Phase Portraits as Movement Primitives for Fast Humanoid Robot Control

Guilherme Maeda[1], Okan Koc[2], Jun Morimoto[1*]

g.maeda@atr.jp, okan.koc@tuebingen.mpg.de, xmorimo@atrj.jp

[1] *ATR Computational Neuroscience Laboratories. Kyoto, Japan*

[2] *Max Planck Institute. Tübingen, Germany*

Abstract

Currently, usual approaches for fast robot control are largely reliant on solving online optimal control problems. Such methods are known to be computationally intensive and sensitive to model accuracy. On the other hand, animals plan complex motor actions not only fast but seemingly with little effort even on unseen tasks. This natural sense of time and coordination motivates us to approach robot control from a motor skill learning perspective to design fast and computationally light controllers that can be learned autonomously by the robot under mild modeling assumptions. This article introduces Phase Portrait Movement Primitives (PPMP), a primitive that predicts dynamics on a low dimensional phase space which in turn is used to govern the high dimensional kinematics of the task. The stark difference with other primitive formulations is a built-in mechanism for phase prediction in the form of coupled oscillators that replaces model-based state estimators such as Kalman filters. The policy is trained by optimizing the parameters of the oscillators whose output is connected to a kinematic distribution in the form of a phase portrait. The drastic reduction in dimensionality allows us to efficiently train and execute PPMPs on a real human-sized, dual-arm humanoid upper body on a task involving 20 degrees-of-freedom. We demonstrate PPMPs in interactions requiring fast reactions times while generating anticipative pose adaptation in both discrete and cyclic tasks.

Keywords: Imitation learning, reinforcement learning, movement primitives, phase estimation, coupled oscillators.

1. Introduction

Humans react to changes in the environment by adapting and coordinating complex motor actions gracefully and skillfully. Consider motor skills that basketball players exhibit when passing and receiving a ball as shown in Figure 1. They naturally coordinate the position of the hands according to the progress of the ball trajectory, quickly adapting both in time and space. Yet, in contrast to the heavy online optimization typically found in robot control (e.g. [1, 2, 3, 4]), no conscious effort seems to be spent on either the ball prediction or the trajectory planning of the hands. This natural agility could be in part

*Corresponding author: xmorimo@atr.jp. ATR. Department of Brain Robot Interface. 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

Figure 1: **Playing with a ball.** Humans are naturally endowed with the ability to estimate the time-to-contact while coordinating multiple degrees of freedom even without knowing the exact dynamics of the flight and contact forces. To endow robots with such capabilities, this article introduces Phase Portrait Movement Primitives (PPMPs). A PPMP is capable of estimating the future temporal states of an interaction by using coupled oscillators as a dynamical model in phase space. By using policy search, the robot learns to combine the estimated phase with the high-dimensional kinematics of the task represented as a phase portrait.

because, from an early age, humans seem to develop a sense of timing[5] that proves essential in fast motor actions. Moreover, the fact that we continuously anticipate the interaction by adapting our pose—e.g. when someone tricks us by rocking a ball towards us, pretending a pass but not doing it—indicates that this sense of timing is given by some predictive mechanism that makes us anticipate poses according to the expected progress of the interaction.

Animal agility also seems to be related to the existence of motor primitives as internal models learned and improved from experience. These primitives provide a mapping from sensory signals to motor commands [6, 7, 8] coordinating the relevant degrees-of-freedom in the system and can be quickly queried to achieve fast adaptation. In robotics, researchers have for many years tried to create the corresponding counterparts as movement primitives acquired via imitation[9]. While the search space decreases considerably by using the templates provided by primitives, its reliance on future state prediction is still a challenge for fast adaptation. For example, Kober et al. [10] designed a specialized Kalman filter to predict the future states of an incoming ball in robotic table tennis. That work used Dynamical Movement Primitives (DMPs) [11], a representation widely used within the robot learning community. A goal of our method is to incorporate a predictive sense of timing (such as the time-of-contact of a flying object) as an intrinsic part of the primitive without relying on the detailed physics models required by traditional state estimators.

A fundamental requirement in tasks involving fast dynamics is that robot control commands must be computed extremely fast, often at the order of tens of Hertz, while accounting for the future states of the interaction. In Figure 2, this difficulty is cartooned by a robot that when observing the flight of a ball at time t_0, must advance its progress as shown at time t_1 such that it can compensate for the many sources of delays (inertia, torque limits, friction, backlash, etc.) when preparing for the landing of the ball at t_2. Here, we refer to the robot's and ball's progresses by their phases, ϕ^{robot} and ϕ^{target}, respectively. Thus, a positive phase shift $\phi^{\text{robot}} > \phi^{\text{target}}$ indicates that the robot is acting in an anticipative manner.

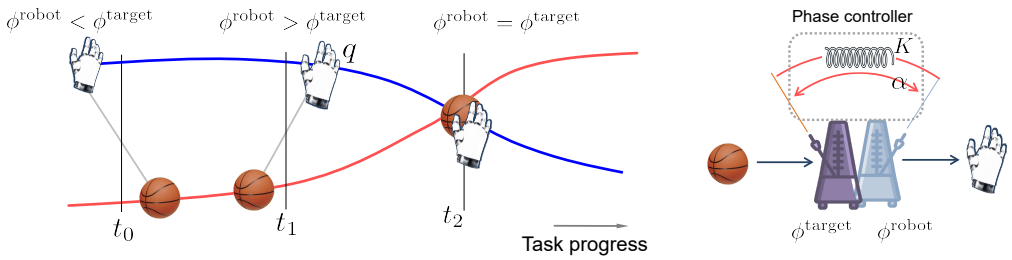


Figure 2: **Temporal prediction and adaptation for ball catching.** An illustration of a motivating problem where a robot must continuously observe the flight of a ball and adapt its phase ϕ^{robot} to the ball phase ϕ^{target} such that a successful catching is possible. Pure reactive feedback tracking on the ball position can not account for the fact that the robot must advance its phase, as shown at time t_1, to preemptively prepare for the ball landing at time t_2. We use the concept of coupled oscillators and let the robot learn a policy that modulates the (nonlinear) impedance between these two oscillators. At runtime, the phase oscillator can assign the robot to future phase states by using a positive phase shift ($\alpha = \phi^{\text{robot}} - \phi^{\text{target}} > 0$).

Using this illustrated case as motivation, in the broadest possible terms, two principal directions to solve this problem are to use models that can be exploited by online optimization or to rely on trial-and-error under the framework of model-free reinforcement learning.

Within the class of methods based on models and online optimization, for many years, Model Predictive Control (MPC) has been considered the gold standard in fields such as chemical processes and more recently in robotics [12, 2, 4]. MPC is characterized by the computation of a look-ahead trajectory (whose length is defined by the horizon) which starts at the current state of the task. This trajectory is executed partially until a new look-ahead trajectory is ready to replace the previous one. By constantly refreshing the reference trajectory, the effects of disturbances and modeling errors are greatly reduced as they are continuously absorbed within the states used to generate the subsequent trajectory. As a consequence, MPC—and online optimal control, in general—not only relies upon the designer's domain knowledge for modeling but also demands fast optimization routines and powerful computing (e.g. [1, 4, 13]). This reliance on models and computational requirements makes such approaches less suited to fulfill the vision of intelligent robots capable of learning tasks autonomously.

On the other spectrum, reinforcement learning is an extremely general approach for robotic autonomous learning[14]. However, the lack of model information leads to an unrealistic amount of experience to be acquired via trial-and-error; reducing its practicality to low dimensional problems. Here, we propose a middle-ground solution in terms of model requirements which we will loosely refer to as *semi-model-free*. We implement and investigate whether the *temporal dimension* of the task can be sufficiently predicted by a general dynamical model borrowed from the locomotion literature [15, 16]. Under the manipulation interpretation, the assumption is that the timing of the manipulation obeys the dynamics of two oscillators—one for the robot, another for the environment—coupled by an impedance feedback law (details in the next section). The *spatial dimension* is represented by a kinematic model valid around the vicinity of demonstrations. These temporal and spatial representations do not rely on domain knowledge or parameter system identification required by model-based methods. Moreover, the dimensionality decreases to a degree that is amenable to the use of policy search reinforcement learning methods.

The main *contribution* of this article is to propose a control method where a mechanism for fast

phase estimation is an intrinsic part of the movement primitive. As the primitive is represented as a phase portrait we name the method Phase Portrait Movement Primitives (PPMP). This close connection between phase estimation and primitives allows for first solving the *timing* of the interaction in the phase space, subsequently reducing *spatial coordination* to a kinematic problem. The benefit is that at runtime, PPMPs have negligible computation load while allowing for fast generation of predictive actions. PPMP is a general method devised to support the vision of future robots capable of learning tasks completely autonomously. This article empirically evaluates the sufficiency of this representation and its efficiency for reinforcement learning by applying it on a variety of tasks; from dynamic manipulation of a flying ball, to handovers with a human, to footstep placing for walking. Our preliminary investigations on the phase oscillator dynamics which led to the PPMP proposed in this article first appeared as a short conference paper [17]. This article has a mature view of the method, in which we more deeply analyze its fundamental components while guiding the reader through the synthesis of PPMPs. In addition to formalizing the concept of phase portraits for the first time, this article also extends experimental cases significantly, comparing and discussing the parallels with other primitive formulations.

2. Coupled Oscillators and Phase Portraits

This section introduces the main components of PPMPs. A phase estimator in the form of coupled phase oscillators is used for estimating the timing of the interaction while a probabilistic phase portrait as a distribution of demonstrated trajectories is used for kinematic coordination. Because the dynamic link between phase oscillators and phase portraits are not observable during humans demonstrations, the last step of PPMP consists of merging the oscillators and the phase portrait as a single policy via reinforcement learning (RL).

