Noname manuscript No.
(will be inserted by the editor)

A Robot Learning from Demonstration Framework to Perform Force-based Manipulation Tasks

Leonel Rozo · Pablo Jiménez · Carme Torras

Received: date / Accepted: date

Abstract This paper proposes an end-to-end learning from demonstration framework for teaching force-based manipulation tasks to robots. The strengths of this work are many-fold: first, we deal with the problem of learning through force perceptions exclusively. Second, we propose to exploit haptic feedback both as a means for improving teacher demonstrations and as a human-robot interaction tool, establishing a bi-directional communication channel between the teacher and the robot, in contrast to works using kinesthetic teaching. Third, we address the well-known *what to imitate?* problem from a different point of view, based on the mutual information between perceptions and actions. Lastly, the teacher's demonstrations are encoded using a Hidden Markov Model, and the robot execution phase is developed by implementing a modified version of Gaussian Mixture Regression that uses implicit temporal information from the probabilistic model, needed when tackling tasks with ambiguous perceptions. Experimental results show that the robot is able to learn and reproduce two different manipulation tasks, with a performance comparable to the teacher's one.

Keywords Programming by demonstration · Imitation learning · Haptic perception · Mutual Information · HMM · GMR · Robotic Manipulation

1 Background and Related Work

One of the main challenges in Robotics is to develop robots that can interact with humans in a natural way, sharing the same dynamic and unstructured environments. Learning from demonstration (**LfD**)[1] is a type of human-robot interaction (**HRI**) whose purpose is to transfer knowledge or skills to the robot. Here, a human user carries out examples of a given task while a robot *observes* these executions and extracts relevant information for learning, representing and reproducing the taught task under unknown conditions [1,2].

HRI requires suitable communication channels for conveying information between a human and a robot [3,4]. In LfD, most existing works rely on vision or on motion sensors as input channels to the robotic system. As for vision-based input, positional information about the objects in the scene is captured with a set of cameras, which are also used to locate and follow markers placed on the teacher's body [5,6]. Most state-of-the-art approaches consider vision as the best choice for extracting information from teacher examples, as human beings do in everyday tasks [7,8]. However, vision-based systems must deal with typical problems as occlusion, appearance changes and complex human-robot kinematics mapping, which can be partly solved by using motion sensors instead. Such type of sensors allows to track the teacher's motion more precisely and to establish a simpler mapping, which make them appropriate to teach tasks to humanoid robots [5,9–11].

In contrast to these works, we are concerned with learning from force-based perceptions. Force conveys relevant information for several tasks where vision or motion sensors can not provide sufficient data to learn a motion or a set of primitives. In many daily tasks, people use force-based perceptions to perform successfully. Examples include assembly processes, opening doors, pulling drawers, cutting slices of bread, etc. Robots

[1] Also known as programming by demonstration or imitation learning.

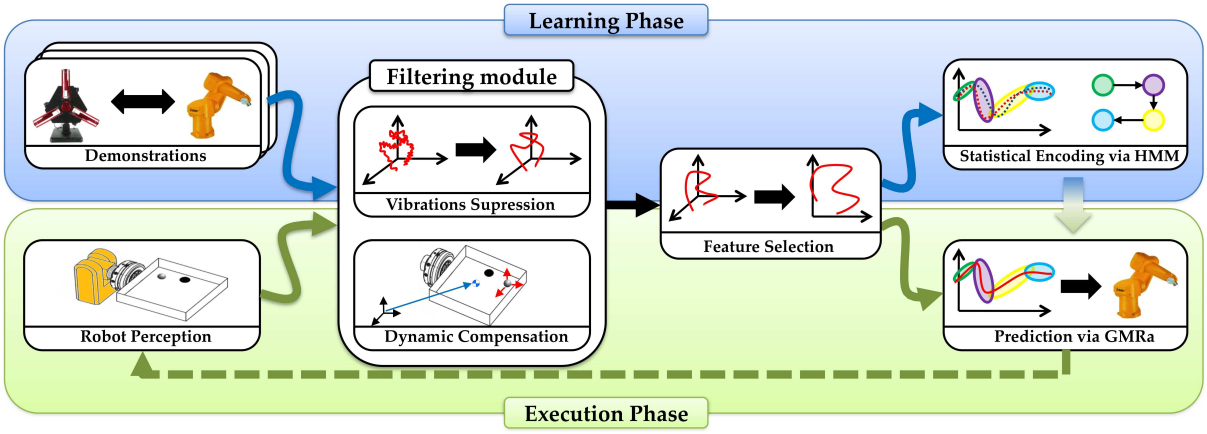


Fig. 1 Entire learning framework. (*Top*) Task learning stage. (*Bottom*) Robot execution stage. The *filtering* module consists of the signal processing to achieve a high fidelity bidirectional communication channel. The *feature selection* block corresponds to the proposed solution for the *What to Imitate?* problem through mutual information analysis.

may also take advantage of force-torque information for learning this kind of tasks. Evrard *et al.* [12] proposed a learning structure similar to ours, where a humanoid robot learns to carry out collaborative manipulation tasks (object-lifting along a vertical axis) using force data. An extension of this research [13], combines LfD and adaptive control for teaching the task, endowing the robot with variable impedance and an adaptive algorithm to generate different reference kinematic profiles depending on the perceived force. Kormushev *et al.* [14] proposed to use a haptic device for defining the force profile of contact-based tasks (ironing and door opening) while the robot follows a previously learned trajectory. This cited work uses kinesthetic guidance and does not exploit haptic feedback as a tool for improving teacher demonstrations, thus avoiding several challenging problems arising when clean and realistic signals are to be displayed to the human during the demonstrations. We contribute a complete force data-based learning framework that includes filtering processes and high-fidelity haptic feedback. This establishes a force-based bidirectional communication channel, which has seldom been exploited as a human-robot interaction tool in LfD in contrast to kinesthetic-based teaching and vision-based systems.

Another point to be addressed in LfD is related to the learning level of the task. Teacher demonstrations may be encoded at a symbolic level, where the task is often represented as state-action pairs in a graph-like structure [15,16], or as motion primitives following hierarchical approaches [17,18]. At trajectory level, on the other hand, the aim is that the robot extracts a generalized trajectory (movement) from slightly different teacher executions [19–21]. Unlike the cited works, our contribution mixes concepts from these two levels as the same goal may be reached from different trajectories or initial states of the task. We propose to encode the demonstrations through a set of states using a Hidden Markov Model (**HMM**) and to execute the skill using a modified version of Gaussian Mixture Regression (**GMRa**) that exploits the temporal coherence of the task at hand.

Regardless of the particular approach, researchers have to deal with three main problems: *what to imitate?*, *how to imitate?* and *when to imitate?* [22]. The first question refers to extracting the most relevant information of the task necessary to learn and reproduce it successfully. The second key question addresses the problem of how the robot can reproduce the task based on the teacher executions. The third problem is related to the timing of the learning phase based on the robot perceptions (observations). In this paper, we propose to solve the first issue through Mutual Information (**MI**) analysis, and to tackle the second problem through an HMM/GMRa-based framework, as shown in Figure 1.

This paper is organized as follows: Section 2 describes our experimental setup and the two manipulation tasks taught to the robot. Section 3 explains how we tackle the *what to imitate?* problem by extracting the most relevant features of the tasks via MI. After, in Sections 4 and 5, the learning and reproduction phases are described respectively, first, illustrating the statistical encoding of the demonstrations by using an HMM and then, showing how the implicit temporal information in HMM is exploited at the execution stage (the entire process is shown in Figure 1). Section 6 shows computational and robot execution results. Finally, the conclusions of this paper and future work are presented in Section 7.

2 Experimental Setups

To test our learning framework based on force perceptions, we constructed an experimental setup to teach a robotic manipulator to carry out two different manipulation tasks using exclusively haptic data. In both scenarios a human user holding the end-effector of a 6-DoF haptic interface (Delta device from Force Dimension) teleoperates a robotic arm (RX60 from Stäubli) which has a force-torque sensor (Shunk FTC-050) placed on its wrist.

Force-based perceptions are feedback to the teacher in order to establish a bidirectional communication channel during the demonstration stage. This implies to work at a minimum frequency of 1000 Hz to have a high fidelity force reflection and a stable teleoperation system, which greatly depend on the executed processes between the position sensing of the haptic device and when the sensed force is reflected on it. Our experimental setup takes such requirement into account and guarantees a high bandwidth communication in the haptic loop.

Two different tasks are proposed to test and analyze the performance of the proposed framework. Their particular features are described below.

2.1 Ball-in-box Task

In this task the robot holds a plastic container with a steel sphere inside it, as shown in Figure 2. At the demonstration phase, the teacher repeatedly carries out the task to be learned, which consists of taking the ball out of the box through the hole, following a specific motion strategy: *Starting at some predefined initial positions, the ball is driven towards the wall adjacent to the hole, and then forced to roll along this wall to the hole* (see Figure 2). During the demonstrations, the teacher feels at the end-effector of the haptic device the force-torque sensed at the robotic wrist. Note that the teacher has an additional information source by watching the scene directly. No visual data are provided to the robot. We like to emphasize that this particular manipulation task has been chosen because it is well-defined and simple enough to permit analyzing each stage of the proposed LfD framework separately and in depth.

2.1.1 Filtering Processes

A first experimental finding derived from the use of haptic feedback in this bidirectional learning framework is the need for filtering. Several people tested the experimental setting, by teleoperating the robotic

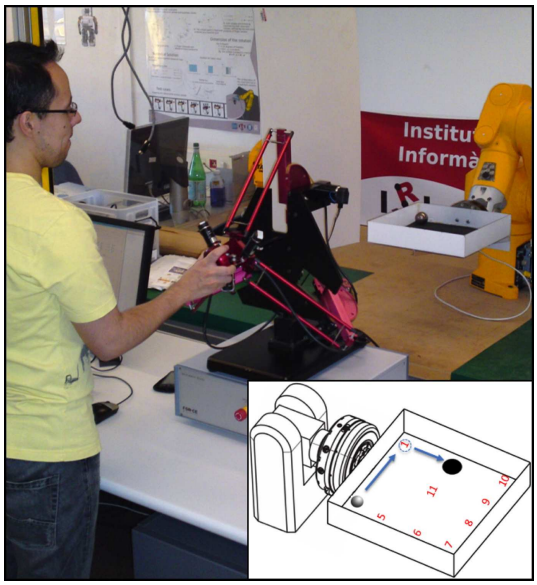


Fig. 2 Experimental scenario of the *ball-in-box* task. (*At the bottom right corner*) Initial positions of the ball for the training phase and motion strategy followed by the teacher. Numbering is counterclockwise.

arm through the haptic interface while receiving force-torque feedback from the sensor mounted on the robotic wrist. Initially, they teleoperated the robot while feeling both the container's mass and the ball's dynamics. Then, they carried out the same task just feeling the ball's dynamics. All the participants argued that the presence of the container's mass was a very distracting factor making the task more difficult to teach. Thus, filtering and dynamic compensation are necessary to obtain better demonstrations and to improve the bidirectional communication channel, as explained below.

