# Demonstration-Free Contextualized Probabilistic Movement Primitives, Further Enhanced with Obstacle Avoidance

Adrià Colomé[1] and Carme Torras[1]

***Abstract*— Movement Primitives (MPs) have been widely used over the last years for learning robot motion tasks with direct Policy Search (PS) reinforcement learning. Among them, Probabilistic Movement Primitives (ProMPs) are a kind of MP based on a stochastic representation over sets of trajectories, which benefits from the properties of probability operations. However, the generation of such ProMPs requires a set of demonstrations to capture motion variability.**

**Additionally, using context variables to modify trajectories coded as MPs is a popular approach nowadays in order to adapt motion to environmental variables. This paper proposes a contextual representation of ProMPs that allows for an easy adaptation to changing situations through context variables, by reparametrizing motion with them. Moreover, we propose a way of initializing contextual trajectories without the need of real robot demonstrations, by setting an initial position, a final position, and a number of *trajectory interest points*, where the contextual variables are evaluated. The parametrizations obtained show to be accurate while relieving the user from the need of performing costly computations such as conditioning. Additionally, using this contextual representation, we propose a simple yet effective quadratic optimization-based obstacle avoidance method for ProMPs. Experiments in simulation and on a real robot show the promise of the approach.**

## I. Introduction

Movement Primitives (MPs) are a common approach to characterize motion in robotics, usually as a compact representation that easily permits learning tasks based on changes in their parameters. Over the last years, Dynamic Movement Primitives (DMPs) have been widely used for motion representation and learning [1], [2]. However, DMPs are a deterministic approach to motion representation, thus they are not capable of representing motion variability. In contrast, the recently proposed ProMPs [3] approach is capable of capturing the variance of a set of demonstrations to a robot of the same task, and then reproducing the trajectory with the same variance over time, thanks to a stochastic model-based linear feedback controller.

The ProMPs representation allows the use of probabilistic operations such as conditioning a trajectory to a certain via-point, and blending several trajectories, among others [3]. The adaptability of such ProMPs can be extended through a model with context variables, i.e., environmental elements that can be encoded with a variable, as a via-point position, goal position, initial position, etc. These contextual representations are then suitable for contextual reinforcement learning via direct Policy Search (PS) [4], [5], [6], [7] or through hierarchical representations [8]. In the case of ProMPs, the contextual parameters can be inserted by reparametrizing the trajectory weights with the context variables, as in [9].

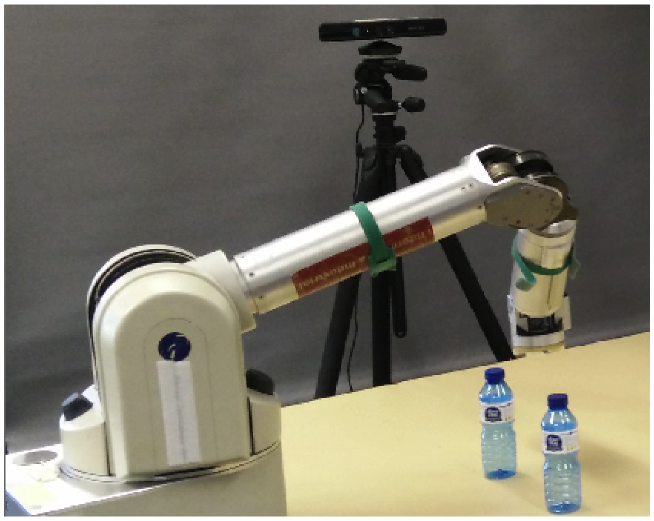
Fig. 1. The experimental setting consisting of a Barrett WAM robot arm and a Kinect camera.

However, ProMPs usually require a set of demonstrations to be trained, and to fully characterize their covariance, a number of samples equal to, at least, the number of parameters, is recommended. However, in robots as the Barrett WAM robot (see Fig. 1) with 7 DoF, using ProMPs, often a few hundreds of parameters are needed to appropriately fit a trajectory. Therefore, having a full-ranked covariance matrix during training is impractical. For this reason, in this paper we propose to build a framework for easily initializing ProMPs with synthetic data, and build a conditioning dataset in order to easily map context variables to motion parameters. This can be used for both exploiting its features by executing such contextualized trajectories, or improving through contextual PS.

Moreover, in this paper we also propose an obstacle avoidance framework for ProMPs. Obstacle avoidance is usually an important matter when learning trajectories with movement primitives, and it is often difficult to adapt trajectories to avoid contact with elements present in the environment. While some approaches clustered a series of recorded trajectories to help avoiding obstacles [10], DMPs can also adapt by setting potential functions in the acceleration domain [2]. In the case of ProMPs, we analytically impose a certain distance from the robot trajectory to the

This work is partially funded by CSIC projects MANIPlus (201350E102) and TextilRob (201550E028), and by the European project I-DRESS (PCIN-2015-147)

[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain. E-mail: acolome@iri.upc.edu, torras@iri.upc.edu.

obstacle, while minimizing the differences wrt. the original trajectory. The resulting optimization problem can be solved fast by quadratic solvers available online, like the *YALMIP* solver [11].

In Section II, we will define the basic elements of ProMPs used throughout this paper. Section III is devoted to the contextual ProMPs and we present the obstacle avoidance with ProMPs in Section IV. Finally, some conclusions and future prospects are sketched.