2.1. Timing the Dynamics of Interactions via Coupled Oscillator

We revisit a locomotion model based on coupled oscillators [15, 16] and reframe it as a predictive model to estimate timing in manipulation tasks. If properly tuned, the oscillators allow the robot to anticipate actions as a function of the evolution of another agent. The illustration at the right in Figure 2 shows the concept of the phase mechanism adopted in this work as two metronomes coupled via a stiffness K and a phase difference α. In feedback terms, the stiffness is a proportional controller whose goal is to decrease the phase tracking error while the phase shift acts as a set point offset. The target oscillator's state ϕ^{target} is a measure of the progress of the interaction with the environment. The second oscillator provides the phase state of the robot ϕ^{robot}, whose value is computed by integrating the dynamics of the coupling

$$\dot{\phi}^{\text{robot}} = \omega + K \sin(\phi^{\text{target}} - \phi^{\text{robot}} + \alpha), \tag{1}$$

where ω is the natural frequency of the system, that is, the velocity at which the robot would move if no correction was applied. An important insight is that while the reactive nature of the coupled oscillators is the essence of the simplicity and fast estimations, the effect of its output in the control task is, in fact, *predictive*. Note from (1) that the oscillators can assign the robot to future phase states in relation to the target state (time t_1 in Figure 2) by using a positive phase shift $\alpha = \phi^{\text{robot}} - \phi^{\text{target}} > 0$. The problem is then to learn what is the optimal value of this phase shift and also the stiffness in which this shift should be tracked according to the input ϕ^{target}.

2.2. Kinematic Coordination via Probabilistic Phase Portrait

The phase is related to the temporal dimension of the task—represented by the horizontal axis in Figure 2—and does not contain any spatial or kinematic information such as the joint configuration of the robot. Here, we describe how the joint positions of the agent can be learned and computed as a function of the target's position via inference. A probabilistic approach for fast robot pose adaptation has two main benefits. First, by assuming a normal distribution, a closed-form solution can be used to instantaneously infer robot poses given observed context positions. Second, the phase portrait can be designed from demonstrations, rendering a methodology that requires little domain knowledge. As an imitation learning approach, the idea is to obtain a few variations of trajectory pairs of duration T of robot joint angles $\boldsymbol{q}_{1:T}^{\text{demo}}$ and the target movements usually observed in Cartesian space (e.g. via motion capture) $\boldsymbol{x}_{1:T}^{\text{demo}}$ from multiple demonstrations. Each paired trajectory is assumed to be a sample from a joint distribution $(\boldsymbol{q}, \boldsymbol{x})_{1:T} \sim P(\boldsymbol{q}, \boldsymbol{x})_{1:T}$. Assuming normal distributions, at each time step $t \in \{1, .., T\}$ we can write

$$P(\boldsymbol{q}, \boldsymbol{x})_t = \mathcal{N}(\{\boldsymbol{\mu}_{q,x}\}_t, \{\boldsymbol{\Sigma}_{q,x}\}_t), \tag{2}$$

where

$$\{\boldsymbol{\mu}_{q,x}\}_t = \Big[\{\boldsymbol{\mu}_q\}_t, \quad \{\boldsymbol{\mu}_x\}_t\Big]^\top = \text{mean}(\{(\boldsymbol{q}', \boldsymbol{x}')_{1:N}\}_t) \quad \text{and} \tag{3}$$

$$\{\boldsymbol{\Sigma}_{q,x}\}_t = \begin{bmatrix} \{\boldsymbol{\Sigma}_{qq}\}_t & \{\boldsymbol{\Sigma}_{qx}\}_t \\ \{\boldsymbol{\Sigma}_{xq}\}_t & \{\boldsymbol{\Sigma}_{xx}\}_t \end{bmatrix} = \text{cov}(\{(\boldsymbol{q}', \boldsymbol{x}')_{1:N}\}_t). \tag{4}$$

At each instant t, a target observation $\boldsymbol{x}_t$ is made and the corresponding joint configuration of the robot is computed as a conditional distribution

$$P(\boldsymbol{q}|\boldsymbol{x})_t = \mathcal{N}(\{\boldsymbol{\mu}_{q|x}\}_t, \{\boldsymbol{\Sigma}_{q|x}\}_t), \tag{5}$$

with

$$\begin{aligned} \boldsymbol{\mu}_{q|x} &= \boldsymbol{\mu}_q + \boldsymbol{\Sigma}_{qx}\boldsymbol{\Sigma}_{xx}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x) \\ \boldsymbol{\Sigma}_{q|x} &= \boldsymbol{\Sigma}_{qq} - \boldsymbol{\Sigma}_{qx}\boldsymbol{\Sigma}_{xx}^{-1}\boldsymbol{\Sigma}_{xq}, \end{aligned} \tag{6}$$

where the subscript t was dropped.

In practice, many tasks cannot be demonstrated directly using a robot either because most robots are not backdriveable (no kinesthetic teaching capability) or because the task is too fast to be executed while moving a robotic arm (like catching a flying ball). As shown in Figure 3(a), a more natural approach is to observe a human executing the task while interacting with a target where paired end-effector and target trajectories are recorded. Later, the robot trajectory is computed via inverse kinematics $\boldsymbol{q}_{1:T}^{\text{demo}} \leftarrow \boldsymbol{x}_{1:T}^{\text{end-eff.}}$ as shown in Figure 3(b).

Figure 3(c) shows the sequence of joint distributions (2) in time-domain. The plane cutting the trajectories in half is used to indicate the difference between cyclic and single-stroke tasks. In the former, the distributions return close to the initial state, and in the latter, only the first half of the distributions exist. PPMPs apply seamlessly in both cases.

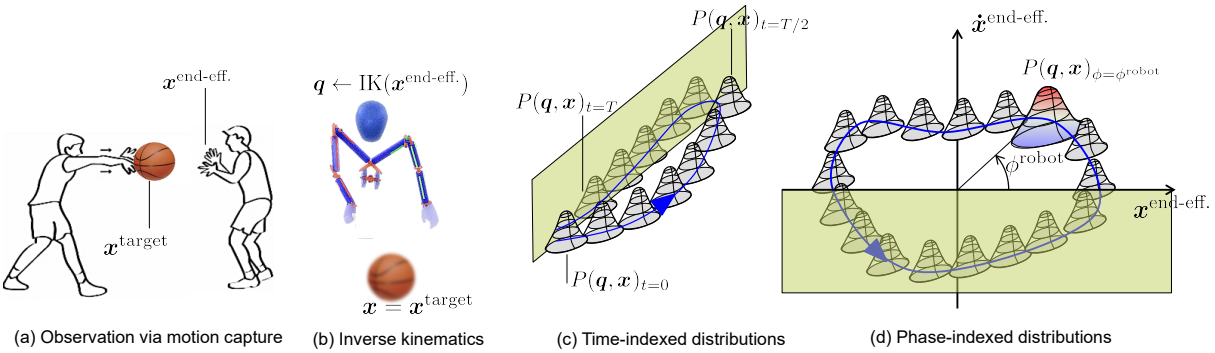


Figure 3: **Representing Kinematics as a Probabilistic Phase Portraits.** The Phase Portrait Movement Primitives (PPMP) consists of a sequence of joint distributions that correlates demonstrator and target positions. These joint distributions are first captured via demonstration (a) and transformed into joint angles via inverse kinematics (b). The trajectory of distributions (c) are then laid on the phase-plane of the end-effector ($\boldsymbol{x}^{\text{end-eff.}}, \dot{\boldsymbol{x}}^{\text{end-eff.}}$) which can then be queried by a phase angle ϕ^{robot}. The semi-transparent plane in (c-d) shows the only distinction between cyclic and single-stroke tasks, where in the latter case only half of the distributions are present as no path of return exists.

To connect oscillators and primitives consistently, we map the sequence of kinematic distributions onto the phase plane of the end-effector motion such that the angle of the phase plane becomes the input from the oscillator. This mapping is illustrated in 3(d) by using the mean value of the end-effector trajectory to compute the phase of the robot as the angle

$$\phi^{\text{robot}} = \phi_t = -\arctan(\dot{\boldsymbol{x}}_t^{\text{end-eff.}}, \boldsymbol{x}_t^{\text{end-eff.}}),\ t \in [1,\ T]. \tag{7}$$

This primitive is a sequence of distributions on the phase plane and can be interpreted as a probabilistic phase portrait. At run-time, the time-based phase ϕ_t is replaced by the phase computed by the oscillator ϕ^{robot}. The idea is to use the angle ϕ^{robot} output by the oscillator to retrieve $P(\boldsymbol{q}, \boldsymbol{x})_\phi$ from the phase portrait. This distribution is then conditioned on the observed target to produce a set of joint angles[1] $\boldsymbol{q} \sim P(\boldsymbol{q}|\boldsymbol{x})_\phi$. In the single stroke case, the phase portrait only covers the first two quadrants of the phase plane while in cyclic tasks all quadrants are covered.

2.3. Consolidating Oscillators and Phase Portraits as a Single Policy

So far, the phase estimation resulting from the dynamics of the coupled oscillators is unrelated to the kinematics of the phase portrait. Not only the oscillators and the portraits represent different policies (temporal and spatial) but they have been designed independently in the previous sections. One issue to connect them is that the relation between oscillators and phase portraits cannot be observed during the demonstration. Another issue is that we want the robot to adapt to a variety of situations for which demonstrations are not available. Thus, it is necessary to consolidate these two components as a single robust dynamical policy. To this end we propose the scheme in Figure 4 where the dynamics between

[1] Alternatively, one could retrieve $\boldsymbol{x}_\phi^{\text{end-eff.}}$ from the phase portrait and use inverse kinematics to compute a solution for the inferred end-effector position online. In principle, the only compelling reason for doing so is when the true distribution cannot be capture as $\mathcal{N}(\{\boldsymbol{\mu}_{q,x}\}_t, \{\boldsymbol{\Sigma}_{q,x}\}_t)$.