Formally, the force-torque signals from the sensor can be expressed as:

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

where $\boldsymbol{F}/\boldsymbol{T}_b$ corresponds to the ball dynamics, $\boldsymbol{F}/\boldsymbol{T}_m$ represents the container mass and ε is the noise (where we include the container vibrations). For achieving a clean and stable communication channel between the human and the robot, it is necessary to display only relevant force-torque signals to the teacher, that is, those corresponding to the dynamics of the ball inside the container. Therefore force-torque produced by noise and the container's mass must be removed before sending force information to the haptic controller.

Since the box is not a perfectly rigid structure, it vibrates as the robot moves. These unwanted vibrations introduce noise in the teleoperation system, leading to unstable behavior. To avoid this, we implemented a low-pass digital filter that cuts out all vibration signals on the force-torque sensor in such a way that the

ε effects in Equation 1 are greatly reduced, in a similar way as done in [23] for suppressing residual vibrations in flexible payloads carried by robot manipulators. We computed the signals' fundamental frequency by subjecting the container – with the ball inside – to vibrations through a force applied perpendicularly to the container's base, at the front edge of it. Then, the frequency spectrum of the generated data was analyzed, from which we obtained the fundamental frequency (7.5Hz) as the cutoff frequency of our low-pass filter. Using $MATLAB^{®}$'s FDAtool, we designed the filter by implementing the *Constrained Least Squares technique* whose order was 75 [24].

2.1.2 Dynamic Compensation

During the demonstration stage, most of the people acting as teachers declared that having to compensate the container's mass while executing the demonstrations makes it considerably harder to focus on the task's goal. Therefore, we model the effects of the container's mass dynamics on the force-torque signals with the aim of removing them and just sending perceptions conveying the ball's motion (similarly as in [25,26]). Considering the system shown in Figure 3, let $\boldsymbol{p}$ denote the position of the center of gravity of the container, $\boldsymbol{\omega}$ its angular velocity, m its mass, $\boldsymbol{I}$ its moment of inertia, $\boldsymbol{F}/\boldsymbol{T}_s$ and $\boldsymbol{F}/\boldsymbol{T}_e$ the sensor and external forces/torques respectively, and $\boldsymbol{r}_s$ and $\boldsymbol{r}_e$ the vectors from the center of gravity of the container to the sensor and external forces frames. Then, using the Newton-Euler equations, we obtain:

$$\sum \boldsymbol{F} = m\ddot{\boldsymbol{p}} = m\boldsymbol{g} + \boldsymbol{F}_e + \boldsymbol{F}_s$$
$$\sum \boldsymbol{T} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{T}_s + \boldsymbol{r}_s \times \boldsymbol{F}_s + \boldsymbol{T}_e + \boldsymbol{r}_e \times \boldsymbol{F}_e$$

If we assume very low linear and angular accelerations as well as very low angular velocities, we obtain the following approximation:

$$\boldsymbol{F}_s = -m\boldsymbol{g} - \boldsymbol{F}_e \quad (2)$$
$$\boldsymbol{T}_s + \boldsymbol{r}_s \times \boldsymbol{F}_s = -\boldsymbol{T}_e - \boldsymbol{r}_e \times \boldsymbol{F}_e \quad (3)$$

Determining the force and torque values generated by the container's mass via the sensor measurements without the ball inside, and using the former equations, it is possible to remove these undesirable force-torque signals from sensor readings and to return just the ball dynamics effects. The users agree that the felt force-torque values at the end-effector after compensation are realistic enough as to provide a clear understanding of how their actions translate into motions of the ball.

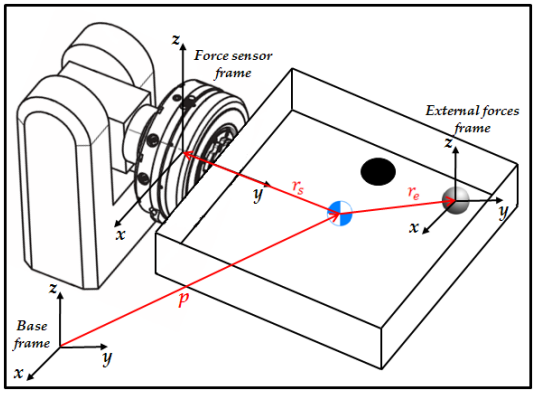


Fig. 3 Dynamic compensation model for removing container's mass effects from the force-torque perceptions sent to the haptic controller.

2.2 Pouring task

The second task consists of pouring drinks. Here, the robotic arm holds a 1 liter plastic bottle full of tiny metallic spheres, which play the role of a fluid (this solution was adopted to avoid spilling liquid during tests). The teacher teleoperates the robot in order to demonstrate how to pour 100 ml drinks into a plastic glass, as shown in Figure 4. Every sample of the task starts from an unique predefined initial pose of the bottle, which is also the stop configuration once the robot has poured a drink. Initially, the bottle is completely full, and the teacher carries out several demonstrations until the bottle is empty. Thus, the initial force-torque values for each example vary according to how much "fluid" has been poured previously. It is worth to highlight that such changes in the input variables at the beginning of the demonstrations are similar to those observed in the *ball-in-box* task for each initial position of the sphere inside the container.

Again, it was necessary to implement a smoothing filter to reduce the noise from the sensor readings, this time generated by the tiny metallic spheres. On the other hand, the dynamic compensation model presented previously was also used here for removing the bottle mass effects from the sensor readings, in order to feedback the teacher with only the external forces-torques generated by the "fluid" at the demonstration phase. Note that in this task, the teacher is also able to watch the scene directly, thus he/she can know the location of the glass in the robot workspace. Such information is not provided to the robot during the execution phase because the glass position is predefined in advance and fixed across the examples.[2]

[2] Note that a camera system may also be used to know the location of the glass in the robot frame, so that the demonstrations would also be dependent on this parameter.

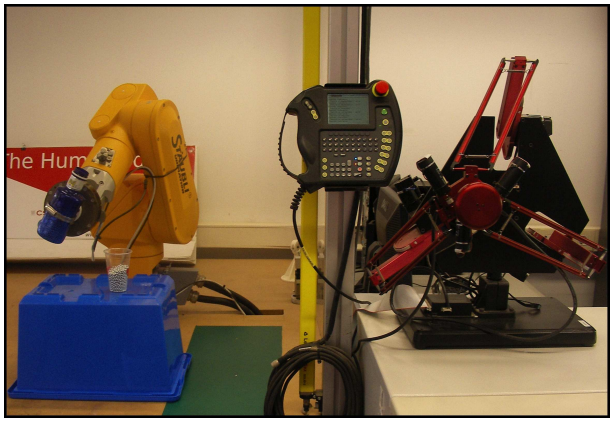


Fig. 4 Experimental scenario of the *pouring task*. The teacher demonstrates the robot how to pour 100 ml drinks into a plastic glass by teleoperation.

Tamosiunaite *et al.* [27] tackled the same task using reinforcement learning, which was applied to improve the initial encoding obtained from human demonstrations modeled through dynamic motion primitives. Thus, this task allows to show that the proposed framework can be used for learning more realistic force-based skills like the ones that an inexperienced human wish to teach to a home service robot [28].

3 Feature Selection through Mutual Information

The *what to Imitate?* problem means to determine which information of the demonstrations is relevant for learning the task at hand successfully [29]. Most works tackle this problem by analyzing the variability across demonstrations of the task at trajectory level. Those parts with large variances do not have to be learned precisely, whereas low variance suggests that the corresponding motion segment is significant and deserves to be learned [14,30]. This approach exploits variance for constructing task constraints [31] as well as for determining secure interaction zones in a robot coaching framework [32]. However, the cited works do not focus on the relative relevance of each individual input dimension for the task to be learned. But irrelevant or redundant information may actually be present across input dimensions, which can increase the computational cost of the learning stage and make the task harder to learn. The point is to select the most relevant subset of input variables. The benefits in computational cost and noise reduction during the learning stage do outperform a hypothetical and marginal loss of information. Furthermore, this approach is compatible with the previously described variance-based analysis criterion.

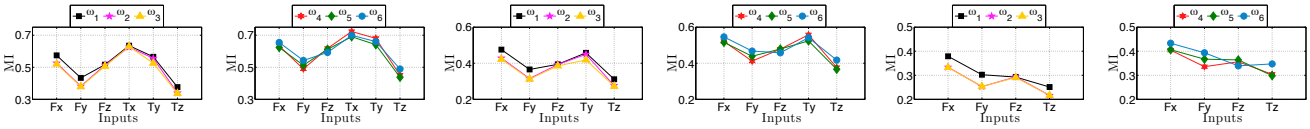
In literature, two types of approaches tackle the problem of selecting a subset of variables from the original data. *Feature selection* methods keep only useful features and discard others, while *feature transform* techniques construct new variables out of the original ones [33]. In LfD, feature selection may be preferable to transforms because it could be essential to retain some of the original features. For instance, in active learning, the robot may let the teacher know which perceptions it has selected, in order to get feedback about how well or how convenient its selection was according to the human knowledge of the task. Such human assistance will not be available if the robot carries out the selection from input transforms. This fact may occur in [30], where the authors propose to project the human samples onto a latent space obtained from a principal component analysis to diminish redundancies, where the transformed variables do not have a clear interpretation for a human teacher anymore. Also, this analysis is applied to the input variables of the problem, without taking into account how these influence the output variables. In addition, when the number of irrelevant perceptions exceeds the number of relevant inputs by orders of magnitude, learning a transform reliably may require excessive amounts of training data.

Since our framework may be used as the basis of LfD structures, it is more generic and suitable if it may provide clear information about what the robot considers that should be imitated. Thus, feature selection methods are preferred in this context. Here we use the Mutual Information (**MI**) criterion, which allows to establish which input variables give more information with respect to their effects on the outputs (i.e., how force-torque perceptions affect the teacher actions). In contrast to other techniques (e.g., correlation criterion), MI detects non-linear dependencies between inputs and outputs [34]. The purpose of this method in feature selection [35] is the reduction of the output data uncertainty provided by each input variable. In our context, depending on how the uncertainty of the output data is reduced, a robot perception gives more or less information about the desired actions. Note that this approach has shown satisfactory results in sensor fusion [36] and vision-based positioning of a robotic arm [37].

