Learning and Adaptation of Periodic Motion Primitives Based on Force Feedback and Human Coaching Interaction

Andrej Gams, Tadej Petrič, Bojan Nemec and Aleš Ude

Abstract—Dynamic movement primitives (DMP) allow efficient learning and control of complex robot behaviors for both periodic and discrete point-to-point movements either in joint or Cartesian space. They also allow efficient modulation by changing of parameters. In this paper we introduce and evaluate the means of adapting periodic DMP trajectories with respect to force feedback. We simultaneously consider two aspects: 1) adaptation of whole trajectories to comply with the constraints set by the environment; and 2) partially modifying the trajectories during the execution based on human intervention to improve the task performance. The latter can either be force-based, i.e. through physical contact, or through predefined gestures. By intervening when necessary the human acts as a tutor, instructing the robot how to modify the trajectory and bypassing the need to learn new trajectories by autonomous exploration. We introduce the approach in the context of wiping a surface, where the robot first has to acquire and maintain contact, and where later the human tutor modifies the originally learned trajectory in order to achieve the desired robot behavior. We present simulation and real world results of wiping a surface with a Kuka 7 degree-of-freedom LWR robot.

I. INTRODUCTION

In this paper we consider two problems of on-line motion adaptation. The first is the adaptation of trajectories to the external environment in order to achieve desired forces of contact throughout the complete trajectory. The second is adapting the trajectories to the interventions of an instructor, who modifies the trajectories with physical contact or with predefined gestures. Thus, he acts as a tutor, coaching the robot into desired behavior. The common point of these two approaches, besides force feedback, is also the use of a unified trajectory representation, i.e. the dynamic movement primitives (DMPs). The learning properties of DMPs can be exploited to realize both adaptation to external forces and response to coaching gestures. The combination creates an intuitive and user-friendly interface to learning and modifying robotic trajectories with the potential of creating complex interaction trajectories with a simple demonstration and a little tutoring.

Dynamic movement primitives, introduced by Ijspeert et al. [1], are the means of encoding a trajectory in the form of a linear second order differential equation with an added nonlinear forcing term, which changes the second order attractor dynamics to the desired behavior. DMPs have many favorable properties for encoding of robotic trajectories, e. g. they contain open parameters that can be used for learning without affecting the overall convergence of the system. They can easily be temporally modulated without requiring an explicit time recalculation, which can be effectively used for synchronization to external signals and devices [2]. They are robust against perturbations and can be spatially modulated to adapt to different requirements [3]. Other trajectory representations include splines [4], mixture models [5] and different dynamical systems, which can also be adapted based on force feedback [6].

Modulation of DMPs with respect to force feedback has been studied for both discrete point-to-point movements, as well as for periodic applications. Discrete trajectories were adapted to force feedback by Pastor et al. [7], who recorded the forces during an execution and used these recordings as the referential signal for a controller, plugged into the acceleration of the DMP. The accelerations were thus altered so that the same force behavior was achieved when the conditions of the task would change. On a very similar basis Kulvicius et al. [8] modified the accelerations of a DMP based on virtual forces defined from the proximity to obstacles. Additionally, the optimal gains of the trajectory adaptation were learned. On the other hand, Gams et al. [9] used force feedback at both velocity and acceleration levels of a DMP to adapt to force feedback. In several discrete iterations the approach also learned a feed-forward term to minimize the feedback signal by means of iterative learning control (ILC). Thus any desired force trajectory (profile) was made achievable if physically feasible. Using ILC requires several repetitions of the same task with resetting of the initial conditions. The latter cannot be achieved in periodic tasks, where the conditions at the start of the period change until steady-state behavior has been reached. Consequently the use of repetitive control (RC) in a feed-forward term was proposed for periodic movements instead of ILC [10].

Both ILC or RC can learn a coupling term, which is fed into the DMP at either velocity level or at both velocity and acceleration levels as a feed-forward component to reduce the feedback error signals [9], [10]. On the other hand, DMPs themselves can be modified to cancel out feedback error signals. This was presented in a wiping task [11], where the feedback force of contact with the environment was used to alter the referential (target) trajectory of DMP learning with a velocity-resolved admittance approach [12]. The wiping motion was expanded to use transient initial motions in [13] and for structural bootstrapping from sensorimotor experience [14], but the force adaptation algorithm was not altered in these works. The force adaptation algorithm from [11] provides the basis for comparison in this paper.
We modify the approach by excluding the change of the reference trajectory itself and by introducing means to online modify desired trajectories using coaching gestures.

