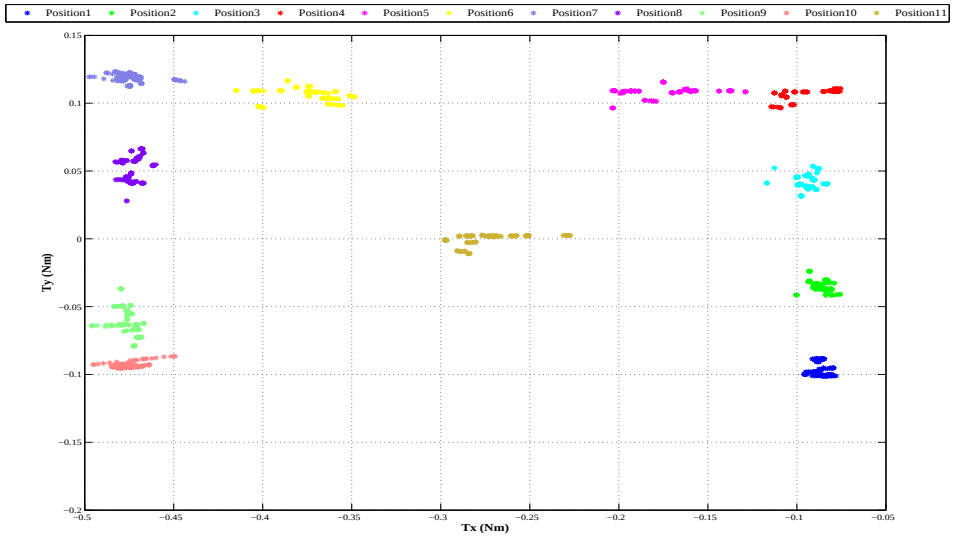

Fig. 4. Torques map for each initial position (Cluster)

demonstrations for each initial position (Figure 5 shows some predicted trajectories for given queries). Finally, MSEs were used as performance measure of the different GMM/GMR-based approaches.

On the other hand, at a second stage a mutual information analysis was carried out on the training data with the aim of selecting those input variables with a high influence on the outputs, keeping in mind that this feature selection process directly reduces the dimensionality of the input space and removes noise generated by irrelevant dimensions as well. It is important to highlight that it can be considered as another approach to solve the paradigmatic question: *What to imitate?*, because the algorithm just learns from that information that is relevant for the task to execute. So, observing MIFS results showed in Figure 3, a reduced input dataset (F_x, F_z, N_x, N_y) was used for evaluating the GMM/GMR performance on this new resulting training data composed of the four input dimensions and the original six outputs – we should stress that N_x, N_y are inputs highly correlated with the position of the ball, which confirms former inferences about which input variables are the most relevant for the task. In a similar way, both batch and incremental versions of GMM/GMR were tested with two Gaussian models.

V. RESULTS

Experiments aimed at testing the overall performance of the learning algorithm and at evaluating the effects of reducing the input space through principal components analysis and mutual information. We tested the complete and reduced versions of GMM/GMR at eleven "strategy datasets", corresponding to experiments beginning at each initial position of the ball. The inputs to the prediction stage consisted in a set of F/N values along each trajectory, and the actual output