## II. Probabilistic Movement Primitives (ProMPs)

### A. Definition

ProMPs are a general approach to learn and encode a set of similar motion trajectories that present time-dependent variances over time as seen in Fig. 2. Given a number of basis functions per DoF, $N_f$, ProMPs use a $N_f \times 2$ time-dependent matrix $\mathbf{\Phi}_t = [\phi_t, \dot{\phi}_t]$ to encode position and velocity, $\phi_t$ being the vector of normalized kernel basis functions (e.g., uniformly distributed Gaussian basis function over time). Thus, the position and velocity state vector $\mathbf{y}_t$ can be represented as

$$\mathbf{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \mathbf{\Phi}_t^T \boldsymbol{\omega} + \boldsymbol{\epsilon}_y, \tag{1}$$

where $\boldsymbol{\epsilon}_y \sim \mathcal{N}(0, \mathbf{\Sigma}_y)$ is a zero-mean Gaussian noise and the weights $\boldsymbol{\omega}$ are also treated as random variables with a distribution

$$p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega). \tag{2}$$

This distribution can be fitted, given a set of demonstration trajectories $\boldsymbol{\tau}_k = \{\mathbf{y}_t^k\}_{t=1..N_t}$, $k = 1..N_k$, by obtaining the weights $\boldsymbol{\omega}_k$ of each demonstration through least squares. Subsequently, the parameters of the distribution $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega, \mathbf{\Sigma}_y\}$, $\mathbf{\Sigma}_y$ being the state covariance, are fitted by means of a maximum likelihood estimate, i.e., we compute the sample mean and the sample covariance of $\boldsymbol{\omega}$. Then the probability of observing a trajectory $\boldsymbol{\tau}$ can be expressed as the product of all timestep probabilities $p(\mathbf{y}_t; \boldsymbol{\theta})$

$$p(\boldsymbol{\tau}; \boldsymbol{\theta}) = \prod_t \int \mathcal{N}(\mathbf{y}_t|\mathbf{\Phi}_t^T \boldsymbol{\omega}, \mathbf{\Sigma}_y)\mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega)d\boldsymbol{\omega} \tag{3}$$

Due to the probabilistic representation, the ProMPs approach can represent motion variability while keeping other MP properties such as rescalation and linear representation wrt. parameters. It also allows for other operations such as modulation via probabilistic conditioning and combination by product [3].

In addition, ProMPs also come with a model-based stochastic controller that reproduces the encoded trajectory distribution. In Fig. 2, we show the average and standard deviation of a ProMP and different sample trajectories from its distribution.

### B. Conditioning and blending of ProMPs

As shown in [3], ProMPs allow the use of probability operations to merge or condition trajectories. In order to assign a via-point, the ProMP represented as $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega\}$ can be conditioned to go through the operational space point $\mathbf{y}_t^\star$ with covariance $\mathbf{\Sigma}_t^\star$ at timestep $t$, resulting in a new set of ProMP Gaussian parameters $\boldsymbol{\theta}^{\text{cond}} = \{\boldsymbol{\mu}_\omega^{\text{cond}}, \mathbf{\Sigma}_\omega^{\text{cond}}\}$ [3]:

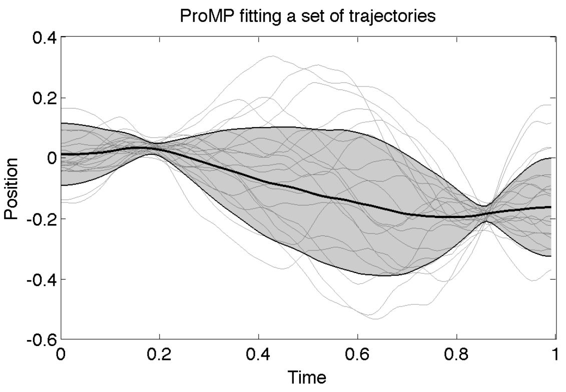

Fig. 2. ProMP fitting a set of trajectories, the mean and standard deviation for each timestep are shown.

$$\begin{aligned} \boldsymbol{\mu}_\omega^{\text{cond}} = & \ \boldsymbol{\mu}_\omega + \\ + & \ \mathbf{\Sigma}_\omega \mathbf{\Phi}_t \left(\mathbf{\Sigma}_t^\star + \mathbf{\Phi}_t^T \mathbf{\Sigma}_\omega \mathbf{\Phi}_t\right)^{-1} \left(\mathbf{y}_t^\star - \mathbf{\Phi}_t^T \boldsymbol{\mu}_\omega\right) \end{aligned} \tag{4}$$

$$\boldsymbol{\mu}_\omega^{\text{cond}} = \boldsymbol{\mu}_\omega + \mathbf{\Sigma}_\omega \mathbf{\Phi}_t \left(\mathbf{\Sigma}_t^\star + \mathbf{\Phi}_t^T \mathbf{\Sigma}_\omega \mathbf{\Phi}_t\right)^{-1} \left(\mathbf{y}_t^\star - \mathbf{\Phi}_t^T \boldsymbol{\mu}_\omega\right) \tag{5}$$

$$\mathbf{\Sigma}_\omega^{\text{cond}} = \mathbf{\Sigma}_\omega - \mathbf{\Sigma}_\omega \mathbf{\Phi}_t \left(\mathbf{\Sigma}_t^\star + \mathbf{\Phi}_t^T \mathbf{\Sigma}_\omega \mathbf{\Phi}_t\right)^{-1} \mathbf{\Phi}_t^T \mathbf{\Sigma}_\omega. \tag{6}$$

Moreover, trajectories can be combined or blended by using the probability product (see [3] for more details). For the purpose of this paper, we consider the combination of two trajectories, i.e., their intersection. Given two trajectories with the same Gaussian basis functions and parameters $\boldsymbol{\theta}_1 = \{\boldsymbol{\mu}_1, \mathbf{\Sigma}_1\}$, $\boldsymbol{\theta}_2 = \{\boldsymbol{\mu}_2, \mathbf{\Sigma}_2\}$, their combination/intersection is obtained as:

$$\boldsymbol{\mu}_{\text{comb}} = \mathbf{\Sigma}_2 \left(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2\right)^{-1} \boldsymbol{\mu}_1 + \mathbf{\Sigma}_1 \left(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2\right)^{-1} \boldsymbol{\mu}_2 \tag{7}$$

$$\mathbf{\Sigma}_{\text{comb}} = \mathbf{\Sigma}_1 \left(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2\right)^{-1} \mathbf{\Sigma}_2, \tag{8}$$

These combinations of ProMPs can also be used to sequentially condition several via-points. If two or more via-points are conditioned by sequentially using Eqs.(4) and (6), conditioning the second point over the result of the first conditioning usually results in a trajectory that does not necessarily preserve the first conditioned point. For this reason, in case of more than one conditioned point, it is more robust to apply Eqs. (4), (6) independently to each point wrt. the original ProMP parameters, and then combine all the resulting conditioned trajectories with Eqs. (7), (8). In Fig. 3, we see the effect of this procedure.

