Reinforcement Learning of Trajectory Distributions: Applications in Assisted Teleoperation and Motion Planning

Marco Ewerton[1,2], Guilherme Maeda[3,4], Dorothea Koert[1], Zlatko Kolev[1], Masaki Takahashi[5] and Jan Peters[1,6]

***Abstract*— The majority of learning from demonstration approaches do not address suboptimal demonstrations or cases when drastic changes in the environment occur after the demonstrations were made. For example, in real teleoperation tasks, the demonstrations provided by the user are often suboptimal due to interface and hardware limitations. In tasks involving co-manipulation and manipulation planning, the environment often changes due to unexpected obstacles rendering previous demonstrations invalid. This paper presents a reinforcement learning algorithm that exploits the use of relevance functions to tackle such problems. This paper introduces the Pearson correlation as a measure of the relevance of policy parameters in regards to each of the components of the cost function to be optimized. The method is demonstrated in a static environment where the quality of the teleoperation is compromised by the visual interface (operating a robot in a three-dimensional task by using a simple 2D monitor). Afterward, we tested the method on a dynamic environment using a real 7-DoF robot arm where distributions are computed online via Gaussian Process regression.**

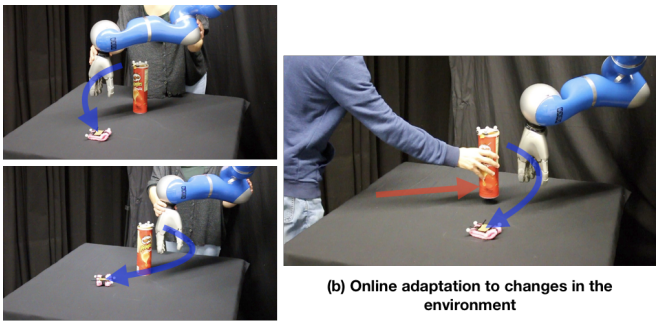


Fig. 1: In this experiment, the robot has to reach the target (pink object) while avoiding the obstacle (Pringles can). **(a)** Demonstrations. **(b)** Our learning system is able to adapt on the fly to changes in the environment.

I. Introduction

Human-robot co-manipulation and teleoperation can greatly benefit from learning from demonstration since users can easily demonstrate trajectories to a robot. These trajectories are then fitted to a model that can be later used during the execution of repetitive tasks to provide assistance and to compensate communication latency and intermittency. This paper provides a reinforcement learning algorithm, **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO), to improve upon demonstrated trajectories when these are suboptimal or when solutions to new situations must be found. The main feature of PRO is that it preserves the variance of policy parameters that are not relevant to the objective component currently being optimized for. The usefulness of this property is easily understood when optimizing with respect to components whose influence are only local, such as the many via-points along a trajectory.

The concept of relevance function first appeared in [1] where the metric of relevance was computed with a computation-intensive iterative algorithm which required the design of basis functions for the relevance. In the present paper, we investigate the use of Pearson correlation [2]. Pearson correlation allows the optimizer to determine relevance functions faster while eliminating some of the open parameters found in [1]. In the later part of the paper, we extend PRO with Gaussian Processes (GP) regression to cope with dynamic environments in an online fashion. PRO is used to optimize the GP inferences as a mapping from environment to trajectory distribution.

The main contribution of this paper is a robot control algorithm to improve suboptimal demonstrations. In detail, we introduce the use of Pearson correlation to compute relevance functions and show how the method can be combined with GP regression to enable online corrections. Experiments in the context of assisted teleoperation and other human-robot interaction tasks are used to validate two different cases where demonstrations are suboptimal. [1]

[1]Intelligent Autonomous Systems Group, Department of Computer Science, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {ewerton, koert, peters}@ias.tu-darmstadt.de, zlatko.kolev@stud.tu-darmstadt.de

[2] Idiap Research Institute, Martigny, Switzerland marco.ewerton@idiap.ch

[3]ATR Computational Neuroscience Laboratory, Department of Brain Robot Interface (BRI), 2-2-2 Hikaridai Seika-sho, Soraku-gun Kyoto 619-0288, Japan g.maeda@atr.jp

[4]Preferred Networks Inc. Tokyo, Japan gjmaeda@preferred.jp

[5]Keio University, Faculty of Science and Technology, Department of System Design Engineering, Takahashi Laboratory, 3-14-1 Hiyoshi Kohoku-ku, Yokohama-shi, Kanagawa 223-8522, Japan takahashi@sd.keio.ac.jp

[6]Max Planck Institute for Intelligent Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany jan.peters@tuebingen.mpg.de

[1]Although the method is not limited to a particular policy representation, in this paper we refer to the term "policy" as a "trajectory" and "policy search" means "optimizing a distribution of trajectories with respect to certain objectives". The optimization objectives are related to via points, obstacles, length and jerkiness of the movement.

