Robot Motor Skill Coordination with EM-based Reinforcement Learning

Petar Kormushev, Sylvain Calinon and Darwin G. Caldwell

***Abstract*— We present an approach allowing a robot to acquire new motor skills by learning the couplings across motor control variables. The demonstrated skill is first encoded in a compact form through a modified version of Dynamic Movement Primitives (DMP) which encapsulates correlation information. Expectation-Maximization based Reinforcement Learning is then used to modulate the mixture of dynamical systems initialized from the user's demonstration. The approach is evaluated on a torque-controlled 7 DOFs Barrett WAM robotic arm. Two skill learning experiments are conducted: a reaching task where the robot needs to adapt the learned movement to avoid an obstacle, and a dynamic pancake-flipping task.**

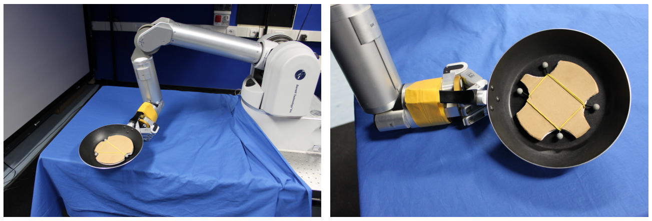


Fig. 1. Experimental setup for the *Pancake-Flipping* task. A torque-controlled 7-DOF Barrett WAM robot learns to flip pancakes in the air and catch them with a real frying pan attached to its end-effector. Artificial pancakes with passive reflective markers are used to evaluate the performance of the learned policy.

I. INTRODUCTION

Acquiring new motor skills involves various forms of learning. The efficiency of the process lies in the interconnections between imitation and self-improvement strategies. Similarly to humans, a robot should be able to acquire new skills by employing such mechanisms.

Some tasks can be successfully transferred to the robot using only imitation strategies [1], [2]. Other tasks can be learned very efficiently by the robot alone using Reinforcement Learning (RL) [3]. The recent development of compliant robots progressively moves the robots from industrial applications to home and office uses, where the role and tasks given to the robots cannot be determined in advance. While some tasks allow the user to interact with the robot to teach it new skills, it is preferable to provide a mechanism for the robot to improve and extend its skills to new contexts on its own.

A tremendous effort has been brought by researchers in machine learning and robotics to move RL algorithms from discrete to continuous domains, thus extending the possibilities for robotic applications [4]–[7]. Until recently, policy gradient algorithms (such as Episodic REINFORCE [8] and Episodic Natural Actor-Critic eNAC [9]) have been a well-established approach to cope with the high dimensionality. Unfortunately, they also have shortcomings, such as high sensitivity to the learning rate. Trying to overcome this drawback, the following two recent approaches were proposed.

Theodorou *et al* proposed in [5] a RL approach for learning parameterized control policies based on the framework of stochastic optimal control with path integrals. They derived update equations for learning which avoid numerical instabilities because neither matrix inversions nor gradient learning rates are required. The approach demonstrates significant performance improvements over gradient-based policy learning and scalability to high-dimensional control problems, such as control of a quadruped robot dog.

Kober *et al* proposed in [10] an episodic RL algorithm called *Policy learning by Weighting Exploration with the Returns* (PoWER), which is based on *Expectation-Maximization* algorithm (EM). One major advantage over policy-gradient-based approaches is that it does not require a learning rate parameter. This is desirable because tuning a learning rate is usually difficult to do for control problems but critical for achieving good performance of policy-gradient algorithms. PoWER also demonstrated high performance in tasks learned directly on real robots, such as underactuated pendulum swing-up and ball-in-a-cup tasks [11].

In order to reduce the number of trials required to learn a skill in a real robot learning scenario, another body of work explored the use of efficient representations of the skill that can be applied to RL. Guenter *et al* explored in [7] the use of *Gaussian Mixture Model* (GMM) and *Gaussian Mixture Regression* (GMR) to respectively encode compactly a skill and reproduce a generalized version of it. The model was initially learned by demonstration through *Expectation-Maximization* techniques. RL was then used to move the Gaussian centers in order to alter the reproduced trajectory by regression. It was successfully applied to the imitation of constrained reaching movements, where the learned movement was refined in simulation to avoid an obstacle that was not present during the demonstration attempts.

