# A Friction-model-based Framework for Reinforcement Learning of Robotic Tasks in Non-rigid Environments

Adrià Colomé, Antoni Planells and Carme Torras

***Abstract*— Learning motion tasks in a real environment with deformable objects requires not only a Reinforcement Learning (RL) algorithm, but also a good motion characterization, a preferably compliant robot controller, and an agent giving feedback for the rewards/costs in the RL algorithm. In this paper, we unify all these parts in a simple but effective way to properly learn safety-critical robotic tasks such as wrapping a scarf around the neck (so far, of a mannequin).**

**We found that a suitable compliant controller ought to have a good Inverse Dynamic Model (IDM) of the robot. However, most approaches to build such a model do not consider the possibility of having hystheresis of the friction, which is the case for robots such as the Barrett WAM. For this reason, in order to improve the available IDM, we derived an analytical model of friction in the seven robot joints, whose parameters can be automatically tuned for each particular robot. This permits compliantly tracking diverse trajectories in the whole workspace.**

**By using such friction-aware controller, Dynamic Movement Primitives (DMP) as motion characterization and visual/force feedback within the RL algorithm, experimental results demonstrate that the robot is consistently capable of learning tasks that could not be learned otherwise.**

## I. Introduction

The interest in human-robot interaction is growing nowadays, thanks to the increasing availability of more compliant robots with low inertia, that makes them safer to move in a soft or fragile environment, in which we could include interaction with humans. Tasks like placing a scarf on a mannequin (see Fig. 1) can be taught to the robot by using Dynamic Movement Primitives (DMP) and then reproduced with high precision. However, in compliant environments, there is usually a tradeoff between precision and safety as, making the robot more precise (commonly with a high error-compensating term) will make its motion *stiff*, which makes it dangerous for a human. For this reason, model-based controllers such as Computed Torque Control (CTC) [1] can be used, for which we need an IDM of the robot, i.e., the mapping from the position, velocity and acceleration to the total torques acting on the robot. The IDM of a robot can be computed by derivating its mechanical energy, or by iterative methods (see [2], Chapter 7), or fitted by a regression model from real execution data with different machine learning methods [3]. Nevertheless, none of these methods generally takes into account the impact of high hystheresis on the dynamics of some robots, as the occurs for the Whole Arm Manipulator (WAM), where for non-high speed motion the friction is usually the second highest torque acting on the robot after gravity.

Not properly representing this hystheresis reduces the accuracy of the IDM. As a consequence, a higher error-compensating gain is needed to ensure good trajectory tracking, thus making the robot less compliant. For this reason, in Sec. II we propose an analytical formulation of the friction, which can be easily tuned with real data and then computed online, providing good results in the whole joint space of the robot.

Once having this IDM, we built a learning framework (see Fig. 2), using DMPs for the motion characterization and a model-based External Force Estimation (EFE) [4] to obtain the interaction torques with the environment. Within this framework, starting from an initial demonstration, exploration on the DMP parameters progressively improves the robot's performance and, using a Policy Improvement with Path Integrals (PI2) [5] algorithm, better robot behaviours are learned relying on a cost function based on vision, kinematics and dynamics.

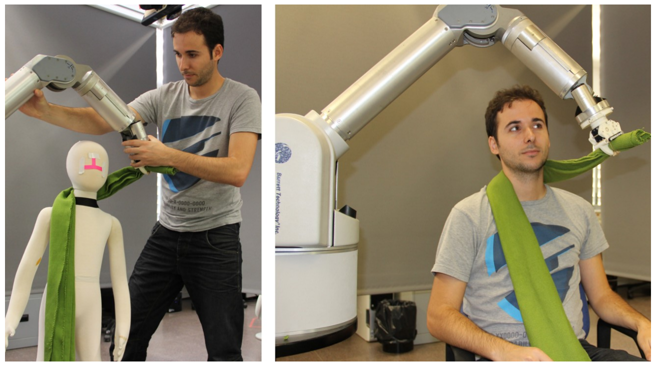
Fig. 1. Kinesthetic teaching of a task to a WAM robot (left) and motion reproduction with a compliant controller (right).

In Sections III and IV, we detail how the friction model is used to track a trajectory provided by a DMP and its embedding in a RL setting, to later show an experiment with a real robot in Section V.