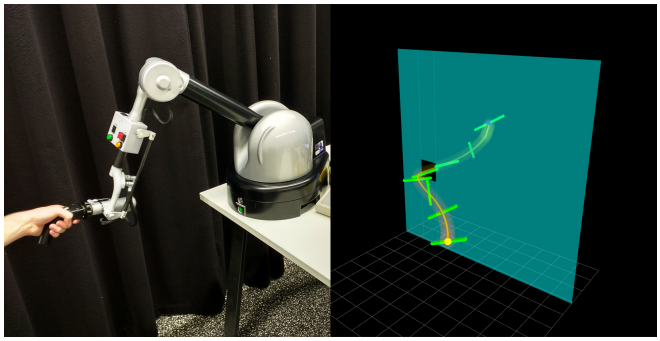


Fig. 2: A haptic device, the Haption Virtuose 6D, is operated by a user to move a beam in a virtual environment. The haptic device uses a trajectory distribution learned with **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) to assist the user in moving the beam from a start position and orientation to an end position and orientation through a window in the wall.

An extended version of this paper with additional experiments and evaluation is under preparation for a journal.

II. Related Work

A method for assisting users in shared-autonomy tasks, e.g. co-manipulation, with probabilistic models learned from demonstrations has been proposed in [3]. In that paper, Gaussian Mixture Models [4] are used to create multiple probabilistic virtual guides which constrain the movements of the user to a region close to the demonstrations. Also using probabilistic models, in [5] users are assisted in teleoperation tasks with shared control. In that work, task-parameterized Gaussian Mixture Models (TP-GMMs) [6] have been used to encode the probability distribution of demonstrated trajectories. Gaussian Mixture Regression (GMR) [6] has been used to generate a behavior according to the learned model. The learning agent assists the user with the teleoperation of a device to scan a surface.

Our work relates to [3] and [5], with the important difference that our approach addresses cases where demonstrations are suboptimal or when the learned model cannot generalize well enough to a new scenario. This is possible due to the use of reinforcement learning as a mechanism to manipulate the original distribution. An approach for improving upon suboptimal initial demonstrations is presented in [7]. Nevertheless, that approach is based on iterative refinement by the human user rather than reinforcement learning.

In the context of motion planning, PRO presents similarities to STOMP [8], a black-box stochastic optimizer for motion planning, in the sense that PRO is gradient-free and also relies on a stochastic trajectory optimization technique. However, while in STOMP trajectories are generated by perturbing an initial trajectory with a certain noise, in our work, a distribution of trajectories based on demonstrations (and potentially also on prior knowledge) is optimized.

Learning from demonstrations has also been applied in supervisory control. In this paradigm, after training, the remote system can execute a task autonomously, needing only high-level task goals to be provided by the user. In [9], task-parameterized hidden semi-Markov models (TP-HSMMs) are used to build probabilistic models of manipulation motions and Model Predictive Control (MPC) is used to execute these motions. In that work, TP-HSMMs have been shown to generalize better to changes in the environment than Probabilistic Movement Primitives (ProMPs) [10], which are used in our work. We believe that our work can contribute to enhancing the generalization capabilities of frameworks using probabilistic models such as TP-HSMM and ProMP by using reinforcement learning to let the remote system look for solutions to new tasks by trial and error.

To support online obstacle avoidance, PRO performs reinforcement learning while making use of Gaussian Processes to output trajectory distributions trained via supervised learning. This process resembles the way Guided Policy Search (GPS) [11] uses trajectory optimization in combination with the constraint that the actions output by a Convolutional Neural Network (CNN) must track the optimized trajectories. In our approach, PRO assumes the role of the trajectory optimizer while the GPs assumes the role of the CNN. In contrast to GPS, while the CNN outputs actions for any given state, in our approach, GP regression outputs ProMPs (distributions of trajectories) for any given environment.

III. Pearson-Correlation-Based Relevance Weighted Policy Optimization

Pearson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) is a stochastic policy search algorithm based on Reward Weighted Regression [12] where, at each iteration, policy parameters are sampled from a probability distribution. The probability distribution is then optimized to maximize the expected reward. Differently from RWR, PRO estimates the relevance of each policy parameter with respect to each objective. This information is then used to adapt the sampling pattern such that undesirable changes in the distribution of policy parameters are avoided. For example, Fig. 3 illustrates the difference between RWR and PRO in a 2D parameter space where the optimal solution is not affected by the value of one of the parameters. In the case of RWR, the optimal parameters collapse to a single point while PRO preserves the distribution of the irrelevant dimensions. In the motion planning context, this larger final distribution reflects as final richer distributions covering larger areas of the workspace.

A. Measuring Relevance via Pearson Correlation

The relevance of the policy parameter w_n to the objective o, denoted by $f_o(n)$, can be represented by the absolute value $|\rho_{n,o}|$ of the Pearson correlation[2] coefficient

$$\rho_{n,o} = \frac{\mathrm{Cov}(w_n, o)}{\sigma_{w_n}\sigma_o}, \tag{1}$$

[2]The Pearson correlation coefficient $\rho_{X,Y}$ of any two random variables X and Y is a measure of the linear correlation between X and Y and $-1 \leq \rho_{X,Y} \leq 1$.