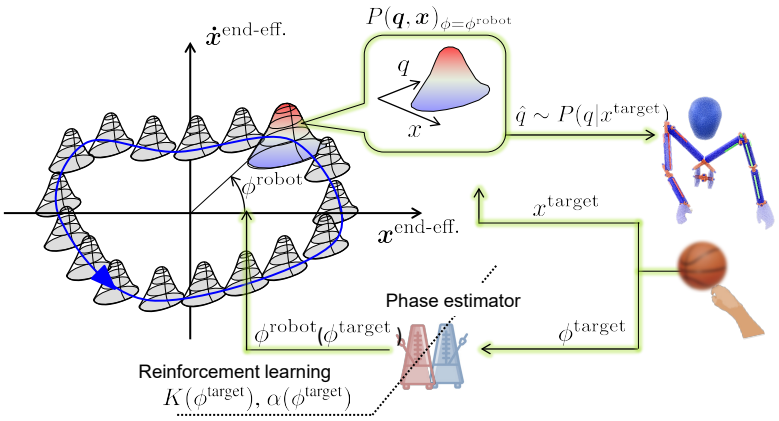


Figure 4: **Learning to combine coupled oscillators with the probabilistic phase portrait as a control policy.** The full method where the oscillator output selects a joint distribution on the phase-portrait. This distribution is then conditioned on the target position to retrieve robot joint angles. The use of policy search RL on the parameters coupling the oscillators allows the robot to consolidate the nonlinear synchronization between the oscillator dynamics and robot poses as a single and consistent policy.

the coupled oscillators affect how the distributions on the phase portraits are queried as a function of the phase of the target ϕ^{target}.

Referring to (1), the only open parameters that allow this consolidation are the coupling elements, stiffness K and phase shift α, between oscillators. The natural frequency ω governs the velocity of the cycle when the robot is free from external inputs. For manipulation tasks, it is natural to assume that the robot should not move when the target does not move, thus we use $\omega = 0$ as the default value for all the experiments in this article. Fixed values of K and α generate a linear control law, which, except for simple single stroke tasks, show insufficient to represent different task regimes. Thus, to allow for nonlinear oscillator dynamics we define the coupling parameters as two functions $K = K(\phi^{\text{target}})$ and $\alpha = \alpha(\phi^{\text{target}})$. Finding the optimal PPMP policy means finding the shapes of $K(\phi^{\text{target}})$, $\alpha(\phi^{\text{target}})$ such that a cost is minimal when the task is finished successfully.

With the exception of simple systems, this optimization is usually not suited for model-based methods as the interaction dynamics are the result of many unknown lumped effects such as the delays in the robot actuation, mechanism nonlinearities (e.g. friction and backlash), bi-manual asymmetric exchange of contact forces between the hands and the manipulated object, inaccuracies of the perception system and its delays, etc. Thus, for the general case, a gradient-free direct policy search approach [18] is better suited. The parameters that govern the shape of the two functions $K(\phi^{\text{target}})$, $\alpha(\phi^{\text{target}})$ are optimized directly based on rewards. As usual in policy search, the optimization is done on weights as parameters that control the amplitude of N radial-basis functions (RBFs), such that

$$[K(\phi^{\text{target}}), \alpha(\phi^{\text{target}})] = [\boldsymbol{\Phi}_{i,1} \ \ldots \ \boldsymbol{\Phi}_{i,N}][\boldsymbol{w}_K^\top, \boldsymbol{w}_\alpha^\top], \tag{8}$$

where $\boldsymbol{w}_K \in \mathbb{R}^N$ and $\boldsymbol{w}_\alpha \in \mathbb{R}^N$ are weight vectors. The matrix of basis functions $\boldsymbol{\Phi}$ has dimensions

$T \times N$ where T is the number of steps on the demonstration and N is the number of bases[2]. As such, the n-th column of $\boldsymbol{\Phi}$ is a RBF centered at $n = (T/N)$ and defined over the interval $[-\pi,\ \pi]$ for cyclic tasks or $[0,\ \pi]$ for single-stroke tasks.

The definition of the number of basis functions is usually an empirical process. Lower N values impose smoother functions while larger values allow for functions requiring rapid transitions. In our experiments, a typical value was $N = 10$. The weights can be computed by solving the normal equations (refer to [19] chap. 3 for detailed procedure)

$$\begin{aligned} \boldsymbol{w}_K &= \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top K(\phi^{\text{target}}_{-\pi:\pi}) \\ \boldsymbol{w}_\alpha &= \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \alpha(\phi^{\text{target}}_{-\pi:\pi}). \end{aligned} \tag{9}$$

As the policy search algorithm we adopted Pi^{BB} [20] which is a black-box optimization[3] algorithm based on Pi^2 [21] to search for the optimal vector $\boldsymbol{w} = [\boldsymbol{w}_K^\top, \boldsymbol{w}_\alpha^\top]$. The updates are of the form

$$\boldsymbol{w}^{\text{new}} \leftarrow \boldsymbol{w}^{\text{old}} + \sum_{r=1}^{R} [P(\boldsymbol{\tau}_r)\boldsymbol{\epsilon}_r], \tag{10}$$

where a parameter update is done every R batches of roll-outs. The old parameters $\boldsymbol{w}^{\text{old}}$ are updated by using the weighted average of the exploration noise $\boldsymbol{\epsilon}_r \in \mathbb{R}^N$, which are sampled from a zero-mean Gaussian distribution. The weight of each roll-out is computed with

$$P(\boldsymbol{\tau}_r) = \frac{\exp[-\lambda C(\boldsymbol{y}_r)]}{\sum_{r'=1}^{R} \exp[-\lambda C(\boldsymbol{y}_{r'})]}, \tag{11}$$

where $C(\cdot)$ is the cost of the roll-out and must be designed for the task at hand. While we motivate the policy search as a way to find an optimal policy, an alternative interpretation is to view the optimization as a parameter estimation problem via RL [22] since the coupling between oscillators is a parameterized control law.

Referring to Figure 4, the execution of a roll-out consists in using the exploratory weights to retrieve corresponding functions $K(\phi^{\text{target}})$ and $\alpha(\phi^{\text{target}})$ with (8). Under continuous target observations $\boldsymbol{x}^{\text{target}}$ the target phase ϕ^{target} is computed and used as the input for the coupled oscillators, from which the robot phase is found as

$$\dot{\phi}^{\text{robot}} = \omega + K(\phi^{\text{target}}) \sin(\phi^{\text{target}} - \phi^{\text{robot}} + \alpha(\phi^{\text{target}})). \tag{12}$$

The phase of the robot indicates which joint distribution from the phase portrait to be used. This distribution is conditioned on $\boldsymbol{x}^{\text{target}}$ to obtain the joint configurations of the robot. This process is repeated online during a roll-out and only requires the computation of the integral (12) and the inference

[2]An alternative to RBFs is to use circular features such as von Mises bases to ensure the continuity of $K(\phi^{\text{target}})$, $\alpha(\phi^{\text{target}})$ when ϕ^{target} wraps around a full cycle. We did not notice the continuity to be important as ϕ^{robot} results from the integration of a differential equation. Thus, even if the parameters are discontinuous, the phase trajectory that results from using these parameters is smooth by construction.

[3]PPMPs do not require a particular policy search algorithm. In principle, it can be used with any black-box optimizer.

in (6). By the end of the roll-out, the cost of using this particular weight vector is then assessed. Once the optimization is finished, the weights are fixed and a roll-out can be run repeatedly in single-stroke cases, or the same roll-out can run indefinitely in cyclic cases.

3. Results

This section describes experimental results on a cyclic basketball pushing task using a real human-sized upper body bi-manual humanoid comprised of 17 DoFs (seven DoFs in each arm, and three DoFs on the waist). While the ball pushing is a cyclic task, we later show experimental evidence on the generality of the method motivated by a single stroke task of handovers. In the ball pushing task, the ball adds three DoFs to the task as Cartesian positions such that the PPMP encodes 20 DoFs.

3.1. Phase Portrait Movement Primitives on a Cyclic Ball Pushing Task

Motivated by the ball passing game described in the introduction, we describe experimental results where a Phase Portrait Movement Primitive (PPMP) was learned to reproduce the skill of dynamically receiving and passing a ball. Since the robot has a limited bandwidth to respond to the fast movements of the ball, under a successfully trained policy, this experiment allows us to validate the anticipative action that was motivated in Figure 2. Also, because the ball can be easily manipulated by an external agent, it is easy to introduce large disturbances into the system to evaluate its robustness. The ball was attached to a 1.5-meter string hanging from the ceiling such that it would sit in front of the robot when resting. The string limited the ball's travel range and ensured its return, facilitating the execution of roll-outs during training. The robot task was to persistently maintain the ball on a limit cycle. To do so, the robot had to repeatedly push the ball as if it was passing it to someone in front of it, and to smoothly decelerate the ball to avoid bouncing during its return while also preparing for the next push.

3.1.1. PPMP Design from Motion Capture

To design the PPMP, we executed the procedure previously illustrated in Figure 3 where demonstration trajectories of a ball push-receive were recorded as Cartesian trajectories of T time steps for both the hands and the ball as shown by the row of snapshots in Figure 5 (a). Markers were attached to the right hand of the demonstrator and on the ball. The right hand trajectory was mirrored across the sagittal plane of the robot to enforce symmetrical dual-arm trajectories such that $\boldsymbol{x}_t^{\text{end-eff.}} = [(x, y, z, q)_{\text{left}}, (x, y, z, q)_{\text{right}}]^\intercal \in \mathbb{R}^{14}$ ($q \in \mathbb{R}^4$ is the quaternion). The plot in Figure 5 (b) shows the recorded trajectories in the Y direction of the left hand and the ball during a full cycle of pushing and receiving. By comparing the amplitudes of the movements, it is noticeable that the ball had a much larger travel range than the hands meaning that during most of the time, the task is underactuated as ball and hands are not in contact.