In order to apply MI analysis to our resulting training data (after filtering and dynamic compensation), let us define the MI value between two continuous variables $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows (more details in [38]):[3]

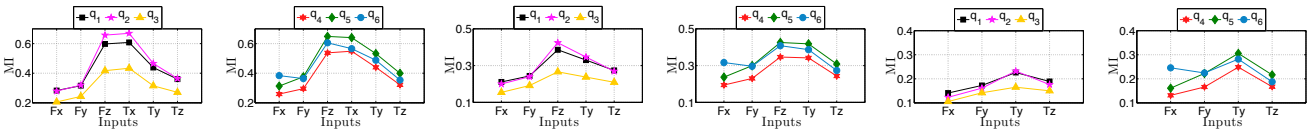
$$I(\boldsymbol{x};\boldsymbol{y}) = \int_x \int_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{4}$$

[3] The basic division and product rules of *log* can be applied for numerical stability.



(a) MI for all the input-output pairs. (b) Conditional MI given T_x. (c) Conditional MI given T_x and T_y.

Fig. 5 MI values at each variable selection phase for the *ball-in-box task*.



(a) MI for all the input-output pairs. (b) Conditional MI given T_x. (c) Conditional MI given T_x and F_z.

Fig. 6 MI values at each variable selection phase for the *pouring task*.

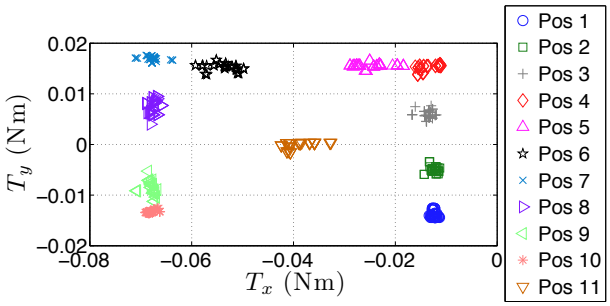


Fig. 7 Torques map representing clusters for each initial position of the ball inside the container. Plotting the first samples of the variables most relevant to the current task, T_x vs. T_y, it is observed they do describe where the ball is in the box.

3.1 Ball-in-box Task

Let us define the inputs of this manipulation task as the wrench $\boldsymbol{\vartheta} = \{F_x,\ F_y,\ F_z,\ T_x,\ T_y,\ T_z\}$, i.e., the sensed forces and torques in the robot's frame, and the outputs as the joint velocities of the robot defined by $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_{N_q}\}$, where N_q is the number of joints of the robot. Using equation 4, the MI value is computed for each input-output pair using entire data streams obtained at the demonstration phase. Both the marginal and joint probabilities are approximated using histogram-based densities, which are computed from discrete partitions of the dataspace.[4] The quantization error in the conversion from continuous variables to discrete ones is bounded by some constant value which depends only on the number of partitions that divide the continuous space [39].

Figure 5(a) shows the different MI values for all the input-output pairs in the task. In general terms, the input variables F_y and T_z show less relevance whereas T_x and T_y are the most correlated variables with the outputs. This 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 7). These results confirm what we intuitively expected about which input variables were the most relevant for this task. Note that this technique can be also applied in more complex problems where the important perceptions can not be easily detected. Nonetheless, once MI values have been computed, the problem is to select a subset $\boldsymbol{\Omega}$ of k perceptions from the original set $\boldsymbol{\Psi}$ of n inputs, that is "maximally informative" about the robot actions. In this context, the computed MI values provide a ranking that can be used for selecting *the most relevant* input. However, to choose the remaining $k-1$ perceptions, the redundancy among inputs must be taken into consideration. To achieve this, we resort to a *greedy selection* algorithm known as "mutual information-based feature selection deduced from uniform distributions" (**MIFS-U**) [39], which was adapted to our learning framework as described in Algorithm 1. The core of this technique is to select the rest of variables by maximizing $I(\boldsymbol{y}; \boldsymbol{x}_i|\boldsymbol{x}_s)$, this means to choose the input $\boldsymbol{x}_i$ that provides most information about $\boldsymbol{y}$ given $\boldsymbol{x}_s$. In this approach, the computation of the conditional mutual information is approximated as follows:

$$I(\boldsymbol{y}; \boldsymbol{x}_i|\boldsymbol{x}_s) = I(\boldsymbol{y}; \boldsymbol{x}_i) - \frac{I(\boldsymbol{y}; \boldsymbol{x}_s)}{H(\boldsymbol{x}_s)} I(\boldsymbol{x}_s; \boldsymbol{x}_i), \tag{5}$$

where $H(\boldsymbol{x}_s)$ represents the entropy of $\boldsymbol{x}_s$ (details in [39]). Note that the algorithm can be extended to a multidimensional output case assuming a set of N_m input variables $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_m}\}$ and output dataspace $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_n}\}$ of dimensionality N_n.

For this task, we set $k = 3$ and carried out the MIFS-U to choose the perceptions to be used in next stages of the learning framework. The selected variables were the subset $\boldsymbol{\Omega} = \{T_x,\ T_y,\ F_x\}$, as shown in Figures 5(b) and 5(c). It should be noted that the MI val-

[4] Other type of non-parametric density may also be used, such as Parzen windows.

Algorithm 1 MIFS-U

1: **Initialization:** Set $\boldsymbol{\Omega} \leftarrow \{\}$, and $\boldsymbol{\Psi} \leftarrow \boldsymbol{X}$
2: **Compute MI:** Obtain $I(\boldsymbol{y}_j, \boldsymbol{x}_i), \forall \boldsymbol{x}_i \in \boldsymbol{X}$, and $\forall \boldsymbol{y}_j \in \boldsymbol{Y}$
3: **Mean MI:** $I(\boldsymbol{Y}, \boldsymbol{x}_i) = \frac{1}{N_n} \sum_{j=1}^{N_n} I(\boldsymbol{y}_j, \boldsymbol{x}_i)$, $\forall \boldsymbol{x}_i \in \boldsymbol{X}$
4: **Select the most relevant input:** Find the input $\boldsymbol{x}_s = \arg\max_{\boldsymbol{x}_i \in \boldsymbol{X}} I(\boldsymbol{Y}, \boldsymbol{x}_i)$, and set $\boldsymbol{\Omega} \leftarrow \{\boldsymbol{x}_s\}$, $\boldsymbol{\Psi} \setminus \{\boldsymbol{x}_s\}$
5: **Greedy selection:**
for $t = 1 \rightarrow k-1$ **do**
  5.1: Compute the conditional MI $I(\boldsymbol{Y}, \boldsymbol{x}_i | \boldsymbol{x}_s)$, $\forall \boldsymbol{x}_i \in \boldsymbol{X}$
  5.2: Find $\boldsymbol{x}_s = \arg\max_{\boldsymbol{x}_i \in \boldsymbol{X}} I(\boldsymbol{Y}, \boldsymbol{x}_i | \boldsymbol{x}_s)$, and set $\boldsymbol{\Omega} \leftarrow \{\boldsymbol{x}_s\}$, $\boldsymbol{\Psi} \setminus \{\boldsymbol{x}_s\}$
end for
6: Output the set $\boldsymbol{\Omega}$

ues for F_x and F_z are very similar for most outputs initially (see Figure 5(a)), however, MIFS-U automatically chooses F_x and discards F_z. This is in accordance to intuition, since F_z is the force along the vertical axis in the robot frame, which represents the gravitational force of the ball. Such force generates the torques about the axes x and y, thus there is a high correlation between F_z and $\{T_x,\ T_y\}$.

3.2 Pouring Task

In this task the input variables also are the wrench $\boldsymbol{\vartheta}$, but the output variables are the joint robot position defined by $\boldsymbol{q} = \{q_1, \ldots, q_{N_q}\}$ at instant $t+1$ for the given $\boldsymbol{\vartheta}$ at t. Such change of output variables for this task is aimed at showing the generic significance of the proposed learning framework for different representations of the task state. Again, the MI value was computed for all the input-output pairs (as shown in Figure 6(a)) in order to select the most important input variable, T_x in this case. Note that T_x and F_z display nearly the same MI value for all the robot joints. This is an expected result because F_z is the vertical force in the robot frame representing the gravitational component of the load (i.e., the bottle and fluid masses), while T_x is approximately the torque generated by such a load. This means that both variables are providing similar information with respect to the robot movements, because they significantly vary as the fluid comes out of the bottle.

Subsequently, Algorithm 1 was applied to select the remaining $k-1$ variables (with $k=3$), from which the resulting "most informative" set of inputs was $\boldsymbol{\Omega} = \{T_x,\ F_z,\ T_y\}$. The selection process and the values of the conditional mutual information are shown in Figures 6(b) and 6(c). There is an interesting aspect to highlight from this result: the third selected variable T_y shows slight variations when the robot rotates the bottle to pour a drink, which are likely produced by the location change of the center of mass of the load due to the "fluid dynamics" into the bottle. Note that such dynamics may be hardly modeled as reported in [27], but the algorithm was able to detect that T_y was non-linearly correlated to the robot motion encapsulating part of the fluid dynamics (also confirmed after a detailed analysis of the data streams). In sum, MI is shown to be a good and advisable tool for extracting the perceptions that are relevant in a LfD framework.

An interesting aspect to highlight is that MIFS-U often gives more preference to redundant variables over irrelevant ones during the selection process, as also noted in [40]. In this case the variables T_x and F_z provide nearly the same information about the task, but both are chosen even being redundant, because their relevance with respect to the outputs keeps high despite one of them has been selected previously. We consider that such behavior is not a drawback in the LfD context, because our aim is to extract the relevant perceptions of the task (even if several of them provide similar information). Nonetheless, this fact opens an attractive issue of research to be tackled in near future works.

3.3 Automatic selection of input variables

As shown previously, the number of inputs to be selected was predefined in advance, however it would be desirable to have a measure to decide on the optimal number of components selected by MIFS-U. In this direction, let us define a new variable ζ^t that computes the ratio at iteration t between the information a candidate input variable $\boldsymbol{x}_c^t$ provides and the one already given by the current subset of selected perceptions $\boldsymbol{\Omega}^t$ as follows,

$$\zeta^t = \frac{I(\boldsymbol{Y}, \boldsymbol{x}_c^t | \boldsymbol{x}_s)}{I(\boldsymbol{Y}, \boldsymbol{\Omega}^t)}, \tag{6}$$

where $\boldsymbol{x}_c^t = \arg\max_{\boldsymbol{x}_i \in \boldsymbol{X}} I(\boldsymbol{Y}, \boldsymbol{x}_i | \boldsymbol{x}_s)$ and $I(\boldsymbol{Y}, \boldsymbol{\Omega}^t) = \sum_{j=1}^{t} I(\boldsymbol{Y}, \boldsymbol{x}_s^j | \boldsymbol{x}_s^{j-1})$. Such *conditional mutual information ratio* shows how much information the next input to be selected provides taking into consideration the accumulated conditional MI given by the current selected variables. In this sense, it is desired that ζ_t is greater than a predefined threshold $0 < \phi \leq 1$, which controls what is the minimum information ratio that allows to select one more input variable (i.e., the minimum mutual information that a variable should provide). It is worth mentioning that this new selection criterion would modify step 5 in Algorithm 1, where the *greedy* selection is now controlled by ζ^t, which is evaluated at each iteration before selecting the next input. Thus, the algorithm keeps selecting variables while

the condition $\zeta^t \geq \phi$ is satisfied. Note that the higher ϕ, the more selective the algorithm.