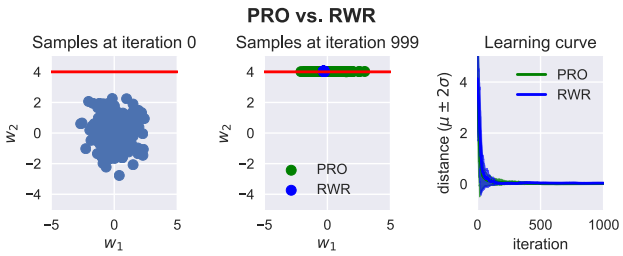


Fig. 3: **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) versus Reward Weighted Regression (RWR). Here, w_1 and w_2 are the policy parameters. The red line represents the region in the space of policy parameters where the reward is the maximum. The reward for any point in this space is $R = \exp(-\beta d)$, where β is a hyperparameter chosen by the user and d is the distance between the point and the red line. Both RWR and PRO were applied to optimize a Gaussian distribution of $\boldsymbol{w} = [w_1, w_2]^T$ with 1000 iterations and 200 samples per iteration. The variances of RWR collapse while PRO is able to keep the variance of w_1 because this parameter is not relevant to this optimization problem. In this case, the difference between the learning curves of these algorithms is nevertheless negligible.

where $\text{Cov}(w_n, o)$ is the covariance between w_n and the value of the objective o, σ_{w_n} is the standard deviation of w_n and σ_o is the standard deviation of the values of the objective o. A policy, e.g. a trajectory, is represented by the vector $\boldsymbol{w} = [w_1, \cdots, w_N]^\top$, where N is the number of policy parameters. For each $\boldsymbol{w}$ sampled from a distribution, e.g. a Gaussian with small and uniform variance such that the assumption of linear correlation holds, the computation of the value of each objective o is made. Subsequently, $\rho_{n,o}$ is computed.

The relevance $f_o(n)$ of w_n to the objective o expresses how strongly changes in w_n are linearly correlated to changes in the values of the objective o. In our implementation we found useful to normalize the relevance function $f_o(n) = f_o(n) / \max_n f_o(n)$.

The distribution of policy parameters is assumed to be Gaussian, i.e. $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})$. As such, PRO samples w_n from the distribution $\mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}\right)$,

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o} = \begin{pmatrix} \sigma_{w_1}^2 f_o(1) & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \sigma_{w_N}^2 f_o(N) \end{pmatrix}, \tag{2}$$

where the initial parameters are estimated from the demonstrations.

B. Policy Optimization using Relevance Functions

Once a number S of policy parameter vectors $\boldsymbol{w}$ have been sampled from $\mathcal{N}\left(\boldsymbol{\mu_w}, \boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}\right)$, RWR is used to optimize the mean $\boldsymbol{\mu_w}$ and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}$ to maximize the reward as

$$\{\boldsymbol{\mu}_{\boldsymbol{w}}^{k+1}, \boldsymbol{C}^{k+1}\} = \underset{\{\boldsymbol{\mu_w}, \boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}\}}{\arg\max} \sum_{i=1}^{S} R_{o,i} \mathcal{N}\left(\boldsymbol{w}_i; \boldsymbol{\mu_w}, \boldsymbol{\Sigma}_{\boldsymbol{w}}^{f_o}\right). \tag{3}$$

Its solution is given by

$$\boldsymbol{\mu}_{\boldsymbol{w}}^{k+1} = \frac{\sum_{i=1}^{S} R_{o,i} \boldsymbol{w_i}}{\sum_{i=1}^{S} R_{o,i}}, \tag{4}$$

$$\boldsymbol{C}^{k+1} = \frac{\sum_{i=1}^{S} R_{o,i} \left(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}}^k\right) \left(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}}^k\right)^\top}{\sum_{i=1}^{S} R_{o,i}}. \tag{5}$$

The variable k in the expressions above represents the iterations of the algorithm. The variable $R_{o,i}$ represents the reward with respect to objective o obtained by the sampled trajectory i. The reward is non-negative and usually has the form $R_{o,i} = \exp(-\beta o(i))$, where $o(i)$ is the value obtained by the sampled trajectory i for objective o and β is a hyperparameter chosen by the user.

Finally, the new covariance matrix $\boldsymbol{\Sigma_w}$ is determined. It is a diagonal matrix with the variances in the diagonal given by

$$\sigma_{w_n,k+1}^2 = (1 - f_o(n)) \sigma_{w_n,k}^2 + f_o(n) \boldsymbol{C}_{nn}^{k+1}, \tag{6}$$

where $\sigma_{w_n,k}^2$ is the variance of w_n in iteration k and $\boldsymbol{C}_{nn}^{k+1}$ is the element at row n and column n of the covariance matrix $\boldsymbol{C}^{k+1}$.