Kober and Peters explored in [12] the use of *Dynamic Movement Primitives* (DMP) [13] as a compact representation of a movement. In DMP, a set of attractors is used to reach a target, whose influence is smoothly switched

The authors are with the Advanced Robotics Department, Italian Institute of Technology (IIT), 16163 Genova, Italy. {petar.kormushev,sylvain.calinon, darwin.caldwell}@iit.it.

along the movement.[1] The set of attractors is first learned by imitation, and a proportional-derivative controller is used to move sequentially towards the sequence of targets. RL is then used to explore the effect of changing the position of these attractors. The proposed approach was demonstrated with pendulum swing-up and ball-in-a-cup tasks.

Pardo *et al* proposed in [14], [15] a framework to learn coordination for simple rest-to-rest movements, by taking inspiration of the motor coordination, joint synergies, and the importance of coupling in motor control [16]–[19]. The authors suggested to start from a basic representation of the movement by considering point-to-point movements driven by a proportional-derivative controller, where each variable encoding the task is decoupled. They then extended the possibilities of movement by encapsulating coordination information in the representation. RL was then used to learn how to efficiently coordinate the set of variables which were originally decoupled. They showed simulation experiments in which a humanoid learns how to stand up by coordinating its joint angles to avoid falling during the rest-to-rest motion, and in which a robot learns how to throw a ball to a desired target.

Rosenstein *et al* proposed in [20] a robot weightlifting experiment, where an appropriate coordination of the joints exploiting the robot's intrinsic dynamics is searched through RL. Their work highlight the advantage of considering off-diagonal elements in gain matrices to enable active coupling of the individual joints. They showed through experiments that skillful movements that exploit dynamics are best acquired by first learning (or specifying) simple kinematic movement,[2] and then using practice to transform that movement into dynamic solution with tighter coupling from the control system.

We propose here to build upon the works above by taking into consideration the efficiency of DMP to encode a skill with a reduced number of states, and by extending the approach to take into consideration local coupling information across the different variables.

II. PROPOSED APPROACH

The proposed approach represents a movement as a superposition of basis force fields, where the model is initialized from imitation. RL is then used to adapt and improve the encoded skill by learning optimal values for the policy parameters. The proposed policy parameterization allows the RL algorithm to learn the coupling across the different motor control variables.

A. Encoding of the skill

A demonstration consisting of T positions x, velocities $\dot{x}$ and accelerations $\ddot{x}$ is shown to the robot (x has $D = 3$ dimensions). By considering flexibility and compactness issues, we propose to use a controller based on a mixture of K proportional-derivative systems (see also [21])

$$\hat{\ddot{x}} = \sum_{i=1}^{K} h_i(t) \Big[K_i^{\mathcal{P}} (\mu_i^{\mathcal{X}} - x) - \kappa^{\mathcal{V}} \dot{x} \Big]. \qquad (1)$$

The above formulation shares similarities with the *Dynamic Movement Primitives* (DMP) framework originally proposed by Ijspeert *et al* [22], and further extended in [13], [23] (see [24] for a discussion on the similarities of the proposed controller with DMP). We extend here the use of DMP by considering synergy across the different motion variables through the association of a full matrix $K_i^{\mathcal{P}}$ with each of the K primitives (or states) instead of a fixed $\kappa^{\mathcal{P}}$ gain.

The superposition of basis force fields is determined in (1) by an implicit time dependency, but other approaches using spatial and/or sequential information could also be used [25], [26]. Similarly to DMP, a decay term defined by a canonical system $\dot{s} = -\alpha s$ is used to create an implicit time dependency $t = -\frac{ln(s)}{\alpha}$, where s is initialized with $s = 1$ and converges to zero. We define a set of Gaussians $\mathcal{N}(\mu_i^{\mathcal{T}}, \Sigma_i^{\mathcal{T}})$ in time space $\mathcal{T}$, with centers $\mu_i^{\mathcal{T}}$ equally distributed in time, and variance parameters $\Sigma_i^{\mathcal{T}}$ set to a constant value inversely proportional to the number of states. α is fixed depending on the duration of the demonstrations. The weights are defined by $h_i(t) = \frac{\mathcal{N}(t;\, \mu_i^{\mathcal{T}}, \Sigma_i^{\mathcal{T}})}{\sum_{k=1}^{K} \mathcal{N}(t;\, \mu_k^{\mathcal{T}}, \Sigma_k^{\mathcal{T}})}$.