## II. Building a friction model

### A. Robot Dynamics Equation

The dynamics equation of a robot in joint space [2] is

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}_{fric} = \mathbf{u}_c - \mathbf{u}_e, \quad (1)$$

*This work was supported by CSIC project MANIPlus (201350E102). A. Colomé is also supported by the Spanish Ministry of Education, Culture and Sport via a FPU doctoral grant (AP2010-1989).

Authors are with Institut de Robòtica i Informàtica Industrial (UPC-CSIC), Barcelona (SPAIN), [acolome,aplanells,torras] at iri.upc.edu

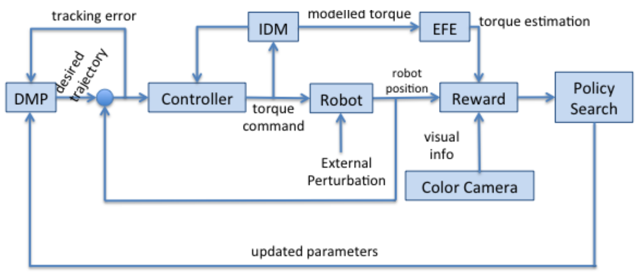

Fig. 2. Global scheme of the proposed framework. The DMP sends the desired trajectory to the feed-forward controller, which uses the inverse dynamic model to track the trajectory compliantly. The robot performs the task, which may include an interaction with the environment, estimated by the external force estimator. The latter is used together with the camera feedback and the acceleration measurements, to obtain a reward/cost function, which is used by a policy search algorithm to, after a certain number of rollouts, update the DMP.

where $\mathbf{M}(\mathbf{q})$ is the inertia tensor of the robot, $\mathbf{C}(\dot{\mathbf{q}},\mathbf{q})$ is the term containing the coriolis and centripetal forces, $\mathbf{G}(\mathbf{q})$ the gravity, $\mathbf{F}_{fric}$ the robot friction, $\mathbf{u}_c$ the controller torque and $\mathbf{u}_e$ the external forces on the robot.

As mentioned, in the absence of external forces, we can model/compute the inertia, coriolis, centripetal and gravity forces, and the controller torques are assumed to be known, thus we can infer the values of friction as:

$$\mathbf{F}_{fric} = \mathbf{u}_c - \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} - \mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) - \mathbf{G}(\mathbf{q}), \tag{2}$$

and use them to fit a friction model. To obtain the terms in (2) in the case of the WAM robot [6], the code provided by the manufacturer already included an application to calibrate the gravity values, while geometry, masses and inertia terms are also provided by the manufacturer and can be used to obtain the coriolis, centripetal and inertia terms.

## B. Basic Friction model

As mentioned in the introduction, the friction torque applied in any joint of the robot when it moves at low speed is usually the second highest torque acting on the robot after gravity. This friction has a high hysteresis in its dynamic behavior which makes it very difficult to model. The reason to fit the friction as an hystheresis function, rather than with a *complete* set of basis functions, is that we know the qualitative behaviour of the friction, thus using a proper fitting function will be more efficient than using any other type of kernel, in terms of precision, number of parameters, and samples required for the fitting process.

At first, this torque was modeled as a viscous friction $F_{fric} = c\dot{q}$, where $\dot{q}$ is the joint's velocity and $c$ is a constant that varies depending on the joint that moves. This model was not very precise and did not model the friction with a high degree of accuracy because some other variables had not been considered, such as the position of the joints.

In [7], the friction is modelled with an initial model as:

$$F^i_{fric} = b_1 atan(s\dot{q}_i) + b_2\dot{q}_i, \tag{3}$$

where $\dot{q}_i$, is the $i$th joint velocity and $b_1$, $b_2$ and $s$ are parameters obtained with least-squares techniques.

However, this model not only is independent from position (which we observed as a fact from data), but also has no hysteresis value (friction is zero for zero velocity). Thus we added a term to model this hystheresis with a parameter $z$ defining its width and the sign of the acceleration of the joint, which we will consider as our basic model:

$$F^i_{fric} = \quad b_1 atan(s\dot{q}_i + z sign(\ddot{q}_i)) + b_2.\dot{q}_i, \text{ for i = 1..7} \tag{4}$$

This basic model did not offer the level of accuracy required because it did not show the linear dependence the friction could have with the position of the joints, along with other phenomena that could be seen. One of this other phenomena that could be observed were some oscillations depending on the position of the joints.