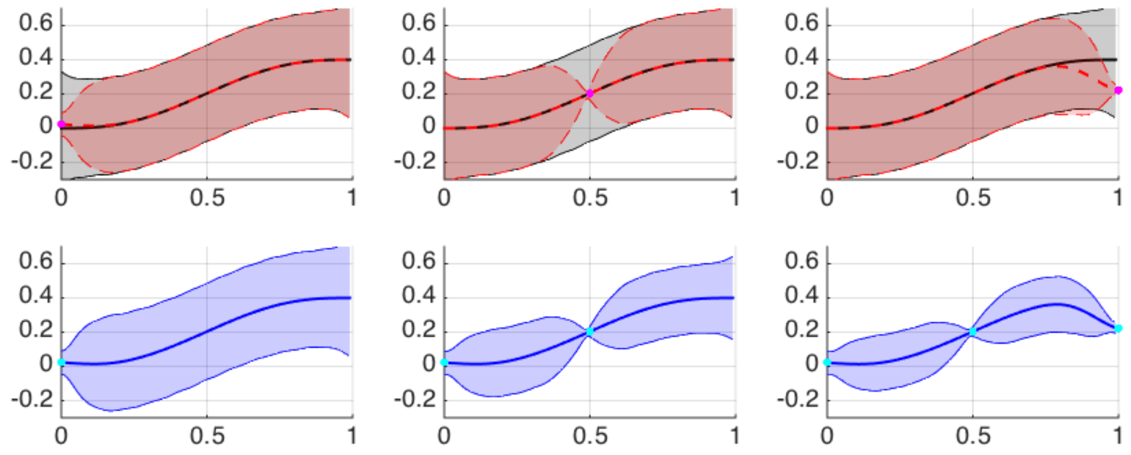
Fig. 3. Sequential conditioning of a ProMP with several via-points. The original trajectory (in black in the top row) is conditioned to three points, the resulting conditioned trajectories applying Eqs.(4), (6) to each point are shown in red, overlapping the original trajectory. The bottom row shows the result of applying Eqs. (7), (8) sequentially to combine the three conditioned trajectories to obtain a trajectory that goes through all the three conditioned points.

### C. Contextual ProMPs

Context variables usually represent elements of the task at hand that can be measured and, therefore, the robot can be told to adapt to such changing situation, the typical case being a changing via-point in a trajectory. In PS, many algorithms already consider such context variables [4], [5], [6]. The application of context variables to ProMPs can be done by having a large covariance in the context part of the trajectory and then conditioning the ProMP to it with Eqs. (4) (6), or by reparametrizing the parameters with context variables (denoted $\mathbf{s}$), using $\boldsymbol{\mu}_\omega(\mathbf{s}) = \boldsymbol{\mu}_\omega + \mathbf{K}_\omega \mathbf{s}$, where $\mathbf{K}_\omega$ is a matrix mapping the context variables into the trajectory mean parameter [9]. However, this representation does not encode a desired covariance change. Therefore, we studied better ways of representing context-dependent ProMP parameters, which will be described in next section.

## III. Demonstration-free Contextual ProMPs

As mentioned in the previous section, we aim at finding a good mapping from context variables into ProMPs parameters. Moreover, while demonstrations can generate a highly valuable ProMP initialization, they are not fully necessary in many cases, where the initial and final points of the trajectory are known, and so are a number of *interest points*, which are considered to be those points of the trajectory that can be contextualized, or are mandatory via-points.

In the case of a demonstration-free trajectory, we will initialize the trajectory, given an initial and a final position, by calculating the minimum-jerk trajectory between those positions. With the obtained trajectory, we will fit the initial ProMP mean $\boldsymbol{\mu}_\omega$. And we will set the initial ProMP covariance $\boldsymbol{\Sigma}_\omega$ arbitrarily, usually by setting $\boldsymbol{\Sigma}_\omega = K\mathbf{I}_{N_f \cdot d}$, where $K$ is the variance along each one of the parameters of the trajectory, and $N_f$ is the number of Gaussian kernels used per each one of them. The resulting trajectory will be similiar to the intial one used for the example in Fig.3, shown in black in the background of the top row.

### A. Trajectory interest points

Besides an initial and a final position, we will set a number $N_p$ of *interest points*. These points are the ones where we will allow context variables to change a trajectory and are tagged by the timestamp at which they occur. As an example, Fig. 3 has three interest points, located at $t = 0, 0.5, 1$. These points will be separated as fixed points or variable points. While fixed points will never change and are *hard* via-points, *exploration* interest points will be parametrized by a context variable that will reparametrize the ProMP depending on its value.

#### 1) Fixed points

We will condition the initial min-jerk trajectory through the fixed points with a corresponding variance by using Eqs. (4) (6), and (7), (8) if more than one fixed point is set. The variance imposed will usually be small. Given $\boldsymbol{\Sigma}_t = \boldsymbol{\Phi}_t^T \boldsymbol{\Sigma}_\omega \boldsymbol{\Phi}_t$ the variance from the initialization at time $t$, we can set the imposed variance as $\boldsymbol{\Sigma}_t^\star = \lambda^2 \boldsymbol{\Sigma}_t$, where $\lambda$ is a scalar, usually around $\lambda = 10^{-2}$ in order to reduce the variance of such via-point.