In (1), $\{K_i^{\mathcal{P}}\}_{i=1}^{K}$ is a set of full stiffness matrices, also called coordination matrices in [15] (in this paper we use the term *coordination matrix* rather than *stiffness matrix*). Using the full coordination matrices allows us to consider different types of synergies across the variables, where each state/primitive encodes local correlation information. Both attractor vectors $\{\mu_i^{\mathcal{X}}\}_{i=1}^{K}$ and coordination matrices $\{K_i^{\mathcal{P}}\}_{i=1}^{K}$ in Eq. (1) are initialized from the observed data through least-squares regression (see [21] for details).

B. Controller

To control the robot, we exploit the torque-feedback properties of the manipulator, where the robot remains actively compliant for the degrees of freedom that are not relevant for the task. We control the 7 degrees of freedom (DOFs) robot through inverse dynamics solved with recursive Newton Euler algorithm [27]. The joint forces f_i at each joint $i \in \{1, \ldots, 7\}$ are calculated as $f_i = f_i^a - f_i^e + \sum_{j \in c(i)} f_j$, where f_i^a is the net force acting on link i, f_j with $j \in c(i)$ are the forces transmitted by the child $c(i)$ of link i. $f_i^e = F_T + F_G$ are the external forces, where $F_T = [f_T, M_T]^\top \in \mathbb{R}^6$ is the vector of force and momentum requested to accomplish the task (only applied at the end-effector, i.e. when $i = 7$), and $F_G = [f_G, 0]^\top \in \mathbb{R}^6$ is the gravity compensation force. Tracking of a desired path in Cartesian space is insured by the force command $f_T = m_T \hat{\ddot{x}}$, where m_T is a virtual mass and $\hat{\ddot{x}}$ is the desired acceleration command defined in (1).

C. Reinforcement Learning

To learn new values for the coordination matrices, we use the state-of-the-art EM-based RL algorithm called PoWER

[1] This approach can similarly be interpreted as a force disturbing a point-to-point reaching movement.

[2] In contrast to DMP, their approach only considers hard-switching among the different sub-controllers.

developed by Kober and Peters [10]. PoWER inherits from EM algorithm two major advantages over policy-gradient-based approaches: firstly, PoWER does not need a learning rate, unlike policy-gradient methods; secondly, PoWER can be combined with importance sampling to make better use of the previous experience of the agent in the estimation of new exploratory parameters.

Similar to policy gradient RL, PoWER uses a parameterized policy and tries to find values for the parameters which maximize the expected return of rollouts (also called episodes or trials) under the corresponding policy. In our approach the policy parameters are represented by the elements of the full coordination matrices $K_i^{\mathcal{P}}$ and the attractor vectors $\mu_i^{\mathcal{X}}$.[3]

The return of a rollout τ is given by the undiscounted cumulative reward $R(\tau) = \sum_{t=1}^{T} r(t)$, where T is the duration of the rollout, and $r(t)$ is the reward received at time t, defined differently according to the goal of the task.

In general, as an instance of an EM algorithm, PoWER estimates the policy parameters θ such as to maximize the lower bound on the expected return from following the policy. The policy parameters θ_n at the current iteration n are updated to produce the new parameters θ_{n+1} using the following rule (see also [11])

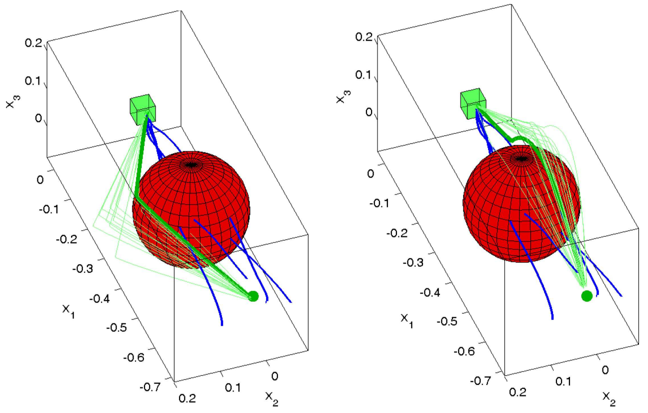
$$\theta_{n+1} = \theta_n + \frac{\Big\langle (\theta_k - \theta_n) R(\tau_k) \Big\rangle_{w(\tau_k)}}{\Big\langle R(\tau_k) \Big\rangle_{w(\tau_k)}}. \qquad (2)$$