## C. Advanced model for the WAM robot

To improve the basic model some changes were made and new variables and terms were added:

- A linear term in position, which was observed through data analysis.
- A basis of Fourier functions on the joints position, divided in two layers, depending on the sign of the velocity of each joint.

These Fourier basis functions were added to model an oscillating curve wrt the position of the friction. However, the authors observed different oscillations for positive and negative velocities, thus two sets of Fourier basis were fitted for them, and activated with the help of a sign function, as we can see in Eq. (5). These Fourier terms approximate the oscillations seen wrt position, ignoring the noise data due to the PID controller that was used to obtain the data.

This brought us to a friction model for each of the joints of the robot with the following expression:

$$\begin{aligned}&F^i_{fric} = b_1 q_i + b_2\dot{q}_i + b_3 atan(s\dot{q}_i + z sign(\ddot{q}_i))\\ &+0.5(1+sign(\dot{q}_i))(b_4 f_1 + b_5 f_2 + b_6 f_3 + b_7 f_4 + b_8 f_5)\\ &+0.5(1-sign(\dot{q}_i))(b_9 f_1 + b_{10} f_2 + b_{11} f_3 + b_{12} f_4 + b_{13} f_5),\\ &\text{ for i = 1..7,}\end{aligned} \tag{5}$$

where $f_j = \dfrac{4}{\pi}\dfrac{sin((2j-1)hq_i)}{2j-1}$, $j = 1..5$ are the sine Fourier basis functions. We neglected the cosine functions assuming an antisymmetric behaviour of the friction. $q_i$, $\dot{q}_i$ i $\ddot{q}_i$ are the position, the velocity and the acceleration of the $i$th joint. All the constraints $b_k$, $k = 1..13$, $s$, $z$ i $h$ were obtained with least-squares techniques and vary according to each joint.

Most of the terms in (5) have a physical interpretation. Firstly, $b_2$ is the viscous friction coefficient of any joint of the robot, while $b_3 atan(s\dot{q}_i + z sign(\ddot{q}_i))$ models the histeresis of the friction through the parameters $b_3$, $s$ and $z$: $b_3$ allows to set the vertical amplitude of the hysteresis, $s$ defines the transition speed between the two levels and $z$ defines the width of the hysteresis. Finally, the terms $b_1 q_i$ and the sine Fourier functions are used to model the motor cogging effect, and some of the model uncertainties. The parameter $h$ is used to select the frequency of the oscillations seen in Fig 3.

This friction model has a high degree of accuracy only when joints were moving separately and did not take into account that some of the joints shared part of their friction due to the coupling between their engines.

Joints 1, 4 and 7 do no present this phenomenon, but joints 2-3 and joints 5-6 do when they move together in pairs. In order to model the friction torque of a pair of joints that share the coupling effect when they move we had to find a new model so that the friction torque was fitted with enough accuracy when this happened.

In this case, after several observation experiments, it could be seen that the friction torque applied on a joint was reduced when the other joint of the pair moved at a higher velocity. This reduced friction torque showed the same non-linear hysteresis behaviour but with a smaller range of values and was modeled with the same expression but with different least squares parameters.

After modeling this reduced friction, a transition between models had to be established so that the friction torque was fitted correctly at every timestep. Defining $modA_i$ as the friction model of the $i$th joint when it moves at a higher velocity than its pair $j$ (non reduced friction torque), $modB_i$ as the friction model of the same $i$th joint when it moves at a lower velocity than its pair $j$ (reduced friction torque) and defining $mod_i(t)$ the friction model of the $i$th joint at timestep $t$:

$$\begin{aligned} mod_i(t) &= modA_i(t), \text{ if } |\dot{q}_i(t)| \geq |\dot{q}_j(t)| + \epsilon \\ mod_i(t) &= modB_i(t), \text{ if } |\dot{q}_j(t)| \geq |\dot{q}_i(t)| + \epsilon \\ mod_i(t) &= mod_i(t-1), \text{ otherwise,} \end{aligned}$$

for the pairs $(i = 2, j = 3)$, $(i = 3, j = 2)$, $(i = 5, j = 6)$ and $(i = 6, j = 5)$, where $\epsilon$ is used to avoid chattering between the friction models of each one of the coupled joints.

After defining this transition when the coupling phenomenon appears, the dynamic friction torques of each of the joints of the robot are modeled with enough accuracy, even if all the joints move together at the same time.

### D. Fitting performance

In order to compare the performance of the different models in equations (3), (4) and (5), we fitted the three models with a dataset of 80 oscillation movements at different speeds, moving joints individually at different positions and also with coupled movements.

