Exploiting Symmetries in Reinforcement Learning of Bimanual Robotic Tasks

Fabio Amadio[1], Adrià Colomé[2] and Carme Torras[2]

***Abstract*— Movement Primitives (MPs) have been widely adopted for representing and learning robotic movements using reinforcement learning policy search. Probabilistic Movement Primitives (ProMPs) are a kind of MP based on a stochastic representation over sets of trajectories, able to capture the variability allowed while executing a movement. This approach has proved effective in learning a wide range of robotic movements, but it comes with the necessity of dealing with a high-dimensional space of parameters. This may be a critical problem when learning tasks with two robotic manipulators, and this work proposes an approach to reduce the dimension of the parameter space based on the exploitation of symmetry. A symmetrization method for ProMPs is presented and used to represent two movements, employing a single ProMP for the first arm and a symmetry surface that maps that ProMP to the second arm. This symmetric representation is then adopted in reinforcement learning of bimanual tasks (from user-provided demonstrations), using Relative Entropy Policy Search (REPS) algorithm. The symmetry-based approach developed has been tested in an experiment of cloth manipulation, showing a speed increment in learning the task.**

I. Introduction

The state-of-the-art method in Reinforcement Learning (RL) for robotic movements is Policy Search (PS) [1], in which a set of parameters defining the motion policy is learned from robot executions, in order to optimize a reward/cost function. The starting policy can be obtained from some demonstrations in which the user kinesthetically guides the robot to perform the task (Fig. 1), under a Learning from Demonstration (LfD) approach [2]. The policy can generate sample trajectories, exploring the space of parameters. Those trajectories are executed and evaluated in order to update the policy through PS. The procedure is repeated until convergence. Movement Primitives (MPs) [3] are a well-established approach for representing basic movements in robotics. They provide a compact representation of the robot's policy that is able to deal with the inherently continuous robot movements. Probabilistic Movement Primitives [4] (ProMPs) are further capable of capturing and reproducing the variance along time of a set of demonstrations. When dealing with bimanual tasks the high complexity of the movement brings the necessity to simplify its representation. For example, in [5] a user-demonstrated bimanual task is reproduced characterizing it through a set of constraints, while in [6] joint synergies are extracted through group factor analysis. Instead, to the best of our knowledge, no direct exploitation of symmetry in motion primitives of bimanual movements has been addressed up to now. By analyzing human behaviour, one can notice the high amount of symmetry between our arms in the daily tasks we perform, such as lifting weights, folding a shirt, etc. In the context of RL, the notion of symmetric Markov Decision Process (MDP) was given in [7], and used in [8], [9], [10] to solve the value function approximation problem relying on symmetry in the state-action space, but in the PS approach followed here we make no use of value functions. In this scenario, it is important to reduce the dimension of the policy parameter space in order to get good solutions with a limited number of real-robot executions [11]. Hence, we face the problem of learning bimanual robotic tasks that are parameterized in high-dimensional spaces by imposing and learning symmetries that will simplify the problem. The approach proposed is to look for such symmetries between the movements executed by the two arms when performing a task, in order to represent the bimanual motion in a reduced manner, with the objective of increasing the speed of the learning process. Initially, we

Fig. 1: *Kinesthetic demonstration of the symmetric bimanual operation to fold a towel.*

This work was supported by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656) and partially developed in the context of the project CLOTHILDE ("CLOTH manIpulation Learning from DEmonstrations"), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Advanced Grant agreement No 741930).

[1]Department of Information Engineering, Università degli Studi di Padova, Italy amadiofa@dei.unipd.it

[2]Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, Spain acolome@iri.upc.edu torras@iri.upc.edu

tackled planar symmetries. However, we noticed that some motions might have other kinds of symmetries, such as a cylindrical or spherical symmetry (for example they could appear when using long tools, such a broom, with two hands). Therefore, we also included these in our work.

The rest of the paper is organized as follows: Section II defines the basic elements of ProMPs used throughout this work, Section III presents the symmetrization methods for Gaussian probability distributions and ProMPs together with some way of estimating the parameters of symmetry surfaces, Section IV is devoted to the description of the RL procedure developed for symmetric bimanual tasks, Section V presents the results obtained in simulation and in a real-robot experiment, and Section VI draws the conclusions.

II. Probabilistic Movement Primitives (ProMPs)

ProMPs use a weight vector $\boldsymbol{w}$ to compactly represent a single generic trajectory $\boldsymbol{\tau} = \{\mathbf{x}_t\}_{t=1\dots N_t}$ ($\mathbf{x}_t \in \mathbb{R}^D$ is the state vector at time *t*, N_t is the total number of samples) as a linear basis function model $\mathbf{x}_t = \boldsymbol{\Phi}_t^T \boldsymbol{w} + \nu$. $\boldsymbol{\Phi}_t^T := [I_D \otimes \phi_t]$, where ϕ_t defines a time-dependent basis vector, composed by the values at time *t* of *M* Gaussian functions with centers uniformly distributed over time, I_D indicates the identity matrix of the same dimension of the state and $\otimes$ the *Kronecker product*. $\boldsymbol{w}$ is a weight vector of dimension *M*, and ν a zero-mean Gaussian noise with variance Σ_ν. $\boldsymbol{w}$ is assumed to be a random variable with Gaussian multivariate distribution $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mu_{\boldsymbol{w}}, \Sigma_{\boldsymbol{w}})$. The resulting trajectory can be expressed as a probability distribution whose expression depends on parameters $\boldsymbol{\theta} = \{\mu_{\boldsymbol{w}}, \Sigma_{\boldsymbol{w}}, \Sigma_\nu\}$. This distribution can be fitted, given a set of K_d demonstrated trajectories $\boldsymbol{\tau}^k = \{\mathbf{x}_t\}_{t=1\dots N_t}$, $k = 1 \dots K_d$, by obtaining the weights $\boldsymbol{w}_k$ for each demonstration using the *Moore-Penrose pseudoinverse*. Parameters $\boldsymbol{\theta}$ are fitted by means of a *maximum likelihood estimation*, i.e. computing the sample mean and the sample covariance of $\boldsymbol{w}$. The distribution $p(\mathbf{x}_t; \boldsymbol{\theta})$ at time *t* is given by

$$p(\mathbf{x}_t; \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t | \boldsymbol{\Phi}_t^T \mu_{\boldsymbol{w}}, \boldsymbol{\Phi}_t^T \Sigma_{\boldsymbol{w}} \boldsymbol{\Phi}_t + \Sigma_\nu) \qquad (1)$$

Then the probability of observing a trajectory can be expressed as the product of all time-step probabilities.

