Accelerating Synchronization of Movement Primitives: Dual-Arm Discrete-Periodic Motion of a Humanoid Robot

Andrej Gams¹, Aleš Ude² and Jun Morimoto³

Abstract—Human-demonstrated motion transferred to a robotic platform often needs to be adapted to the current state of the environment or to modified task requirements. Adaptation, i.e. learning of a modified behavior, needs to be fast to enable quick utilization of the robot either in industry or in future household-assistant tasks. In this paper we show how to accelerate trajectory adaptation based on learning of coupling terms in the framework of dynamic movement primitives (DMPs). Our method applies ideas from feedback error learning to iterative learning control (ILC). By taking into account the actual physical constraints of the synchronous motion – through synchronization of both positions (or forces) and velocities – it is not only a more faithful representation of actual real-world processes, but it also accelerates the speed of convergence. To show the applicability of the approach in the framework of DMPs, we tested it on a formulation which encodes an initial discrete motion, followed by a periodic behavior, all in a single system. Modifications of the original discrete-periodic formulation now also allow for the use of DMP temporal scaling property. In the paper we also show how the DMP coupling can be implemented in joint space, whereas the measured forces and previous approaches always remained in the task space. We applied our approach to an example dual-arm synchronization task on Sarcos humanoid robot CB-I.

I. INTRODUCTION

Robot programming by demonstration has been extensively used for the generation of robotic motions [1], including full body motions [2]. Neurological findings motivated also the use of action primitives [3]. Amongst the various methods of encoding demonstrated motion using primitives, dynamic movement primitives (DMP) have emerged for both discrete point-to-point motion as well as for periodic motion [4]. Ernesti et al. [5] presented a combined DMP representation for both discrete transient motion and subsequent periodic behavior, in a single system. In our study we build on the same, but slightly modified computational model that combines discrete and rhythmic movement primitives in a unified representation. In this paper we show how the initially trained "discrete-transition-into-periodic" behaviour can be modified to account for the constraints of the task. The main contribution of the paper is in accelerating the speed of convergence through the modification of the error signal, based on the feedback error learning approach.

Combining discrete and periodic motion has been studied from the perspective of common neural control systems in humans [6], [7]. Besides the approach of Ernesti et al. [5] within the DMP framework, other approaches have been suggested for robotics applications. Motor primitives and central pattern generators were utilized for a combined rhythmic-discrete motion representation by Degallier et al. [8], who applied their framework to infant-like crawling and drumming on a humanoid robot. As an underlying structure they used nonlinear oscillators, which allowed the transition between different types of behaviors. Oscillators were also used in a control structure by Ajallooeian et al. [9], who showed how nonlinear phase oscillators can be morphed to an arbitrary limit cycle, including an initial transient motion.

Despite the rich and favorable properties of DMPs for robotic control, direct replication of the demonstrated movement on the robot usually does not produce the desired task behavior. Different methods were previously reported for modification and synchronization of trajectories encoded as motor primitives. Direct demonstration using DMPs can be modified by external feedback [10], [11], or can serve as an initial approximation for learning of the correct signal, e.g. using reinforcement learning [12], [13]. In order to reduce the error of task adaptation through a feedback controller, Gams et al. [14] proposed using iterative learning control (ILC) to learn task-appropriate coupling signals, which minimized feedback errors. The nature of ILC makes it applicable to discrete processes [15], and consequently repetitive control (RC) was proposed for periodic DMPs [16]. One of the contributions of this paper is in showing how coupling and ILC can be used for combined discrete-periodic tasks, represented by a single DMP system. In this paper we name this kind of a DMP a discrete-periodic dynamic movement primitive (DP-DMP). Periodic DMPs were modified using feedback error learning also in [17]. The authors learned the feed-forward accelerations provided to an inverse dynamics model in order to minimize the feedback control effort of an additional stabilizing control law. DMPs have also been extended to Interaction Primitives [18], incorporating tools for synchronizing, adapting, and correlating motor primitives between cooperating agents. Interaction forces were learned from human demonstrations in [19], [20].

In this paper we show how the method of synchroniz-
ing DMP trajectories based on learning of task-appropriate
coupling signals [14] can be 1) accelerated by incorporating
the ideas of feedback error learning and 2) applied to joint
space motion in contrast to the previous applications that
were limited to the task space. Through the first contribu-
tion, the modification of the feedback error to include both
position and velocity components, we also depict a more
faithful representation of actual-real world behavior, where
synchronized dual-arm behavior requires both. The second
contribution enables the modification of trajectories that were
initially acquired in joint space. This has the advantage of
preserving the configuration of the robot, which might be
important for redundant robots such as humanoids.

Two additional contributions show 1) how our accelerated
synchronization of DMPs through the modified feedback
error can be applied to the combined discrete-periodic DMPs,
i.e. a DP-DMP; and 2) a modification of the original DP-
DMP formulation to implement the basic DMP property of
modulating the speed of execution.