Equation (6) keeps the variance of irrelevant policy parameters unchanged and updates the variance of relevant policy parameters. If $f_o(n) = 0$, for example, $\sigma_{w_n,k+1}^2 = \sigma_{w_n,k}^2$, i.e. the variance of w_n at iteration $k+1$ is the same as at iteration k. On the other hand, if $f_o(n) = 1$, $\sigma_{w_n,k+1}^2 = \boldsymbol{C}_{nn}^{k+1}$, i.e. the variance of w_n at iteration $k+1$ is the result of the RWR optimization at iteration k, yielding $\boldsymbol{C}_{nn}^{k+1}$. For other relevance values, which must lie by definition between 0 and 1, the new variance is a weighted average of its previous value and the optimized one.

Note that differently from the algorithm presented in [1] where multiple iterations were required to estimate the relevance, here a one-shot approach for computing relevance functions is presented. Fig. 4 presents a comparison between these algorithms.

C. Online Adaptation with Gaussian Process

In this work we use parametrized trajectory distributions in the form of Probabilistic Movement Primitives (ProMPs) as the trajectory generation mechanism (see [10] for details). To adapt the distributions of trajectories online as the environment changes, the learning system must be able to compute solutions quickly. However, since PRO requires several iterations to find the optimal solution, it is not well suited for such problems. Thus, we propose Gaussian Process

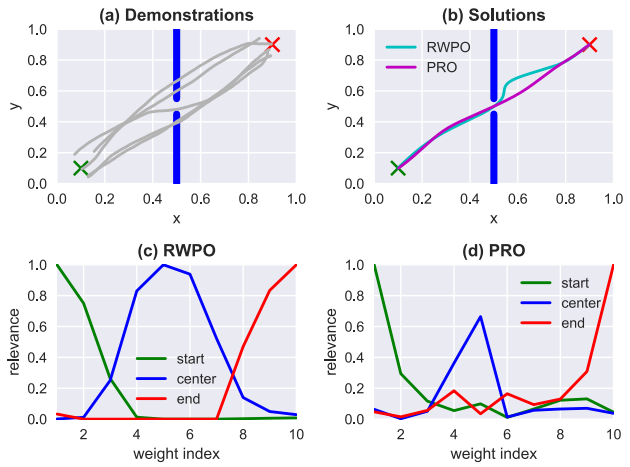


Fig. 4: Comparison between Relevance Weighted Policy Optimization (RWPO) [1] and PRO, which is proposed in this paper. In this problem, a trajectory must be found that starts at the green $\times$, goes through the window in the center and ends at the red $\times$. RWPO took 84.47s to learn the relevance functions while PRO took only 0.02s. Both algorithms were implemented in Python. The machine used for both computations was the same. The hyperparameters for RWPO are the same used in [1]. PRO used 200 trajectory samples to compute the relevance function. The vast difference in execution time is due to the iterative nature of the relevance computation in RWPO while PRO has a one-shot approach to determine the relevance. **(a)** 5 demonstrations provided by a user. **(b)** Both algorithms find solutions that satisfy the given criteria. The larger deviation from the direct path observed in the solution by RWPO does not represent an error because there is no cost for larger deviations from the direct path. The cost function used by both algorithms is the same and depends only on the distances to the start, window center and end. **(c, d)** Relevance for the 10 weights that parameterize the x trajectory computed by RWPO and by PRO, respectively. Differently from RWPO, PRO does not use basis functions for the relevance, which contributes to the more irregular shape of its relevance function.

(GP) regression to map variables describing the environment to mean vector $\boldsymbol{\mu_w}$ and covariance matrix $\boldsymbol{\Sigma_w}$ of a ProMP.

The entire training consists of the following steps: 1) Initialization via demonstrations 2) Given a random state of the environment, infer a ProMP using GP regression; 3) Use PRO to optimize upon the inferred ProMP; 4) Update dataset of environment states and corresponding ProMPs with the solution provided by PRO. Steps 2 to 4 are repeated several times until the learning system can solve a given task for a range of possible configurations of the environment. Essentially, this is a self-supervised learning process where trajectories sampled from the current GP are then optimized by PRO and fed back as training data.

The vector of variables describing the current state of the environment is denoted by $\boldsymbol{e}$. The elements of this vector can be for example obstacle positions, via points, target positions, etc.

The user is asked to initialize our learning system by providing multiple demonstrations for each environment configuration $\boldsymbol{e}_m$ in the set $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_M\}$ containing M different configurations. Based on these demonstrations, the variables $\mu_{w_{n,m}}$ and $\sigma^2_{w_{n,m}}$ are computed through Maximum Likelihood Estimation (MLE), where $n \in \mathbb{N}, m \in \mathbb{N}, 1 \leq n \leq N, 1 \leq m \leq M$. The variables $\mu_{w_{n,m}}$ and $\sigma^2_{w_{n,m}}$ are the mean and the variance, respectively, of weight w_n based on the demonstrations for environment configuration $\boldsymbol{e}_m$. The set of demonstrations can be augmented for additional environment configurations by trajectories based on prior knowledge, as previously mentioned. In this case, the trajectories based on prior knowledge are treated just as the demonstrations directly provided by the user.