We have made the choice of representing robot movements as the trajectories drawn by the end-effector. Only its position is considered, regardless of its orientation, by executing trajectories keeping the end-effector aligned with the robot-arm. Thus, the problem is reduced to a three-dimensional positioning of the end-effector point. Being (x_t, y_t, z_t) the end-effector coordinates at time t, ProMPs have been used to model trajectories of the state vector $\mathbf{x}_t := [x_t, y_t, z_t]^T$. Fitting noise ν has been neglected. Demonstrations in task space are derived from joint values via forward kinematics [12], and ProMP's rollouts are executed by solving inverse kinematics for the robotic manipulators, like in [13], [14], [15] and passing joints commands to the robot arm.

III. Symmetrization methods

Given a two-arms motion encoded as a ProMP, the movement of the second arm can be expressed as a symmetrization of the ProMP of the first one. In this way, the bimanual task policy is represented by only a single MP and a surface, reducing considerably the number of parameters. The most common type of symmetry is defined by a plane, but also cases of curved surfaces are taken into account.

A. Gaussian Distribution Symmetrization

Being ProMPs a representation that describes a trajectory as a time-dependent multivariate Gaussian probability density function, first of all, it is necessary to find a method to symmetrize a Gaussian with mean μ and covariance Σ with respect to a given surface. In order to symmetrize the Gaussian, we will symmetrize the ellipsoid given by the covariance matrix. This will be the base on which the generalized ProMP symmetrization method will be built upon. Firstly, we will introduce a methodology for planar symmetries. However, we generalize to other kinds of shapes, such as cilindrical or spherical symmetries. From now on we will use superindex s to indicate symmetric elements.

1) *Planar Symmetries*

A Gaussian distribution is characterized by two elements, its expected value and its covariance matrix. Being the mean value $\mu = [x_\mu, y_\mu, z_\mu]^T$, finding its symmetric μ^s with respect to a plane $ax + by + cz + d = 0$ is a trivial geometric problem. The covariance matrix Σ is completely defined by its eigenvalues λ_i and eigenvectors $\boldsymbol{v}_i$ ($i = 1, 2, 3$). The symmetrization of Σ with respect to a plane can be seen as a roto-translation of its eigenvectors, keeping the eigenvalues unchanged. Such symmetrized eigenvectors, $\boldsymbol{v}_i^s$ (see Fig. 2) can be then used to reconstruct the symmetric covariance matrix $\Sigma^s = V^s \; D \; (V^s)^T$ where $V^s = [\boldsymbol{v}_1^s | \boldsymbol{v}_2^s | \boldsymbol{v}_3^s]$ and $D = diag(\lambda_1, \lambda_2, \lambda_3)$.

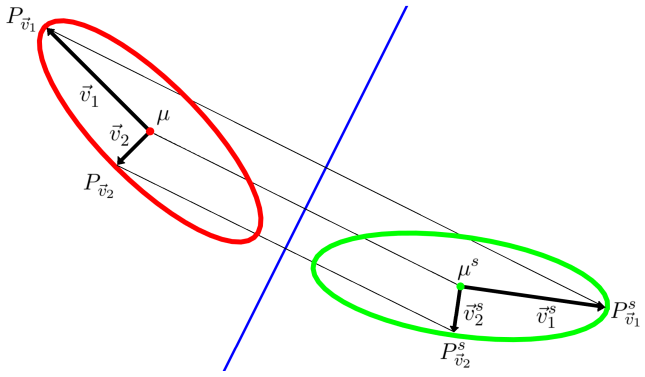

Fig. 2: *Planar symmetrization method applied to a two-dimensional Gaussian distribution represented as an ellipse. Note how the symmetric eigenvectors can be easily obtained from the symmetrization of points μ, $P_{\vec{v}_1}$ and $P_{\vec{v}_2}$.*

2) *Curved Symmetries*

Representing a covariance matrix as an ellipsoid of uncertainty, it is clear that its symmetrization w.r.t. a curved surface will end in a non-elliptical shape. Due to the necessity of obtaining always a normal distribution after the symmetrization (also in the case of curved symmetries),

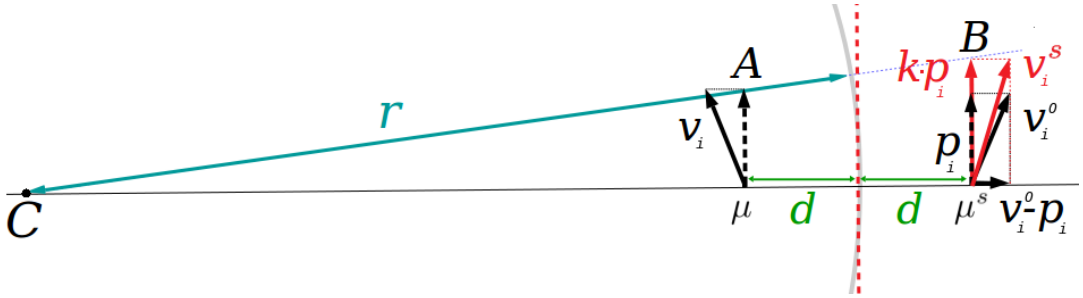

Fig. 3: *Reshaping of an eigenvector in a two-dimensional representation. Factor k is obtained from the Thales proportion $\overline{\mu A} : (r-d) = \overline{\mu^s B} : (r+d)$. Vector $\boldsymbol{v}_i^0$ (obtained from the initial planar symmetrization, for $i = 1, 2, 3$) is decomposed into the projection $\boldsymbol{p}_i = \frac{\boldsymbol{v}_i^0 \cdot \boldsymbol{t}}{\boldsymbol{t} \cdot \boldsymbol{t}} \cdot \boldsymbol{t}$ onto the curvature direction $\boldsymbol{t}$, and the difference $\boldsymbol{v}_i^0 - \boldsymbol{p}_i$. Then, the component $\boldsymbol{p}_i$ is multiplied by the scaling factor k to obtain the reshaped symmetric eigenvector $\boldsymbol{v}_i^s$, as $\boldsymbol{v}_i^s = k \cdot \boldsymbol{p}_i + (\boldsymbol{v}_i^0 - \boldsymbol{p}_i)$.*

this effect has been modeled with a deformation of the eigenvectors and eigenvalues of the symmetric covariance matrix, which depend on the degree of curvature of the surface:

- The first step is to perform a planar symmetrization, in the same way as described before, considering the tangent plane going through the projection of $\mu = [x_\mu, y_\mu, z_\mu]^T$ on the surface. Let's call the resulting covariance matrix Σ_0^s, that will be adapted taking into account the degree of curvature of the symmetry surface used. Eigenvectors $\boldsymbol{v}_i^0$ and eigenvalues λ_i^0 $(i = 1, 2, 3)$ are obtained from Σ_0^s, and it is necessary to modify them to reshape the covariance matrix.
- The eigenvectors are projected onto the directions tangent to the symmetry surface, and only the tangent components are modified by a scaling factor k that depends on the radius r and the distance from the surface d, being $k = \frac{r+d}{r-d}$.
- The considered components are multiplied by k, if μ is inside the surface, or divided by k, if outside. The factor depends on the value of d with respect to r, its expression is based on the proportionality between segments defined in *Thales theorem* (see Fig. 3).
- Eigenvalues are also updated multiplying them by the ratio between the norm of the associated reshaped eigenvector and the original one, $\lambda_i^s = \lambda_i^0 \cdot \frac{|\boldsymbol{v}_i^s|}{|\boldsymbol{v}_i|}$. The symmetric covariance matrix is obtained from $\Sigma^s = V^s \, D \, (V^s)^{-1}$, V^s being the matrix whose columns are the new eigenvectors $\boldsymbol{v}_i^s$ normalized, and D is a matrix with the new eigenvalues λ_i^s in the diagonal and zeros in all the other entries.