In the above equation, $(\theta_k - \theta_n) = \Delta\theta_{k,n}$ is a vector difference which gives the relative exploration between the policy parameters used in the k-th rollout and the current ones. Each relative exploration $\Delta\theta_{k,n}$ is weighted by the corresponding return $R(\tau_k)$ of rollout τ_k, and the result is normalized using the sum of the same returns.[4]

In order to minimize the number of rollouts which are needed to estimate new policy parameters, we use a form of importance sampling technique adapted for RL [3], [10] and denoted by $\langle \cdot \rangle_{w(\tau_k)}$ in Eq. (2). It allows the RL algorithm to re-use previous rollouts τ_k and their corresponding policy parameters θ_k during the estimation of the new policy parameters θ_{n+1}. The importance sampler we use is defined as

$$\Big\langle f(\theta_k, \tau_k) \Big\rangle_{w(\tau_k)} = \sum_{k=1}^{\sigma} f(\theta_{\mathrm{ind}(k)}, \tau_{\mathrm{ind}(k)}), \qquad (3)$$

where σ is a fixed parameter denoting the number of rollouts used by the importance sampler, and $\mathrm{ind}(k)$ is an index function which returns the index of the k-th best rollout in the list of all past rollouts sorted by their corresponding returns, i.e. for $k = 1$ we have $\mathrm{ind}(1) = \mathrm{argmax}_i\, R(\tau_i)$, and $R(\tau_{\mathrm{ind}(1)}) \geq R(\tau_{\mathrm{ind}(2)}) \geq \ldots \geq R(\tau_{\mathrm{ind}(\sigma)})$. The effect of the importance sampler is significant because it allows the RL algorithm to re-use the top-σ best rollouts so far in order to calculate the new policy parameters. This helps to reduce the number of required rollouts and makes the algorithm applicable to online learning, which we demonstrate with the *Pancake-Flipping* task described in Section III-B.



(a) Using only the main diagonal of $K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$

(b) Using the full coordination matrices $K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$

Fig. 2. Simulation of a *Reaching task* with obstacle avoidance, using two primitives to represent the trajectories. In (a), only the diagonal values of $K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$ are used, and in (b), the full matrices $K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$ are used as parameters to be optimized by the RL algorithm. In the figures, the red sphere represents the obstacle, the green box is the target for the reaching task, the 4 blue lines are the demonstrations recorded on the real robot, and the green dot is the starting position of the end-effector for all rollouts. Some of the rollout trajectories generated during the RL process are shown with thin green lines. The final learned trajectory is shown with thick dark green line.

III. EXPERIMENTS

The proposed method is evaluated on two experiments: *Reaching task* with obstacle avoidance learned in simulation using data from real-world demonstrations, and *Pancake-Flipping* task performed both in simulation and on a real physical robot.

A. Reaching task with obstacle avoidance

The goal of the *Reaching task* is for the robot to reach with its end-effector towards a target, while at the same time trying to avoid collision with a fixed obstacle.

1) Experimental setup: The demonstrations needed for the initialization with imitation learning were recorded on a gravity-compensated robot via kinesthetic teaching, i.e. a human demonstrator is holding the arm of the robot and manually guiding the robot to execute the task. These demonstrations were done without any obstacle. The recorded trajectories are then represented with the model described in Sec. II-A. The model is initialized with least squares regression (see [21] for details), and is later altered by the RL algorithm in order to avoid a newly appeared obstacle between the robot's end-effector and the target for the reaching task.

To better emphasize the differences between the proposed method of using the full coordination matrices $K_i^{\mathcal{P}}$ and attractor points $\mu_i^{\mathcal{X}}$, and using only the main diagonal of

[3] Note that the magnitudes of the values in $K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$ are different, which is taken into account when determining the exploration variance for the policy parameters.