Once learned, a periodic DMP trajectory can be modulated by changing the frequency, amplitude, and/or center of oscillation. However, in order to alter just a part of the trajectory, for example to go more left only at a certain point, one would have to re-learn the entire trajectory. To modify only parts of learned trajectories, we propose to again use the learning of DMPs, by externally changing the error of learning, similar as proposed in [15]. This way DMPs act as an efficient coaching system. The novelty in this paper is twofold: 1) we use physical contact, i.e. force feedback instead of visual coaching gestures, and 2) it is combined with learning of complete periodic trajectories for contact with the environment.

Coaching of robot trajectories has been previously applied to various tasks in different settings. For example, voice commands of a human coach were used as a reward function in the learning algorithm by Gruebler et al. [16]. Verbal instructions, applicable also for non-experts, were used to modify movements obtained by human demonstration [17]. Physical contact was also used, for example Lee & Ott [18] used kinesthetic teaching with iterative updates to modify the behavior of a humanoid robot.

The paper is organized as follows. In Section II we provide a recap of dynamic movement primitives and of learning the forcing term, i.e. the weights of the DMP. In Section III we then show how the learning of the DMP weights can be used when the reference is changed. Section IV shows how we can alter complete trajectories without external algorithms of changing the reference, also showing results. Section V expands on this notion by changing only parts of trajectories, culminating in an efficient force adaptation and coaching algorithm. A discussion which highlights the benefits and differences to other approaches concludes the paper.

II. PERIODIC DYNAMIC MOVEMENT PRIMITIVES

We provide a brief recap of the periodic notation of dynamic movement primitives (DMP), with the formulation based on [19]. Only the basics are provided as DMPs have been thoroughly discussed [1]. We also provide the equations for learning of the DMPs, i.e. the weights of DMPs, as this is the basis for both force learning and coaching.

The following applies for a single degree of freedom (DOF), i.e. one of the external task-space coordinates. It is denoted by \( y \), while \( z \) stands for the (scaled) velocity. Note that DMPs can be applied to joint space coordinates as well.

Periodic DMPs are defined as a nonlinear system of differential equations

\[
\begin{align*}
\dot{z} &= \Omega (\alpha_z (\beta_z (g - y) - z) + f(\phi)), \quad (1) \\
\dot{y} &= \Omega z. \quad (2)
\end{align*}
\]

The nonlinear part of (1), \( f(\phi) \), is comprised of a linear combination of radial basis functions \( \Gamma_i(\phi) \), defined by

\[
f(\phi) = \sum_{i=1}^{N} w_i \Gamma_i(\phi), \quad (3)
\]

\[
\Gamma_i(\phi) = \exp \left( h_i (\cos (\phi - c_i) - 1) \right). \quad (4)
\]

The variable \( r \) is the amplitude control parameter, while \( h_i > 0 \) are the widths of the kernels and \( c_i \) equally spaced along the phase \( \phi \) from 0 to \( 2\pi \) in \( N \) steps. The phase variable \( \phi \) bypasses explicit dependency on time and is assumed to increase with constant rate, where the parameter \( \Omega \) denotes the frequency

\[
\dot{\phi} = \Omega. \quad (5)
\]

The frequency does not have to remain constant, but the phase has to be evaluated on-line, for example with adaptive frequency oscillators, as in [3] and [2].

The parameters \( \alpha_z, \beta_z, > 0 \) and \( \alpha_z = 4\beta_z \) make the system (1) – (2) converge to the oscillations given by \( f(x) \) around the goal \( g \) in a critically damped manner. To realize multiple DOFs we use separate sets of (1) – (2), and a single canonical system given by (5) to synchronize them through the common phase.

The linear part of (1) – (2) defines convergence to the goal \( g \). It is only the weights \( w_i, i = 1, ..., N \), where \( N \) is the number of kernel functions, given in the vector \( w \), that define the actual shape of the encoded periodic trajectory. Only one period of motion is encoded and it repeats as the phase resets at \( 2\pi \). To encode a trajectory as a DMP, we have to learn the weight vector. The latter is accomplished using incremental locally weighted regression, where the target data for fitting is

\[
f_{\text{targ}} = \frac{1}{\Omega^2} \ddot{y}_{\text{ref}} - \alpha_z \left( \beta_z (g - y_{\text{ref}}) - \frac{1}{\Omega} \dot{y}_{\text{ref}} \right), \quad (6)
\]

obtained by matching \( y \) from (1) – (2) to \( y_{\text{ref}}, z \) to \( \dot{y}_{\text{ref}}/\Omega \), and \( \dot{z} \) to \( \ddot{y}_{\text{ref}}/\Omega \). This means that basically we learn the accelerations needed to force the otherwise critically damped spring-mass system given by the linear part of (1) – (2) to follow the desired trajectory.