We again subjected the training data of both manipulation tasks to MI analysis, but this time using the modified version of Algorithm 1 with $\phi = 0.3$. Regarding the ball-in-box task, the resulting subset of selected inputs was again $\boldsymbol{\Omega} = \{T_x,\ T_y,\ F_x\}$, supporting the analysis explained in Section 3.1. In contrast, for the pouring task, the resulting selected perceptions were $\boldsymbol{\Omega} = \{T_x,\ F_z\}$. This tells us that T_y might not provide enough information about the robot actions when T_x and F_z have been already selected. We will assess the framework performance using these last subsets of input variables in Section 6.

4 Learning the Task

Previous research in LfD [41], has proposed to use Gaussian Mixture Models (**GMM**) for encoding manipulation tasks. However, this algorithm does not extract temporal information from data, and time must explicitly be considered as an input variable if required by the type of task to be learned (as in Calinon *et al.* [30]). Force-torque signals tend to show very large time discrepancies, which may be tackled using techniques as dynamic time warping, but increasing the complexity of the learning framework. To avoid including this explicit temporal dependency in the model we resort to HMM, which treats and exploits the sequential patterns in the data and it is therefore more appropriate to encode the features of our tasks without using time as an additional input variable, which would significantly constrain the generalization capabilities. HMM can be interpreted as an extension of GMM in which the choice of the mixture component for each observation depends also on the choice of the component for the previous observation. This technique has been widely used in several computer science areas as speech recognition [42], human motion patterns encoding [43] and LfD applications [32,44], among others.

Most of LfD works using HMM address the problem by learning trajectories from human demonstrations [12] or by encoding a task with predefined states as in assembly processes that can be represented at a symbolic level [45]. However, our problem differs from these and other works in the following points:

- The task goal may be achieved by executing different trajectories depending on the initial conditions of the task (e.g., initial position of the ball inside the container for the *ball-in-box* task or the fluid quantity inside the bottle for the *pouring* skill).
- We do not use time as an additional input variable.

Formally, given our experimental setting described in Section 2 and following the notation of [46], let us denote a training datapoint as $\boldsymbol{d}_p^m \in \Re^D$, with $m = 1, 2, \ldots, M$ and $p = 1, 2, \ldots, P$, where M is the number of demonstrations, P is the number of datapoints collected along demonstration m, and D is the total number of input and output variables. We used an N-states ergodic HMM defined as $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$ where:

- $\boldsymbol{A} = \{a_{ij}\}$ is the state transition probability matrix, with $1 \leq i, j \leq N$.
- $\boldsymbol{B} = \{b_j(k)\}$ is the observation symbol probability matrix, with $1 \leq k \leq (M * P)$ and assuming continuous observation densities defined as normal distributions $\mathcal{N}(O; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$.
- $\boldsymbol{\pi} = \{\pi_i\}$ is the initial state probability vector, with $1 \leq i \leq N$.
- N is the number of model states.

The main idea is to adjust the model to maximize $P(\boldsymbol{O}|\lambda)$, where $\boldsymbol{O} = \{O_1,\ O_2,\ \ldots,\ O_T\}$ is an observation sequence with each O_t corresponding to a training datapoint $\mathbf{d}_p^m$. The *Baum-Welch* method is used to achieve such an objective (more details in [46]). In order to describe the procedure for re-estimation of HMM parameters, it is necessary to define the following variables:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \tag{7}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{8}$$

where α and β are called *forward* and *backward* variables, respectively, and are defined as:

$$\alpha_1(i) = \pi_i b_i(O_1)$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})$$

$$\beta_T(i) = 1$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

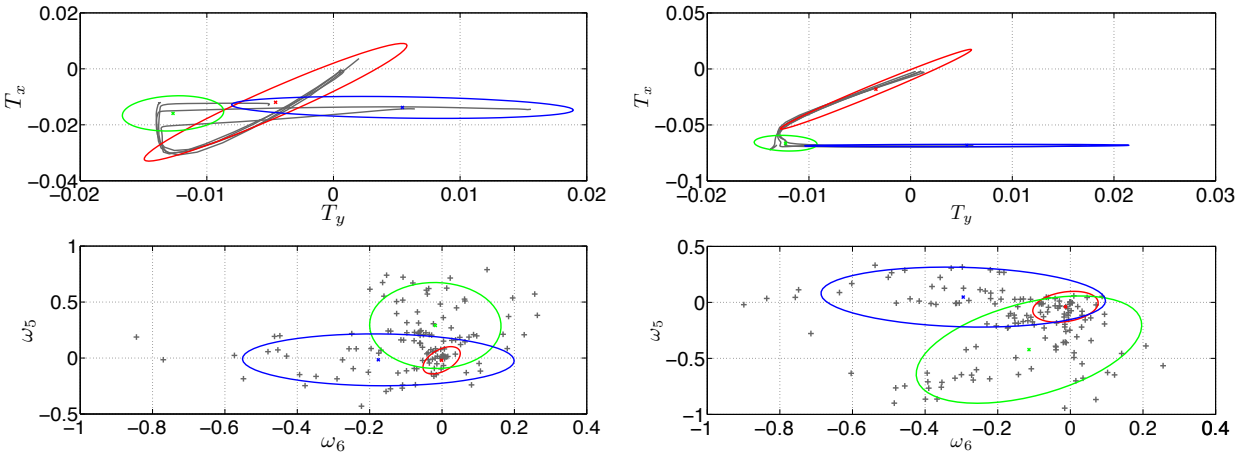
From equations 7 and 8, the HMM parameters are iteratively estimated as follows:

$$\overline{\pi}_i = \gamma_1(i)$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\overline{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) O_t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

$$\overline{\sum}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(O_t - \mu_{jk})(O_t - \mu_{jk})^\top}{\sum_{t=1}^{T} \gamma_t(j,k)}$$



(a) 3-states HMM trained with positions {1,2,3,4}

(b) 3-states HMM trained with positions {7,8,9,10}

Fig. 8 Resulting HMM for two different training datasets of the *ball-in-box* task. *Top*: Input space composed of the most relevant inputs $\{T_x, T_y\}$. *Bottom*: Output space composed of robot joint velocities playing the most important role for the given task. For both cases, the hidden *left-to-right* structure is obtained after convergence (having an ergodic HMM at the beginning).

These equations permit obtaining a suitable trained HMM that represents the teacher demonstrations statistically through a states model capturing the robot motion for given force-based perceptions and taking temporal coherence into account from the resulting matrix $\boldsymbol{A}$.

4.1 Ball-in-box Task

For encoding this task, inputs are the force-torque sensed at the robotic wrist and outputs are the velocity commands ω_l at each robot joint q_l with $l = 1, \ldots, N_q$. Note that joint velocities were chosen as outputs because they do represent the robot actions to be performed according to the force-torque perceptions. As explained in Section 3, we found the subset $\boldsymbol{\Omega}$ of selected inputs as those needed to learn the task successfully, because they showed to contain the most relevant information about the task outputs. Thus, each training datapoint is defined as $\boldsymbol{d}_p^m = \{T_x,\ T_y,\ F_x,\ \omega_1,\ \ldots,\ \omega_{N_q}\}$.

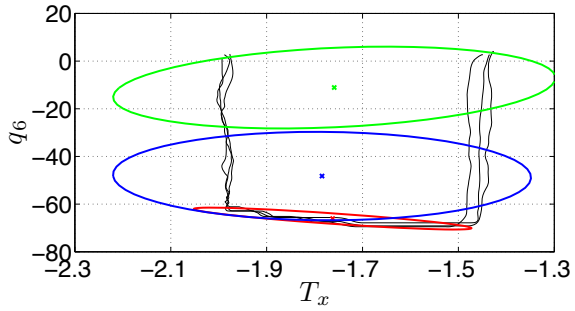
In other words, λ is encoding the joint distribution $P(\boldsymbol{\Omega}, \boldsymbol{\omega})$. To understand better this idea, Figure 8 shows the HMM convergence for two different datasets: Figure 8(a) displays a 3-states HMM trained with similar demonstrations starting from positions {1,2,3,4}, while Figure 8(b) shows another 3-states model trained with samples starting from positions {7,8,9,10}. Observe how the hidden *left-to-right* structure is obtained after convergence (having an ergodic HMM at the beginning), which is the appropriate topology for learning these datasets separately. For both cases, the resulting vector $\boldsymbol{\pi}$ gives as initial state the blue Gaussian, that corresponds to the first movement carried out by the teacher (i.e., when the user orients the robot in such a way that the ball rolls towards the wall adjacent to the hole). In Figure 8(a), blue and red states intersect each other in input space, covering the same segments of trajectories. In this case, the temporal information is essential to determine what velocity command has to be provided, which is not clear using a GMM-based approach.

4.2 Pouring Task

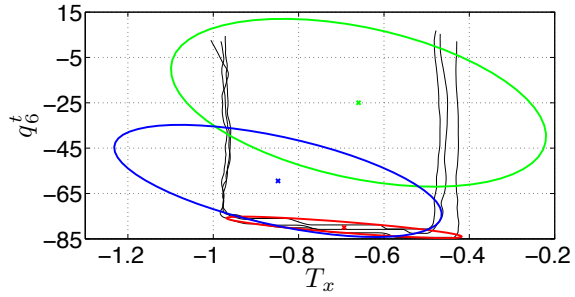
In order to show the flexibility of the proposed framework, we use a different representation of the task state for learning the pouring skill. Specifically, inputs are the selected subset of variables $\boldsymbol{\Omega}$ previously obtained in Section 3 and the current joint positions $\boldsymbol{q}$ at time step t.[5] Outputs are the desired robot state to be achieved at $t+1$. Thus, each training datapoint is defined as $\boldsymbol{d}_p^m = \{T_x,\ F_z,\ q_1^t,\ \ldots,\ q_{N_q}^t,\ q_1^{t+1},\ \ldots,\ q_{N_q}^{t+1}\}$. Then, in this case the model is encoding the joint distribution $P(\boldsymbol{\Omega}, \boldsymbol{q}^t, \boldsymbol{q}^{t+1})$.

Figure 9 shows two single HMM encoding different sets of demonstrations of the *pouring* task. The displayed projection of the models corresponds to the most relevant input T_x and the robot joint q_6 that is rotated to pour the drinks. Note that there is one state encapsulating the start and end of the skill at the same time

[5] It should be noted that $\boldsymbol{q}^t$ was not considered in the MI-based analysis because it is known that $\boldsymbol{q}^{t+1}$ is highly correlated to its values at time step t because the dynamics of the task.