[4] Intuitively, this update rule can be thought of as a weighted sum of parameter vectors where higher weight is given to the vectors which result in higher returns.

$K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$, we deliberately use a low number of primitives (or states). To avoid a simple spherical obstacle, it is in this case possible to use only two primitives.

For each rollout τ_k the end-effector trajectory is simulated for a fixed number of steps $T = 200$. In case a collision is detected, the end-effector remains still until the end of the current episode. The reward function $r(t)$ is defined based on two criteria: to reach the goal and to stay as close as possible to the original demonstrations

$$r(t) = \begin{cases} \frac{w_1}{T}\, e^{-||x_t^R - x_t^D||}, t \neq t_e \\ w_2\, e^{-||x_t^R - x^G||}, t = t_e \end{cases} , \tag{4}$$

where t_e is the end of the rollout, $x_t^R \in \mathbb{R}^3$ is the position of the robot's end-effector at time t, $x_t^D \in \mathbb{R}^3$ is the initial demonstrated position at time t, x^G is the position of the target, and $|| \cdot ||$ is Euclidean distance. The first term is maximized when the rollout trajectory matches the demonstrated trajectory. The second term is maximized when the goal is reached at the end of the task. The weights used are $w_1 = 0.5$ and $w_2 = 0.5$.

2) Experimental results: A visualization of the *Reaching task* with obstacle avoidance is shown in Fig. 2. Using only the main diagonal of $K_i^{\mathcal{P}}$ and $\mu_i^{\mathcal{X}}$ with two primitives can produce only trajectories which have at most one turn along them, as shown in Fig. 2(a). On the other hand, the proposed method of using the full coordination matrices $K_i^{\mathcal{P}}$ and attractor points $\mu_i^{\mathcal{X}}$ is capable of producing more complex trajectories, as shown in Fig. 2(b). The learned trajectory has two turns and smoother curvature around the obstacle, which allows it to be closer to the demonstrated trajectories without colliding with the obstacle. Using the proposed method, the expected return after 100 rollouts (averaged over 10 experiments) is increased from 0.61 to 0.73.

B. Pancake-Flipping task

The real-world evaluation of the presented method is done on a dynamic *Pancake-Flipping* task. The goal of the task is to toss a pancake in the air so that it rotates 180 degrees before being caught. Due to the complex dynamics of the task, it is unfeasible to try to learn it directly using *tabula rasa* RL. Instead, a person presents a demonstration of the task first, which is then used to initialize the policy.

1) Experimental setup: The experimental setup is shown in Fig. 1. The experiment is conducted with a torque-controlled *Barrett WAM* 7 DOFs robotic arm. Using a gravity-compensation controller, the *Pancake-Flipping* task is first demonstrated via kinesthetic teaching. The number of states is fixed at 8, which is determined empirically by examining the quality of the initial reproduced trajectories with different number of states.

Custom-made artificial pancakes are used, which have 4 highly-reflective passive markers, in order to track both the position and the orientation of the pancakes during the task execution (See Fig. 1). For easier visual inspection, the two sides of the pancakes are colored in different colors - white and yellow. The pancake weights only 26 grams, which makes it susceptible to air flow influence and makes its motion less predictable.

The pancake's position and orientation are tracked by a marker-based *NaturalPoint OptiTrack* motion capture system with 12 cameras. It tracks the position x^p and orientation (q^p in quaternion representation, M^p in direction cosine matrix representation) of the pancake at a rate of 30 frames per second.

The return of a rollout τ is calculated from the timestep reward $r(t)$. It is defined as a weighted sum of two criteria (orientational reward and positional reward), which encourage successful flipping and successful catching of the pancake

$$r(t_f) = w_1 \Big[\frac{\arccos(v_0 . v_{t_f})}{\pi} \Big] + w_2 e^{-||x^p - x^F||} + w_3 x_3^M, \tag{5}$$

where t_f is the moment when the pancake passes with downward direction the horizontal level at a fixed height Δ_h above the frying pan's current vertical position, v_0 is the initial orientation vector of the pancake (unit vector perpendicular to the pancake), v_{t_f} is the orientation vector of the pancake at time t_f, x^P is the position of the pancake at time t_f, x^F is the position of the frying pan at time t_f, and x_3^M is the maximum reached altitude of the pancake. The first term is maximized when the pancake's orientation vector at time t_f goes in the opposite direction to the initial orientation vector, which corresponds to a successful flip. The second term is maximized when the pancake lands in the center of the frying pan. The weights we use are $w_1 = w_2 = w_3 = 0.5$. For all other time steps $t \neq t_f$ we define $r(t) = 0$.