Given \( f_{\text{targ}} \), \( w_i \) is updated incrementally for each time-step \( j \)

\[
w_{i,j+1} = w_{i,j} + \Psi_j P_{i,j+1} e_j \quad (7)
\]

\[
P_{i,j+1} = \frac{1}{\lambda} \left( P_{i,j} - \frac{P_{i,j}^2 r^2}{\lambda + P_{i,j} r^2} \right) \quad (8)
\]

\[
e_j = f_{\text{targ},j} - w_{i,j} r \quad (9)
\]

\( P_i \), in general, is the inverse covariance of \( w_i \) [20]. The recursion is started with \( w_i = 0 \) and \( P_i = 1 \). \( r \) is the amplitude gain. \( \lambda \) provides the forgetting factor. If \( \lambda < 1 \), then the incremental regression gives more weight to recent data, meaning that it tends to forget older ones.
III. ADAPTATION OF DMPs BY CHANGING THE REFERENCE

In this section we provide and modify the algorithm from [11], which serves as comparison for two possibilities of modifying DMP trajectories using incremental locally weighted regression, which is at the core of the DMP learning.

The algorithm in [11] provides the means to adapt to external contact by adapting the referential trajectory that a DMP encodes in the weight vector $w$. The error of learning $e_j$ in (9), updated incrementally in every time step, is calculated using (6), where the referential trajectory $y_{\text{ref}}$ (along with $\hat{y}_{\text{ref}}$ and $\dot{y}_{\text{ref}}$) in (6) is provided by (13). Omitting the details given in [11], the algorithm uses a velocity-resolved approach. For one degree of freedom denoted by $y$, it calculates the resolved velocity $\dot{y}_r$ with

$$\dot{y}_r = k_p(F_{y0} - F_y). \quad (10)$$

Here parameter $k_p$ is the force gain, $F_y$ the measured and $F_{y0}$ the desired force in this degree of freedom. The new referential trajectory $y_{\text{ref}}$ is integrated using this velocity and the starting position $y_0$ by

$$y_{\text{ref}} = y_0 + \int \dot{y}_r dt. \quad (11)$$

Instead of a simple proportional law in (10), we can also use a proportional-derivative law to increase the damping, resulting in

$$\dot{y}_r = k_p(F_{y0} - F_y) + k_d \frac{d}{dt}(F_{y0} - F_y). \quad (12)$$

Given that the measured force signal is very noisy, this might result in an even noisier resolved velocity. Inserting (12) into (11) yields

$$y_{\text{ref}} = y_0 + \int k_p(F_{y0} - F_y) dt + k_d(F_{y0} - F_y). \quad (13)$$

This has the advantage that there is no derivative term and that the system has an additional proportional term directly on the force (and not integrated force), which results in faster responses.

Even with the formulation in (13), the change of the reference $y_{\text{ref}}$ is still subject to $k_p$ and $k_d$ and will always have some delay. It is still a feedback controller, where an error has to appear in order to change the reference. Even though a DMP is incrementally encoding the reference, it can only exactly encode (without delay) adaptation to a flat surface, where the output of the integral in (13) provides the exact reference. The results are shown in Fig. 1 and 2, left. The robot adapts its height in order to maintain the desired contact force $F_{y0}$ with the surface. All of our experiments are shown also in the accompanying video.

If the surface is not constant, for example a slope, or even an arbitrary surface, the DMP will only encode what (13) will provide, and this always has a slight delay. This is evident in Fig. 3, where we show the adaptation of a DMP to a sideways tilted flat surface, performed by a real robot. The bottom plot shows that a hysteresis of force is present, depending if the robot is moving up or down the slope. The hysteresis of learned motion is also seen in Fig. 2, right, where we can

Fig. 1. The top plot shows real world results of adaptation of motion in $p_z$ direction (downwards). The resulting forces with the referential force of contact given at 6 N (dashed line) is shown in the bottom plot. As the robot was performing left-right wiping motion, some oscillations due to the contact are visible in the force measurement. Fig. 2 left shows the complete 3-D trajectory for this experiment.

Fig. 2. The complete 3D trajectories for 2 experiments where the adaptation of the trajectory in $p_z$ direction was implemented using the velocity-resolved approach. We can see in the left plot the adaptation to a flat surface. The results also depict initial learning of motion. The force results are presented in Fig. 1. The right plot shows the adaptation to an inclined surface. The wiping motion was not learned, but re-used from the experiment in the left plot. The forces, which show a clear hysteresis, are depicted in Fig. 3.