This procedure does not guarantee the symmetry of the matrix obtained, because reshaped eigenvectors are no longer orthonormal. Hence to ensure symmetry, Σ^s is corrected by substituting it with $(\Sigma^s + (\Sigma^s)^T)/2$, averaging each pair of symmetric non-diagonal entries. The effects of the reshaping effect can be visualized in Fig. 4.

B. ProMP Symmetrization

End-effector trajectory $\boldsymbol{\tau} = \{\mathbf{x}_t\}_{t=1 \dots N_t}$ in cartesian position coordinates can be expressed with a ProMP as

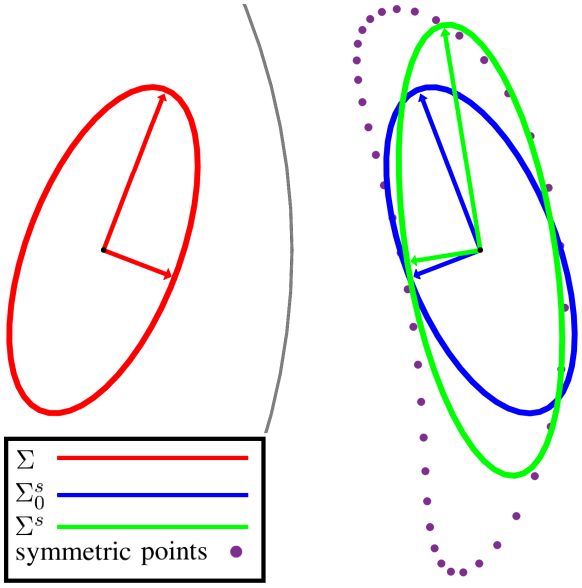

Fig. 4: *Curved symmetrization method applied to a two-dimensional Gaussian distribution. The radial symmetrization of ellipse-points produces a result that is not an ellipse anymore (purple points), hence curved symmetry of a Gaussian is approximated with the presented reshaping effect on the eigenvectors and eigenvalues.*

$\mathbf{x}_t = \boldsymbol{\Phi}_t^T \boldsymbol{w}$ (recalling notation used in Section II and remembering that fitting noise has been neglected). Thus, the three-dimensional Gaussian probability distribution for each time t is given by

$$p(\mathbf{x}_t; \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t | \mu_t, \Sigma_t) := \mathcal{N}(\mathbf{x}_t | \boldsymbol{\Phi}_t^T \mu_{\boldsymbol{w}}, \boldsymbol{\Phi}_t^T \Sigma_{\boldsymbol{w}} \boldsymbol{\Phi}_t) \quad (2)$$

A sequence of symmetric distributions $\mathcal{N}(\mathbf{x}_t^s | \mu_t^s, \Sigma_t^s)$ can be computed from $\mathcal{N}(\mathbf{x}_t | \mu_t, \Sigma_t)$ (for $t = 1 \dots N_t$) using the methods described previously (depending on the symmetry surface considered). The objective is to use this new sequence of Gaussians to build ProMP^s $(\boldsymbol{w}_s \sim \mathcal{N}(\mu_{\boldsymbol{w}}^s, \Sigma_{\boldsymbol{w}}^s))$, the symmetrization of the given ProMP. Considering the expected value of the symmetric trajectory $\mu_t^s = \mathbb{E}[x_t^s, y_t^s, z_t^s]^T \ \forall t$, we define the vector of positions for each direction as $\bar{\mathbf{x}}^s := \mathbb{E}[x_1^s \cdots x_{N_t}^s, y_1^s \cdots y_{N_t}^s, z_1^s \cdots z_{N_t}^s]^T$, and the block

basis function matrix $\Psi := [\phi_1 \dots \phi_{N_t}]$. $\mu^s_{\boldsymbol{w}}$ can be computed with expression (4).

$$\bar{\mathbf{x}}^s = \boldsymbol{\Psi}^T \mu^s_{\boldsymbol{w}} \quad \text{with} \quad \boldsymbol{\Psi}^T := I_3 \otimes \Psi^T \tag{3}$$

$$\mu^s_{\boldsymbol{w}} = (\boldsymbol{\Psi}^T)^+ \mathbf{x}^s \tag{4}$$

Concerning the computation of $\Sigma^s_{\boldsymbol{w}}$, N_t different equations $\Sigma^s_t = \boldsymbol{\Phi}^T_t \Sigma^s_{\boldsymbol{w}} \boldsymbol{\Phi}_t$ must be taken into account, one for each instant $t = 1 \dots N_t$. The equations for all time steps can be composed in a unique block matrix equation, which it is solved using the *Moore-Penrose pseudoinverse*:

$$\begin{bmatrix} \vdots \\ \Sigma^s_t(\boldsymbol{\Phi}_t)^+ \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \boldsymbol{\Phi}^T_t \\ \vdots \end{bmatrix} \Sigma^s_{\boldsymbol{w}}, \; \Sigma^s_{\boldsymbol{w}} = \begin{bmatrix} \vdots \\ \boldsymbol{\Phi}^T_t \\ \vdots \end{bmatrix}^+ \begin{bmatrix} \vdots \\ \Sigma^s_t(\boldsymbol{\Phi}_t)^+ \\ \vdots \end{bmatrix} \tag{5}$$

It is necessary to modify the matrix $\Sigma^s_{\boldsymbol{w}}$ at the end, to guarantee its symmetry, otherwise it cannot be a valid covariance matrix. This is done by substituting it with $(\Sigma^s_{\boldsymbol{w}} + (\Sigma^s_{\boldsymbol{w}})^T)/2$, averaging each pair of symmetric non-diagonal entries. The ProMP symmetrization method developed can be applied to all the kinds of symmetries considered, planar and curved. It depends only on the time-dependent probabilistic representation of the symmetric trajectory, regardless of the nature of the symmetry surface. In this section we tackled the problem of finding the ProMP symmetric to another. In the next part, we propose a way of estimating from the demonstrations the most suitable symmetry for the task.