The PPMP was designed using inverse kinematics (IK) to find the corresponding joint angle trajectories of the robot based on the recorded human's hand trajectories $\boldsymbol{q}_{1:T}^{\text{robot}} = \text{IK}(\boldsymbol{x}_{1:T}^{\text{demo}})$. These joint trajectories were paired with the trajectory of the target ball $(\boldsymbol{q}^{\text{robot}}, \boldsymbol{x}^{\text{ball}})_{1:T}$. To avoid multiple demonstrations we artificially generated perturbed simulations using the real demonstration as a nominal trajectory, a procedure that is explained in detail in the Appendix. Figure 5 (c) shows the corresponding demonstrated phases of the agent and the ball, both computed with (7). Note that we used the Y

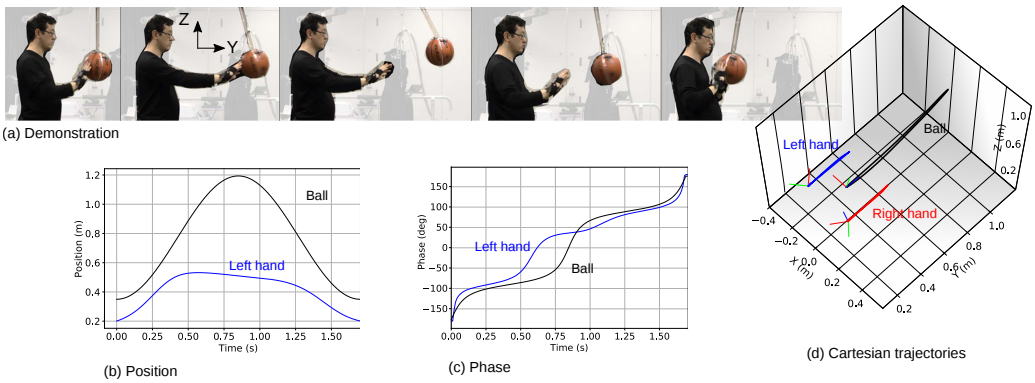


Figure 5: **Demonstration of a ball pushing.** (a) Demonstration of a ball push-receive task and the resulting Cartesian trajectories (right). Note that the ball hangs from the ceiling by a rope. The trajectories along the Y direction (b) and their phases (c).

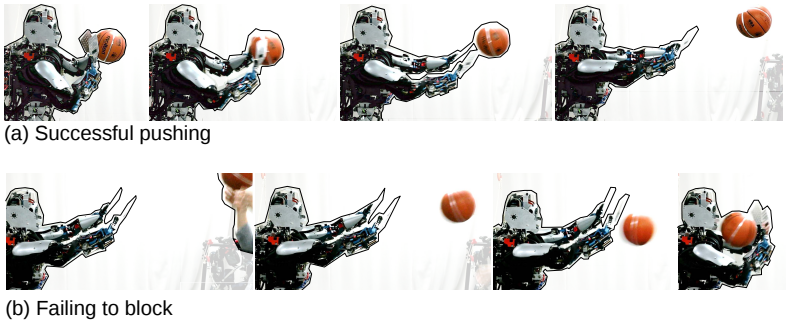


Figure 6: **Full cycle with linear coupling.** Using constant coupling parameters K and α the robot could push the ball but could not block it in time during its return indicating that a linear phase predictor does not suffice to cover different regimes of the cycle.

coordinates of the movement as it is the direction that measures the distance between the ball and the robot, and thus describes the phase of the interaction.

3.1.2. Policy Search on the Real Robot

Our initial attempts in using constant coupling parameters rendering a linear phase predictor were unsuccessful, an indication that a nonlinear parametrization is necessary for this task. As shown in Figure 6, while the dynamics of the coupled oscillators could be tuned to properly push the ball (upper row), the same settings did not succeed when receiving the ball (bottom row). In fact, as results show, to forcefully push the ball away, the robot must aggressively track the ball phase with a positive phase shift which sets its phase to be ahead of the ball. Conversely, to softly decelerate the incoming ball and avoid bouncing, a smooth and compliant phase tracking where the robot starts largely advanced in phase but recedes as the ball approaches is necessary. It is important to keep in mind that although the optimization is only on the parameters of coupled oscillators, as it was shown in Figure 4, its effect passes through the kinematics of phase portraits, such that the RL procedure consolidates the full spatio-temporal spaces as a single policy.

We follow the procedure of Section 2.3 to optimize the policy. Since the goal of the task was to

maintain the ball on a persistent limit cycle we transcribed this requirement with the cost

$$c_t(y_t) = v_1 \sum_{j=1}^{17} (\ddot{q}_t)_j^2 + v_2 \, |y^{\text{goal}} - y_t^{\text{ball}}| + v_3 \, (0.5|y_t^{\text{left}} - y_t^{\text{ball}}| + 0.5|y_t^{\text{right}} - y_t^{\text{ball}}|). \tag{13}$$

where $\ddot{q}_t$ is the joint acceleration at time step t, y^{goal} is a goal value set to 3 meters which encouraged the robot to strongly push the ball far away from itself. The left and right end-effector positions are given by y_t^{left} and y_t^{right}. The last term rewards the robot for keeping its end-effectors and the ball distant from each other, which is only possible if the robot learns to constantly push the ball away from itself. The weights of each component $\{v_1, v_2, v_3\} = \{10, 5, -20\}$ were tuned by hand. Since the task is cyclic, a roll-out was defined by its duration and its cost C was computed as the average of the instantaneous costs

$$C(\boldsymbol{y}_{1:M}) = \sum_{t=1}^{M} c_t(y_t)/M \tag{14}$$

where M is the total number of time steps during the roll-out. Each roll-out was set to last 30 seconds allowing the robot to attempt 15 to 17 pushes per trial.

To allow for fast and aggressive initial explorations, the first 10 policy updates—each update consisting of 10 roll-outs—were run in a simulated environment under large exploration noise. Subsequently, seven additional policy updates—each consisting of five roll-outs—were run on the real setup totaling 17 minutes of training. Figure 7 (a) shows the reduction of cost first in simulation and subsequently using the real robot in (b). At each update cycle, the larger circles (red in simulation and blue in real experiments) represent the cost of a "clean" roll-out, that is, the roll-out where the best current policy was evaluated. The small light circles represent the cost of "exploration" roll-outs, that is, the roll-outs where the base policy was added with additional noise in an attempt to generate better (lower cost) versions of the base policy. The variance of costs was larger in simulation than in the real setup due to the larger exploration of the parameters. Compared to the costs in simulation (which aggressively decreased from -2 to -9 due to larger exploration noise) the absolute cost on the real robot was lower (decreasing from -14 to -16) as the ball had a larger travel range on the real setup.

Figure 7 (c) and (d) show the changes in the policy in simulation (light red) and subsequently after switching to the real system (light blue). The final solutions in simulation and on the real robot are shown as the thick red and blue curves, respectively. The final coupling functions indicate that the stiffness and phase differences have large values at the beginning of the cycle. The pushing occurred at the beginning, requiring a large phase difference to put the motion of the robot ahead of the ball. The high stiffness also enforces a close phase tracking at this critical stage. As the ball returns to the robot (between 50-150 degrees), the phase difference increases again such that the hands returned faster than the ball, and the difference decreases at the end, an indication that the robot attempts to approximate its phase with the ball phase for a synchronized deceleration under high tracking gains. From Figure 7 (c) it is noted that the coupling gain K increased substantially when the optimization switched from simulation to the real case. This increase suggests compensation for the dynamics of the real robot, which presents delays, tracking error, and compliance; while in the simulator the robot was as an ideal reference tracker with infinite bandwidth.

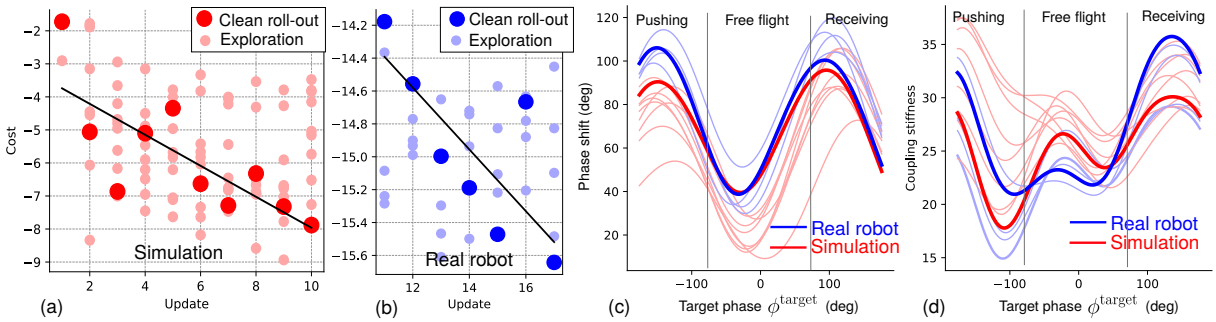


Figure 7: **Cost and policy improvements on the repetitive ball pushing task**. (a-b) Cost decrease in simulation followed by optimization in the real robot. The big and small circles represent the cost of exploitation (clean) and exploration (noisy) roll-outs, respectively. The lines are first-order polynomials fit to the clean roll-outs. Due to the stochastic nature of the exploration, it is not possible to make strong guarantees on the convergence (e.g. monotonic). However, the trend lines show clearly that the improvement was achieved within the set of updates. The optimization histories of the phase difference $\alpha(\phi^{\text{target}})$ and of the stiffness $K(\phi^{\text{target}})$ are shown in (c) and (d), respectively. The thick red curves represent the optimized functions in simulation and the thick blue line its final result after optimization using the real robot.