Given $\mu_{w_{n,m}}$ and $\sigma^2_{w_{n,m}}, \forall m, 1 \leq m \leq M$, the variables μ_{w_n} and $\sigma^2_{w_n}$ are computed. These variables represent the average mean $\mu_{w_n} = \frac{1}{M}\sum_{m=1}^{M} \mu_{w_{n,m}}$ and the average variance $\sigma^2_{w_n} = \frac{1}{M}\sum_{m=1}^{M} \sigma^2_{w_{n,m}}$ for each weight w_n.

A new environment $\boldsymbol{e}_{\text{new}}$ is sampled at random from a set containing both the environments $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_M$ for which there were demonstrations as well as environments for which there were no demonstrations. Gaussian Process (GP) regression is used to infer the trajectory parameters $w_{n,\text{new}}$ for the new configuration $\boldsymbol{e}_{\text{new}}$, given the demonstrations. There is one Gaussian Process (GP) for each parameter w_n. The GPs use the squared exponential kernel $k(\boldsymbol{e}_i, \boldsymbol{e}_j) = \exp\left(-\alpha (\boldsymbol{e}_i - \boldsymbol{e}_j)^\top (\boldsymbol{e}_i - \boldsymbol{e}_j)\right), \alpha \in \mathbb{R}, \alpha > 0$. The variables $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ represent any two arbitrary environment configurations. The posterior is $p(w_{n,\text{new}}|w_{n,1:M}) = \mathcal{N}\left(\mu_{w_{n,\text{new}}}, \sigma^2_{w_{n,\text{new}}}\right)$, where

$$\mu_{w_{n,\text{new}}} = \mu_{w_n} + \boldsymbol{K}_{\text{new},1:M}\left(\boldsymbol{K}_{1:M,1:M} + \boldsymbol{\Sigma}_{w_n}\right)^{-1}\left(\boldsymbol{\mu}_{w_{n,1:M}} - \boldsymbol{\mu}_{w_n}\right), \quad (7)$$

$$\sigma^2_{w_{n,\text{new}}} = \boldsymbol{K}_{\text{new,new}} + \sigma^2_{w_n} - \boldsymbol{K}_{\text{new},1:M}\left(\boldsymbol{K}_{1:M,1:M} + \boldsymbol{\Sigma}_{w_n}\right)^{-1}\boldsymbol{K}_{1:M,\text{new}}, \quad (8)$$

$\boldsymbol{\mu}_{w_n} = \left[\mu_{w_n}, \cdots, \mu_{w_n}\right]^\top$ is a column vector with μ_{w_n} repeated M times, $\boldsymbol{\mu}_{w_{n,1:M}} = \left[\mu_{w_{n,1}}, \cdots, \mu_{w_{n,M}}\right]^\top$, the covariance matrix of each GP is

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_{\text{new,new}} & \boldsymbol{K}_{\text{new},1:M} \\ \boldsymbol{K}_{1:M,\text{new}} & \boldsymbol{K}_{1:M,1:M} \end{pmatrix} \quad (9)$$

and

$$\boldsymbol{\Sigma}_{w_n} = \begin{pmatrix} \sigma^2_{w_{n,1}} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \sigma^2_{w_{n,M}} \end{pmatrix}. \quad (10)$$

The covariance matrix $\boldsymbol{K}$ comprises four blocks. $\boldsymbol{K}_{\text{new,new}}$ is here just the scalar $k(\boldsymbol{e}_{\text{new}}, \boldsymbol{e}_{\text{new}})$. $\boldsymbol{K}_{\text{new},1:M}$ is a row vector with elements $k(\boldsymbol{e}_{\text{new}}, \boldsymbol{e}_j), j \in \mathbb{N}, 1 \leq j \leq M$. $\boldsymbol{K}_{1:M,\text{new}}$ is a column vector with elements $k(\boldsymbol{e}_i, \boldsymbol{e}_{\text{new}}), i \in \mathbb{N}, 1 \leq i \leq M$. Finally, $\boldsymbol{K}_{1:M,1:M}$ is a matrix with elements $k(i, j)$.

After computation of the posterior distribution given by $\mu_{w_{n,\text{new}}}$ (Eq. 7) and $\sigma^2_{w_{n,\text{new}}}$ (Eq. 8), PRO optimizes upon this distribution as described in Section III. The dataset of known environments and corresponding trajectory distributions is updated with $\boldsymbol{e}_{\text{new}}$, $\mu_{w_{n,\text{new}}}$ and $\sigma^2_{w_{n,\text{new}}}$, $\forall n, 1 \leq n \leq N$. Subsequently, this entire process is repeated for another $\boldsymbol{e}_{\text{new}}$. After several iterations, as it will be shown in the experimental section, the learning system is able to generate successful distributions of trajectories for a pre-defined range of environment configurations.