The learning process is based on the PoWER algorithm implementation provided by Kober *et al* [11]. $\sigma = 6$ is used as parameter for the importance sampler. The parameters θ_n for the RL algorithm are composed of two sets of variables: the first set contains the full 3×3 coordination matrices $K_i^{\mathcal{P}}$ with the positional error gains in the main diagonal and the coordination gains in the off-diagonal elements; the second set contains the vectors $\mu_i^{\mathcal{X}}$ with the attractor positions for the primitives. The RL algorithm is stopped when a successful and reproducible pancake flipping is achieved with return $R(\tau) \geq 0.9$.

2) Experimental results: At each iteration of the RL loop, the trajectory generated by the current policy is transferred to the real robot for execution. While the rollout is performed, the trajectory of the pancake is being recorded by the motion capture system, and the trajectory of the end-effector (obtained through forward kinematics) is recorded by the robot controller. At the end of the rollout, the two trajectories are transferred back to the RL algorithm. The rollout is then evaluated using Eq. (5) and the data from the two recorded trajectories. Using the update rule in Eq. (2), new values for the policy parameters θ_{n+1} are estimated, taking into account previous experience via the importance sampler. Then, a new trajectory for the end-effector is generated and the loop starts over. Fig. 3 shows one sample rollout performed during the online RL phase, during which the pancake rotated only

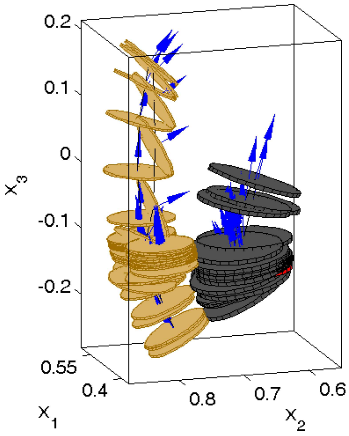


Fig. 3. Semi-successful real-world pancake flipping rollout performed on the robot. The pancake (in yellow) was successfully tossed and caught with the frying pan (in grey), but it only rotated 90 degrees (for better visibility of the pancake's trajectory, the frying pan's trajectory has been shifted to the right). The calculated return of the rollout is 0.7. The normal vectors perpendicular to the pancake and the frying pan are shown with blue arrows.

90 degrees before falling on the frying pan. The estimated return of this rollout was 0.7, because the positional reward was high in this case. Fig. 4 shows another rollout from the online RL phase, during which the pancake rotated fully 180 degrees and was caught successfully with the frying pan. The estimated return of this rollout was 0.9. Fig. 5 shows the average expected return over rollouts.

A video of the *Pancake-Flipping* experiment accompanies the submission (see also Fig. 6). It is interesting to notice the up-down bouncing of the frying pan towards the end of the learned skill, when the pancake has just fallen inside of it. The bouncing behavior is due to the increased compliance of the robot during this part of the movement. This was produced by the RL algorithm in an attempt to catch the fallen pancake inside the frying pan. Without it, a controller being too stiff would let the pancake bounce off from the surface of the frying pan and fall out of it. Such undesigned discoveries made by the RL algorithm highlight its important role for achieving adaptable and flexible robots.

IV. DISCUSSION

The *Pancake-Flipping* task is difficult to learn from multiple demonstrations because of the high variability of the task execution, even when the same person is providing the demonstrations. Extracting the task constraints by observing multiple demonstrations, as in [21], is not appropriate in this case for two reasons: (1) when considering such skillful movements, extracting the regularities and correlations from multiple observations would be too long, as consistency in the skill execution would appear only after the user has mastered the skill; and (2) the generalization process may smooth important acceleration peaks and sharp turns in the motion. Therefore, in such highly dynamic skilful tasks, early trials have shown that it was more appropriate to select a single successful demonstration (among a small series of trials) to initialize the learning process.