#### 2) Exploration points

Exploration points will be treated similarly to fixed points, with the difference that, for each interest point to contextualize, $p = 1..N_p^e$, we will sample trajectories artificially to build a relation model with the ProMPs parameters, as explained next in Section III-B.

### B. Synthetic data generation

Once the ProMP has been initialized by setting an initial position, a final position, and all the Gaussian kernels with their initial weight covariance matrix, and it has been conditioned to the fixed interest points, we will build the mapping from the context variables, i.e., exploration interest points, to the trajectory weights. In order to do so, we will generate a certain amount of synthetic data. Under the assumption that the desired operational space covariance matrix $\boldsymbol{\Sigma}_{t_p}^\star$

will always be that of the initial trajectory multiplied by a scalar, i.e., $\mathbf{\Sigma}^{\star}_{t_p} = \lambda_p^2 \mathbf{\Sigma}_{t_p}$, for each exploration interest point $p = 1..N_p^e$. Then, eqs. (4),(6) can be simplified to:

$$\boldsymbol{\mu}_\omega^{\text{cond}} = \boldsymbol{\mu}_\omega + \mathbf{\Phi}_t^{T\dagger} \frac{\mathbf{s}_p}{1+\lambda_p^2} \tag{9}$$

$$\mathbf{\Sigma}_\omega^{\text{cond}} = \mathbf{\Sigma}_\omega \left( \mathbf{I}_{d\cdot N_f} + \frac{\mathbf{\Phi}_t \left( \mathbf{\Phi}_t \mathbf{\Sigma}_\omega \mathbf{\Phi}_t^T \right)^{-1} \mathbf{\Phi}_t^T \mathbf{\Sigma}_\omega \mathbf{\Phi}_t^T}{1+\lambda_p^2} \right), \tag{10}$$

where $\mathbf{s}_p = \mathbf{y}^{\star}_{des} - \mathbf{\Phi}_t^T \boldsymbol{\mu}_\omega$ is the context variable, centered on the initial trajectory value $\mathbf{\Phi}_t^T \boldsymbol{\mu}_\omega$.

While the conditioned covariance does not simplify much wrt. Eq.(6), the mean in Eq. (9) does, showing a linear dependency on both terms $\mathbf{s}_p$ and $\xi_p = \frac{1}{1+\lambda_p^2}$. This allows for a faster conditioning in order to generate synthetic data.

Therefore, for each of the $N_p^e$ exploration interest points, we have the covariance ratio $\xi_p$, encoding the covariance variation, and the context variable $\mathbf{s}_p$ (with a dimension equal to the number of dimensions $d$ of the ProMP). Appending such variables we can define $\boldsymbol{\xi} = \left[\xi_1; ...; \xi_{N_p^e}\right]$ and $\mathbf{s} = \left[\mathbf{s}_1; ...; \mathbf{s}_{N_p^e}\right]$.

In order to generate synthetic data, we will define an exploration ratio $\sigma_\lambda$, so that $\lambda_p \sim \mathbf{U}[0, \sigma_\lambda]$ is a uniform distribution. Similarly, we will use $\mathbf{s}_p \sim \mathcal{N}\left(\mathbf{y}_{t_p}, \mathbf{\Sigma}_{t_p}\right) = \mathcal{N}\left(\mathbf{\Phi}_{t_p}^T \boldsymbol{\mu}_\omega, \mathbf{\Phi}_{t_p}^T \mathbf{\Sigma}_\omega \mathbf{\Phi}_{t_p}\right)$.

We will generate a number $N_{\text{data}}$ of sample trajectories, and fit a contextual ProMP as described in the following section III-C.

## C. Contextual ProMPs fitting

When trying to fit a contextual ProMP model, we started by the one defined in [9], and expanded it so as to better represent covariance. In order to fit the trajectory mean, we define the context-dependent ProMP mean as in [9]:

$$\boldsymbol{\mu}_\omega(\mathbf{s}) = \boldsymbol{\mu}_\omega^0 + \mathbf{K}_\omega \hat{\mathbf{s}}, \tag{11}$$

where $\boldsymbol{\mu}_\omega^0$ is the initial ProMP mean after conditioning the fixed interest points. $\mathbf{K}_\omega$ is a matrix that maps $\hat{\mathbf{s}}$ into the new mean weights by using the pseudoinverse operator †:

$$\mathbf{K}_\omega = \begin{bmatrix} \hat{\mathbf{s}}^1 \\ ... \\ \hat{\mathbf{s}}^{N_k} \end{bmatrix}^{\dagger} \begin{bmatrix} (\boldsymbol{\mu}_\omega^1 - \boldsymbol{\mu}_\omega^0)^T \\ ... \\ (\boldsymbol{\mu}_\omega^{N_k} - \boldsymbol{\mu}_\omega^0)^T \end{bmatrix} \tag{12}$$

Moreover, $\hat{\mathbf{s}}$ includes the powers of the values of $\mathbf{s}$ up to a degree $\alpha$, representing a polynomial fitting of the mean.

In order to capture the covariance properly, given $\boldsymbol{\omega} \sim \mathcal{N}\left(\boldsymbol{\mu}_\omega(\hat{\mathbf{s}}), \mathbf{\Sigma}_\omega(\hat{\mathbf{s}}, \boldsymbol{\xi})\right) = \mathcal{N}\left(\boldsymbol{\mu}_\omega^0 + \mathbf{K}_\omega \hat{\mathbf{s}}, \mathbf{\Sigma}_\omega(\hat{\mathbf{s}}, \boldsymbol{\xi})\right)$, we know from Eq. (37) in [12], that the transferred effect of the variance $\mathbf{\Sigma}_{\hat{s}}$ will be $\mathbf{K}_\omega \mathbf{\Sigma}_{\hat{s}} \mathbf{K}_\omega$. However, the transferred effect of $\boldsymbol{\xi}$ remains unknown and, if not considered, a variance that we might expect to be smaller that $\mathbf{\Sigma}_\omega^0$ would actually be larger. Therefore, we will fit a model of the form