(a) 3-states HMM encoding the 2^{nd} set of pouring samples



(b) 3-states HMM encoding the 4^{th} set of pouring samples

Fig. 9 Resulting HMMs for two different set of demonstrations of the *pouring* task.

(i.e., the green Gaussian), while the red ellipse is encoding when the fluid comes out from the bottle while the robot slightly rotates q_6. The complete model of this skill and the reproduction results are shown and analyzed later on.

5 Task Reproduction

Since the tasks are neither strictly learned as a sequence of discrete actions nor as simple trajectories, it is necessary to find a suitable way to reconstruct the output commands, given a perception and the resulting trained HMM. To achieve this goal, a modified version of GMR (here named GMRa) is used for computing the robot actions to be sent to the controller as the desired robot state to be achieved, as described next.

Recent works [11,41] proposed to use GMM/GMR for learning tasks at trajectory level, where the main idea is to model data from a mixture of Gaussians and to compute predictions for a given set of queries through regression by applying the original version of GMR. In this approach, standard GMR averages the different observations, even if they have been observed at different parts of the skill. Formally, for each Gaussian component i, both input $\boldsymbol{x}$ and output $\boldsymbol{y}$ are separated by expressing the mean and covariance matrix as:

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\boldsymbol{x}} \\ \boldsymbol{\mu}_i^{\boldsymbol{y}} \end{bmatrix}, \; \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\boldsymbol{xx}} & \boldsymbol{\Sigma}_i^{\boldsymbol{xy}} \\ \boldsymbol{\Sigma}_i^{\boldsymbol{yx}} & \boldsymbol{\Sigma}_i^{\boldsymbol{yy}} \end{bmatrix}$$

Then, the conditional expectation $\hat{\boldsymbol{y}}$ given the input vector $\boldsymbol{x}$, for a mixture of N Gaussians is:

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{N} \beta_i \left[\boldsymbol{\mu}_i^{\boldsymbol{y}} + \boldsymbol{\Sigma}_i^{\boldsymbol{yx}} (\boldsymbol{\Sigma}_i^{\boldsymbol{xx}})^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i^{\boldsymbol{x}}) \right] \tag{9}$$

where $\beta_i = \frac{p(i)p(\boldsymbol{x}|i)}{\sum_{j=1}^{N} p(j)p(\boldsymbol{x}|j)}$ is a weight exclusively based on the input variables (mainly force-torque data in our tasks).

We aim at predicting the necessary robot commands as a function of its force-based perceptions in order to follow the taught strategy for each task as close as possible. We adopt the approach proposed by Calinon *et al.* in [47], where the robot's actions are computed from a modified version of the well-known regression technique GMR (which we name GMRa). This version computes the predictions from a mixture of Gaussians (e.g., the HMM states) taking the encapsulated temporal information by the HMM (i.e., the variable α) into account along with the given inputs (i.e., the robot perceptions). In this way, our learning framework is able to handle perceptual aliases, this means that the robot may be able to carry out the correct action if more than one output exist for the same perception pattern, by taking advantage of the sequential information of the task. This makes the proposed structure more generic and versatile, thus being useful for a wider set of manipulation skills.

In GMRa, the weights are estimated using the actual values of the inputs, and also implicitly their previous values, through the transition probabilities related to the forward variable α. Formally, the definition of the new GMRa based on temporal information is given by:

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{N} \alpha(i) \left[\boldsymbol{\mu}_i^{\boldsymbol{y}} + \boldsymbol{\Sigma}_i^{\boldsymbol{yx}} (\boldsymbol{\Sigma}_i^{\boldsymbol{xx}})^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i^{\boldsymbol{x}}) \right] \tag{10}$$

where $\alpha(i)$ is the forward variable for the i-th Gaussian in the HMM. This variable expresses the probability of observing the partial sequence, $\boldsymbol{O} = \{O_1, O_2, \ldots, O_t\}$ and of being in state S_i at time t. Now, for a given force-torque perception, the predicted command is based on current and past observations, which makes sense for those tasks where more than one output exists for a given input pattern.

Note that Lee and Ott's work [32] proposes a similar framework that encodes the demonstrations through

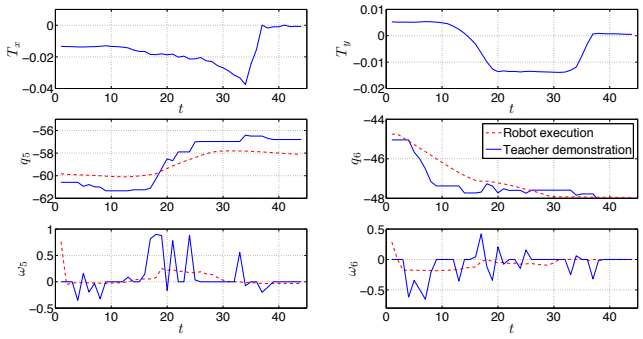


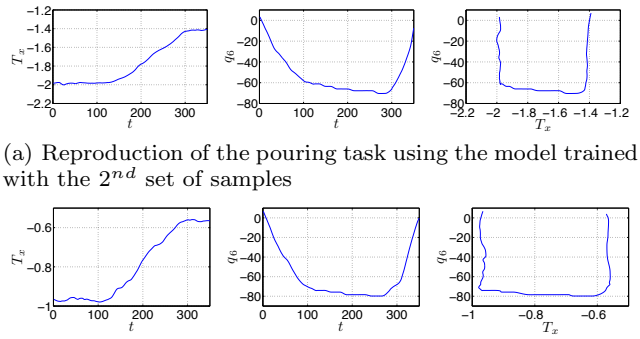
Fig. 10 *Top*: Inputs pattern for T_x and T_y when the ball starts at position number 3. *Middle*: Trajectories of joints q_5 and q_6 corresponding to both teacher demonstration and robot's execution obtained from the velocity profiles ω_5 and ω_6 shown at *bottom*.

an HMM and retrieves the robot actions using a time-driven version of the classical Gaussian Mixture Regression (**GMR**). In contrast to the *forward variable*-based weights, the weighting mechanism used by GMR exclusively depends on time, and neither previous observations nor sequential information are taken into account in this approach. Such approach might show an unsatisfactory performance when force data present large time discrepancies, because the explicit use of time at the reproduction phase.

5.1 Ball-in-box Task

Figure 13 shows the robot joint trajectories and velocities obtained while the teacher demonstrates how to take the ball out of the box, when starting at position 7. The trajectories and velocity profiles of the robot in the execution phase are also displayed. These predictions have been computed via GMRa for the inputs displayed in the first row of the figure and using the HMM displayed in Figure 8(b). It can be observed that the learning framework is able to compute the correct velocity commands to follow the teacher's strategy as well as to accomplish the task's goal. In addition, every joint trajectory is very similar to the *desired* one, even for those robot joints that are not playing a relevant role in the task (e.g., q_1 or q_3).

The predictions in Figure 10 were obtained from the HMM shown in Figure 8(a). The most relevant feature of this example is how the learning framework performs successfully even when the input data lie simultaneously on two HMM states. Figure 8(a) shows two overlapping states, where GMR may likely retrieve a wrong velocity command if a given input datapoint lies in this zone. Instead, GMRa performs correctly as it takes not only the given perception into account, but also the sequence of states through α.



(a) Reproduction of the pouring task using the model trained with the 2^{nd} set of samples

(b) Reproduction of the pouring task using the model trained with the 4^{th} set of samples

Fig. 11 *Left*: Torque pattern around the axis x during the reproduction. *Middle*: Trajectory of the robot joint q_6 that rotates the bottle to pour the drinks. *Right*: T_x vs. q_6 plot showing a reproduction pattern quite similar to the ones observed in the demonstrations set.

5.2 Pouring Task

Figures 11(a) and 11(b) show the obtained reproductions using the models previously displayed in Figures 9(a) and 9(b), respectively. For both executions, the initial force-torque perception was slightly different from the ones sensed during the demonstration phase, for which the robot performed successfully, as evidenced from the trajectories followed by q_6. Note that such trajectories show that the robot comes back to the starting configuration after having poured a drink, as expected.

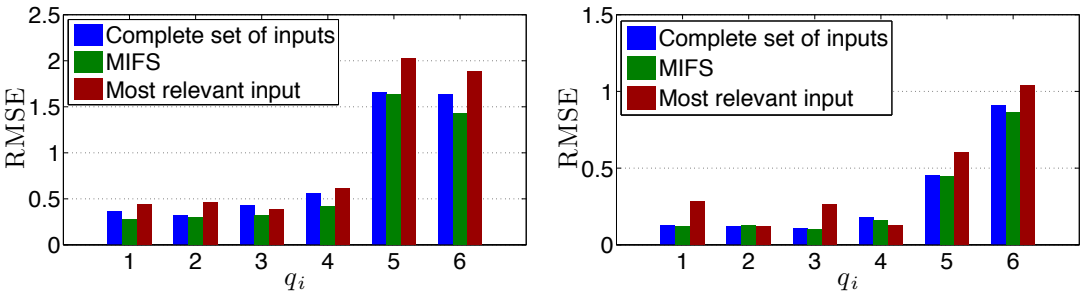
These results provided a good starting point to evaluate the encoding and reproduction capabilities of the proposed framework in more real force-based tasks. In Section 6 results of the complete model encoding all the provided demonstrations of the pouring skill are analyzed. The Matlab source codes of the proposed learning framework will be made publicly available at the time of publication.

6 Experimentation with the complete systems

In this section we show how MIFS-based inputs selection influence the robot performance, and also we analyze the encoding and reproduction results of the proposed framework for both manipulation tasks previously described in Section 2.

6.1 Assessing the mutual information criterion

In order to assess the interest of using MIFS within our learning framework, we evaluate the robot performance in terms of the root mean squared error (**RMSE**) of the joint trajectories. The objective is to observe how



(a) Ball-in-box task (b) Pouring task

Fig. 12 Root mean squared error of robot reproductions for a given set of query vectors. Different sets of input variables are used in order to test the robot performance for three different cases: *(a)* when the original input set is used (blue bar), *(b)* applying MIFS (green bar) and *(c)* using only the most relevant input (red bar).

the robot performance varies for three different cases, namely: *(a)* when all the input variables are used, *(b)* when only the perceptions selected by the modified MIFS-U compose the input space (see Section 3), and *(c)* when the most relevant input is solely used. To achieve this, three different HMMs were trained using the aforementioned datasets, and a query vector for every initial ball position was extracted from the demonstrations. The mean RMSE for each robot joint was computed across all the initial positions.