We built another dataset to validate the models and our proposed friction model in Eq. (5) showed to perform the best in all cases. In Fig. 3, we can see an example validation trajectory with the data obtained from Eq. (2). Also, in Table I, we show the error indicators used in order to validate the advanced model against the initial and basic model with the new validation dataset. The error indicators used were the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the maximum sample error in the dataset. It can be clearly seen that this approach outperforms previous ones in terms of numerical error.

TABLE I
FRICTION VALIDATION RESULTS

| | Initial Model | Basic model | Adv. model |
|---|---|---|---|
| MAE | 0.5907 | 0.5115 | 0.4277 |
| MSE | 0.7613 | 0.5611 | 0.3511 |
| Max error | 4.7281 | 3.6624 | 2.5849 |

Performance indicators for a set of N validation trajectories for the Barrett WAM joint 1. We indicate the MAE, MSE and maximum error, all of them averaged over 8 validation trajectories.

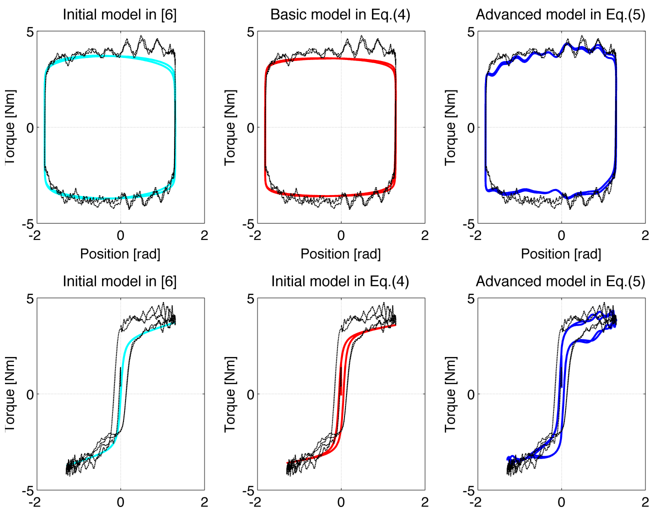

Fig. 3. Example of the fitting of the three models for a trajectory of the first joint of the Barrett WAM. Our proposed function outperforms the previous approaches.

## III. CONTROLLING AND REPRODUCING TRAJECTORIES

Given the friction model in Section II, we can build an IDM that can be used to track a desired trajectory, In this section, we explain how to build such trajectories and the controller used for tracking them.

### A. DMP

In order to learn a task, a robot must have a framework characterizing its motion. Along this work we used Dynamic Movement Primitives (DMP) [8], [9], which characterize a movement by means of a second order dynamical system, using a position error, a velocity term and an excitation function for obtaining the acceleration profile generating the movement:

$$\begin{gathered} \dot{\mathbf{z}}/\tau = \alpha_z \left(\beta_z \left(\mathbf{y_g} - \mathbf{y}\right) - \mathbf{z}\right) + \mathbf{f}(x) \\ \mathbf{f}(x) = \mathbf{\Theta}\mathbf{g}(x), \end{gathered} \tag{6}$$

where $\mathbf{y}$ is the joint position vector, $\mathbf{y_g}$ the goal/ending joint position, $\tau$ a time constant, $x$ is a transformation of time verifying $\dot{x} = -\alpha_x x/\tau$ and $\mathbf{z} = \dot{\mathbf{y}}/\tau$ a rescaled velocity vector. In addition, $\mathbf{\Theta}$ is a parameter matrix, where each row represents the parameters used for each joint to learn an initial move, applied to a set of basis functions $\mathbf{g}(x)$ defined as:

$$g_i(x) = \frac{\phi_i(x)}{\sum_j \phi_j(x)} x, i = 1..N_f, \tag{7}$$

where $\phi_i(x) = exp\left(-0.5(x-c_i)^2/d_i\right)$, and $c_i$, $d_i$ represent the fixed center, usually taken equally spaced in time, and width of the *i*th Gaussian used of a total of $N_f$ per degree-of-freedom. The DMP gives, at each timestep, a value of desired acceleration, which can be integrated to obtain a desired position and velocity.