3.1.3. Evaluating the Optimal Policy with Spatio-Temporal Disturbances

Figure 8 (a-b) shows the use of the optimized policy on undisturbed ball trajectories as the robot repeatedly pushed and received the ball for an entire minute. While the plotted trajectories were obtained in real experiments, the figures are overlaid with a graphical image of the robot to facilitate interpretation. As shown in the side-view in (a), the strongest pushes moved the ball almost 1.5 meters away from the robot in the Y direction. Although the ball was not externally disturbed, the lateral swing of the ball barely stayed on the sagittal plane of the robot (the plane that cuts the body symmetrically in right and left sides) creating elliptical paths 44 cm wide and forcing the robot hands to move sideways (see subplot (a)). The large variation of ball trajectories is the result of uneven dual-arm pushes, in part, due to the difficulties in setting a perfectly symmetric scenario. Subplot (b) shows two segments where the robot moves to block the ball (left), and subsequently pushes it (right). The left and right hand's non-trivial and dissimilar paths act in concert to achieve a successful manipulation of the ball. These complex actions are the result of imitating the coordination implicit in the human demonstration.

We validated the fast adaptive mechanism of PPMPs employing spatio-temporal disturbances. The robot reacted almost immediately—with a bandwidth between 30-60 Hz which was limited by the frequency of the RGB-D camera. Figure 8 (c) shows a trial where a person often disturbed the task. The sequence of snapshots shows the moment someone grasped the ball mid-flight and pretended a pass to the right and then to the left of the robot. Note from frames 1 and 4 that the arms of the robot are fully extended as the ball is moving away from the robot. In frames 3 and 6, the robot preemptively positioned its hands to receive the ball at the appropriate positions as the phase of the robot was largely advanced with respect to the ball. This behavior is the exact one described in our initial motivation in Figure 2 where the robot advances its phase to wait for the ball landing. The normalized curves at the right of Figure 8 (c) show the trajectories of the ball and the robot's hand moving along the pushing direction. In the first segment, the ball flight was disturbed by someone vigorously rocking the ball back and forth. In the second half, the ball was released and the robot could graciously recover and bring the ball back to a limit cycle (refer to the video here `https://gjmaeda.github.io/videos/PPMP/PPMP_ball_pushing.mp4`, to

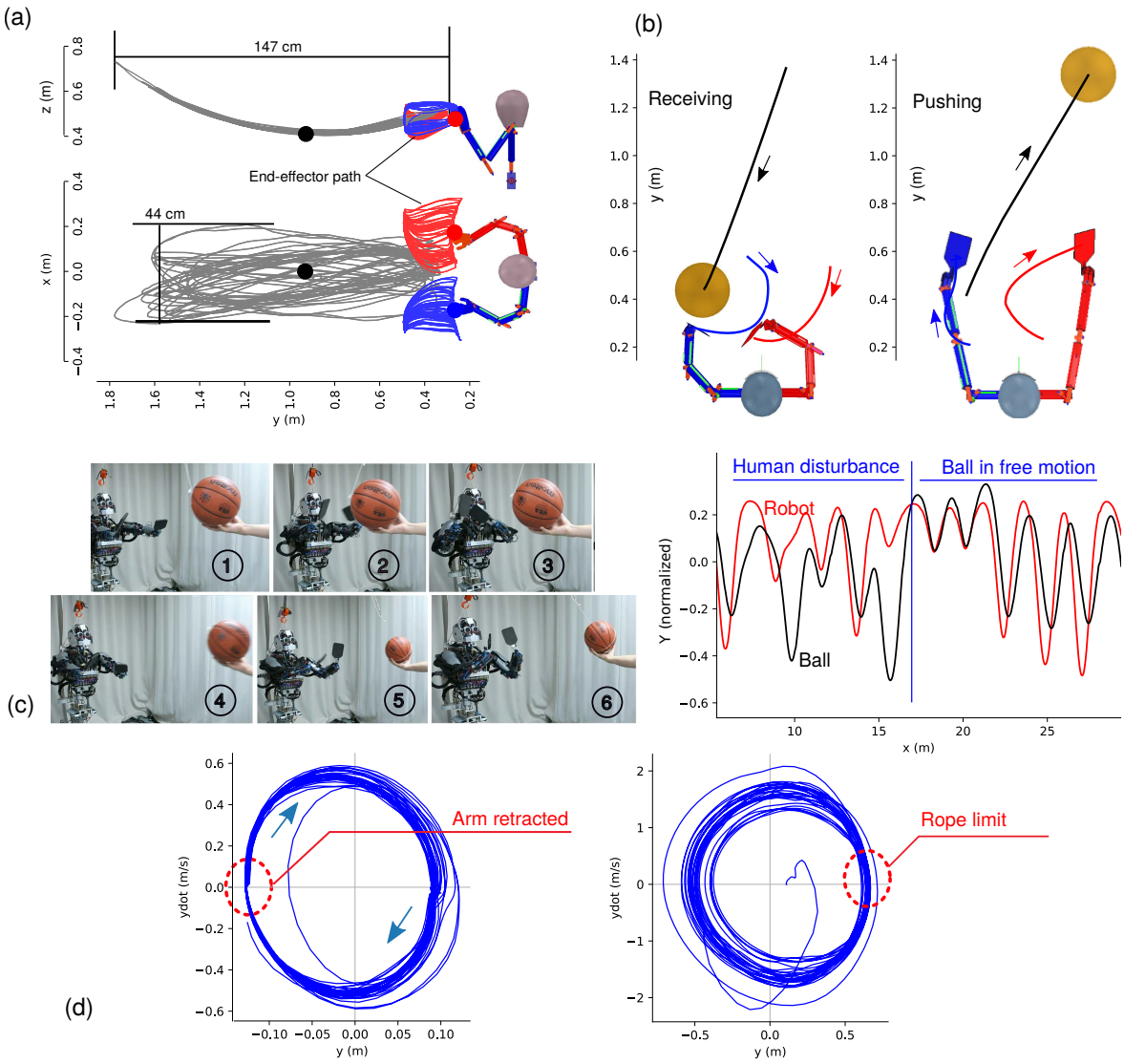


Figure 8: **Extended experiments using the final policy.** (a) The paths of the ball and hands as the robot interacted with the ball for one minute. (b) Two real trajectories of the ball receiving and pushing the ball. Note the non-trivial asymmetry of hand trajectories. The robot's illustration is shown to facilitate interpretation. (c) A one-minute trial where a person grabbed the ball and pretended to pass the ball but reversed the trajectory a few times. Note from the snapshots that the robot tried to adapt by changing its pose. The curves show the left-hand trajectory in the Y direction as the robot responded to the changes in the ball flight. The phase plane trajectories of an undisturbed experiment are shown in (d). The highlighted areas represent the physical constraints of the robot joint limits and the length of the rope. A video of the ball pushing experiment can be accessed with the link `https://gjmaeda.github.io/videos/PPMP/PPMP_ball_pushing.mp4`.

better understand the dynamics and intensity of disturbances applied during the experiments).

Figure 8 (d) shows the phase plane trajectories corresponding to an undisturbed limit cycle of the ball. The low variance at the leftmost part of the phase of the robot is due to the arm being in a fully retracted pose, ready for a push. In the case of the ball phase plane, the variance is lower at the rightmost part of trajectories as the ball was achieving its maximum travel range constrained by the length of the rope. The phase-plane trajectories of the ball evidence a stable limit cycle. In extended experiments, the robot could maintain the ball on a limit cycle for more than five minutes, which resulted in more than 120 consecutive pushes.

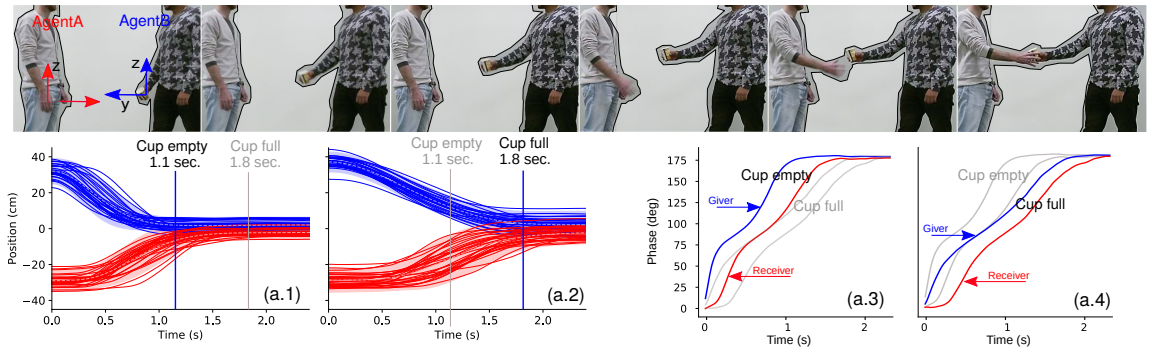
This experiment allowed us to evaluate the responsiveness of the PPMPs when adapting to the extremely large and arbitrary spatio-temporal disturbances introduced by a human manipulating the ball. Also, the experiments provided empirical evidence that the dimensionality of the PPMPs makes it feasible to train the policy with reinforcement learning and to run the algorithm at 30 Hz on a system comprised of 20 DoFs in total (17 DoF robot of the robot and three DoFs of the moving ball). It is worth noting that the PPMP was learned in a semi-model-free setting, where the only modeling assumption was the two coupled oscillators. In contrast, in online optimal control, the designer is first met with the challenge of modeling, identifying parameters, and validating the predictions of the bi-manual contact forces which, per se, is arguably more challenging (if not impractical depending on the required accuracy) than the entire design of the PPMP itself. Only after the system identification step is overcome, the designer would be able to proceed with the implementation of the necessary optimization routines.