Fig. 3. The trajectory of motion when adapting to a flat but tilted surface in the top plot. The resulting forces show a clear hysteresis resulting from moving up or down the slope in the bottom plot. Fig. 2-right shows the complete 3-D trajectory arising in this experiment.
note that the bottom left-right line has a hysteresis, albeit not a very pronounced one.

IV. DIRECT ADAPTATION OF TRAJECTORIES

To exclude the delay of the algorithm in the previous section, we exploit the weight-fitting algorithm of the DMP. Let’s assume a given trajectory encoded as a DMP, where the weights w encode the trajectory. The trajectory is following the demonstrated trajectory with the error of learning in (9) at \( e_j \equiv 0 \), meaning that w does not change anymore.

By changing (9) into

\[
e_j = k_l (F_y0 - F_y),
\]

and using it in (7), the fitting, i.e. the incremental locally weighted regression will update the weights whenever the error of force in (14) will not be zero. In other words, it will adapt the trajectory to fulfill the condition of (14), which is that the actual force of contact \( F_y \) is the same as the desired force \( F_y0 \). Parameter \( k_l \) is a positive constant. Note that the implementation of adaptation should take care that the values of the inverse covariance \( P_i \) do not decrease to \( P_i \equiv 0 \) as this will stop the adaptation, given that the update of weights is multiplied by \( P \).

A feedback term can be added to the acceleration level of the DMP in order to account for noise and non-repeating disturbances, changing (1) into

\[
\dot{z} = \Omega (\alpha z (\beta_z (g - y) - z) + f(\phi) + d(F)).
\]

The feedback term can be a simple proportional control law with gain \( k_{fb} > 0 \), for example \( d(F) = k_{fb} (F_y0 - F_y) \).

In this paper we name the trajectory adaptation method based on (14) the Direct method. In simulation, where we can model the forces of contact with displacement of the elastic environment with stiffness \( k_{env} \), we can rewrite (14) into \( F = k_{env} (y_0 - y) = k_{env} \ell \). Any error of forces at end-effector will therefore introduce a position difference \( k_l k_{env} (y_0 - y) \), which will through (7) reflect in \( f(\phi) \). From (1) – (2) we can see that through integration of the DMP differential equations, \( f(\phi) \) (and consequently \( y_0 - y \)) is integrated twice, which results in a slight delay.

From a physical standpoint, the linear part of (1) represents accelerations of a spring-mass system, while \( f(\phi) \) provides the modification for accelerations that force the system to follow the desired trajectory. In order to exclude the above mentioned delay from position-difference integration, we need to change (14) so that it provides proper accelerations. These cannot be just any accelerations, but accelerations to drive the given DMP spring-mass system, which are calculated according to (6). We therefore write

\[
e_j = \frac{1}{\Omega^2} k_2 \gamma - \alpha z \left( \beta_z (g - k_2 \gamma) - \frac{1}{\Omega} k_2 \gamma \right),
\]

where \( k_2 \gamma = k_l \) \( k_{env} (y_0 - y) \) models the forces. In this paper we name the trajectory adaptation method based on error signal (16) the Diff method.

Figure 4 shows simulated results of trajectory adaptation on a tilted flat surface. We can see in the top plot the location of the surface (red) and the trajectory of the robot in three different control settings. The green plot is the result of the velocity-resolved approach, i.e. (13) provides the referential trajectory for (9). The delay of adaptation is clearly seen. The black plot shows the results of using (14), i.e. the Direct method. The blue plot shows results of using (16), i.e. the Diff method. The plots show that the Diff plot converges faster and to a smaller error that the Direct method. The errors of adaptation are shown in the bottom plot.

Real-world results of adaptation to a sinusoidal reference force, where we used the Direct method and no feedback term in (15) are shown in Fig. 5. The experiment required that the robot, after establishing contact with the table, presses on it with a changing, sinusoidal force. We can see from the results that the trajectory was adapted to the desired shape.

![Fig. 4. The results of simulated trajectory adaptation using three different control methods, with a tilted flat surface as a reference. The experiment started with the robot already in contact with the surface. We can see the reference (red) and the three resulting trajectories in the top plot. The errors of adaptation are shown in the bottom plot. See the text for a description of separate lines.](image)

![Fig. 5. The results of adapting the robot trajectory using 14, with a sinusoidal referential force. The experiment started with the robot already in contact with the surface. The referential and resulting forces are in the top plot, while the error signal, which is an input to the weight fitting, is in the bottom plot.](image)
V. ADAPTING PARTS OF TRAJECTORIES

During trajectory learning the demonstrator repeats several periods of motion and the collected data are given as reference to the incremental locally weighted regression. The trajectory is learned, but it might not exactly do what the demonstrator intended, as is often the case when giving instructions to another person on how to perform something. When the resulting motion is not satisfactory, the demonstrator can coach the other person, specifying how to alter the motion in certain parts, or simply showing the complete motion again.