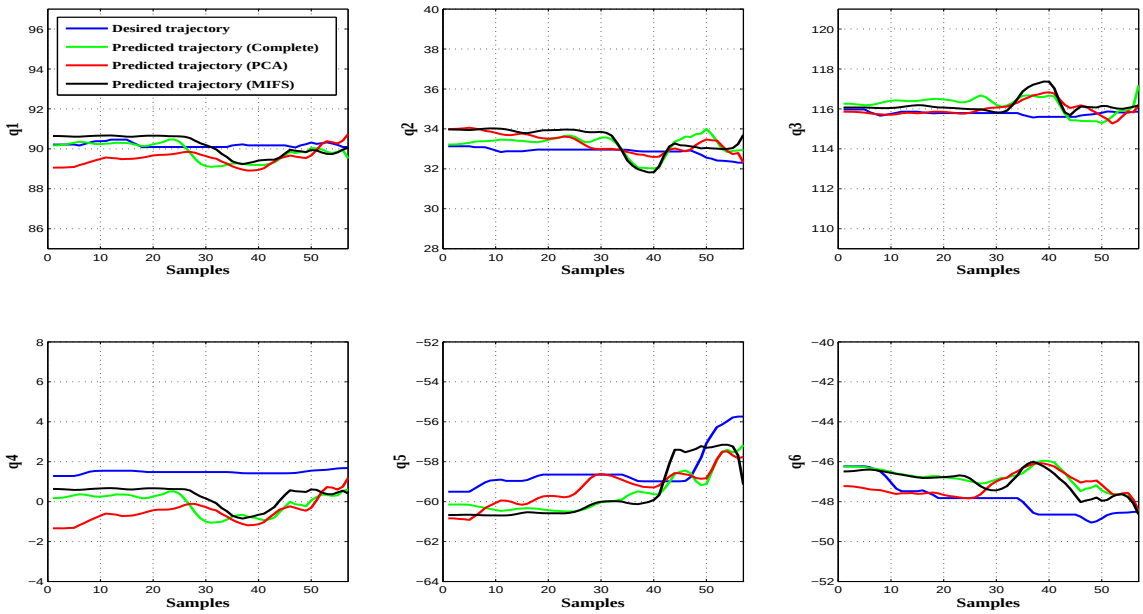

Fig. 5. Predicted and desired trajectories for initial position 3

robot coordinate values were subtracted from the obtained predictions in order to compute the prediction errors. These prediction errors (for each input data point), were used to compute the mean squared error for each output dimension in each experiment. These MSEs constitute a measure of the success in learning a trajectory: the lower they are, the more similar the predicted trajectory is to the demonstrated one. As we wanted to test also whether the learning method was able to generalize creating a set of trajectories for the joints for given inputs, using a training dataset where no predefined strategy exists apparently, we did also carry out similar experiments with further eleven "random datasets".

Prediction times (i.e., time inverted in computing each prediction) were also measured. Considering less variables allows to spare up to one third of prediction time, as shown in Table I

TABLE I

PREDICTION TIMES FOR TESTED ALGORITHMS

Algorithm	Prediction time(s)
GMM/GMR(Complete)	0.032
GMM/GMR(PCA)	0.021
GMM/GMR(MI)	0.026

The computed MSEs, summarized in Figures 6 and 7 allow us to evaluate the following items:

- Incidence of each output dimension of the learned actions on performance.
- Relevance of the original and reduced datasets on training and prediction phases.
- Influence of the starting position of the ball (more or less difficult trajectories to learn).
- Learning a specific motion strategy versus learning random trajectories.

After analyzing these figures, we can afirm the following: in general, the performance of the learning algorithm in terms of prediction errors is quite good. Most average MSEs are below 0.06 for any one of the tested versions (i.e. complete training data, reduced training data through PCA and MIFS) and types of data sets. The predicted values for each robot joint are close to the actual ones, the actions based on these predictions are very similar to those taught by the demonstrator. Moreover, the strategy implicitly proposed by the teacher as well as the random trajectories were learned successfully.

On the other hand, Figure 6 shows that the prediction errors for the three last robot joints are lower than for the first positioning joints, which can be expected from the fact that these variables are the least relevant to achieve the task's goal, as they do not affect the orientation of the container.

As for Figure 7, regarding the effect of applying principal component analysis as a pre-processing stage, it is clear that this process increases considerably the MSE at each robot joint and initial position of the ball in the maze. However, as it is expected the prediction time is lower than when using the original data set (see Table I). On the other hand, in general terms the application of MIFS on data increases slightly the MSE in comparison with those obtained with original data. However, there are some cases where this method leads to lower MSEs, e.g. those for q_2 in Figure 6, initial positions 2, 3 and 6 with "random" dataset in Figure 7.

Interestingly, the overall performance of generalizing the "random dataset", without a predefined set of actions, is better than learning the "strategy dataset", considering the MSEs obtained for each robot joint. A possible explanation may consist in that the teacher possibly developed a *taking the shortest way* strategy –maybe without being aware of it– for leading the ball out of the container, which has been better