3.2. PPMPs on Single Stroke Tasks: Handover Case

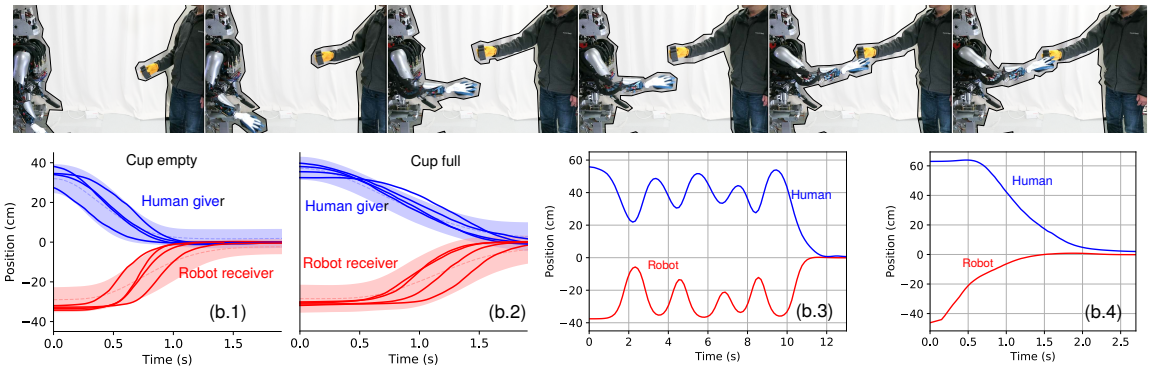
PPMPs are not exclusive to cyclic tasks. The methodology can be applied without modifications to single-stroke tasks as well. As in the cyclic case, the only mild assumption is that the dynamics of the coupled oscillators are sufficient to describe the temporal interactions of the task at hand. Experiments in handovers were chosen not only because handovers are one of the most studied tasks in the field of physical human-robot interaction [23] but also because it is a task where *timing* is very relevant for fluid and natural interactions.

3.2.1. Handover Demonstrations

We implemented a single PPMP to interact with a human partner under different timings in handovers. To this end, we recorded a total of 30 demonstrations of a cup handover where the cup was empty, and 30 demonstrations where the cup was filled with water. The upper row in Figure 9 (a) shows a sequence of snapshots of one demonstration instance. The right hand trajectories of each demonstrator were recorded as a sequence of Cartesian coordinates $\boldsymbol{x}^{\mathrm{A}}_{1:T}, \boldsymbol{x}^{\mathrm{B}}_{1:T}$, with $\boldsymbol{x}_t = [x, y, z]^\intercal \in \mathbb{R}^3$ via motion capture. The first snapshot shows the convention of coordinate frames where the horizontal and vertical axes are on the sagittal plane of the agent. Figures 9 (a.1, a.2) show the two sets of demonstrations as a distribution (mean $\pm$ two standard deviations) for the case when the cup was empty and full, respectively. The figures also indicate the mean settling time of the handover. As expected, when the cup is empty the settling time is shorter than when the cup is full (see indications on plots (a.1, a.2)). In the context of phase dynamics control, the underlying hypothesis is that the giver acts as the temporal reference, and the receiver adapts its phase according to the giver's progress.



(a) Human-human demonstrations



(b) Human-robot experiments

Figure 9: **PPMP in discrete handover tasks**. (a) Snapshots show one instance of human demonstrations using markerless skeleton tracking. (a.1, 2) Show 30 demonstrations for the cases where the cup is empty and filled with water, respectively. (a.3, 4) The progress of the phases using the averaged values on the empty and full cup cases. The grey curves represent the opposite cup state to facilitate comparison. (b) Experiments using the real robot where agent A was replaced by the humanoid. (b.1,2) Examples of handovers using the real robot as a receiver. Subplot (b.3) shows the case where the human tricks the robot by pretending passes but retracting his hand a few times. (b.4) The robot acting as a giver where its phase evolves in open-loop. A video of the handover experiment can be accessed via the link `https://gjmaeda.github.io/videos/PPMP/PPMP_handover.mp4`.

Subplots (a.3-4) summarizes the phase relationship by the average progress of the human phases for both empty and full cup cases. The phase of each agent was computed with (7). To facilitate comparison, the figure shows the opposite cup condition (full/empty) in grey. It is noticeable that when the cup is empty, the phase of the slower agent (the receiver) reaches 180^0 at around 1.5 seconds, while when the cup is filled with water its phase achieves the same value at around 2 seconds. In this single stroke case, only the first and second quadrants of the phase plane are traversed as no returning path exists.

3.2.2. Handover Experiments with the Real Robot

Compared to the cyclic ball pushing, controlling the handover is possible with a linear control law with constant coupling parameters K and α. In this case, it is not hard to tune these two values by hand. The plots in Figure 9 (b.1) show the trajectories of four different experiments where the trajectories of the human and the robot hands movement are overlaid on the distribution of demonstrations corresponding to the empty cup case. The values of the coupling were set to $K = 30$ and $\alpha = -65^0(\pi/180^0)$. Figure 9 (b.2)

show four similar cases for slower movements of the giver handing over a cup full of water. Compared to the empty cup response, the more sluggish response of the robot was achieved by decreasing the value of the coupling stiffness to $K = 20$.

The humanoid upper body was used to replace the role of agent A as the receiver of the object. At each updated position of the giver's hand, estimated using the color of the glove and a depth sensor, the PPMP provided the vector of joint angles $\boldsymbol{q}_t$ that defined the corresponding pose of the robot. The computed joint angles were used as reference angles for the low-level position tracking controller of the robot. Figure 9 (b) shows a sequence of snapshots of a handover where the robot receives the object from the human giver. All experiments used the same phase portrait with different coupled oscillator parameters.

The fast phase adaptation and prediction of PPMP make the robot react as if it had a sense of time similar to humans. This feature is shown in Figure 9 (b.3) where during the handover the human purposely rocked his hand back and forth before finally handing over the object. The robot could adaptively advance and retract its hand, in the same way, people would do in such a situation. To better understand the dynamics of the experiments, an extensive sequence of handovers can be watched in the linked video.

For completeness, and to demonstrate the flexibility of the oscillator dynamics formulation, we swapped the roles between the human and the robot by running the phase estimator in open-loop. That is, the indexes of the phase-portrait $P(\boldsymbol{x}, \boldsymbol{q})_{\phi^{\text{robot}}}$ were run as a function of time where timing was reproduced from one of the demonstrations, $\phi^{\text{robot}}_{1:T} = \phi^{\text{A}}_{1:T}$. In this way, the robot moved independently of the human partner's progress, although the position of its hands was still being coordinated with that of the human. One particular instance is shown in Figure 9 (b.4).

3.2.3. PPMP vs Hard-Coded Trajectories

As discussed, one of the main advantages of PPMPs is to be a semi-model-free approach which only requires the general dynamics of coupled oscillators to generate predictive adaptation. In contrast, take for example the recent work of Pan et al. [24] who implemented a handover controller where the joint trajectories were hard-coded for each degree of freedom of a 7-DoF arm. To account for the case where the robot needed to retract its arm, Pan et al. used a kind of finite-state machine approach to activate the action for arm retraction. In PPMPs, the arm retraction is achieved without additional high-level rules, as it is simply the result of the adaptation provided by the coupled oscillators. Also, PPMPs dynamically control the timing along the entire trajectory while in [24] only the initial delay can be manually adjusted. The reader is invited to watch the video `https://www.youtube.com/watch?v=w1Ff4nqcUvk` of Pan et al., and compare with the accompanying handover video of this paper `https://gjmaeda.github.io/videos/PPMP/PPMP_handover.mp4` while qualitatively observing the similarities in the robot's reaction. While in [24] a robot acting based on engineered trajectories and human-made rules allowed the authors to focus on the effect of timing in social human-robot interaction, our focus with PPMPs is to eliminate hard-coding or engineering of tasks to pave the way for autonomous learning.

4. Discussion

This section discusses aspects of PPMPs concerning other primitive representations. In particular, we apply the method in the cyclic task of walking to compare it against a periodic Dynamical Movement

Primitive used as a Central Pattern Generator. We also present some other observations related to direct feedback tracking and the temporal smoothness of the PPMP.

4.1. Phases in other Movement Primitive Representations

In manipulation, many authors have proposed different ways to provide primitives with time-independence. These can vary from the use of Hidden Semi-Markov Models to learn the transition dynamics of the movement[25], to scaling the velocity of a learned dynamics used for trajectory prediction [26], to using the ratio between the current robot state and the remaining path till the goal [27], to cite a few (a concise review with many approaches can be found in [3]). Dynamical Movement Primitives (DMPs) [11] have long suggested the explicit use of phases to replace time, and more recently, Probabilistic Movement Primitives (ProMPs) [28] also followed the same idea. However, in real robots, the use of phases in existing formulations has been quite simplistic; mainly as an open-loop signal to synchronize multi-DoF systems and to adapt the speed of movements. Here, we state three advantages of PPMPs with respect to existing movement primitive formulations.