C. Symmetry Estimation

It is necessary to estimate a possible symmetry surface from the demonstrated movements, whether it is a plane or a curved surface. With data obtained from real demonstrations, the symmetry cannot be absolutely perfect, but the distribution of midpoints in the space can be analyzed in order to get the surface that best fits the data. We assume a normal distribution for the parameters and their estimation is given with a mean and a covariance matrix, which can be used for successful policy exploration and learning.

1) *Planar Symmetries*

The method proposed to estimate the parameters of a symmetry plane is based on the distribution of the midpoints $\boldsymbol{m}_t$ between the mean trajectories demonstrated with the two robots $\mu^1_t = [x^1_t, y^1_t, z^1_t]^T$, $\mu^2_t = [x^2_t, y^2_t, z^2_t]^T$ at each instant $t = 1 \dots N_t$.

$$\boldsymbol{m}_t = \frac{\mu^1_t + \mu^2_t}{2} \qquad t = 1 \dots N_t \tag{6}$$

Principal Component Analysis (PCA) has been used to analyze the set of midpoints and find a viable symmetry plane. In a perfectly symmetric bimanual movement, all the midpoints must belong to the same plane, and PCA would result in a third component describing no variance of data. Thus, dealing with real demonstrations, a possible symmetry plane can be defined as the plane orthogonal to the vector defining the third principal component (direction with the least variation) $\boldsymbol{v}_3$, passing through the center of the midpoints $\mu_m := \frac{1}{N_t} \sum_t^{N_t} \boldsymbol{m}_t$. Thus, if $\boldsymbol{v}_3 = [x_{\boldsymbol{v}_3}, y_{\boldsymbol{v}_3}, z_{\boldsymbol{v}_3}]^T$, the estimated parameters of the symmetry plane equation $\hat{a}\ x + \hat{b}\ y + \hat{c}\ z + \hat{d} = 0$ are $\hat{a} = x_{\boldsymbol{v}_3}$, $\hat{b} = y_{\boldsymbol{v}_3}$, $\hat{c} = z_{\boldsymbol{v}_3}$, and $\hat{d} = -\hat{a}x_{\mu_m} - \hat{b}y_{\mu_m} - \hat{c}z_{\mu_m}$. It is also possible to calculate the covariance matrix associated to the estimates of the parameters. This can be done by applying PCA to each pair of demonstrated trajectories independently, obtaining a set of parameters $\{\hat{a}_k, \hat{b}_k, \hat{c}_k, \hat{d}_k\}$ for $k = 1 \dots K_d$ (where K_d is the total number of demonstrations). Covariance matrix of plane's parameters can then be computed with *maximum likelihood estimation*.

2) *Curved Symmetries*

Estimation of curved surfaces is possible too, and the case of a spheric symmetry is presented. The spheric symmetry is defined by 4 parameters $[x_C, y_C, z_C, R]$, center and radius of the sphere. To perform their estimation, the lines $l_{t,k}$ connecting two end-effectors at each time *t*, for every demonstration, are taken into account. The center of the sphere is chosen as the point with minimum quadratic distance from all the lines $l_{t,k}$:

$$\hat{C} = (\hat{x_C}, \hat{y_C}, \hat{z_C}) = \underset{P:=(x,y,z)}{\operatorname{argmin}} \sum_{\forall t,k} d^2(P, l_{t,k}) \tag{7}$$

Then, the radius can be obtained as the distance between $\hat{C}$ and the center of the demonstrations' midpoints μ_m: $\hat{R} = d(\hat{C}, \mu_m)$. Like in the case of the plane, it is possible to calculate also the covariance matrix associated to the estimates of parameters. This can be done by finding the center and radius of the sphere for each pair of demonstrated trajectories independently, obtaining a set of parameters $\{\hat{x_C}^k, \hat{y_C}^k, \hat{z_C}^k, \hat{R}^k\}$ for $k = 1 \dots K_d$. Covariance matrix of sphere's parameters is obtained with *maximum likelihood estimation*.

3) *Optimization of Parameters*

The surface estimated might not be the optimal one, hence this initial estimation needs to be refined using optimization methods. To do so, the *Kullback–Leibler divergence* (KL divergence) is used as a measure of how the probability distributions of the weights of the ProMP_2 (ProMP obtained from demonstrations of the second arm) diverges from the one obtained with the symmetrization of ProMP_1 (ProMP obtained from demonstrations of the first arm). KL divergence is defined as a non-symmetric indicator of the difference between two probability distributions p, q over a random variable x, defined as $\text{KL}(p||q) = \int p(x) log \frac{p(x)}{q(x)} dx$ generally. In our case, where multivariate Gaussian distributions are considered, KL divergence between ProMP_2 and the symmetrized ProMP_1 can be computed as follows.

$$KL(\text{ProMP}_2 || \text{ProMP}^s_1) = \frac{1}{2}(\text{tr}[(\Sigma^s_{\boldsymbol{w}})^{-1}\Sigma^2_{\boldsymbol{w}}] +$$

$$+ (\mu^s_{\boldsymbol{w}} - \mu^2_{\boldsymbol{w}})^T (\Sigma^s_{\boldsymbol{w}})^{-1} (\mu^s_{\boldsymbol{w}} - \mu^2_{\boldsymbol{w}}) - 3 \cdot M + \ln \frac{det(\Sigma^s_{\boldsymbol{w}})}{det(\Sigma^2_{\boldsymbol{w}})}) \tag{8}$$

The surface parameters can then be optimized to minimize the resulting KL divergence between the weight vector distribution of ProMP_1^s and ProMP_2. We used the `fmincon` function in MATLAB to optimize the parameters, using PCA or least-squares solutions to initialize them. Note that using symmetry surfaces, a mirror effect is generated, i.e. an opposed orientation in the symmetric trajectories (to have an intuitive idea think to what happens to letters reflected in a mirror) . Such mirror effect can be removed by considering more than one plane of symmetry, as considered in the Appendix.

IV. Learning of symmetric bimanual tasks

The symmetrization methods developed are used to represent bimanual robotic tasks with a reduced number of parameters defining the policy. The aim is to see if using this symmetry-based approach in the learning process can lead to a faster convergence in the policy search.