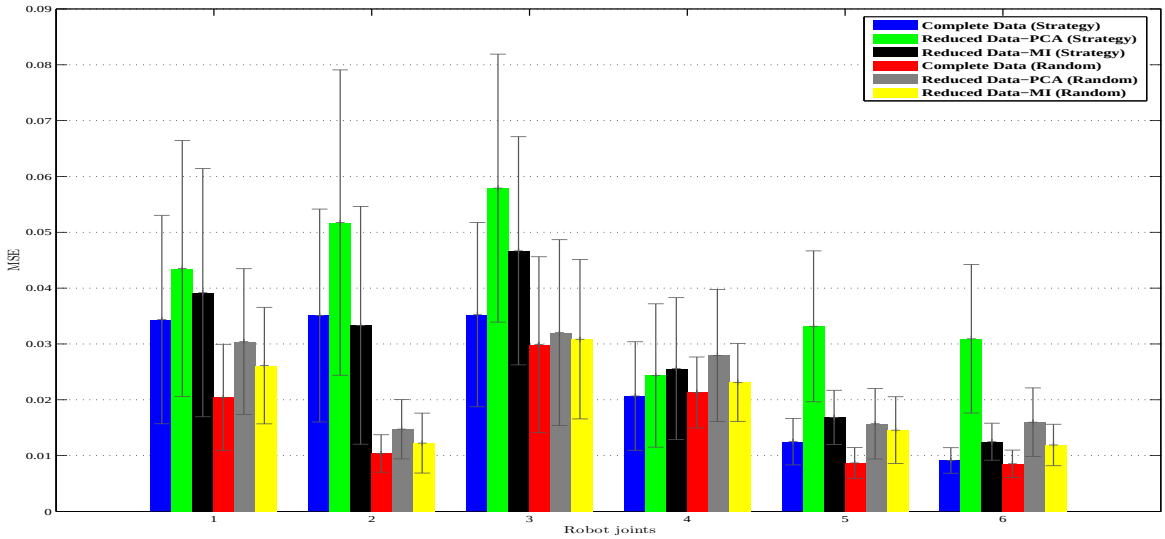

Fig. 6. Average MSEs for each robot's joint with their corresponding standard deviations

learned than that with a predefined set of movements (i.e. the "strategy dataset"). Information which is latent in the user's mind and not directly observable by the robot may be used by the teacher in an undeliberate fashion, being present while demonstrating unknown robot tasks through teleoperation. This includes user preferences as to how a task should be performed or state information observable to the human but not the robot (in our case, the visual input on the position of the ball inside the container, not available to the robot).

Analyzing Figure 5, the learning algorithm learns a better approximation of the "desired trajectory" (as the one carried out by the demonstrator of the task, which it does not imply that this is the only one for achieving the goal) for most robot joints, when it uses the complete data or a reduced dataset through mutual information, than when using reduced data via PCA. It is important to highlight that for q_5 and q_6, the predicted trajectories using complete and MI datasets approximate better the desired movement, as compared to the predictions for the rest of robot joints, which are not so relevant in this task.

VI. CONCLUSIONS AND FUTURE WORK

A. Conclusions

The application described in this paper 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 next subsection) should 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 GMM/GMR techniques. On the other hand, since no programming expertise should be required from the teacher –as the final goal is to develop the tools that allow to teach robots in domestic settings–, we have opted for a PbD 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 specific motion strategy.

It has been shown that F/N feedback constitute valuable input sources for learning manual skills, specially in the absence of visual information. MSEs have been computed as a measure of the discrepancy between real and predicted (as output of the learning process) trajectories, and the low MSE values obtained confirm that GMM/GMR, in its different versions, has been able to learn simple rigid-container emptying skills. Furthermore, as for dimensionality reduction techniques, we have seen that in most cases the price of a slight increment in the MSE values has to be paid for considering less variables. Comparing the results obtained for PCA and MI, the better performance of the second technique probably lies in the fact that here the correlation between input and output drives the selection of variables, whereas PCA prunes both input and output spaces independently by projecting data on a latent space where reduction dimensionality takes place. It should be also noted that MI selects precisely the variables that provide a better discrimination on the position of the ball inside the container.

Despite a few outliers, the different versions of the learning algorithm produced a similar pattern of results regarding both the involved robot joints and the different initial positions. Tests where the teacher was instructed to follow a strategy compared to others where he was not provided useful expertise that permits devising the new research stages, where the taught motion will be refined using reinforcement learning or coaching.