Fast Predictions in Phase-Space. PPMP provides a principled and efficient way to control phases. Also, note that its oscillator can be used with existing movement primitives explicitly parameterized by phases such as DMPs and ProMPs. Compare the phase mechanism of PPMP, DMP, and ProMP cases,

$$\begin{aligned}
\dot{\phi}^{\text{robot}} &= \omega + K\sin(\phi^{\text{target}} - \phi^{\text{robot}} + \alpha) \quad \text{PPMP,} \\
\tau\dot{\phi}^{\text{robot}} &= -\alpha_s\phi^{\text{robot}} \quad \text{DMP in the discrete case,} \\
\tau\dot{\phi}^{\text{robot}} &= 1 \quad \text{DMP in the cyclic case,} \\
\phi^{\text{robot}} &= f(t) \quad \text{ProMP,}
\end{aligned} \tag{15}$$

where α_s is a positive constant value, τ is the system time constant, and $f(t)$ is any monotonically increasing function. Since the phase of DMPs and ProMPs evolve in open-loop, temporal adaptation relies on external mechanisms to correct the phase (e.g. by using Kalman filter when model parameterization is possible [10], or by learning models for unknown object dynamics as it was done in [3]). In contrast, Phase Portrait Movement Primitives (PPMPs) exploits the use feedback in the form of the coupled oscillators for tracking and predicting phases, thus achieving much faster adaptation under much lower computation and modeling costs. On the other hand, the reactive feedback nature of the coupled oscillators should not be confused with the inability to predict states in the future as a positive phase shift advances the robot ahead of the current temporal evolution of the interaction.

Scalability for Joint-Space Control. A significant computational advantage over DMPs for joint space control is the fact that a single PPMP is used regardless of the number of degrees-of-freedom of the robot. This is possible because the same PPMP encodes the correlation of all degrees-of-freedom of the robot, which is a feature also provided by ProMPs and GMMs. On the other hand, for joint-space DMPs and joint-space GPs [29, 30], the number of primitives scales with the number of joints.

Fast Spatial Adaptation on Cyclic Tasks. The third advantage of our method is evident in cyclic tasks. PPMPs natively allow the robot pose to be adjusted instantaneously, at each time step. Periodic DMPs require optimization over the entire limit cycle as its goal attractor can only modify the averaged behavior of the cycle but not the instantaneous position at each time step. We illustrate this difference

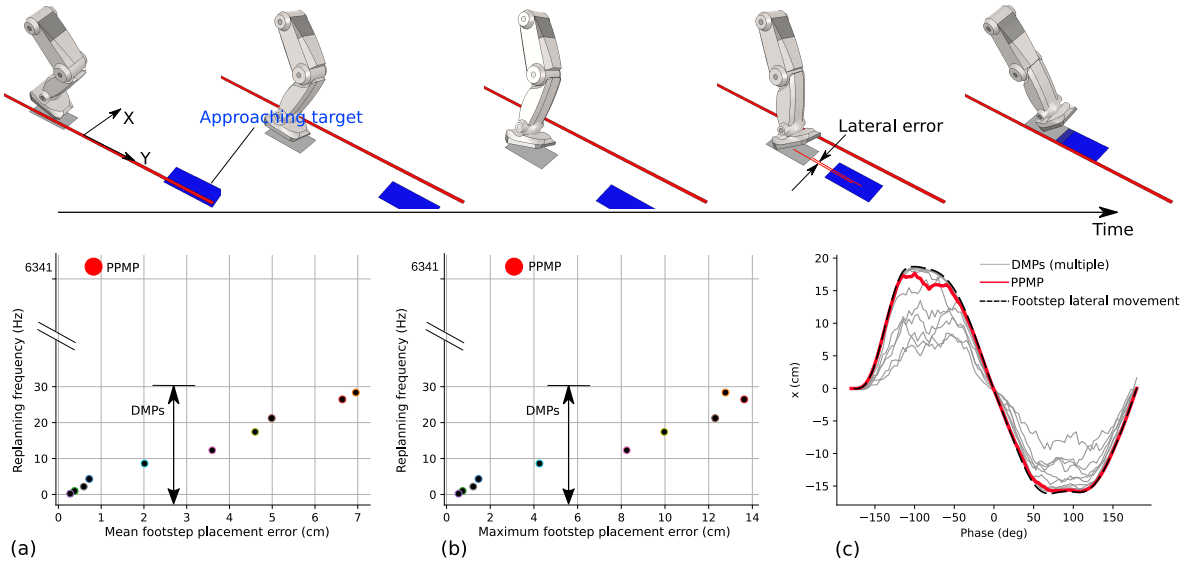


Figure 10: **DMP vs PPMP in a rhythmic task.** Upper row: a sequence of snapshots where PPMP is used to infer the joint angles of a robotic leg given the footstep placement while executing a walking cycle. The phase oscillator dynamics define the progress of the walking pattern while the lateral target motion dictates the foot placement. (a-b) when using DMPs, smaller foot placement errors (mean and maximum) can be obtained by allowing the optimizer to converge. In turn, waiting for convergence leads to low re-planning frequencies. By controlling the maximum number of allowed iterations, the re-planning frequency can be increased up to 30 Hz, at the expense that the error of the DMP also increases as the optimization process is truncated. In (c) the effect of the planning frequency is observed in terms of the DMP trajectories. Trajectories with larger errors were planned faster. In general, PPMP plans at kHz order while achieving the same error of DMPs planned at 10 Hz.

by a robotic walking task as shown in Figure 10 where PPMPs and DMPs are compared quantitatively. As shown by the snapshots, in this task, the incoming desired footstep placement (assumed given by a perception system) moves sideways and the robot must adapt the foot position laterally while the walking cycle evolves. On rhythmic DMPs, this adaptation requires optimizing the forcing function parameters with a cost that penalizes for lateral error position In PPMPs adaptation requires solving $\boldsymbol{q} \sim P(\boldsymbol{q}|\boldsymbol{x}^{\text{target}})$ which has a closed-form solution for Gaussian distributions. Figure 10 (a) show that the DMP error in footstep placement increases with the increase of the replanning frequency[4]. In the case of PPMP, no optimization is required and the frequency of replanning runs two orders of magnitude faster[5]. Finely optimized DMPs can achieve less error than the PPMP at the expense of slower updates (less than 5 Hz). This is because PPMP computes the robot pose by inference. As such, its accuracy depends on how close the true distribution fits the assumption of normally distributed spatial models. For fine tasks that demand accuracy, a mixture of PPMPs or optimization on the PPMP solution may be necessary.

[4]As an optimizer, we used the Pi^{BB} [20] algorithm and controlled the replanning frequency by changing the maximum number of parameter updates.

[5]Both methods were implemented in Python and run on the same computer.

4.2. Direct Tracking via Position Feedback

In general, direct feedback tracking of the position of the ball or the hand of a human (e.g. with a Cartesian controller under visual servoing) cannot accomplish the same level of task complexity of movement primitives, including PPMPs. In a sense, direct feedback provides robotics reflexes: given a stimulus, it outputs an instantaneous action that does not involve reasoning over future states. On the other hand, PPMPs make use of the kinematic distributions and positive phase shifts to advance the pose of the robot with respect to the current phase of the target. For example, the motor actions of receiving and pushing the ball are not solvable by pure reflexes given by a Cartesian tracking controller. Since a feedback controller attempts to decrease the tracking error, the robot would try to move the hands until they touch the ball, but not to push it. Also, because the ball moves faster than the arm, direct tracking of position is prone to fail. In contrast, the learned PPMP policy allowed the robot to act in a predictive manner, by positioning the hand at the right location before the ball arrived and by later pushing the ball far away, a feature that no visual servoing controller can provide.

4.3. Spatio-Temporal Smoothness of PPMPs

One distinct characteristic of PPMPs is that each of the joint distributions on the phase portrait is independent of each other. That means that, in principle, the temporal states of the robot are allowed to "jump" or "skip" in time. This is different from most primitive formulations that usually rely on some mechanism to guarantee temporal smoothness to generate suitable robot commands. In the case of DMPs, the smoothness is due to the use of radial basis functions used to encode the forcing function. ProMPs similarly achieve this smoothness with the difference that the basis functions encode the positions and velocities. Other methods where the smoothness is provided by construction due to the appropriate choice of kernels or features are Gaussian Processes [29] and Gaussian Mixture Models [25] as primitives.

The mechanism responsible for temporal smoothness in PPMPs is the dynamics of the coupled oscillators, which is governed by a differential equation (1). Since the phase is the result of an integral, it can only evolve continuously over time which enforces the temporal smoothness of robot commands. The spatial smoothness in PPMPs is a natural consequence that the robot motion is conditioned on the target motion. In the experiments of this article, the targets (the ball, the human hand, the footstep placement) moved smoothly in space, that is, they did not "teleport", and as a consequence, the conditioned robot movement behaved accordingly. PPMPs shows promise in future applications under hybrid controllers where hard-switches may occur as PPMPs do not enforce smoothness regarding spatial transitions.

5. Conclusions

Real-world implementations of fast humanoid control executing bi-manual manipulation have been extremely scarce and existing cases have usually relied on domain knowledge, carefully engineered solutions, and heavy computation, all of which are not suited for autonomous learning. This article proposed PPMP, a learning control method suited for fast and anticipative tasks with a native capability to estimate temporal dynamics. Coupled oscillators provides the robot with predictive adaptation to the timings of the interaction while the associated joint distributions are used as priors to spatially correlate all degrees-of-freedom in the task. The only open parameters of the method are the coupling components between the oscillators, rendering a low dimensional representation that is amenable to the use of

reinforcement learning in real robots. While this approach is inspired by observed motor skill characteristics found in animals[5, 31] our main goal is not to reproduce biological systems per se. Rather, the PPMP goal is to be a fast humanoid control method for autonomous learning that can be designed with minimal domain knowledge and run under a low computational budget. In regards to the literature of movement primitives for robot control, building the method from scratch to include a phase predictor led to a method that is not only faster and simpler to implement particularly for rhythmic tasks, but whose scalability is not affected by the number of degrees-of-freedom of the system. The semi-model-free approach means that PPMPs are predictive in nature without relying on online optimization while being efficient enough to have its policy optimized via reinforcement learning on real, high dimensional tasks.