Ball-in-box task

Figure 12(a) shows the RMSE values (given in degrees) for the three different cases. On the one hand, note that the RMSE across all the robot joints shows nearly the same values for cases *(a)* and *(b)*, which proves that the MIFS-based dimensionality reduction does not affect the robot performance because the unselected input variables do not influence the robot behavior (e.g., F_y and T_z are weakly correlated to the robot actions). The RMSE even slightly decreases in case *(b)* for some joints, which might mean that the removed perceptions were introducing noise (or at least no useful information) into the system, making a bit harder to reproduce the task satisfactorily.

On the other hand, looking at the RMSE of the robot joints q_5 and q_6 (those playing the most relevant role to carry out this task), it is observed that in case *(c)*, i.e., when the learning framework uses exclusively the most relevant perception T_x, the robot performance deteriorates, which might indicate that the robot does not have enough information to perform successfully. Here, it should be mentioned that the robot is not able to carry out the task when perceiving only T_x, because this variable does not describe entirely the location of the ball inside the box (see Figure 7).

Pouring task

The same set of experiments carried out to assess robot performance for the ball-in-box task was also carried out for this task. Again, three HMMs were trained and a query vector of every pouring demonstration (four in this task, as described in Section 2.2) was used to compute the RMSE of the resulting robot joint trajectories. The three same cases *(a)*, *(b)* and *(c)* above were used to analyze how MIFS-U may influence the robot performance in this task. Figure 12(b) shows the mean RMSE obtained for all the robot joints across the four reproductions. Again, it is observed that the robot performance is almost the same for cases *(a)* and *(b)*, thus MIFS-U does not deteriorate robot execution while it reduces data dimensionality and saves computational resources. For case *(c)*, RMSE values are a bit greater than the ones observed for cases *(a)* and *(b)*, however, the robot was also able to carry out the task successfully. This may be explained by the fact that F_z nearly provides the same information given by T_x. They provided redundant information about the task as shown in Figure 6, but both of them are relevant in the sense of their correlation with the robot output commands (see Section 3.2).

6.2 Encoding and reproduction results

Computational and experimental results of the two manipulation task are explained and analyzed in the next paragraphs, where the models were trained using the input dataspace reduced through MIFS.

Ball-in-box Task

To evaluate the performance of the proposed learning framework in this scenario, the teacher carried out four demonstrations for ten different initial ball positions

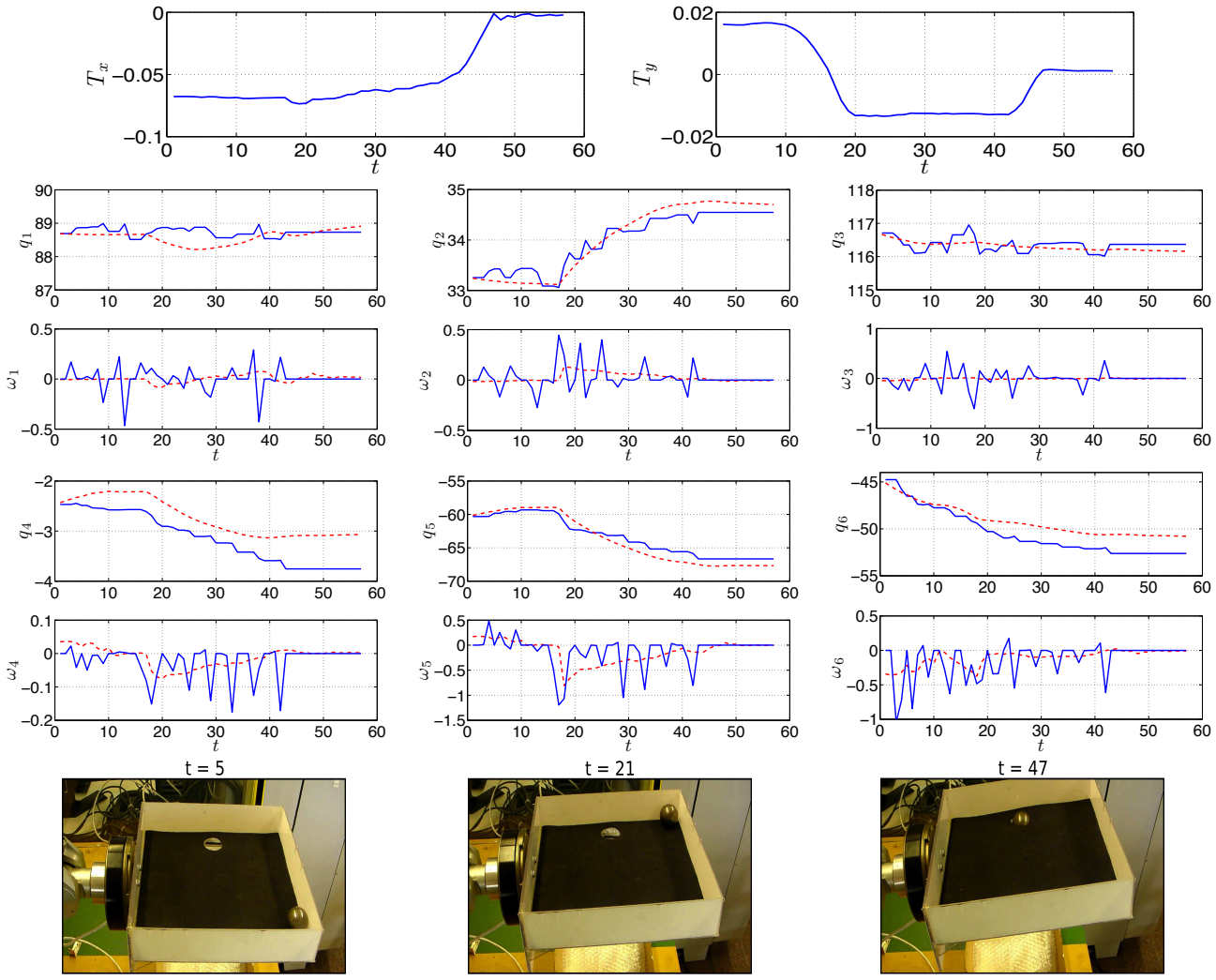


Fig. 13 *Top*: The first row shows the pattern of inputs T_x and T_y when the ball starts at position number 7, and the remaining rows display the robot joint trajectories and velocity profiles when the teacher demonstrated the task (solid blue line) and when the robot executed its motions based on predictions given by GMRa (dashed red line). *Bottom*: Left image shows a snapshot of the beginning of the learned task. The center image displays the moment where the robot has completed the first stage of the strategy and starts to orient the box for taking the ball towards the hole. The right image shows the successful completion of the task.

placed along the box edges. Every demonstration was executed by teleoperating the robotic arm through the 6-DoF haptic device (as shown in Figure 2) and following the motion strategy explained in Section 2. The resulting training dataset consisted of all datapoints $\mathbf{d}_p^m$, which were used to train several HMMs by applying the *Baum-Welch* method until convergence. To find the "best" model, we resort to the Bayesian Information Criterion (**BIC**), which allows to find a trade-off between optimizing the model's fitting and the number of states [48]. Note that the selected HMM will be a model that can fit the data well, with no overfitting in BIC sense. Figure 14 displays the different BIC values for the set of models tested, and Figure 15 shows the selected 5-states HMM. The execution and generalization capabilities were tested for some of these models using query data extracted from the demonstrations and real experiments. The 2-states HMM showed the worst performance, this model was not able to carry out the task starting at any place, even if it did it from a pre-trained initial position. The HMMs with 4, 8 and 9 states could achieve the goal from every pre-trained positions but sometimes failed starting at non-trained initial configurations, showing poor generalization capabilities. Finally, the models with 5, 6 and 7 states showed very similar performances with no clear differences, and all of them performed the task successfully.

Observing the selected 5-states HMM, it is interesting to highlight how the proposed framework is able to learn a multiple solution task by taking advantage

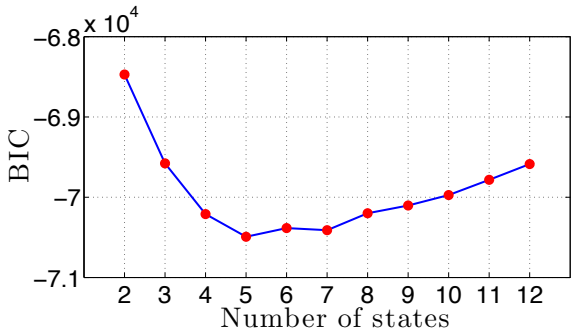


Fig. 14 BIC values for models with different number of states.

of the HMM properties. The model is shown in Figure 15, where the blue state in the input space covers the beginning of all demonstrations whose initial positions are placed on the wall opposite to where the hole is. At these starting positions, a larger velocity command is required to draw the ball out of its resting configuration by moving the robot joint q_6 (Figure 15, output space projection). After, the green and light-blue states represent the movements to force the ball to role to the hole, through q_5 and depending on whether the ball is up or down with respect to the hole (i.e., positive or negative velocity commands, respectively). The yellow Gaussian can be considered as an intermediate state the system goes through to reach the final state (red ellipse) at which the velocity commands are zero (i.e., when the ball is getting out of the box) in input space.

As for the prediction phase, one teacher's demonstration for each initial position was removed from the training examples and used as "query data" for evaluating the learning framework performance by comparing its results with the teacher executions. All robot joint trajectories obtained from velocity commands synthesized by our HMM/GMRa approach are smoother than the teacher's demonstrations (as shown for initial positions 3 and 7 in Figures 10 and 13, respectively). By observing the obtained velocity profiles for each robot joint, one sees that they are also smoother than teacher ones, because human user executions show several abrupt changes, which are not over-fitted by our learning framework. This can be attributed to the fact of using GMRa to retrieve the velocity command, because this type of regression takes the covariance information into account for computing the estimation of the output, outperforming techniques that only use the mean of the Gaussians. Thus, we can conclude in this context that the robot performs better than the teacher. In addition, all synthesized trajectories follow the same motion pattern as that of the teacher's executions, which indicates that the strategy applied by the human user was learned successfully.

Once computational results were satisfactory, we validated our framework on the experimental setup. First, the robot had to perform the task with the ball starting at the already trained initial positions (see Figure 2). For all experiments, the robot was able to carry out the task effectively. After this, a second set of tests was executed, where the ball was located at random positions inside the container. For these tests, the robot was also able to achieve the task's goal, executing the motions learned for the closest initial position, by identifying the corresponding HMM state. It was observed that in some executions the ball reached and surpassed the hole, without falling through it. This behavior may be justified by the fact that we are assuming a "quasi-static" case in our task.[6] However, the robot was always able to take the ball out of the box after some more executions, as it correctly identified the HMM state corresponding to the current and past input patterns (taking into account the temporal information). This means that the robot generates its actions as a function of its current and past perceptions, following the taught motion strategy. If the robot fails to reach the goal, the ball goes to another position inside the box, providing new perceptions from which the robot can compute new movements. Videos showing executions of learned trajectories are available online at `http://dl.dropbox.com/u/15185273/JISR/JISR.html`.