**Algorithm 1** Contextualized ProMP generation

**Input:**
$\boldsymbol{\mu}_\omega^0$, $\mathbf{\Sigma}_\omega^0$ initialized ProMP with fixed interest points already conditioned.
$t_1, ..., t_{N_p^e}$ timestamps of exploration condition points
$\alpha$ polynomial fitting degree
$\sigma_\lambda$ allowed precision deviation

```
1: for $k = 1..N_k$ do
2:     Sample $\mathbf{s}_p^k \sim \mathcal{N}\left(\mathbf{\Phi}_{t_p}^T \boldsymbol{\mu}_\omega^0, \mathbf{\Phi}_{t_p}^T \mathbf{\Sigma}_\omega^0 \mathbf{\Phi}_{t_p}\right)$, $p = 1..N_p^e$, and $\lambda_p^k \sim \mathbf{U}[0, \sigma_\lambda]$
3:     Generate $\boldsymbol{\mu}_\omega^k$, $\mathbf{\Sigma}_\omega^k$ by sequentially conditioning and combining the exploration interest points with $\{\mathbf{s}_p^k, \xi_p^k = \frac{1}{1+(\lambda_p^k)^2}\}_{p=1..N_p^e}$
4:     Store $\hat{\mathbf{s}}_p^k$ with up to the $\alpha$-th power of $\mathbf{s}_p^k$ for each exploration interest point
5:     Store $\hat{\mathbf{s}}^k = [\hat{\mathbf{s}}_1^k; ...; \hat{\mathbf{s}}_{N_p^e}^k]$ and $\boldsymbol{\xi}^k = [\boldsymbol{\xi}_1^k; ...; \boldsymbol{\xi}_{N_p^e}^k]$
6: end for
7: Compute $\mathbf{\Sigma}_{\hat{s}} = \text{covariance}(\hat{\mathbf{s}})$
8: Compute $\mathbf{K}_\omega$ with Eq.(12)
9: Compute $\mathbf{\Lambda}$ with Eq.(14)
10: Compute $\mathbf{\Lambda}_p$ with Eq.(15), $\forall p = 1..N_p^e$
```

$$\mathbf{\Sigma}_\omega(\mathbf{s}, \boldsymbol{\xi}) = \mathbf{\Sigma}_\omega^0 + \mathbf{K}_\omega \mathbf{\Sigma}_{\hat{s}} \mathbf{K}_\omega^T + \sum_{p=1}^{N_p} \xi_p \mathbf{\Lambda}_p, \tag{13}$$

where $\mathbf{\Lambda}_p$ are matrices with the size of $\mathbf{\Sigma}_\omega$ that we will fit through least squares:

$$\mathbf{\Lambda} = \begin{bmatrix} (\boldsymbol{\xi}^1)^T \\ ... \\ (\boldsymbol{\xi}^{N_k})^T \end{bmatrix}^{\dagger} \begin{bmatrix} (\text{vec}(\mathbf{\Sigma}_\omega^1 - \mathbf{\Sigma}_\omega^1 - \mathbf{K}_\omega \mathbf{\Sigma}_{\hat{s}} \mathbf{K}_\omega^T))^T \\ ... \\ (\text{vec}(\mathbf{\Sigma}_\omega^{N_k} - \mathbf{\Sigma}_\omega^1 - \mathbf{K}_\omega \mathbf{\Sigma}_{\hat{s}} \mathbf{K}_\omega^T))^T \end{bmatrix}, \tag{14}$$

where vec is the vectorization of a matrix, and then

$$\mathbf{\Lambda}_p = \text{vec}^{-1}(\mathbf{\Lambda}(p, :)), \tag{15}$$

where $\text{vec}^{-1}$ is the inverse operation of the vectorization (transforming the vectorized matrix into a matrix again).

The procedure for fitting the model provided by Eqs.(11) and (13) is summarized in Algorithm 3.

## IV. Obstacle avoidance for ProMPs

Another feature to be added to ProMPs is the capability of avoiding obstacles in a trajectory. While obstacle avoidance has been studied through different perspectives on DMPs [2], its application on ProMPs is not as clear. In [13], a collision-free ProMP is learned by maximizing the distance to the obstacle while staying close to the data by using the demonstrated data. Here, we propose a method for obtaining a collision-free trajectory within the ProMP distribution obtained directly from the ProMP parameters, which can be obtained without the need of data samples.

When trying to avoid an obstacle located at a position $\mathbf{y}_o$, a simple approach can be to condition the ProMP so as to go through a point which is at a certain threshold distance $D$ from it. However, there are infinite many solutions of points at a distance $D$ of an obstacle. Therefore, a criterion for deciding which one to take is needed.

In our case, given a ProMP characterized by $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega\}$, we find a weight $\boldsymbol{\omega}$ such that is as close as possible to the mean trajectory $\boldsymbol{\mu}_\omega$, while at a distance $D$ from the obstacle at a certain time $t$, corresponding to the closest point of the trajectory to the obstacle. However, this might result in a trajectory with very low probability wrt. the ProMP. Therefore, we also limit the Mahalanobis distance (usually with one or two standard deviations) in order to obtain a trajectory within the distribution. Therefore, we will find the solution $\boldsymbol{\omega}$ of:

$$\begin{aligned} &\text{argmin}_{\boldsymbol{\omega}} \left(\boldsymbol{\omega} - \boldsymbol{\mu}_\omega\right)^T \cdot \left(\boldsymbol{\omega} - \boldsymbol{\mu}_\omega\right) \\ &\text{s.t.} \ \left(\mathbf{y}_o - \boldsymbol{\Phi}_t^T \boldsymbol{\omega}\right)^T \mathbf{M} \left(\mathbf{y}_o - \boldsymbol{\Phi}_t^T \boldsymbol{\omega}\right) = D^2 \\ &\qquad D_M(\boldsymbol{\Phi}_t^T \boldsymbol{\omega}, \boldsymbol{\Phi}_t^T \boldsymbol{\mu}_\omega) \leq b^2 \end{aligned} \tag{16}$$