References

[1] B. Bäuml, T. Wimböck, G. Hirzinger, Kinematically optimal catching a flying ball with a hand-arm-system, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 2592–2599.

[2] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, E. Todorov, An integrated system for real-time model predictive control of humanoid robots, in: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2013, pp. 292–299. doi:10.1109/HUMANOIDS.2013.7029990.

[3] S. Kim, A. Shukla, A. Billard, Catching Objects in Flight, IEEE Transactions on Robotics 30 (EPFL-ARTICLE-198748) (2014).

[4] M. M. G. Ardakani, B. Olofsson, A. Robertsson, R. Johansson, Real-time trajectory generation using model predictive control, in: 2015 IEEE International Conference on Automation Science and Engineering (CASE), 2015, pp. 942–948. doi:10.1109/CoASE.2015.7294220.

[5] D. N. Lee, 16 Visuo-Motor Coordination in Space-Time, in: Advances in Psychology, Vol. 1, Elsevier, 1980, pp. 281–295.

[6] D. M. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control, Neural networks 11 (7) (1998) 1317–1329.

[7] K. A. Thoroughman, R. Shadmehr, Learning of action through adaptive combination of motor primitives, Nature 407 (6805) (2000) 742.

[8] D. Wolpert, J. Diedrichsen, J. Flanagan, Principles of sensorimotor learning, Nature Reviews Neuroscience (2011).

[9] S. Schaal, Is imitation learning the route to humanoid robots?, Trends in cognitive sciences 3 (6) (1999) 233–242.

[10] J. Kober, K. Mulling, O. Kromer, C. Lampert, B. Scholkopf, J. Peters, Movement templates for learning of hitting and batting, in: Proceedings of the 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 853–858.

[11] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: Learning attractor models for motor behaviors, Neural computation 25 (2) (2013) 328–373.

[12] C. E. García, D. M. Prett, M. Morari, Model predictive control: Theory and practice—A survey, Automatica 25 (3) (1989) 335–348. doi:10.1016/0005-1098(89)90002-2.

[13] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, R. Tedrake, Approximate hybrid model predictive control for multi-contact push recovery in complex environments, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017, pp. 31–38. doi:10.1109/HUMANOIDS.2017.8239534.

[14] J. Kober, D. Bagnell, J. Peters, Reinforcement Learning in Robotics: A Survey, International Journal of Robotics Research (IJRR) (2013).

[15] A. H. Cohen, P. J. Holmes, R. H. Rand, The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model, Journal of mathematical biology 13 (3) (1982) 345–369.

[16] J. Morimoto, G. Endo, J. Nakanishi, G. Cheng, A biologically inspired biped locomotion strategy for humanoid robots: Modulation of sinusoidal patterns by a coupled oscillator model, IEEE Transactions on Robotics 24 (1) (2008) 185–191.

[17] G. Maeda, O. Koc, J. Morimoto, Reinforcement Learning of Phase Oscillators for Fast Adaptation to Moving Targets, in: Proceedings of The 2nd Conference on Robot Learning, Vol. 87 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 630–640.

[18] M. P. Deisenroth, G. Neumann, J. Peters, et al., A Survey on Policy Search for Robotics., Foundations and Trends in Robotics 2 (1-2) (2013) 1–142.

[19] C. Bishop, Pattern Recognition and Machine Learning, Vol. 4, springer New York, 2006.

[20] F. Stulp, O. Sigaud, et al., Policy improvement methods: Between black-box optimization and episodic reinforcement learning (2012).

[21] E. Theodorou, J. Buchli, S. Schaal, Reinforcement learning of motor skills in high dimensions: A path integral approach, in: Robotics and Automation (ICRA), 2010 IEEE International Conference On, IEEE, 2010, pp. 2397–2403.

[22] J. Morimoto, K. Doya, Reinforcement learning state estimator, Neural Comput. 19 (3) (2007) 730–756. doi:10.1162/neco.2007.19.3.730.

[23] K. W. Strabala, M. K. Lee, A. D. Dragan, J. L. Forlizzi, S. Srinivasa, M. Cakmak, V. Micelli, Towards seamless human-robot handovers, Journal of Human-Robot Interaction 2 (1) (2013) 112–132.

[24] M. Pan, E. Knoop, M. Bacher, G. Niemeyer, Fast handovers with a robot character: Small sensorimotor delays improve perceived qualities, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[25] S. Calinon, A. Pistillo, D. G. Caldwell, Encoding the time and space constraints of a task in explicit-duration hidden Markov model, in: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference On, IEEE, 2011, pp. 3413–3418.

[26] S. Kim, E. Gribovskaya, A. Billard, Learning motion dynamics to catch a moving object, in: Proceedings of the IEEE/RAS International Conference on Humanoids Robots (HUMANOIDS), 2010, pp. 106–111.

[27] P. Englert, M. Toussaint, Reactive phase and task space adaptation for robust motion execution, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pp. 109–116.

[28] A. Paraschos, C. Daniel, J. Peters, G. Neumann, Probabilistic Movement Primitives, in: Advances in Neural Information Processing Systems (NIPS), 2013, pp. 2616–2624.

[29] M. Schneider, W. Ertel, Robot learning by demonstration with local gaussian process regression, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2010, pp. 255–260.

[30] G. Maeda, M. Ewerton, T. Osa, B. Busch, J. and Peters, Active Incremental Learning of Robot Movement Primitives, in: Proceedings of Machine Learning Research (PMLR) 1st Annual Conference on Robot Learning (CoRL), Vol. 78: Conference on Robot Learning (CoRL), 2017, pp. 37–46.

[31] M. T. Turvey, Coordination., American psychologist 45 (8) (1990) 938.

[32] P. Abbeel, A. Coates, A. Y. Ng, Autonomous helicopter aerobatics through apprenticeship learning, The International Journal of Robotics Research (2010).

[33] A. Vakanski, I. Mantegh, A. Irish, F. Janabi-Sharifi, Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (4) (2012) 1039–1052. doi:10.1109/TSMCB.2012.2185694.

[34] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, S. C. Jacobsen, CB: A humanoid research platform for exploring neuroscience, Advanced Robotics 21 (10) (2007) 1097–1114.

Appendix

Data Augmentation on Ball Pushing Experiments

The design of the probabilistic phase portrait asks for variations of the task to reveal the correlation between positions during training. These variations are usually achieved by multiple demonstrations. However, demonstrations are not only time-consuming, but spatial correlations can only be correctly computed if all demonstrations are free from time misalignment, which usually demands an extra step to eliminate time warping as done by many usually via dynamic-time warping (e.g. [32, 33]). To avoid

these problems, in the ball pushing task we used an artificial data augmentation procedure described as follows.

Assume paired trajectories consisting of robot joint angles and the respective target movements are available $(\boldsymbol{q}^{\text{demo}}, \boldsymbol{x}^{\text{demo}})_{1:T}$ from a single demonstration. At each time step, define the mean of a multivariate Gaussian distribution as $\boldsymbol{\mu}_t = [\boldsymbol{q}_t^{\text{demo}}, \boldsymbol{x}_t^{\text{demo}}]$. Using the known kinematics of the robot (no dynamical models are required), we can then simulate N spatial variations by perturbing the target with a $\boldsymbol{x}'_{t,n} = \boldsymbol{x}_t^{\text{demo}} + \epsilon$ where ϵ is zero-mean Gaussian noise. The respective robot pose is found by using inverse kinematics on the perturbed targets where $\boldsymbol{q}'_{t,n} \leftarrow \text{IK}(\boldsymbol{x}'_{t,n}, \boldsymbol{q}_t^{\text{demo}})$. The mean pose $\boldsymbol{q}_t^{\text{demo}}$ is used as the initial guess for the IK solver in an attempt to obtain similar joint configurations and thus preserve the normal distribution also in joint space—note that is is a heuristic for which not guarantees can be made, but in practice has worked well. This process is repeated for each time step along the recorded nominal trajectory, providing a training set of N sample variations of length T, such that $\{(\boldsymbol{q}', \boldsymbol{x}')_{1:N}\}_{1:T}$. The procedure described in Section 2.2 can then be applied to estimate the distribution parameters of the PPMP.

Experimental Setup

We used the upper body of a human-sized hydraulic humanoid comprised of seven DoFs in each arm, and three DoFs on the waist (details in [34]). The robot is position-controlled at the joint level, with PD controllers running at 500 Hz on a desktop computer (Intel Core i7-4790) using Ubuntu 16.04 with a patched real-time kernel. For simulations, we implemented a kinematic model of the robot together with a dynamic model of a ball swinging like a pendulum.

The estimation of the position of the human hand in the handover experiment and of the ball position in the pushing experiment were done by color segmentation using OpenCV methods. The perception routines were run on the same control loop of the PPMP implemented in Python 2.7. The PPMP runs on a conventional Ubuntu 18.04 laptop (Intel Core i7-7700HQ). The joint angles output by the PPMP were transmitted asynchronously to the real-time controller using socket communication.

The effective frequency of joint updates was limited by the RBG-D camera in use. When using a Kinect2 camera, the planning frequency was approximately 30 Hz. When using a Real Sense D450 camera, the planning frequency was approximately between 45 to 60 Hz. For the quantitative evaluation of the performance, only the Kinect2 camera was used. The Kinect2 was positioned externally, at the side of the robot. In the video, it is also possible to see a version of the experiment under the robot's point of view when the D450 was used instead and mounted on the robot's head.