As the robot was able to accomplish the desired goal in every test, even when the ball reached and surpassed the hole, we evaluated the performance of the robot executions using a time-based criterion [49]. Here, the idea is to determine how much time the robot takes to complete the task successfully by executing the commands obtained from the proposed framework compared with the three following cases: *(i)* the robot executes hand-coded actions according to pre-programmed *if-then* rules, *(ii)* the teacher carries out the task by teleoperation following the mentioned strategy, *(iii)* the robot performs random movements that may take the ball towards the hole. Figure 16 shows execution times for the aforementioned cases. As expected, the teacher's executions show to the lowest times, except for position number 2 where the robot was faster than the human. A relevant aspect to discuss is the fact that the robot execution times are much larger than the teacher's ones for positions 3 to 8. Regarding positions 3 to 5, higher times are due to the fact that the robot starts the task by moving the joint q_6 as expected, however it also

[6] On the one hand, the model variables are force-torque and joint velocities at the given time step, thus no information about the past is explicitly provided. On the other hand, the robot controller only allows position-based control, thus it is not possible to send the desired velocity commands directly.

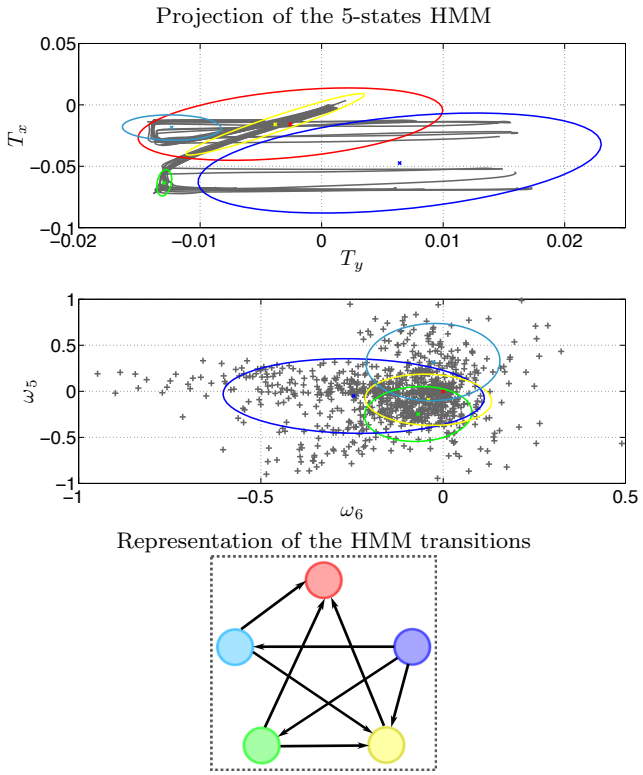


Fig. 15 Resulting 5-states HMM trained with demonstrations starting at every position inside the box. *Top*: Input space composed of the most relevant inputs $\{T_x \quad T_y\}$. *Middle*: Output space composed of robot joint velocities playing the most important role for the given task. *Bottom*: Representation of the resulting transition probabilities matrix. As expected, the most likely transitions from the blue state take the robot to the light blue or green ellipses. Moreover, the transitions from these Gaussians and the yellow one take the system towards the final state.

moves q_5 slightly which sometimes causes the ball to go to the bottom of the box, justifying higher standard deviations for positions 3 and 4. This is a normal effect because the first state of the learned HMM covers non-zero angular velocities for the variable ω_5. Thus, in these cases, the robot identifies the new state where the ball is and changes its motion strategy according to the given input data for reaching the target.

In the case of positions 6 to 8, the robot does also move the joint q_5, however it is because the teacher demonstrations showed that the human tries to guarantee "a stable motion" by taking the ball towards the wall adjacent to the hole along the wall at the bottom of the box. This causes that, when the ball reaches the wall adjacent to the hole, the robot has to carry out more movements in order to take the metalic sphere towards the hole, since the robot must compensate the initial inclination of the box given by the wrong motion of q_5. Thus, the high robot execution times are mainly a consequence of two factors: first, there is a delay between the sensing and execution phases that increases the time measures as the ball is farther from the target, and second, the joint velocity profiles of the robot execution show lesser magnitudes than the teacher ones (as observed in Figures 13 and 10), implying that when these velocity commands are translated into desired positional configurations of the robot, the joints rotation is lower and more velocity commands are needed to orient the box.

Regarding the times shown for the hand-coded actions, several robot learned executions outperformed the hand-coded ones (e.g., starting at positions 1, 2, 4, 7, 9 and 10). This mainly happened because the hand-coded actions also suffered the "surpassing" effect, that is, the ball did not go out through the hole at the first attempt. Moreover, it is important to emphasize that the *if-then* rules programming was tedious and time-consuming, even for this simple task. On the one hand, it was essential to determine how the input space could be transformed to discrete regions to set the *if* conditions. On the other hand, a tuning process was needed to specify the velocity commands that the robot executed. One may think that the higher the velocity, the less time the robot might take to accomplish the task, however the "surpassing" effect may occur more often, increasing the time execution significantly. Thus, the learning-based approach is preferred because being similarly efficient, it is friendlier and can be applied by non-expert users.

Finally, execution times for a "random" strategy show that trying to accomplish the goal by chance is possible, nevertheless this implies much higher times and variances in comparison with when the robot carried out the task by using the taught strategy. These high values occur because the random strategy does not impose movement constraints to the robot and, therefore, a huge set of available motions can be executed, leading to very varied and long trials. This constitutes a reference (lower bound) for comparison purposes, against which the improvement attained by different learning techniques and teaching strategies can be evaluated.

Pouring Task

In order to teach the robot to pour drinks, three "complete executions" of the task are provided to the robot by teleoperation as described in Section 2. Such executions consist of starting with the bottle full of fluid and pouring four 100ml drinks. Note that after each drink is poured, the initial force-torque value changes for the next demonstration, which conditions the robot movements as shown in Figure 17 where the gray lines

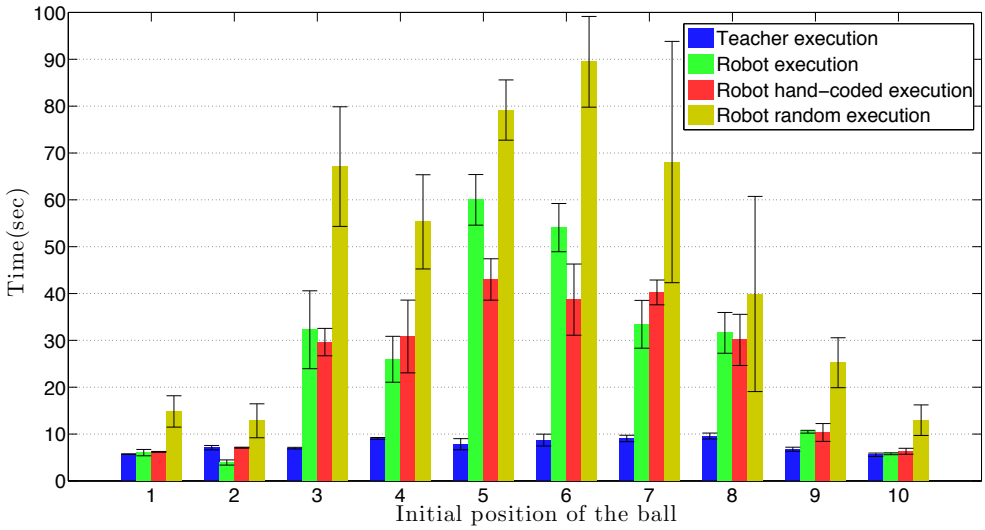


Fig. 16 Mean times for teacher executions, robot learned, hand-coded and random executions starting at each pre-defined initial position of the ball inside the box.

represent the teacher demonstrations. Observe that the less quantity of fluid, the more the robot rotates the bottle.

The resulting training dataset was used to train a 5-states HMM by applying the *Baum-Welch* method until convergence. Note that the number of states was chosen according to the BIC score, similarly to the *ball-in-box* task. Figure 17 shows the model encoding the *pouring* skill, where the yellow state covers the beginning and the end of all the demonstrations, whereas the light blue and green ellipses are encapsulating the phases corresponding to when the fluid is coming out of the bottle. The other two Gaussians can be considered as intermediate states of the task. It is worth highlighting that the resulting states distribution provides very good generalization capabilities (discussed below), which are exploited when the robot has to pour a drink starting from a force-torque value not previously observed (e.g., in between two demonstrated starting values).

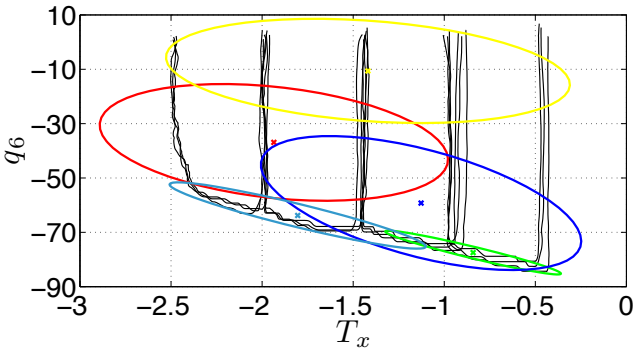
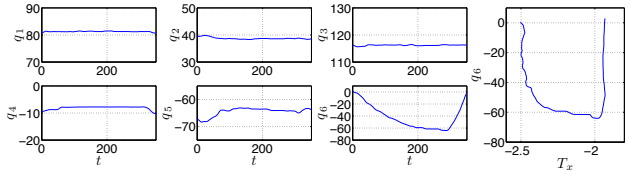
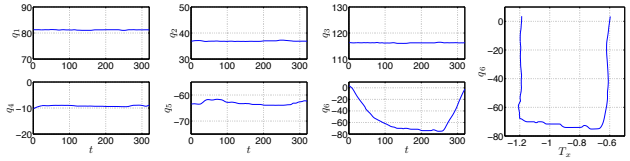


Fig. 17 Resulting 5-states HMM trained with four different sets of demonstrations of the *pouring* task. Each set of provided samples of the skill shows a different initial force-torque value given by the quantity of "fluid" inside the bottle. According to these initial conditions, the less quantity, the more the robot rotates the bottle.