With this motion representation, the robot can be taught a demonstration movement, to obtain the weights and Gaussians of the motion by using least squares techniques on the isolated excitation function $f$:

$$f_{de}^{(j)}(x) = \dot{z}_{de}^{(j)}/\tau + \alpha_z\left(\beta_z\left(y_{de}^{(j)} - y_g^{(j)}\right) + z_{de}^{(j)}\right) = \boldsymbol{\theta}_{(j)}^T\mathbf{g}(x), \tag{8}$$

where $(j)$ indicates the *j*th joint, the subscript *de* represents the demonstration movement taught to the robot and $\boldsymbol{\theta}_{(j)}^T$ the $j$-th row of $\boldsymbol{\Theta}$. The DMP representation of trajectories has good scaling properties wrt. trajectory time and initial/ending positions, has an intuitive behaviour, does not have an explicit time dependence and is linear in the parameters, among other advantages over alternative motion characterizations [8]. For these reasons, DMP are being widely used with policy optimization RL, where the general goal is to optimize the policy parameters $\boldsymbol{\Theta}$ so that the expected reward (cost) is maximal (minimal). After each rollout, the reward/cost function is evaluated and used to search for a set of parameters that improve the performance over the initial movement.

When executing a motion with a robotic arm, the robot can run into an obstacle, or a person may be manually preventing it to move. If using a controller with an error-proportional torque component, the error will increase over time while the robot is prevented to move, making the controller send an increasing torque signal to the motors. If the robot is then released, the robot will move quickly - and dangerously - towards the far away goal position. For this reason, the phase variable $x$ must be slowed down when the controller's positioning error is large. A common approach is to modify $\dot{x}$ to make it dependent on the positioning error: $\tau\dot{x} = -\frac{\alpha_x x}{1+\mathbf{e}_p^T\boldsymbol{\Lambda}\mathbf{e}_p}$, given a positive-definite matrix $\boldsymbol{\Lambda}$, that will indicate how strong is the influence of the error on the time-slowing effect. Also, for complex motions, fitting a demonstrated robot trajectory often requires a large number of Gaussian kernels to properly encode it with DMPs. For this reason, in [10], the authors proposed to separate the layer reproducing the initial trajectory from the exploration layer. In this work, we used the kernels in Eq. (7) for both layers, with less and wider kernels for the exploration layer.

## B. Controller

Once given a desired trajectory, provided by the DMP or a similar framework, we need a proper controller to track it in a way that the robot is not dangerous. The default controller provided with the WAM robot is a PID controller with large gains: $\mathbf{u}_c = \mathbf{K}_p\mathbf{e} + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_I\int_{\tau=0}^{\tau=t}\mathbf{e}_\tau$. However, as already mentioned before, a pure PID controller may not be suitable to interact with humans or deformable objects, thus we needed to implement another controller.

*1) Feed-Forward Controller:* In order to properly track the reference trajectories generated by the DMPs, we include a Computed Torque Controller [1], in which we add an Inverse Dynamic Model (IDM), that allows us to use a low-gain error-compensating term, i.e.: we reduce the stiffness of the robot while performing the motion.

$\mathbf{u}_c = \mathbf{u}_{PD} + \mathbf{u}_{IDM}$ where $\mathbf{u}_{PD} = \mathbf{K}_p\mathbf{e}_p + \mathbf{K}_v\dot{\mathbf{e}}_p$ is a PD controller and the IDM is approximated by

$$\mathbf{u}_{IDM} \simeq \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}_{fric}, \tag{9}$$

where the Inertia, Coriolis, centripetal terms, as well as gravity were obtained with linear parameter identification (see Sec 7.2 in [2]). Using this controller, the robot will have a much more compliant, or *soft* behaviour, as the IDM would just provide the necessary torque to follow the desired commands, while the PD part of the controller takes care of error compensation. In fact, the PD torque only acts to compensate model errors and external perturbations.

## C. Force Estimation

In [4], the authors proposed an External Force Estimation based on a disturbance observer which, by using a previously learned dynamic model, estimates the external forces on the robot with the kinematics data and the control commands sent to the robot. This can be useful not only to predict interaction with the environment or detecting whether the robot is holding something, but also to try to minimize the interaction forces between the robot and its environment so as to reduce the stress on the manipulated objects.

## IV. Learning Framework

To learn a task, we use the scheme in Fig. 2, with a Policy Search [11] algorithm called Policy Improvement with Path Integrals (PI2) [5], [12], relying on a parametrization of the task with DMPs and three different feedbacks from the robot: desired acceleration, external force estimate, and visual scarf location.