IV. Experiments

Three experiments demonstrate the efficacy of our proposed framework. The first experiment demonstrates that PRO, which determines relevance functions based on Pearson correlation, can be applied to optimize upon initial failed attempts to solve a teleoperation task. Differently from Relevance Weighted Policy Optimization (RWPO) [1], PRO does not need predefined task-specific basis functions to determine the relevance functions. The last two experiments demonstrate how PRO in combination with Gaussian Process (GP) regression can tackle online motion planning problems. Please see the accompanying video.

A. Assisting Humans in a Teleoperation Task

In this experiment, the user manipulates the Haption Virtuose 6D to move a beam in a virtual environment (See Fig. 2). This experiment can be seen as a teleoperation task, where the haptic device is the master and the beam is the slave. The goal of the user is to move the beam from a start position and orientation to an end position and orientation through the window without hitting the wall. This task is hard for humans in part due to the difficulty in visually estimating the 3D position and the orientation of the beam. Already performing a similar task with a small cube without the need to control orientation has been difficult for most users as demonstrated by user studies in [1].

First, the user tries 10 times to perform the task without force feedback. A distribution of trajectories based on the trials of the user is created using a ProMP. Subsequently, PRO is used to optimize this ProMP such that sample trajectories from the optimized ProMP pass through four via points with the right beam orientations to avoid collisions with the wall.

In this experiment, the original optimization problem has been separated into two optimization problems: one taking into consideration only the Cartesian coordinates of the via points and another taking into consideration only the orientation of the beam at each via point. This separation helped PRO to find successful trajectories in this problem. The reward function for both problems is $R_{\text{obj}} = \exp\left(-\beta\left(\text{obj} + \beta_{\text{length}}\text{len} + \beta_{\text{jerk}}\text{jerk}\right)\right)$, with $\beta = 200$, $\beta_{\text{length}} = 0.1$ and $\beta_{\text{jerk}} = 10^5$. The value for β was empirically determined by trying a few values between 1 and 300. The values for β_{length} and β_{jerk} were determined by trying different powers of 10. The variable obj represents the distances to each of the four via points. The two via points closest to the window are computed given the current position of the window. The variables len and jerk represent the length and the average jerk of the trajectory, respectively, and are computed by using finite differences. The terms β_{length} and β_{jerk} are used to regulate the importance of the length and average jerk to the reward. PRO optimized the ProMP in 150 iterations with 200 trajectory samples per iteration.

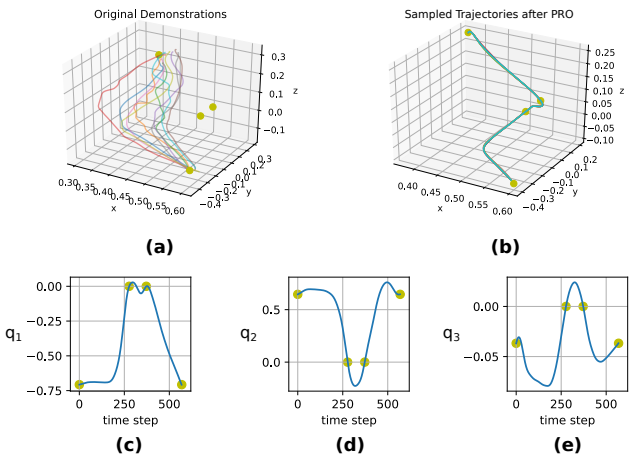


Fig. 5: **(a)** Cartesian coordinates of demonstrated trajectories. **(b)** Cartesian coordinates of trajectories sampled from the ProMP optimized with PRO. **(c - d)** Orientations of the mean trajectory of the ProMP optimized with PRO. The orientations are here represented by the three first variables (q_1, q_2, q_3) of a unit quaternion. The yellow dots represent via points.

Fig. 5(a) shows the initial trials of a user to solve the task for a given scenario without the assistance of the haptic device. Fig. 5(b-e) represents the optimized ProMP, which is used by the haptic device to guide the user with force feedback inverse proportional to the standard deviation.

B. Adaptation in Dynamic Environments — Point Particle

The problem addressed in this section is depicted in Fig. 6. First, a human was presented with 30 random environments. The environments differed in $\boldsymbol{c} = (x_c, y_c)$, the position of the center of the hole in the wall, and in $\boldsymbol{g} = (x_g, y_g)$, the position of the end goal (red $\times$ in Fig. 6). By using a computer mouse, the human provided three demonstrations for each environment.

The random environments $\boldsymbol{e} = [x_c, y_c, x_g, y_g]^\top$ were uniformly distributed in the range $2 \leq x_c \leq 8$, $1 \leq y_c \leq 9$, $x_c + 1.5 \leq x_g \leq 10$, $0 \leq y_g \leq 10$. The start position $\boldsymbol{s} = (x_s, y_s)$ was always the same.