where $\mathbf{M}$ is a metric matrix that will be used to prioritize the dimensions in which we want to avoid the obstacle, and will usually be defined as a diagonal matrix $\mathbf{M} = \text{diag}(\mathbf{v})/\|\mathbf{v}\|$. $D_M(\boldsymbol{\Phi}_t^T \boldsymbol{\omega}, \boldsymbol{\Phi}_t^T \boldsymbol{\mu}_\omega)$ is the Mahalanobis distance between $\boldsymbol{\Phi}_t^T \boldsymbol{\omega}$ and the ProMP distribution $\mathcal{N}\left(\boldsymbol{\Phi}_t^T \boldsymbol{\mu}_\omega, \boldsymbol{\Phi}_t^T \boldsymbol{\Sigma}_\omega \boldsymbol{\Phi}_t\right)$:

$$\begin{aligned} &D_M(\boldsymbol{\Phi}_t^T \boldsymbol{\omega}, \boldsymbol{\Phi}_t^T \boldsymbol{\mu}_\omega) = \\ &= \left(\boldsymbol{\Phi}_t^T \boldsymbol{\omega} - \boldsymbol{\Phi}_t^T \boldsymbol{\mu}_\omega\right)^T \left(\boldsymbol{\Phi}_t^T \boldsymbol{\Sigma}_\omega \boldsymbol{\Phi}_t\right)^{-1} \left(\boldsymbol{\Phi}_t^T \boldsymbol{\omega} - \boldsymbol{\Phi}_t^T \boldsymbol{\mu}_\omega\right) \end{aligned} \tag{17}$$

In Fig. 4, we show the behavior of the proposed obstacle avoidance framework for a 2-dimensional ProMP with $N_f = 12$ Gaussians per DoF. We initialized the trajectory with one fixed point at $t = 0.5$, resulting in the ProMP in yellow, with initial position 0.0 and final position 0.4 for each DoF. We simulated $N_{\text{data}} = 500$ trajectories, with $\sigma_\lambda = 0.2$ and used a third-order fitting ($\alpha = 3$). We obtained the model in Eqs. (11), (13) and conditioned to a new point, resulting in the black ProMP. Then, with the same obstacle (in red), using a distance $D = 0.15$ and Mahalanobis distance of $b = 1$ standard deviation, we can see the results of the optimization. Those are the same when using the Mahalanobis constraint or not in the first case, where both components have equal importance. However, if we give more importance to the $x$ component in a $8:1$ ratio (middle column), the green line (without the Mahalanobis constraint) presents a very large deviation from the mean conditioned trajectory (in black). Using the Mahalanobis constraint in this case results in a trajectory with less deviation from the mean. If all responsibility for avoiding the obstacle is given to the first component (right column), we cannot find a solution with the Mahalanobis constraint, due to it being more than one standard deviation away from the mean, as is the green line.

To conclude, in the case of an already contextualized ProMP that we want to use to avoid obstacles, considering the ProMP parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega\}$ and the resulting parameters after using the model in Eqs. (11), (13), $\boldsymbol{\theta}^{\text{cond}} = \{\boldsymbol{\mu}_\omega^{\text{cond}}, \boldsymbol{\Sigma}_\omega^{\text{cond}}\}$, We then minimize $\left(\boldsymbol{\omega} - \boldsymbol{\mu}_\omega^{\text{cond}}\right)^T \cdot \left(\boldsymbol{\omega} - \boldsymbol{\mu}_\omega^{\text{cond}}\right)$ in the system defined in Eq. (16).

The application to several obstacles can be performed by solving independently the system in Eq. (16), adding one constraint for each obstacle present:

$$\left(\mathbf{y}_{o_i} - \boldsymbol{\Phi}_t^T \boldsymbol{\omega}\right)^T \mathbf{M} \left(\mathbf{y}_{o_i} - \boldsymbol{\Phi}_t^T \boldsymbol{\omega}\right) = D_i^2. \tag{18}$$

We also tested the proposed methods in this paper in the set-up in Fig. 1, where the Barrett WAM arm had to follow a nominal straight-line, consisting on a 2-dimensional trajectory on the X-Y plane with a fixed initial point and a variable end point. Using our approach, we conditioned the end point to be 10cm away and then added two obstacles (small bottles) observed by a Kinect camera. The resulting well-conditioned obstacle-avoiding trajectories were computed by giving all the priority to the Y component, and are shown in Fig. 5. Real-robot execution can be seen at `http://www.iri.upc.edu/groups/perception/#ProMPobstacles`.

## V. Conclusions

In this paper, we presented a framework for generating Probabilistic Movement Primitives (ProMPs) with context variables, without the need of a large number of human demonstrations. While a few demonstrations can be used to obtain an initial guess of a motion encoded as a ProMP, its generalization to context variables often requires a much larger number of motion samples. The proposed method for defining interest points, sampling them and building a model efficiently saves a lot of effort to the user.

Moreover, we proposed an obstacle avoidance method for ProMPs, based on a quadratic optimization with a small computational cost. The application of such obstacle avoidance over a conditioned trajectory was tested in both simulated and real-robot experiments, showing the correct conditioning of the trajectory by using the synthetic model, as well as the effectiveness of the method in avoiding obstacles. While the method scales well to higher dimensions, a deeper study on the effect of the ratio matrix $M$ in Eq. (16) will be developed further in the future.

## References

[1] A. J. Ijspeert, J. Nakanishi and S. Schaal, “Movement Imitation with Nonlinear Dyamical Systems in Humanoid Robots”. *IEEE ICRA*, pp 1398-1403, 2002.

[2] 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.