## A. Policy Improvement with Path Integrals

The PI2 algorithm [5] is a Policy Search (PS) algorithm derived from stochastic optimal control principles that performs well in many situations. It executes several rollouts and gathers a timestep reward associated with each parameter variation for each rollout, and then updates the policy (in this case, the DMP weights) according to the rewards obtained. One of the advantages of the PI2 algorithm is that it only requires two extra parameters: the cost comparison eliteness $\lambda$, used to weight the Gaussian parameters according to the rollout cost, and the exploration variance. This exploration variance had been set manually by trial-and-error until the CMA [13] approach was presented, in which the exploration covariance matrix was updated after each epoch (set of rollouts). This approach helps to modify the exploration magnitudes in cases where one does not have a good initial guess, but it does not significantly improve performance when such initial guess is available. CMA needs more rollouts per policy update, and it is recommended to be used with a base-level exploration to avoid premature convergence or that the algorithm favours too much a single parameter direction if the eliteness parameter is too large.

## V. Experimentation

As an experimental setup for the scheme in Fig.2, we decided to put a scarf to a boy-sized mannequin, placed at a fixed position wrt. the robot, and use a color camera to check if the scarf was properly placed after each rollout. This is a very hard to learn problem, due to the scarf dynamics being very difficult to model and, thus, only real-robot experiments provide information to the task.

To that purpose, we initialized the DMP with 25 Gaussian kernels per degree-of-freedom, fitting a poor-performing motion which does not achieve such a task, and improve it over 20 epochs of 12 rollouts each. Using the CTC defined previously, which includes the friction model in Section II, we reproduced the motions and obtained the costs by evaluating the cost function. The PI2 algorithm, with the CMA and a dual layer of Gaussians as proposed in [4], with 8 kernels per degree-of-freedom, was then used to update the policy. The variance for exploration was initialized with a value $\Sigma_\Theta = 10 I_{8 \cdot dof}$ and updated with a filtered version of the CMA algorithm in [13], keeping only the diagonal part of the matrix for simplicity.

### A. *Cost function*

As a reward function to optimize, we use a timestep cost $c_t$, $t = 1..T$ and a final cost $c_T$, which depend on the acceleration, interaction torques and visual feedback.

The total cost for a trajectory is then $C_T = c_T + \sum_{t=1}^{T} c_t$, where $c_t$ is a timestep cost given by a quadratic form of the acceleration commands sent to the robot by the DMP, and the terminal cost $c_t = c_{cam} + c_{torque}$ is the sum of the costs given by the color camera and the EFE at the end of the rollout.

*1) Desired Acceleration:* To penalize those task attempts where the policy (DMP parameters) tells the robot to move with high acceleration, we added a penalizing term to the cost function, consisting of a quadratic form on the acceleration at each timestep of the trajectory.

*2) External Force Estimation:* In order to punish those motions in which the robot tries to push or pull the scarf and/or mannequin too much, we estimate the external torques resulting from the interaction of the robot and the scarf, which will also include the transfered torque from the scarf-mannequin interation to the robot at each timestep. For simplicity, we used the average of the absolute value of the estimated interaction torques all through the motion.

*3) Visual Feedback:* At the end of each rollout, we use a color camera to check if the scarf was properly placed by the robot. Using HSV color segmentation to distinguish the colors with the code provided in [14], such as the mannequin color, scarf, and a mark placed on its nose. As a descriptive element, we used the distance $d$ from the nose of the mannequin to the color-segmented scarf, in the vertical line from the nose. If the scarf is well-placed, this distance will be close to a reference value $d_{ref}$. Otherwise, the scarf might be hanging too close to the nose or not covering the neck. We defined the scarf position cost as $c_{cam} = 10 \max\left((d - d_{ref})/d_{ref}, 1\right)^\alpha + 3 I_{hanging}$, for a given exponent $\alpha$ (a value of 1.2 was used throughout this paper), and a penalizing factor with the indicator function $I_{hanging}$, which is 1 if the scarf is not hanging on the right side of the mannequin or it is hanging in its left side, assuming we want the scarf only to be hanging on the mannequin's right side (see Fig. 4). In this case, the reference distance $d_{ref}$ was 62 pixels for the color camera placed at a $1m$ distance from the mannequin.

This visual feedback cost is very rudimentary and was implemented only to demonstrate the performance of the entire system. Of course it can be replaced by most elaborate task-dependent cost measures.