A. Relative Entropy Policy Search (REPS)

REPS [16] is the algorithm that has been adopted to perform the policy search for symmetric bimanual tasks. It aims to find the policy π^* that maximizes the expected reward for a given task. A *ProMP* policy $\pi(\boldsymbol{w})$ can then be represented by a normal distribution with mean μ_w and covariance Σ_w generating samples $\boldsymbol{w}_k \sim \mathcal{N}(\mu_w, \Sigma_w)$. Given the previous policy $\pi_{old}(\boldsymbol{w})$, REPS obtains the new policy $\pi(\boldsymbol{w})$ by adding a KL divergence bound ε between the newly obtained policy and the previous one to the optimization of the expected reward. The bound on the KL divergence limits the variation on the new policy and prevents the algorithm from being too greedy.

$$\pi^* = \underset{\pi}{\text{argmax}} \int \pi(\boldsymbol{w}) \text{R}(\boldsymbol{w}) d\boldsymbol{w} \tag{9}$$

$$\text{s.t.} \quad \varepsilon \geqslant \text{KL}(\pi(\boldsymbol{w}) || \pi_{old}(\boldsymbol{w})) \quad \text{and} \quad 1 = \int \pi(\boldsymbol{w}) d(\boldsymbol{w})$$

where $\text{R}(\boldsymbol{w})$ is the reward for the samples weights (it is always negative, and closer to 0 the better is the result of the rollout examined). The constrained optimization problem (9) can be solved efficiently by the method of *Lagrangian multipliers*.

B. Symmetric Learning Approach

The standard way of proceeding, which here is called *Double Learning* (DL), would be to model the movements of the two robotic arms independently with two ProMPs subject to two separated learning processes. Employing the symmetrization techniques presented in this work it is possible to develop a new learning approach, capable of exploiting the symmetric nature of a task, that will be called *Symmetric Learning* (SL). Before starting the learning process, some demonstrations of the task are recorded to build the initial ProMP_1, $\boldsymbol{w}^1 \sim \mathcal{N}(\mu^1_{\boldsymbol{w}}, \Sigma^1_{\boldsymbol{w}})$, modeling movements of the first robot's end-effector, and ProMP_2, $\boldsymbol{w}^2 \sim \mathcal{N}(\mu^2_{\boldsymbol{w}}, \Sigma^2_{\boldsymbol{w}})$, modeling movements of the second robot's end-effector. From these demonstrations, the symmetry surface ρ_0 is estimated.

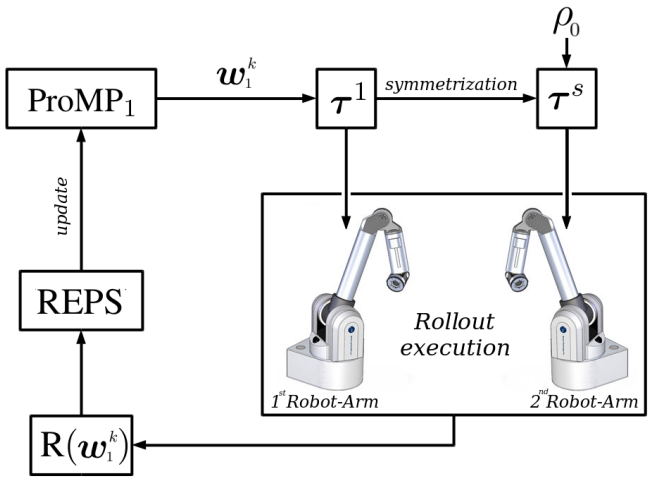

Fig. 5: *Symmetric Learning conceptual scheme.*

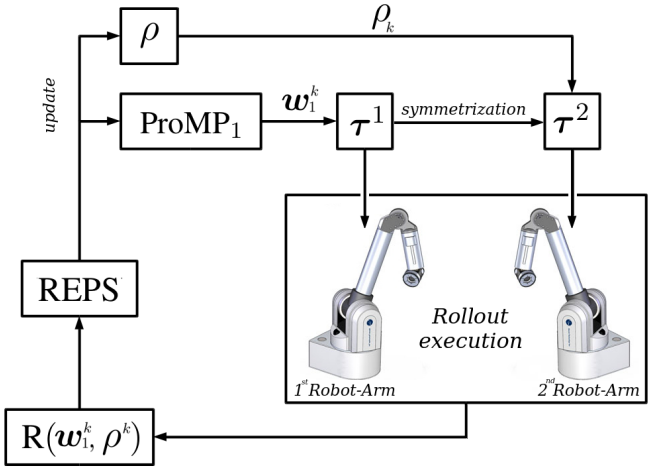

Fig. 6: *Robust Symmetric Learning conceptual scheme.*

Symmetric Learning (SL): *only ProMP_1 is updated during the learning process. At each iteration the first arm executes the movement sampled from ProMP_1, while the second arm executes the symmetric trajectory. Once completed the policy search, movements of the first arm are described by the updated ProMP_1 and movements of the second by its symmetrization ProMP_1^S.*

SL reduces the number of parameters by a half, but its success depends critically on the accuracy of the symmetry surface estimation ρ_0, and if it is not correct, SL cannot have good results. We propose a variation of SL, called *Robust Symmetric Learning* (RSL), that solves the problem by learning, not only ProMP_1, but also the symmetry surface's parameters, represented as a Gaussian probability distribution, $\rho \sim \mathcal{N}(\rho_0, \Sigma_\rho)$. The probability distribution updated by REPS is $\begin{bmatrix} \boldsymbol{w}_1 \\ \rho \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_{\boldsymbol{w}_1} \\ \rho_0 \end{bmatrix}, \begin{bmatrix} \Sigma_{\boldsymbol{w}_1} & * \\ * & \Sigma_\rho \end{bmatrix} \right)$, where weights and the surface's parameters are combined in the same vector (non-diagonal blocks are initialized with zeros).

Robust Symmetric Learning (RSL): *at each learning iteration k, a different surface ρ_k is sampled from the probability distribution, and used to obtain the symmetric*

of the first-arm's trajectory executed by the second one. At the end of the learning process, the surface's probability distribution will converge to the optimal set of parameters to use in the symmetrization of ProMP$_1$ to obtain the policy for the second arm ProMP$_1^s$.

For both SL and RSL, if the symmetry of the task is known a priori, the kind of surface adopted in the learning algorithm can be directly imposed. Otherwise, if this information is not available, it can be derived from demonstrated trajectories: symmetric ProMPs are computed for each kind of possible surfaces in order to choose the one with minimum KL(ProMP$_2$||ProMP$_1^s$) as the most suitable symmetry for the task.

V. Experimental Results

The proposed strategies, Double Learning (DL) and Symmetric Learning (SL), with its variant Robust Symmetric Learning (RSL), have been tested in simulation and in a real-case scenario, in order to verify their capacity to handle the learning of bimanual symmetric movements. For each test, initial ProMPs and a symmetry plane estimate have been computed from some kinesthetic demonstrations of the task. The *MATLAB* code used and all the results obtained, together with a video of the experiment with real-robots, are available at http://www.iri.upc.edu/groups/perception/#symmetricLearning.