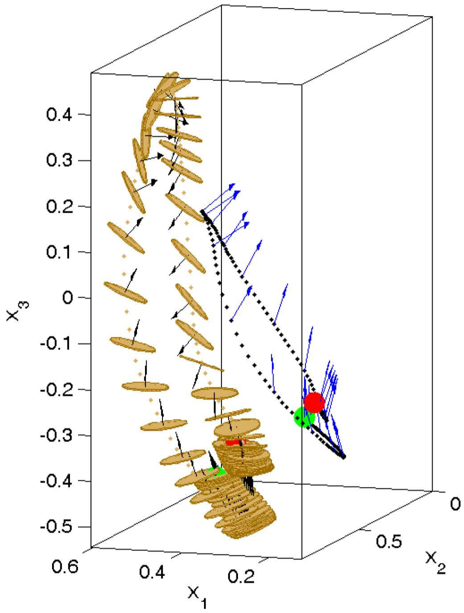


Fig. 4. Successful real-world pancake flipping rollout performed on the robot. The pancake (in yellow) was successfully tossed and caught with the frying pan, and it rotated 180 degrees (for better visibility of the pancake's trajectory, the frying pan is not displayed here). The calculated return of the rollout is 0.9. The trajectory of the end-effector is displayed with black dots, and its orientation with blue arrows. The normal vectors perpendicular to the pancake are shown with black arrows.

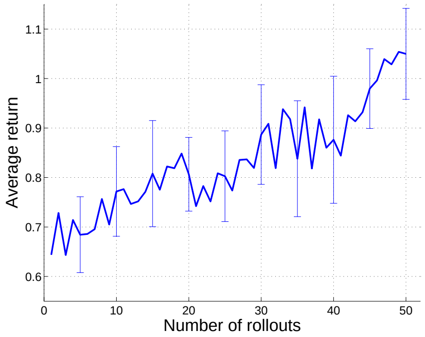


Fig. 5. This figure shows the expected return of the learned policy for pancake flipping averaged over 6 sessions with 50 rollouts in each session.

The importance sampling technique proved to be extremely helpful for the online learning because it re-uses efficiently previous rollouts. In practice, less than 100 rollouts were necessary to find a good solution for the *Pancake-Flipping* task. Importance sampling is also computationally efficient. It needs only linear $O(n)$ memory to store the return $R(\tau_k)$ and the RL parameters θ_k for all rollouts, without need to store the whole rollout trajectories. The value of the σ parameter for the importance sampler was set manually, but a possible future extension would be to have an automatic mechanism to select σ or even dynamically change it during the learning.

In the experiments presented here, imitation learning is used as an initialization phase, and afterwards RL is used to explore for better solutions. Both processes could, however, be interlaced. Depending on his/her availability, the user can,

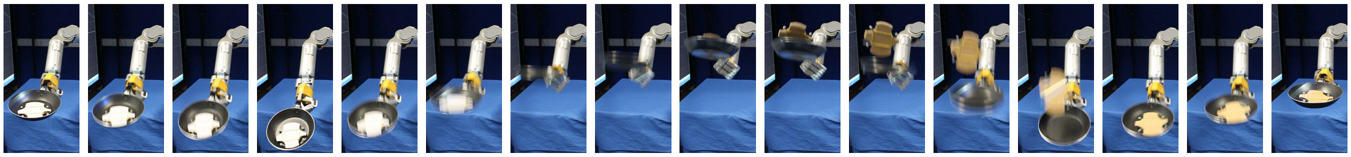


Fig. 6. Sequence of video frames showing a successful pancake flipping (after 50 rollouts), performed on the WAM robot.

for example, occasionally participate in the evaluation of new policies explored by the robot. For example, the user can manually give reward or punishment signals to the RL module. He/she can also provide new examples in case the robot's improvement is too slow, or if the robot is looking for inappropriate solutions. We plan to consider such interaction in future work.