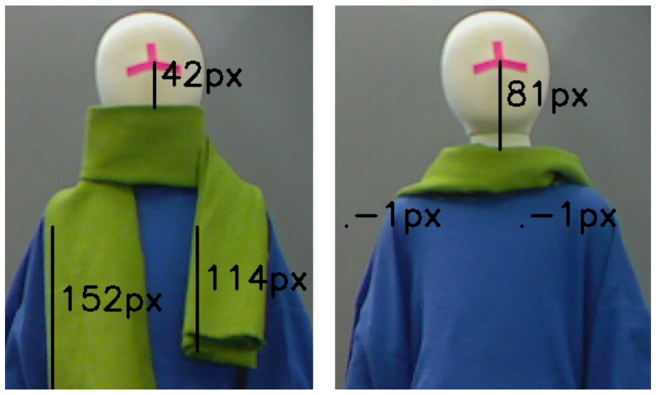

Fig. 4. Two examples of the camera output, measuring the distance (in pixels) from the marker to the hanging scarf, and the lenght of the part hanging on the sides up to a certain level.

### B. *Results*

In Fig. 6, we show the learning curve of the scarf-placing task, in which we can consider that a cost below 3 represents a motion that effectively placed the scarf around the mannequin's neck, hanging on the right side but not on the left side (see Fig. 5). We observe that, after 10 policy updates (i.e., 120 rollouts), the task has already been learned and, after that, the policy gradually reduces its variance, despite not being refined further, mainly due to all the uncertainties in the process of manipulating clothes. This experiment illustrates that certain difficult tasks can be achieved with the provided framework. Other similar experiments have been performed, also placing the scarf on a person instead of a mannequin, as shown in the video included as supplemental material and available at `http://www.iri.upc.edu/groups/perception/#ScarfTask`.

## VI. Conclusion

In this paper, we provided a simple but efficient way of going all through the process of learning a robotic task with a real robot, using a compliant controller that ensures human safety in physical interaction tasks involving deformable objects. To that purpose, we developed a new friction model

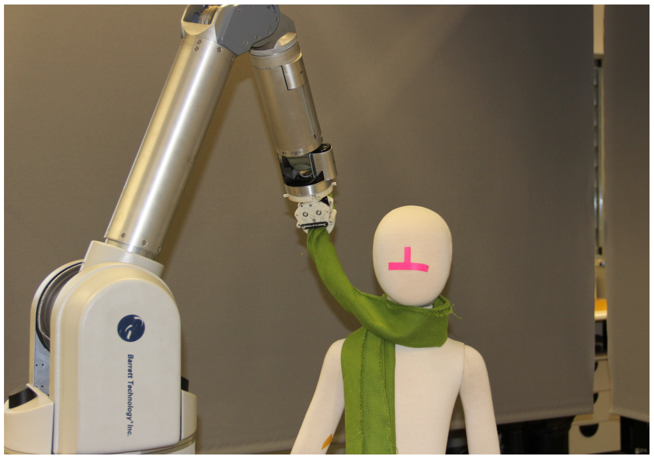

Fig. 5. WAM after placing the scarf around the mannequin's neck.

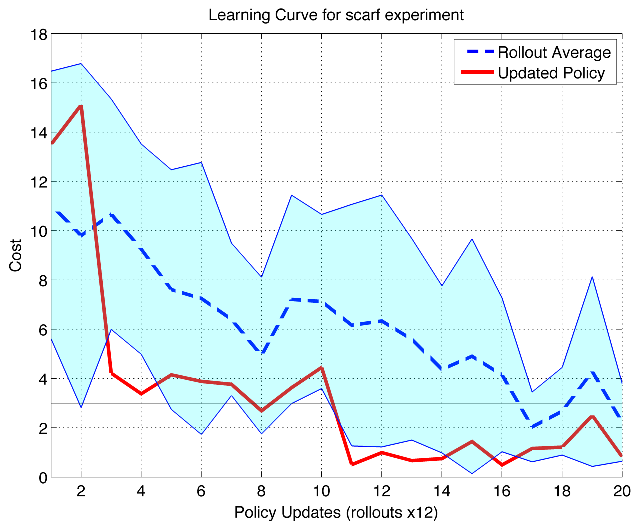

Fig. 6. Learning curve for the scarf experiment. The red continuous line shows the exploration-free policy cost after each update, while the blue shaded area shows the mean and standard deviation for all the exploration rollouts at each epoch. The horizontal black line represents the cost value of 3, which is considered by the authors to be the threshold indicating whether the task was successfully completed or not.

that is well suited for the case of the Barrett WAM robot and does not require as many samples as other IDM fitting aproaches [3]. The IDM resulting from that friction model proved to be precise enough to compliantly but precisely track reference commands with a CTC. Such controller was a key element to learn this kind of tasks, as a high-gain controller may likely harm the mannequin or a person interacting with the robot.