A. Simulation

In the simulation, the midpoint between the two robot end-effectors at each time step $\boldsymbol{m}_t = (x_t^{\boldsymbol{m}}, y_t^{\boldsymbol{m}}, z_t^{\boldsymbol{m}}) := \left(\frac{x_t^1+x_t^2}{2}, \frac{y_t^1+y_t^2}{2}, \frac{z_t^1+z_t^2}{2}\right)$ for $t = 1 \dots N_t$ is made to follow a certain path $\{x_t^{ref}, y_t^{ref}, z_t^{ref}\}_{t=1\dots N_t}$ belonging to a surface. The learning process in this case consists of 500 policy update steps evaluating 10 rollouts each. Three different scenarios have been considered: reference path belonging to a plane, a sphere or a cylinder. This surface defines a symmetry in the movements, and it can be known and equal to the one estimated from demonstrations ρ^0; or unknown and derived from a perturbation of ρ^0's parameters. ProMPs have $M = 10$ basis functions for each direction (i.e., $3 \cdot M = 30$ weights in total) and amplitude parameter $h = 0.02$. The reward function considered (10) penalizes the sum of squared errors from the desired path on the surface.

$$\mathrm{R} = 100 \sum_{t=1}^{N_t} \left(x_t^{ref} - x_t^{\boldsymbol{m}}\right)^2 + \left(y_t^{ref} - y_t^{\boldsymbol{m}}\right)^2 + \left(z_t^{ref} - z_t^{\boldsymbol{m}}\right)^2 \tag{10}$$

An example of the results of the simulation can be seen in Figs. 7 and 8, where the final trajectories and rewards are shown for DL and SL applied to the case of a known planar symmetry. Moreover, Table II shows a summary of results after applying Double Learning (DL), Symmetric Learning (SL), and Robust Symmetric Learning (RSL) to problems with planar, spherical or cylindrical symmetries, both knowing the symmetry a priori or not. Having a look at the results, we see that SL shows a faster convergence in the policy search, mostly in the early phases of the learning process

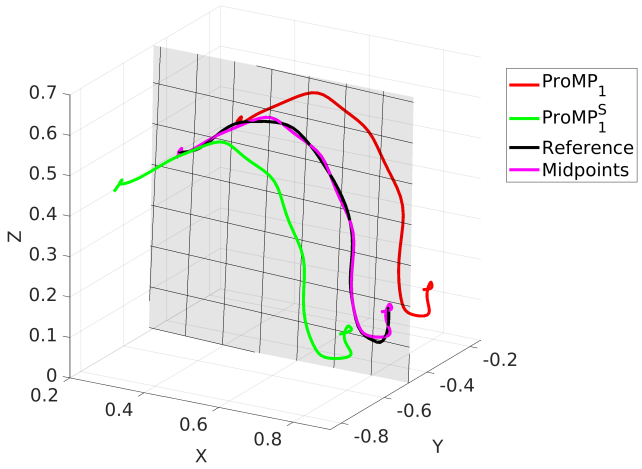

Fig. 7: *Trajectories for ProMP$_1$ and ProMP$_1^s$, obtained by SL after learning to follow a path whose end-effectors' midpoints belong to a known plane.*

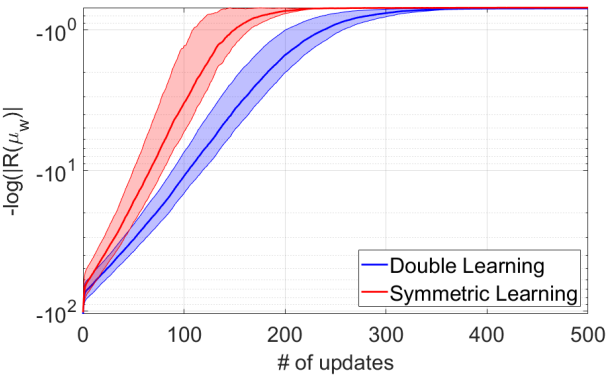

Fig. 8: *Rewards, drawn as mean value and 95% C.I., in the case of a reference path belonging to the a known plane (values in logarithmic scale).*

(roughly the first 100-150 updates), exactly the kind of results sought, but it is only applicable when the symmetry surface is perfectly known. RSL instead shows a slower convergence than SL, because of the need to learn the plane's parameters too. RSL is still faster than DL at the beginning, while later, around 150 updates, it cannot manage to increase rewards as much as DL does. This is still a good result, because it shows that using symmetry in the learning, even in the presence of uncertainties, permits reaching faster a good solution. The fact that later, with more updates, RSL looses effectiveness w.r.t. DL is due the fact that using two separate ProMPs allows to optimize better the rewards, having more parameters at disposal.

B. Real-robot experiment: Folding a towel

The symmetry-based RL approach has been tested in a real-case application. The aim is to make two robotic arms learn how to fold a towel on a table (see Fig. 1). In order to evaluate the quality of the execution, the experimental setup includes a rooftop-placed *Kinect* camera covering the

TABLE I: *Reward function values obtained while learning to follow a desired path of the end-effectors' midpoint.*

			START	10 updates	50 updates	100 updates	200 updates	500 updates
Plane	*known*	DL	-104.8 ± 0.0	-63.9 ± 10.2	-31.0 ± 6.7	-10.98 ± 3.57	-1.511 ± 0.505	-0.7054 ± 0.0001
		SL	-102.0 ± 0.0	-53.7 ± 12.1	-16.6 ± 6.8	-3.30 ± 2.00	-0.735 ± 0.033	-0.6964 ± 0.0001
	unknown	DL	-165.3 ± 32.5	-110.3 ± 26.9	-63.6 ± 20.2	-31.23 ± 13.05	-8.001 ± 5.319	-1.2525 ± 0.7899
		RSL	-169.3 ± 34.4	-82.2 ± 11.2	-31.2 ± 4.3	-8.69 ± 1.43	-2.181 ± 0.564	-1.0438 ± 0.1751
Sphere	*known*	DL	-41.4 ± 0.0	-26.5 ± 4.6	-7.8 ± 1.9	-1.51 ± 0.43	-0.259 ± 0.037	-0.1890 ± 0.0001
		SL	-39.6 ± 0.0	-25.5 ± 4.8	-2.8 ± 1.3	-0.26 ± 0.10	-0.094 ± 0.009	-0.0737 ± 0.0068
	unknown	DL	-61.6 ± 10.4	-37.4 ± 8.6	-12.9 ± 4.0	-2.67 ± 1.02	-0.374 ± 0.120	-0.2001 ± 0.0096
		RSL	-64.9 ± 15.3	-43.0 ± 15.1	-7.4 ± 3.8	-1.07 ± 0.37	-0.186 ± 0.027	-0.1154 ± 0.0180
Cylinder	*known*	DL	-31.5 ± 0.0	-18.7 ± 3.1	-4.0 ± 0.9	-0.66 ± 0.14	-0.232 ± 0.008	-0.2018 ± 0.0001
		SL	-37.1 ± 0.0	-25.8 ± 5.6	-3.3 ± 1.0	-0.38 ± 0.11	-0.090 ± 0.009	-0.0630 ± 0.0071
	unknown	DL	-76.7 ± 22.5	-55.4 ± 15.1	-23.2 ± 8.8	-7.26 ± 3.69	-0.902 ± 0.428	-0.2432 ± 0.0247
		RSL	-87.9 ± 27.9	-56.6 ± 12.5	-12.1 ± 3.3	-2.07 ± 0.54	-0.336 ± 0.149	-0.1868 ± 0.0694