[3] A. Paraschos, G Neumann, C. Daniel, and J. Peters, “Probabilistic movement primitives". *In Advances in NIPS*, Cambridge, MA: MIT Press., 2013.

[4] A. Fabisch and J. H. Metzen, "Active Contextual Policy Search". *J. Mach. Learn. Res.*, vol. 15, no. 1, pp 3371-3399, 2014.

[5] A. Abdolmaleki, D. Simões, N. Lau, L. P. Reis and G. Neumann, "Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation," *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 94-99, 2016.

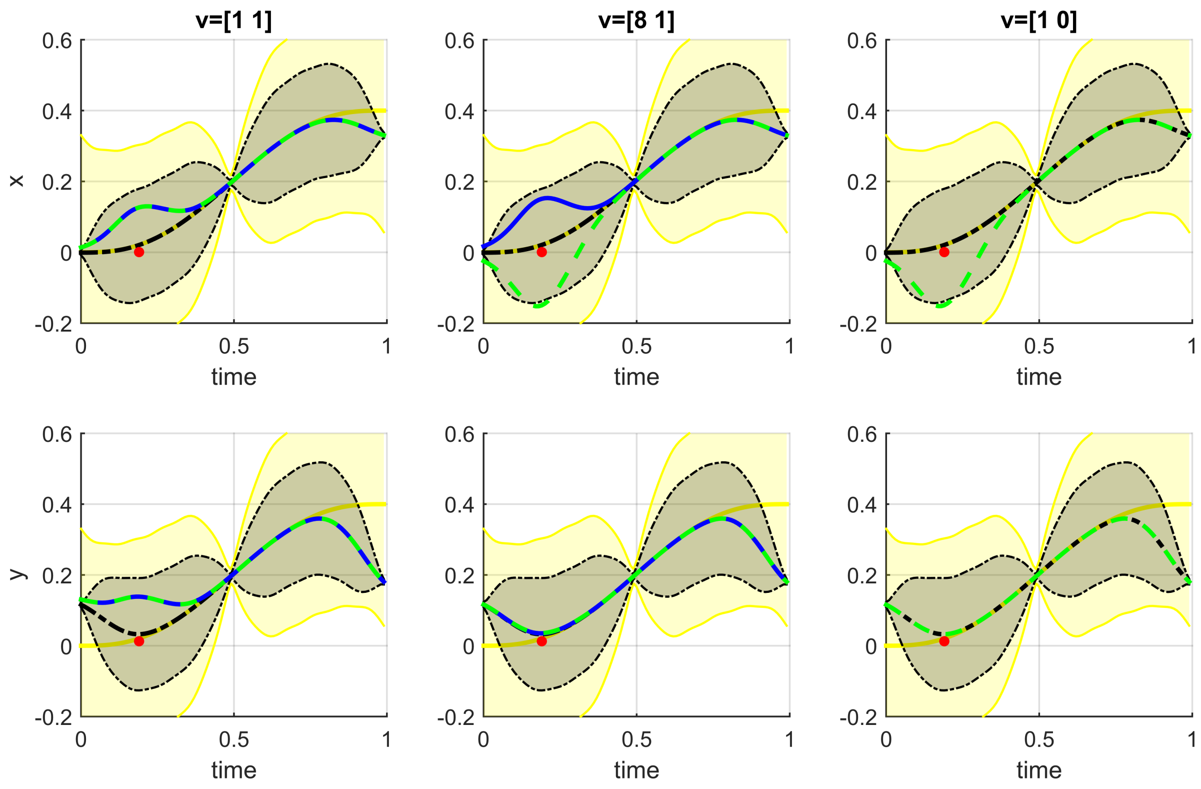

Fig. 4. Proposed obstacle avoidance for a 2-dimensional ProMP with different metrics matrices. Top row shows the first component, while second row shows the second component. The black curves and area represent the ProMP after conditioning the exploration points, the red dot represents the obstacle, while the blue line corresponds to the solution of the obstacle avoidance system in Eq.(16) (where there is a solution), and the green line is the result without the Mahalanobis constraint.

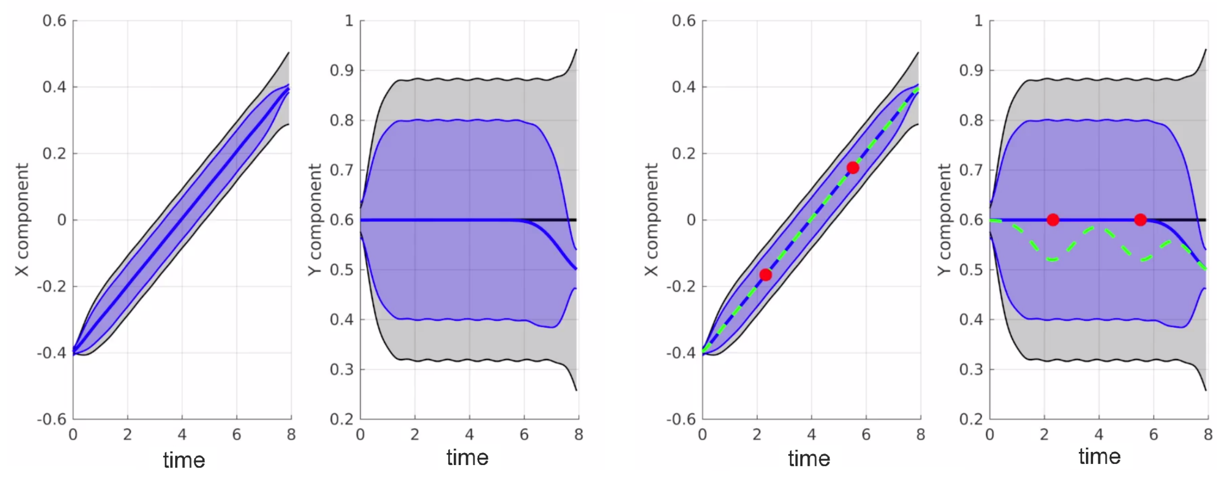

Fig. 5. Experiment with the Barrett WAM robot in the set-up shown in Fig. 1. X and Y components of the generated trajectories. Plots on the left show the initial trajectory with a fixed point at $t = 0$ (in black), and conditioned to an exploration point in $t = 8$ (in blue). On the right we see the X and Y components of the same trajectories, and the result after using obstacle avoidance (in discontinuous green) on the contextual trajectory to avoid two obstacles (red dots) in the Y domain.