PRO used reward functions of the form $R_{\text{obj}} = \exp\left(-\beta\text{obj}\right)$, where $\beta = 20$ was empirically determined by observing the policies learned by PRO for a few values of β: 1, 10, 20, 30, 40 and 50. In this problem, three objectives need to be minimized: the distance to the start position $\text{obj}_1 = ||\boldsymbol{\tau}(0) - \boldsymbol{s}||$, the minimum distance to the center

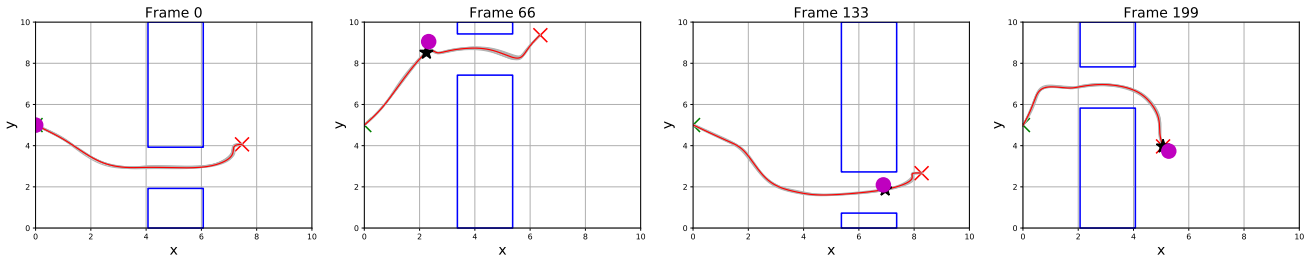


Fig. 6: The magenta point particle has to move from the start position (green $\times$) to the target position (red $\times$) without hitting the walls (blue rectangles). The position of the hole in the wall and the target position change with time. As these positions change, our learning system computes the corresponding trajectory distributions on the fly to solve this task. The red line corresponds to the mean of the computed trajectory distribution. The light gray trajectories are samples from the computed distribution. The black star-shaped marker moves forward along the mean of the current distribution. The magenta point particle tracks the black star-shaped marker with a PD controller. This figure depicts four frames of a test case.

of the hole in the wall $\text{obj}_2 = \min_t ||\boldsymbol{\tau}(t) - \boldsymbol{c}||$ and the distance to the end goal $\text{obj}_3 = ||\boldsymbol{\tau}(T) - \boldsymbol{g}||$. The term $\boldsymbol{\tau}(t)$ represents the position along trajectory $\boldsymbol{\tau}$ at time step t, $\boldsymbol{\tau}(0)$ is the first position and $\boldsymbol{\tau}(T)$ is the last position.

After the initialization with the demonstrations, the self-improvement loop was repeated 1000 times, each time with a new random environment. Each PRO optimization took at most 200 iterations (less if convergence was achieved sooner) and used 100 trajectory samples per iteration. The kernel of the GPs used in this problem had parameter $\alpha = 10^{-3}$. This value was empirically determined by trying a few different powers of 10 and observing the GP inferences. During test, our learning system can compute ProMPs on the fly, solving the task in a dynamic environment (See Fig. 6).

C. Adaptation in Dynamic Environments — Robot Arm

The problem addressed in this section is depicted in Figs. 1 and 7. The obstacle (Pringles can) and the target (pink object) are tracked by using a motion capture system (OptiTrack). First, a human provides demonstrations by moving the 7-DoF robot arm in gravity compensation mode. Demonstrations were provided for 20 different environment configurations. There were three demonstrations for each environment. The configurations were determined by arbitrarily choosing positions on the table for the obstacle and the target. The start position for the robot arm was always the same.

Each environment in this problem was represented by the vector $\boldsymbol{e} = [x_p, y_p, x_g, y_g]^\top$, where (x_p, y_p) was the position of the Pringles can and (x_g, y_g) was the end goal position.

We have noticed that initializing our learning system only with the human demonstrations for 20 different situations was not enough to learn a mapping capable of dealing with any obstacle and target positions on the table. For this reason, we have decided to extend the set of demonstrations with ProMPs based on prior knowledge. These ProMPs had mean trajectory going directly from the start position to the target position irrespective of the obstacle position and the variance of each ProMP weight was the average variance for that weight based on the demonstrations. The GPs were thus initialized with ProMPs for 2024 different environments (including environments for which human demonstrations were given). The additional 2004 environments were generated by taking obstacle and target positions of a grid inside a range of possible positions delimited by the corners of the table.