In order to avoid re-learning of the complete trajectory, we can exploit the same mechanism as in Section IV to change only parts of the trajectory. We again rely on changing (9). If \( e_j = 0 \), there is no learning and the robot just repeats the trajectory it learned during the demonstration. Again, for a single degree of freedom, we change (9) into

\[
e_j = C(\text{input}), \tag{17}
\]

making the error a function of the input, where input can be either the force applied to the robot or the demonstrator’s pointing gesture, visually illustrating in which direction to change the trajectory. For the case of force input, (17) changes into

\[
e_j = k_F F_y, \tag{18}
\]

where parameter \( k_F \) scales the measured force \( F_y \). The measured force in this case should be the force exerted by the human tutor to the robot. If the robot is in contact with an object, for example when wiping the table, one must distinguish between the forces that arise from the contact with the table and as a result of friction, and the forces applied by the human operator. A simple solution is to decouple forces by direction and to include sufficient dead zones.

Fig. 6 shows the robot end-effector trajectory before and after coaching. The initial robot wiping motion is in green. The blue line shows the trajectory of the robot during coaching, i.e. while the human was pushing on it. The measured forces of contact are shown in Fig. 7. Four clear peaks of force show where the human pushed on the robot. The final wiping motion of the robot after coaching is shown in red. The initial motion was performed using a previously learned DMP, the one from Fig. 2, left. The robot found and maintained a contact with a flat surface from the start of the experiment. Coaching was applied in \( x \) direction only.

When using pointing gestures, we can use active motion capture markers to first demonstrate a motion and later use the same markers and their relative positions for tutoring. This was the case in our experiment. We defined the following repulsive force field

\[
e_j(x) = \begin{cases} 
0 & p > 0.1 \\
(0.001/p^2 - 0.1)/40 & p \leq 0.1, \ p_{1z} > p_{2z} \\
(-0.001/p^2 - 0.1)/40 & \text{otherwise}
\end{cases} \tag{19}
\]

where \( p \) stands for the distance between the robot and the closest marker. Index \( z \) is the \( z \) axis location of the \( i \)-th marker. The given force field has no effect on the robot if the closest marker is more than 10 cm away, whereas its effect increases quadratically with proximity, effectively pushing the robot away if \( p \approx 0 \). The relative location of the markers also defines if the robot is being pushed away or pulled towards the tutor. The given force field was determined empirically. This experiment is only shown in the accompanying video.

Fig. 8 shows the still images of a person coaching the motion of a robot through interaction, i.e. by force.

VI. DISCUSSION AND CONCLUSIONS

In this paper we presented an alternative to the velocity-resolved approach applied to DMPs in [11]. The approach utilizes iterative locally weighted regression of DMPs to change the weights of the DMP and therefore the trajectory of the robot based on external input. This external input is not the target trajectory, but the output of a control law, which provides a force reference for the robot. The results show that the approach reduces the error of achieved forces for arbitrary surfaces more than what is achievable with the velocity-resolved approach. The latter is limited with the
bandwidth of the admittance controller, which relies on the integral part and therefore always introduces a delay.

While we have not discussed rotations, these can be modified just as the positions, as was described also in [11]. In the case of wiping, one needs to make sure to distinguish between the forces that arise from the contact, for example wiping, and the forces and torques that should actually change the orientation. For example, when wiping a surface, the friction will produce a force in the opposite direction of wiping. This force should not change the trajectory of motion or the orientation.

Another drawback of the velocity-resolved approach is in the introduction of an external control system, which is out of scope of the well defined and stable DMP framework. This complicates the overall system structure and introduces additional stability criteria, which have to be met.

Our results have shown that the proposed approach is applicable for learning of complete trajectories as well as for adapting just parts of trajectories using physical human-robot interaction and visual pointing gestures for coaching. Thus, the proposed approach provides a user-friendly and intuitive framework for learning of periodic motions in contact with the environment. It is intuitive in the sense that it is first demonstrated and then altered by pushing on it, the more it is pushed, the more it is altered. The user can teach the desired trajectories, alter their parts, and the robot finds and maintains the desired contact with the surfaces, all in one system.

**REFERENCES**