**Algorithm 2** Generic ProMP initialization

```
Input:
Initial and final position $\mathbf{y}_0$, $\mathbf{y}_{N_t}$
Interest points timestamps $t_1, ..., t_{N_p}$
Precision at timestamps $\lambda \sim 10^{-2}$
1: Generate min-jerk trajectory $\boldsymbol{\tau}_0$ from $\mathbf{y}_0$ to $\mathbf{y}_{N_t}$
2: Use $\boldsymbol{\tau}_0$ to obtain $\boldsymbol{\mu}_\omega^0$.
3: $\boldsymbol{\Sigma}_\omega^0 = K \cdot \mathbf{I}$
4: for $p = 1..N_p$ do
5:     Condition ProMP $\{\boldsymbol{\mu}_\omega^0, \boldsymbol{\Sigma}_\omega^0\}$ to $\mathbf{y}_p$ and variance $\boldsymbol{\Sigma}_{t_p}^\star = \lambda_p^2 \boldsymbol{\Sigma}_{t_p}$, obtaining $\{\hat{\boldsymbol{\mu}}_\omega^p, \hat{\boldsymbol{\Sigma}}_\omega^p\}$
6:     $\{\boldsymbol{\mu}_\omega^p, \boldsymbol{\Sigma}_\omega^p\} = \text{Blend}\left(\{\hat{\boldsymbol{\mu}}_\omega^p, \hat{\boldsymbol{\Sigma}}_\omega^p\}, \{\boldsymbol{\mu}_\omega^{p-1}, \boldsymbol{\Sigma}_\omega^{p-1}\}\right)$
7: end for
```

**Algorithm 3** Contextualized ProMP generation

```
Input:
$\boldsymbol{\mu}_\omega^0$, $\boldsymbol{\Sigma}_\omega^0$ initialized ProMP with fixed interest points already conditioned.
$t_1, ..., t_{N_p^e}$ timestamps of exploration condition points
$\alpha$ polynomial fitting degree
$\sigma_\lambda$ allowed precision deviation
1: for $k = 1..N_k$ do
2:     Sample $\mathbf{s}_p^k \sim \mathcal{N}\left(\boldsymbol{\Phi}_{t_p}^T \boldsymbol{\mu}_\omega^0, \boldsymbol{\Phi}_{t_p}^T \boldsymbol{\Sigma}_\omega^0 \boldsymbol{\Phi}_{t_p}\right)$, $p = 1..N_p^e$, and $\lambda_p^k \sim \mathbf{U}[0, \sigma_\lambda]$
3:     Generate $\boldsymbol{\mu}_\omega^k$, $\boldsymbol{\Sigma}_\omega^k$ by sequentially conditioning and combining the exploration interest points with $\{\mathbf{s}_p^k, \xi_p^k = \frac{1}{1+(\lambda_p^k)^2}\}_{p=1..N_p^e}$
4:     Store $\hat{\mathbf{s}}_p^k$ with up to the $\alpha$-th power of $\mathbf{s}_p^k$ for each exploration interest point
5:     Store $\hat{\mathbf{s}}^k = [\hat{\mathbf{s}}_1^k; ...; \hat{\mathbf{s}}_{N_p^e}^k]$ and $\boldsymbol{\xi}^k = [\boldsymbol{\xi}_1^k; ...; \boldsymbol{\xi}_{N_p^e}^k]$
6: end for
7: Compute $\boldsymbol{\Sigma}_{\hat{s}} = \text{covariance}(\hat{\mathbf{s}})$
8: Compute $\mathbf{K}_\omega$
9: Compute $\boldsymbol{\Lambda}$
10: Compute $\boldsymbol{\Lambda}_p$, $\forall p = 1..N_p^e$
```

[6] A. Kupcsik, M. Deisenroth, J. Peters and G. Neumann, "Data-Efficient Generalization of Robot Skills with Contextual Policy Search". *Proceedings of the AAAI Conference*, pp. 1401-1407, 2013.

[7] M. P. Deisenroth, G. Neumann and J. Peters, “A survey on Policy Search for Robotics". *Foundations and Trends in Robotics*, vol 2, pp 1-142, 2013.

[8] C. Daniel, G. Neumann,O. Kroemer and J. Peters, "Hierarchical Relative Entropy Policy Search". *Journal of Machine Learning Research*, vol. 17, no. 93, pp 1-50, 2016.

[9] A. Colomé, G. Neumann, J. Peters and C. Torras "Dimensionality Reduction for Probabilistic Movement Primitives". *IEEE-RAS Humanoid Robots*, pp. 794-800, 2014.

[10] F. Stulp, E. Oztop, P. Pastor, M. Beetz and S. Schaal, "Compact models of motor primitive variations for predictable reaching and obstacle avoidance," *IEEE-RAS Humanoid Robots*, pp. 589-595, 2009.

[11] J. Löfberg, "YALMIP Matlab Library", [online] available: `https://yalmip.github.io/download/`.

[12] M. Toussaint, “Lecture Notes: Gaussian identities". [online] Available: `http://ipvs.informatik.uni-stuttgart.de/mlr/marc/notes/gaussians.pdf`

[13] D. Koert, G. Maeda, R. Lioutikov, G. Neumann and J. Peters, "Demonstration Based Trajectory Optimization for Generalizable Robot Motions", *IEEE-RAS Humanoid Robots*, pp. 515-522, 2016.