TABLE II: *Reward function values obtained while learning to fold a towel.*

	START	1 update	2 updates	3 updates	4 updates	5 updates
DL	-4.82 ± 2.73	-3.23 ± 2.08	-1.75 ± 0.24	-1.48 ± 0.07	-1.46 ± 0.04	-1.48 ± 0.03
RSL	-6.80 ± 3.04	-2.62 ± 1.15	-1.52 ± 0.18	-1.38 ± 0.09	-1.33 ± 0.08	-1.21 ± 0.05

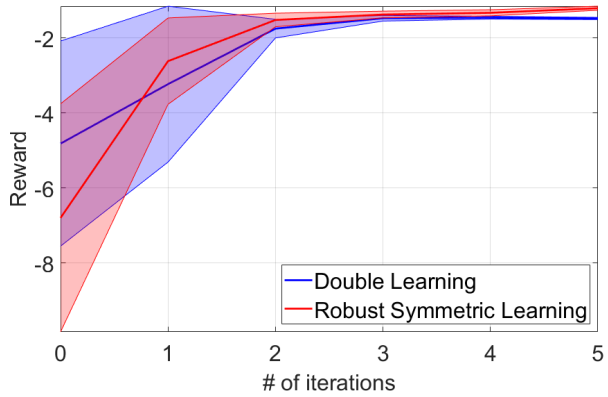

Fig. 9: *Rewards, drawn as mean value and 95% C.I., obtained during the experiment of folding a towel.*

workspace, providing RGB and depth information. ProMPs with $M = 15$ basis functions for each direction have been chosen (i.e., $3 \cdot M = 45$ weights in total), with an amplitude parameter of $h = 0.0167$. In this experiment, the learning process consists of 5 policy updates of 12 rollouts each, with a reuse of the previous 12 sampled weights and rewards in the PS update. Inverse kinematics is used to transform end-effector positions into joint trajectories, which then a computed-torque controller [17] would compliantly track. As reward, a function that penalizes large joint accelerations is adopted, together with an indicator of how well the towel has been folded. This indicator takes into account how well the resulting shape approximates a rectangle, and how many wrinkles the towel shows. Therefore, the reward function used is:

$$\mathrm{R} = \mathrm{R}_{acc} + \mathrm{R}_{shape} + \mathrm{R}_{wrinkle} \tag{11}$$

where R_{acc} is a term penalizing large acceleration commands at joint level to avoid sudden movements. The folding correctness of shape is evaluated by color-segmenting the towel image on the table and fitting a bounding rectangle to the obtained result. R_{shape} has a large penalizing value if the result after motion does not have a rectangular shape on the table, and it is expressed as:

$$\mathrm{R}_{shape} = -\frac{\#\text{ rectangle pixels}}{\#\text{towel pixels}}\left[(A - A_r)^2 + (B - B_r)^2\right] \tag{12}$$

where *A*, *B* are the measured side lengths of the bounding rectangle, and A_r , B_r are their reference values. $\mathrm{R}_{wrinkle}$ penalizes the outcome if the towel presents too many wrinkles after the folding (the mean gradient of the depth is used as the wrinkleness indicator). It is computed using the code available from [18].

RSL and DL approaches succeed both in folding well the towel at the end of the learning process. Exploration of parameter space was problematic in the first iterations due to the physical constraints of task. In fact, movement variability cannot be too big, because if the two arms go too far one from each other, end-effectors cannot hold the towel anymore. These situations lead to some bad rollouts from the initial policies, in particular in DL, when the two arms move independently. After some updates this issue is not a problem anymore, thanks to the convergence to more effective folding movements. Evolution of reward values throughout the learning process are shown in Fig. 9 and Table II. RSL at the beginning shows worse results than DL, which is able to model better movements of the second robot arm thanks to its higher number of parameters. This initial disadvantage is recovered in just one learning update, confirming a faster convergence speed for the RSL method, together with a more significant reduction in reward's variance than the one obtained with DL. Furthermore, DL converges to a slightly worse reward for the final policy.

VI. Conclusions

In this paper, we successfully developed a symmetric approach for the learning of bimanual robotic tasks using ProMP-based policies. The use of symmetries has proven effective in reducing the dimension of the parameter space, although, the standard approach (that uses two distinct policies for each robot) could fit more finely robotic trajectories, because

of its higher number of parameters. The great advantage of the symmetry-based approach is the fastest convergence during the learning process. Thanks to the reduced number of parameters used and the policy exploration limited to symmetric trajectories, the proposed method manages to reach good results with few updates. It could be a promising framework to reach fast and effective learning for symmetric bimanual tasks. Further experiments are needed to test its performance in other scenarios, like assisted dressing, objects lifting and usage of tools. Other significant results of this work are the symmetrization methods derived for multivariate Gaussian distributions, that can find applications also outside of the context of movement representation.

APPENDIX

Composition of Symmetry Planes

Different symmetry planes can be combined to couple robotic trajectories that are impossible to define with only a single symmetry. The problem of representing a rotation of a trajectory around a given point as the combination of different symmetrizations is faced. Consider an initial trajectory $\boldsymbol{\tau}^0 = \{\boldsymbol{\tau}_t^0\}_{t=1\dots N_t}$, and a desired trajectory $\boldsymbol{\tau}^{goal} = \{\boldsymbol{\tau}_t^{goal}\}_{t=1\dots N_t}$, result of a rotation of $\boldsymbol{\tau}^0$ (the origin has been taken as the center of rotation without loss of generality). The objective is to find the combination of symmetries that transform $\boldsymbol{\tau}^0$ into $\boldsymbol{\tau}^{goal}$.

A method to obtain $\boldsymbol{\tau}^{goal}$ from $\boldsymbol{\tau}^0$ as a result of two symmetrizations has been developed, and it proved successful in several trials, with different trajectories and rotation matrices. The first symmetrization is used to obtain a trajectory $\boldsymbol{\tau}^1$ centered with $\boldsymbol{\tau}^{goal}$ (same centroid), the second one to align it correctly to the desired $\boldsymbol{\tau}^{goal}$.