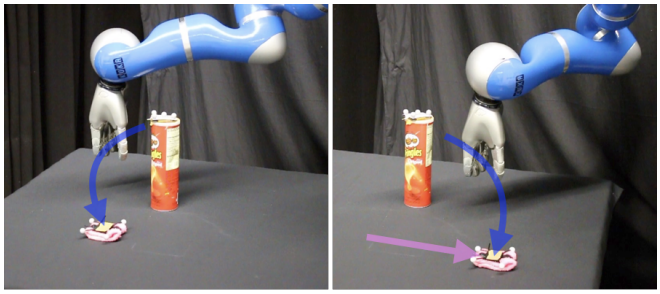
PRO used reward functions of the form $R_{\text{obj}} = \exp(-\beta \text{obj})$, where $\beta = 200$ was empirically determined by observing the policies learned by PRO for a few values of β: 1, 10, 20, 30, 40, 50, 100, 200 and 300. In this problem, three objectives need to be minimized: $\text{obj}_1 = ||\boldsymbol{\tau}(0) - \boldsymbol{s}|| + d$, $\text{obj}_2 = \max(-\min_t ||\boldsymbol{\tau}(t) - \boldsymbol{p}||, -0.2) + d$ and $\text{obj}_3 = ||\boldsymbol{\tau}(T) - \boldsymbol{g}|| + d$.

The term $d = \frac{1}{T}\sum_{t=0}^{T} ||\boldsymbol{\tau}(t) - \boldsymbol{\tau}_{\text{direct}}(t)||$ is the average distance to the direct path $\boldsymbol{\tau}_{\text{direct}}$ from the start to the end goal. This term was added to each of the objectives to avoid large deviations from the direct path to the end goal. Apart of this term, obj_1 and obj_3 are very similar to objectives described in Section IV-B. The variable $\boldsymbol{p} = (x_p, y_p)$ in obj_2 is the position of the Pringles can. Minimizing obj_2 has the effect of avoiding the Pringles can without going too far away from it because distances to the Pringles can larger than 20 cm do not result in additional reward.

The self-improvement loop had 2024 iterations (one for each environment in the initialization data set). Each PRO optimization took at most 50 iterations (less if convergence was achieved sooner) and used 100 trajectory samples per iteration. The kernel of the GPs used in this problem had parameter $\alpha = 1$. This value was empirically determined by trying a few different powers of 10 and observing the GP inferences. During test, the robot can successfully execute the reaching task even when the human moves the obstacle or the target while the robot is moving (See Figs. 1 and 7).

V. Conclusion and Future Work

This paper presented the **P**earson-Correlation-Based **R**elevance Weighted Policy **O**ptimization (PRO) algorithm. In this algorithm, the concept of Pearson correlation is used to determine the relevance of each policy parameter to each optimization objective. The proposed algorithm is computationally much more efficient than the one presented



(a) Around the obstacle from the right (b) Around the obstacle from the left

Fig. 7: Our learning system infers a distribution of trajectories (ProMP) given the current state of the environment. An inverse-dynamics-based feedback controller tracks the mean of the inferred distribution. **(a)** The robot goes around the obstacle (a Pringles can) from the right side to reach the target (a pink object). **(b)** Given that the user changes the position of the target while the robot is moving, the robot switches on the fly to another ProMP, going around the obstacle from the left side.

in [1]. Moreover, a framework which uses PRO and GP regression has been presented, which can compute trajectory distributions on the fly to solve tasks in dynamic environments. PRO can be used to optimize upon suboptimal demonstrated trajectories. An application of PRO to assisted teleoperation has been demonstrated. Our full framework can solve online planning problems in an experiment involving a point particle and in a real robot experiment with a 7-DoF robot arm.

The next step in this research is to apply our framework to assisted teleoperation in dynamic environments. We will also investigate other reward functions for the assisted teleoperation task to avoid the necessity of intermediate via points and the separation in two optimization problems.

VI. Acknowledgments

The research leading to these results has received funding from the German Federal Ministry of Education and Research (BMBF) in the project 16SV7984 (KoBo34), from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 640554 (SKILLS4ROBOTS), from a project commissioned by the Japanese New Energy and Industrial Technology Development Organization (NEDO) and from the Swiss National Science Foundation through the HEAP project (Human-Guided Learning and Benchmarking of Robotic Heap Sorting, ERA-net CHIST-ERA).

References

[1] M. Ewerton, D. Rother, J. Weimar, G. Kollegger, J. Wiemeyer, J. Peters, and G. Maeda, "Assisting movement training and execution with visual and haptic feedback," *Frontiers in neurorobotics*, vol. 12, p. 24, 2018.

[2] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[3] G. Raiola, X. Lamy, and F. Stulp, "Co-manipulation with multiple probabilistic virtual guides," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 7–13.

[4] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 2, pp. 286–298, 2007.

[5] I. Havoutis and S. Calinon, "Learning assistive teleoperation behaviors from demonstration," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 258–263.

[6] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[7] F. Abi-Farraj, T. Osa, N. P. J. Peters, G. Neumann, and P. R. Giordano, "A learning-based shared control architecture for interactive task execution," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 329–335.

[8] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4569–4574.

[9] I. Havoutis and S. Calinon, "Supervisory teleoperation with online learning and optimal control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. IEEE, 2017.

[10] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[12] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 745–750.