B. Future Work

More involved strategies should arise if obstacles are included inside the container. For example, the walls of a

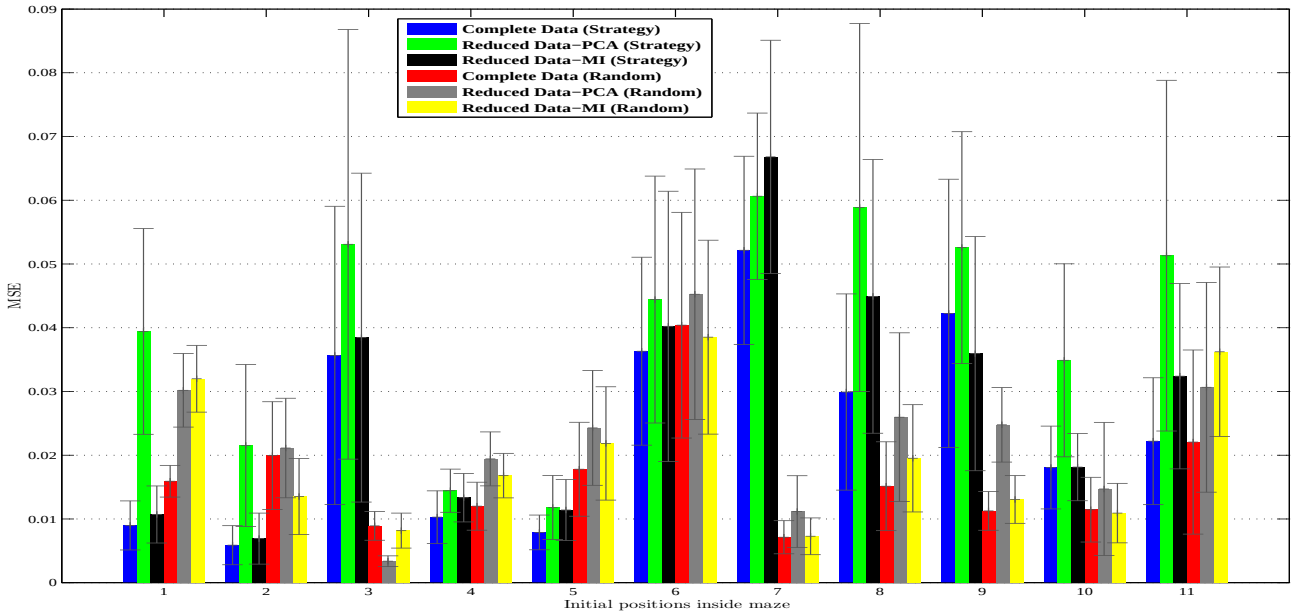

Fig. 7. Average MSEs for each initial positions of the ball in the container with their corresponding standard deviations

maze should favour the success of learning a specific strategy in front of pure random motions. The described experimental setting should be considered as a first step towards taking into account alternative sensorial input sources in learning. Emptying a pill box, e.g., where the weight of the last pills may not be significant enough, as compared to the box, should lead to consider instead finer touch/impact sensors or even sound (together with a sensing directed action as shaking). Moreover, although not considered in the present setting, the use of a haptic display introduces the possibility of enhancing the teaching process by appropriate scaffolds provided by a computer model, that may help the teacher in the execution of his/her own motions. Still another setting that can be regarded as a natural extension of the present work –in that the robot needs to resort to non-visual information– consists in emptying deformable containers like bags. Such flexible containers 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]. Even daily tasks such as opening a door where key-lock interaction forces/torques are of high relevance, may be good target applications for our approach.

REFERENCES

[1] 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*, vol. 37, no. 2, pp. 286–298, 2007.

[2] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Intl. Conf. on Human-Robot Interaction*, 2007, pp. 255–262.

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

[4] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning by demonstration," *Robotics and Autonomous Systems*, no. 57, pp. 469–483, 2009.

[5] 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.

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

[7] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: Object-lifting case study with a humanoid," in *Proc. Intl Conf. on Humanoid Robots*, 2009.

[8] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar, "Learning collaborative manipulation tasks by demonstration using a haptic interface," in *Proc. Intl Conf. on Advanced Robotics*, 2009.

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

[10] L. Rozo, P. Jimenez, and C. Torras, "Learning force-based robot skills from haptic demonstration," in *13th Intl Conf. of the Catalan Association for Artificial Intelligence*, 2010.

[11] 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*, November 2000, pp. 244–249.

[12] 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.

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

[14] A. Dempster, M. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the *EM* algorithm," *Journal of the Royal Statistical Society, Series B*, no. 1, pp. 1–38, 1977.

[15] G. Wells and C. Torras, "Assessing image features for vision-based robot positioning," *Intelligent Robotics Systems*, vol. 30, no. 1, pp. 95–118, 2001.

[16] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. on Neural Networks*, vol. 5, pp. 537–550, 1994.

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