V. CONCLUSION

We have presented an approach based on a mixture of dynamical systems for learning the couplings across multiple motor control variables. An extension of Dynamic Movement Primitive encapsulating synergy information is used to compactly encode a demonstrated skill, where Reinforcement Learning is used to refine the coordination matrices and attractor vectors associated with the set of primitives. The paper provides a mechanism to learn the local coupling information across the different variables. It highlights the advantages of considering probabilistic approaches in RL, and of applying importance sampling to reduce the number of rollouts required to learn the skill. The proposed method was successfully implemented in two experiments. A *Reaching task* experiment with obstacle avoidance illustrates the advantages of using the full coordination matrices to learn a skill with minimum number of states. A *Pancake-Flipping* experiment that demonstrates the fitness of the proposed approach to cope with real-world highly-dynamic tasks.

REFERENCES

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Secaucus, NJ, USA: Springer, 2008, pp. 1371–1394.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*, ser. Adaptive computation and machine learning. Cambridge, MA, USA: MIT Press, 1998.

[4] J.Peters and S.Schaal, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[5] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: a path integral approach," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2010.

[6] A. Coates, P. Abbeel, and A. Y. Ng, "Apprenticeship learning for helicopter control," *Commun. ACM*, vol. 52, no. 7, pp. 97–105, 2009.

[7] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.

[8] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 229–256, 1992.

[9] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomput.*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[10] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, May 2009, pp. 2112–2118.

[11] J. Kober, "Reinforcement learning for motor primitives," Master's thesis, University of Stuttgart, Germany, August 2008.

[12] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems*, 2009, vol. 21, pp. 849–856.

[13] S. Schaal, P. Mohajerian, and A. J. Ijspeert, "Dynamics systems vs. optimal control a unifying view," *Progress in Brain Research*, vol. 165, pp. 425–445, 2007.

[14] D. Pardo, "Learning rest-to-rest motor coordination in articulated mobile robots," PhD thesis, Technical University of Catalonia (UPC), 2009.

[15] D. E. Pardo and C. Angulo, "Collaborative control in a humanoid dynamic task," in *Proc. Intl Conf. on Informatics in Control, Automation and Robotics, Robotics and Automation (ICINCO)*, Angers, France, May 2007, pp. 174–180.

[16] T. Flash and N. Hogan, "The coordination of the arm movements: an experimentally confirmed mathematical model," *Neurology*, vol. 5, no. 7, pp. 1688–1703, 1985.

[17] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature Neuroscience*, vol. 5, pp. 1226–1235, 2002.

[18] R. Huys, A. Daffertshofer, and P. J. Beek, "The evolution of coordination during skill acquisition: the dynamical systems approach," in *Skill Acquisition in Sport: Research, Theory and Practice*, A. M. Williams and N. J. Hodges, Eds. Routledge, 2004, pp. 351–373.

[19] M. Bernikera, A. Jarcb, E. Bizzic, and M. C. Trescha, "Simplified and effective motor control based on muscle synergies to exploit musculoskeletal dynamics," in *Proc. Natl Acad. Sci. USA*, vol. 106, no. 18, 2009, pp. 7601–7606.

[20] M. T. Rosenstein, A. G. Barto, and R. E. A. Van Emmerik, "Learning at the level of synergies for a robot weightlifter," *Robotics and Autonomous Systems*, vol. 54, no. 8, pp. 706–717, 2006.

[21] S. Calinon, I. Sardellitti, and D. G. Caldwell, "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[22] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Proc. IEEE Intl Conf. on Intelligent Robots and Systems (IROS)*, 2001, pp. 752–757.

[23] H. Hoffmann, P. Pastor, D. H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2009, pp. 2587–2592.

[24] S. Calinon, F. D'halluin, D. G. Caldwell, and A. G. Billard, "Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework," in *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, Paris, France, December 2009, pp. 582–588.

[25] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[26] M. Khansari and A. G. Billard, "BM: An iterative method to learn stable non-linear dynamical systems with Gaussian mixture models," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, May 2010, pp. 2381–2388.

[27] R. Featherstone and D. E. Orin, "Dynamics," in *Handbook of Robotics*, B. Siciliano and O. O. Khatib, Eds. Secaucus, NJ, USA: Springer, 2008, pp. 35–65.