- compute centroid of $\boldsymbol{\tau}^0$: $\mu_{\boldsymbol{\tau}^0} = \frac{1}{N_t}\sum_{t=1}^{N_t} \boldsymbol{\tau}_t^0$
- compute centroid of $\boldsymbol{\tau}^{goal}$: $\mu_{\boldsymbol{\tau}}^{goal} = \frac{1}{N_t}\sum_{t=1}^{N_t} \boldsymbol{\tau}_t^{goal}$
- **I symmetry plane** orthogonal to $\boldsymbol{p}_1 = \mu_{\boldsymbol{\tau}}^{goal} - \mu_{\boldsymbol{\tau}}$ and passing through the point $\frac{\mu_{\boldsymbol{\tau}}^{goal} + \mu_{\boldsymbol{\tau}}^0}{2}$ $\rightarrow$ resulting trajectory: $\boldsymbol{\tau}^1$
- compute the trajectory $\boldsymbol{m}$ described by the midpoints of $\boldsymbol{\tau}^1$ and $\boldsymbol{\tau}^{goal}$: $\boldsymbol{m}_t = \frac{\boldsymbol{\tau}_t^{goal} + \boldsymbol{\tau}_t^0}{2}$ for $t = 1 \dots N_t$
- **II symmetry plane** orthogonal to $\boldsymbol{p}_2$ the third principal component vector obtained from PCA applied to the set $\{\boldsymbol{m}_t\}_{t=1\dots N_t}$ (the third component has null variance, because all the midpoints belong to the same plane orthogonal to it) $\rightarrow$ resulting trajectory: $\boldsymbol{\tau}^2 = \boldsymbol{\tau}^{goal}$

The origin (that is the center of rotation considered) will belong to both planes. The symmetrization of a point P with respect to a plane orthogonal to unit vector $\boldsymbol{p}$ that contains the origin can be defined as a linear transformation: $P^s = H \cdot P$ with $H := \left[I - 2\boldsymbol{p}\boldsymbol{p}^T\right]$. H is known as *Householder transformation*. Thus, $\forall R$ rotation matrix, $\exists$ $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ defining two *Householder transformations* $H_1 = \left[I - 2\boldsymbol{p}_1^T\boldsymbol{p}_1\right]$ and $H_2 = \left[I - 2\boldsymbol{p}_2^T\boldsymbol{p}_2\right]$ s.t. $H_2 \cdot H_1 = R$. Moreover, *Householder transformation* H has $\det(H) = -1$, meaning there is a change of orientation: a *mirror effect*. Therefore, those trajectories whose transforms are pure rotations R, having $\det(R) = 1$, will need a pair number of symmetries to be represented. A Symmetric learning approach can then be extended naturally to the combination of symmetry planes, in order to apply it also in the cases in which the bimanual task is not symmetric, for instance an anti-symmetric bimanual movement, that can be described by two orthogonal symmetry planes (an example could be the opening of a valve using two hands).

REFERENCES

[1] M.P. Deisenroth, G. Neumann and J. Peters, "A Survey on Policy Search for Robotics". *Foundations and Trends in Robotics*, 2(1–2), pp. 1–142, 2011.

[2] B.D. Argall, S. Chernova, M. Veloso, B. Browning, "A survey of robot learning from demonstration". *Robotics and Autonomous Systems*, 57(5), pp. 469-483, 2009.

[3] S. Schaal, J. Peters, J. Nakanishi, A.J. Ijspeert, "Learning Movement Primitives". *Robotics Research. The Eleventh International Symposium*, pp. 561-572, Springer Berlin Heidelberg, 2005.

[4] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic Movement Primitives". *Advances in Neural Information Processing Systems (NIPS)*, pp. 2616-2624, 2013.

[5] L.P. Ureche, and A. Billard, "Constraints extraction from asymmetrical bimanual tasks and their use in coordinated behavior". *Robotics and Autonomous Systems*, 103, pp. 222-235, 2018.

[6] K.S. Luck, and H.B. Amor, "Extracting bimanual synergies with reinforcement learning". *International Conference on Intelligent Robots and Systems (IROS)*, pp. 4805-4812, 2017.

[7] M. Zinkevich, and T. Balch, "Symmetry in Markov Decision Processes and its Implications for Single Agent and Multiagent Learning". *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pp. 632-640, 2001.

[8] A. Agostini, and E. Celaya, "Exploiting Domain Symmetries in Reinforcement Learning with Continuous State and Action Spaces". *8th International Conference on Machine Learning and Applications*, pp. 331-336, 2009.

[9] M.A.S. Kamal, and J. Murata. "Reinforcement Learning for Problems with Symmetrical Restricted States". *Robotics and Autonomous Systems*, 56(9), pp. 717-727, 2008.

[10] G. Wang, Z. Fang, B. Li, and P. Li, "Integrating Symmetry of Environment by Designing Special Basis functions for Value Function Approximation in Reinforcement Learning". *14th International Conference on Control, Automation, Robotics & Vision (ICARCV)*, pp. 1-6, 2016.

[11] A. Colomé and C. Torras. "Dimensionality Reduction for Dynamic Movement Primitives and Application to Bimanual Manipulation of Clothes". *IEEE Transactions on Robotics*, 34(3), pp 602-615, 2018.

[12] M. Spong, S. Hutchinson, M. Vidyasagar, "Robot Modeling and Control". John Wiley and Sons, 2006.

[13] M. Shimizu, H. Kakuya, W.K. Yoon, K. Kitagaki, and K. Kosuge, "Analytical Inverse Kinematic Computation for 7-dof Redundant Manipulators with Joint Limits and its Application to Redundancy Resolution". IEEE *Transactions on Robotics*, 24(5), pp. 1131–1142, 2008.

[14] G.K. Singh, and J. Claassens, "An Analytical Solution for the Inverse Kinematics of a Redundant 7-dof Manipulator with Link Offsets". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2976–2982, 2010.

[15] A. Colomé and C. Torras. Closed-Loop Inverse Kinematics for Redundant Robots: Comparative Assessment and Two Enhancements. *IEEE/ASME Transactions on Mechatronics*, 20(2), pp. 944-955, 2015.

[16] J. Peters, K. Mülling, and Y. Altün, "Relative Entropy Policy Search". *24th National Conference on Artificial Intelligence*, 15, pp. 182-189, 2011.

[17] A. Colomé, A. Planells and C. Torras, "A Friction-Model-Based Framework for Reinforcement Learning of Robotic Tasks in Non-Rigid Environments". *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5649-5654, 2015.

[18] A. Ramisa, G. Alenyà, F. Moreno-Noguer, and C. Torras. "A 3D Descriptor to Detect Task-Oriented Grasping Points in Clothing". *Pattern Recognition*, 60, pp. 936-948, 2016.