In order to test the reproduction performance of the model, four demonstrations were removed out of the training data to be used as query datapoints. All the robot joint trajectories were quite similar to the ones obtained from the teacher examples as well as the input-output pattern, as shown in Figure 18(a) (corresponding to the first drink). After this, the following tests were aimed at evaluating the generalization capabilities of the trained HMM. In this case, the bottle contained quantities of fluid different from the ones used at the demonstration phase, but remaining within the range of force-torque measured in that stage. For all the tests where the starting force-torque perception was covered by the initial HMM state (i.e., the yellow ellipse in Figure 17), the robot performed successfully. The robot joint trajectories and the input-output pattern for one of these tests is shown in Figure 18(b). Nevertheless, as the starting perception significantly differs from the values encapsulated by the initial state, the robot performance deteriorates. In others words, the actual model shows good interpolation competences but a poor extrapolation performance. This feature will be tackled in future work as discussed below. Nonetheless, the obtained results evidence the generality of the proposed approach as well as its usefulness in more realistic scenarios. We thus confirm the suitability of our framework to learn manipulation tasks using only force-based perceptions.



(a) Reproduction starting from an already trained force-torque perception



(b) Reproduction starting from an unobserved force-torque perception

Fig. 18 Reproduction of the *pouring* task for two different quantities of fluid. The robot joint trajectories during the execution are shown in the three first columns. Last graph displays the reproduction pattern for q_6 as a function of T_x.

7 Conclusions and Future Work

This work presents a suitable end-to-end system for learning manipulation skills where force-torque data constitute the most relevant input source, specially in the absence of visual information. In the first place, we addressed the problem of conditioning a good bidirectional communication channel between the teacher and the robot when using a haptic device as an information transferring tool. This device provides an useful interaction means between the human and the robot, which becomes more relevant and necessary when the robot is located in a remote place, for instance in space applications. In this specific area, it may be very useful to refine already learned robot actions while feeling the robot's perceptions as a consequence of the teacher's refinement commands. In other words, haptic devices may not be used just in the teaching process, they may also be used as refinement tools as well, for instance as an alternative when kinesthetic refinement is not possible for robots working in remote places. Moreover, the inclusion of a haptic interface improves the teacher's demonstrations when the task relies on force-based perception, because the user can feel how his/her actions affect the robot's surroundings.

We solved the *what to imitate?* problem from a new perspective, by using MI-based inputs selection. Results showed that this technique is appropriate to find which input variables are the most relevant to learn a task. This presents several advantages in LfD settings: reduction of input space dimensionality, less computational cost and probably faster training and execution stages. This approach can also be applied before finding the demonstration segments with low variability that indicate those sections that must be learned. Note that the variance-based approach is more suitable for trajectory learning (i.e., low-level encoding), whereas our solution is more generic and may be applied to a wider set of tasks.

The reduction of the input space obtained through the modified MIFS-U showed not to deteriorate the robot performance when this was compared to the results obtained from the case where the learning framework used all the input variables. Moreover, as an additional contribution, the number of components to be retained is now automatically determined by the selection algorithm according to the proposed conditional mutual information ratio. This selection criterion took advantage of the conditional MI used in MIFS-U. Nonetheless, the threshold ϕ is still an open parameter that depends on how selective the method is desired to be. Future work will consider techniques to tune this variable using the provided demonstrations.

On the other hand, our framework performs efficiently when the teacher's demonstrations exhibit a multi-valued function behavior (e.g., the *ball-in-box* task), which means there may be more than one appropriate action – velocity command – for the same perception (i.e., force-torque input pattern). This was achieved by means of a GMRa tool using temporal information encapsulated by an HMM without explicitly considering time as another input variable and avoiding to deal with very large time discrepancies. Time, or rather sequential information, is already implicitly present along the teacher's demonstrations.

After having carefully designed and tested the several stages of our LfD framework on the well-defined and easy to analyze task of taking a ball out a container, a more realistic task was designed to evaluate the generality of the proposed approach by learning to pour drinks based on force data exclusively. As observed, the feature selection, encoding and reproduction methods showed satisfactory results as well.

As future work we would like to apply force-based skills learning to compliant robots in an active learning environment as a refinement or correction phase, needed to improve the robot performance. This type of robots would allow us to extend our approach to human-robot collaborative tasks by taking advantage of their compliance features. In this context, force-torque perceptions about the task and the haptic communication between the partners are a rich source of information to carry out manipulation tasks cooperatively, where our framework may be used as the core of a learning structure oriented to this kind of scenarios.

Moreover, it was observed that, in some force-based tasks, certain parameter strongly conditions the subse-

quent robot actions. For instance in the *pouring* task, the robot movements were conditioned by the initial force-torque perception (indicating the amount of fluid in the bottle), which may be considered as a parameter of the skill at hand. This feature opens the door to the use of parametric learning methods to encode such type of tasks. Specifically, we plan to enhance the proposed learning framework by introducing such parameters in the model through parametric Hidden Markov models [50].

References

1. A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Springer Handbook of Robotics*, chapter 59. Robot Programming by Demonstration, pages 1371–1394. Springer, 2008.
2. B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning by demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.
3. M. Goodrich and A. Schultz. Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.
4. N. Najmaei and M. Kermani. Applications of artificial intelligence in safe human–robot interactions. *Trans. on Systems, Man and Cybernetics, Part B*, 41(2):448–459, 2011.
5. R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, 2004.
6. M. Riley, A. Ude, C. Atkeson, and G. Cheng. Coaching: An approach to efficiently and intuitively create humanoid robot behaviors. In *Intl. Conf. on Humanoid Robots*, pages 567–574, 2006.
7. D. Bentivegna, C. Atkeson, and G. Cheng. Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47(2-3):163–169, 2004.
8. D. Grollman and O. Jenkins. Dogged learning for robots. In *Intl. Conf. on Robotics and Automation*, pages 2483–2488, 2007.
9. S. Calinon and A. Billard. What is the teacher's role in robot programming by demonstration? toward benchmarks for improved learning. *Interaction Studies*, 8(3):441–464, 2007.
10. T. Inamura, N. Kojo, T. Sonoda, K. Sakamoto, K. Okada, and M. Inaba. Intent imitation using wearable motion capturing system with on-line teaching of task attention. In *Intl. Conf. on Humanoid Robots*, pages 469–474, 2005.
11. S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Intl. Conf. on Humanoids Robots*, pages 255–262, 2007.
12. P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Khedda. Teaching physical collaborative tasks: Object-lifting case study with a humanoid. In *Intl. Conf. on Humanoids Robots*, pages 399–404, 2009.
13. E. Gribovskaya, A. Kheddar, and A. Billard. Motion learning and adaptive impedance for robot control during physical interaction with humans. In *Intl. Conf. on Robotics and Automation*, pages 4326–4332, 2011.
14. P. Kormushev, S. Calinon, and D. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603, 2011.
15. D. Grollman and O. Jenkins. *From Motor to Interaction Learning in Robots*, chapter Can We Learn Finite State Machine Robot Controllers from Interactive Demonstration?, pages 407–430. Springer, 2010.
16. S. Cabras, M. Castellanos, and E. Staffetti. Contact-state classification in human-demonstrated robot compliant motion tasks using the boosting algorithm. *Trans. on Systems, Man and Cybernetics, Part B*, 40(5):1372–1386, 2010.
17. S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Trans. of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2005.
18. D. Kulić, W. Takano, and Y. Nakamura. Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains. *Intl. Journal of Robotics Research*, 27(7):761–784, 2008.
19. C. Atkeson and S. Schaal. Robot learning by demonstration. In *Intl. Conf. on Machine learning*, pages 12–20, 1997.
20. A. Ijspeert, J. Nakanishi, and S. Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *Intl. Conf. on Intelligent Robots and Systems*, pages 752–757, 2001.
21. T. Cederborg, M. Li, A. Baranes, and P. Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. In *Intl. Conf. on Intelligent Robots and Systems*, pages 267–274, 2010.
22. A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.
23. 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*, pages 244–249, 2000.
24. K. Dines. Constrained least squares filtering. *Trans. on Acoustics, Speech and Signal Processing*, 25(4):346–350, 1977.
25. M. Uchiyama and K. Kitagaki. Dynamic force sensing for high-speed robot manipulation using kalman filtering techniques. In *Intl. Conf. on Decision and Control*, pages 2147–2152, 1989.
26. J. Garcia, A. Robertsson, J. Ortega, and R. Johansson. Generalized contact force estimator for a robot manipulator. In *Intl. Conf. on Robotics and Automation*, pages 4019–4024, 2006.
27. M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59(11):910–922, 2011.
28. M. Cakmak and A. Thomaz. Designing robot learners that ask good questions. In *Intl. Conf. on Human-Robot Interaction*, pages 17–24, 2012.
29. C. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. *Interdisciplinary Approaches to Robot Learning, World Scientific Series in Robotics and Intelligent Systems*, 24:136–161, 2000.
30. S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *Trans. on Systems, Man and Cybernetics, Part B*, 37(2):286–298, 2007.
31. S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling constraints

in joint space and task space. In *Intl. Conf. on Intelligent Robots and Systems*, pages 367–372, 2008.
32. D. Lee and C. Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31:115–131, 2011.
33. K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.
34. I. Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
35. R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Trans. on Neural Networks*, 5(4):537–550, 1994.
36. T. Ikeda and H. Ishiguro M. Asada. Adaptive fusion of sensor signals based on mutual information maximization. In *Intl. Conf. on Robotics and Automation*, pages 4398–4402, 2003.
37. G. Wells and C. Torras. Assessing image features for vision-based robot positioning. *Journal of Intelligent and Robotics Systems*, 30(1):95–118, 2001.
38. C. Shannon. A mathematical theory of communication. *SIGMOBILE Mobile Computing and Communications Review*, 5:3–55, 2001.
39. N. Kwak and C. Choi. Input feature selection for classification problems. *Trans. on Neural Networks*, 13(1):143–159, 2002.
40. P. Estévez, M. Tesmer, C. Perez, and J. Zurada. Normalized mutual information feature selection. *Trans. on Neural Networks*, 20(2):189–201, 2009.
41. L. Rozo, P. Jiménez, and C. Torras. Sharpening haptic inputs for teaching a manipulation skill to a robot. In *Intl. Conf. on Applied Bionics and Biomechanics*, pages 370–377, 2010.
42. L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
43. A. Billard, S. Calinon, and F. Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54:370–384, 2006.
44. D. Kulić and Y. Nakamura. Incremental learning of human behaviors using hierarchical hidden Markov models. In *Intl. Conf. on Intelligent Robots and Systems*, pages 4649–4655, 2010.
45. S. Dong and F. Naghdy. Application of hidden Markov model to acquisition of manipulation skills from haptic rendered virtual environment. *Robotics and Computer-Integrated Manufacturing*, pages 351–360, 2007.
46. L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.
47. S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard. Learning and reproduction of gestures by imitation. *Robotics and Automation Magazine*, 17(2):44–54, 2010.
48. S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009.
49. A. Steinfeld, T. Fong, D. Kaber, M.Lewis, J. Scholtz, A. Schultz, and M. Goodrich. Common metrics for human-robot interaction. In *Intl. Conf. on Human-Robot Interaction*, pages 33–40, 2006.
50. A. D. Wilson and A. F. Bobick. Parametric hidden Markov models for gesture recognition. *Trans. on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.