Given the model-based controller, we represented trajectories with DMP, which were demonstrated to the robot by kinesthetic teaching and encoded, to later be used within an RL algorithm, where we separated the initial fitting from the exploration layer of Gaussians in order to overcome the possible curse of dimensionality. We also implemented a PI2 algorithm with CMA, which helped to automatically update the variance of the DMP parameters for exploration.

Finally, we used a combination of kinematic, dynamic and visual feedback within a cost function in order to take all possible factors into account: Accelerations telling how smooth the desired trajectory was, estimated contact torques evaluating if there was a too hard interaction with the mannequin, probably due to the robot hitting or wrongly pushing it, and also a simple color-based segmentation that efficiently measured whether the robot was successful at the performed task.

The difficulties of simulating deformable objects and human environment force RL algorithms to require all these elements to be successful but safe while learning. In this work, we provided an initial framework to safely learn tasks involving deformable objects in close proximity to humans.

As future work, we plan to include a timestep torque feedback in the cost function, as well as try to use other RL algorithms within this framework to accomplish diverse complex tasks.

## ACKNOWLEDGEMENTS

We would like to thank Felip Martí for his help in the implementation of the visual detection part in the experiments.

## REFERENCES

[1] Duy Nguyen-Tuong, M. Seeger and J. Peters. "Computed torque control with nonparametric regression models", *American Control Conference*, pp. 212-217, 2008.
[2] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo. "Robotics. Modelling, Planning and Control", Advanced Textbooks in Control and Signal Processing, Springer-Verlag, 2009.
[3] D. Nguyen-Tuong and Jan Peters. "Model Learning for Robot Control: A Survey", *Cognitive Processing*, pp. 1-22, 2011.
[4] A. Colomé, D. Pardo, G. Alenyà and C. Torras. "External force estimation during compliant robot manipulation", *2013 IEEE Int. Conference on Robotics and Automation*, pp. 3535-3540, 2013.
[5] E. Theodorou, J. Buchli and S. Schaal. "A Generalized Path Integral Control Approach to Reinforcement Learning", *Journal of Machine Learning Research*, vol. 11, pp. 3137-3181, 2010.
[6] W.T. Townsend, J.K. Salisbury, P. Dario, G. Sandini and P. Aebischer. "Mechanical Design for Whole-Arm Manipulation",Robots and Biological Systems: Towards a New Bionics?".*NATO ASI Series, Springer*, Berlin Heidelberg, pp. 153-164, 1993.
[7] D. Mitrovic, S. Nagashima, S. Klanke, T. Matsubara and S. Vijayakumar. "Optimal Feedback Control for Anthropomorphic Manipulators", *IEEE Int. Conference on Robotics and Automation*, pp. 4143 - 4150, 2010.
[8] A. J. Ijspeert, J. Nakanishi and S. Schaal. "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots", *IEEE Int. Conference on Robotics and Automation*, pp. 1398-1403, 2002.
[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal. "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviours", *Neural Computation*, vol. 25, no. 2, pp. 328-373, 2013.
[10] A. Colomé and C. Torras. "Dimensionality Reduction and Motion Coordination in Learning Trajectories with Dynamic Movement Primitives", *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 1414-1420, 2014.
[11] M. Deisenroth, G. Neumann and J. Peters. "A Survey on Policy Search for Robotics", *Foundations and trends in Robotics*, vol. 2, no. 1-2. pp. 1-142, 2011.
[12] F. Stulp, E. Theodorou and S. Schaal. "Reinforcement Learning with Sequences of Motion Primitives for Robust Manipulation", *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1360-1370, 2012.
[13] F. Stulp and O. Sigaud. "Path Integral Policy Improvement with Covariance Matrix Adaptation", *Int. Conf. on Machine Learning*, pp. 281-288, 2012.
[14] C++ Library to check if a scarf is well placed on a mannequin. Available online at https://github.com/FelipMarti .