The paper is organized as follows. In Section II we show
how to modify the feedback error based on the ideas from
feedback error learning framework initially proposed by
Kawato [21]. The application of the DMP modulation in
joint space is provided in Section III. Section IV explains
the structure of the DP-DMP, including the modifications
of the original contribution of Ernesti et al. [5]. Section V
discusses simulated and real-world results. Discussion and
conclusions follow.

II. ACCELERATING THE SYNCHRONIZATION
OF MOVEMENT PRIMITIVES

When two independently controlled robots or agents, i.e.,
without a central controller, have to work in a cooperative
task, their motion needs to be synchronized. Synchronizing
two independent motion primitives can be achieved by con-
necting them, that is by coupling them so that the forces
exhibited through the actions of one have an effect of the
control of the other. In this Section we summarize how we
can couple two DMPs as was described in [14] and most
importantly, how we accelerate the learning of a feedforward
part of the coupling term \( u_{ff} \) by using ideas from feedback
error learning [21]. We first provide basics of the DMPs.

A. DMP Basics

Standard dynamic movement primitives (DMPs) are based
on a damped spring model [4]

\[
\begin{align*}
\dot{z} &= \Omega (\alpha_z (\beta_z (y - y) - z) + f), \\
\dot{y} &= \Omega z,
\end{align*}
\]

(1)

(2)

where \( \Omega \) is a constant governing the speed of execution
(the frequency) and \( \alpha_z = 4\beta_z \) are positive constants that make
the system critically damped. The parameter \( y \) is one of the
degrees of freedom used to control the robot and \( z \) is an
auxiliary variable. The goal parameter \( y \) defines the unique
attractor point, i.e. the point to which \( y \) converges if the
forcing term \( f \) tends to zero as a function of time. The robot
is controlled by integrating system (1) - (2) with the given
initial parameters \( y = y_0, z = \dot{y}_0/\Omega. \) The choice of the
forcing function \( f \) determines whether the movement encoded
by a DMP is periodic or discrete.

To couple a DMP to an external signal we add the control
signal \( u \) to Eq. (2)

\[
\dot{y} = \Omega z + u,
\]

(3)

The control signal, also referred to as the coupling term,
is defined as a combination of a feedback and a feedforward
term \( u = u_f + u_{ff}. \) ILC can be used to learn the feedforward
component, as proposed in [14]. ILC takes a few iterations
to learn. In the next Sections we show how we can accelerate
the learning and apply it to joint space.

B. Modification of Feedback Error

Let’s consider the task of synchronizing the motion of
two robot arms in the context of dual arm manipulation.
We considered the task of lifting an unknown object with
two arms in such a way that the relative object position
and orientation between the two arms remain constant. This
happens if the robot holds the object so that the force
acting on the arms is constant. If the movement of
the two arms is specified by two DMPs given by (1) - (2)
and an appropriate canonical system (see Section IV for the
DP-DMP), synchronous arm behavior can be achieved by
modifying one of the two DMPs with (3) so that the forces
and torques acting on both arms are constant, i.e. \( F(t) = F_d \)
and \( M(t) = M_d. \)

If the initial, not synchronized arm movements are spec-
ified in task coordinates, then the sensory feedback is pro-
vided by measuring gravity-compensated forces and torques
acting on both arms. If the task is to modify the movement
of the second arm so that it lifts the object together with
the first arm, then the feedback signal can be defined as

\[
e(t) = \begin{bmatrix} F_d - F_2(t) \\ M_d - M_2(t) \end{bmatrix},
\]

(4)

where \( F_d, M_d \) are the desired forces and torques and \( F_2(t) \)
and \( M_2(t) \) the actual forces and torques acting on the second
arm. In a simulated setting, we can model the forces and
torques with a spring, i.e.,

\[
e(t) = \begin{bmatrix} K_p(x_d(t) - x_2(t)) \\ K_o \log(q_d(t) * q_2(t)) \end{bmatrix},
\]

(5)

where \( K_p, K_o \) stand for the translational and rotational
positive definite stiffness matrices, respectively. The values
can differ for separate directions. \( x \) is the position vec-
tor and \( q \) the orientation quaternion. Note that the term
\( 2 \log(q_d(t) * q_2(t)) \) corresponds to the angular velocity
that rotates quaternion \( q(t) \) to \( q_d(t) \) in unit time, just like \( x_d(t) - x_2(t) \)
corresponds to the linear velocity that translates \( x_2(t) \)
to \( x_d(t) \) in unit time. Note also that standard DMP equations
(1) - (2) need to be modified to enable the integration of
orientational motion represented by quaternions. See [20],
[22] for more details and formal definition of the logarithmic
map \( \log \) and quaternion DMPs. For simplicity we assume
that everything is in the world coordinate system.
Ideally, one would use the full inverse dynamic model to calculate the motor command error given the task space error. Since accurate inverse dynamics models are difficult to obtain, feedback error learning approach [21] uses the output of a simple controller consisting of proportional, differentiation, and acceleration feedback to iteratively improve the motor command. The proportional part is provided by (5). We propose to improve the coupling performance by adding a differentiation term. Such an improvement is also a more faithful representation of the physical behavior. If the positions are to be synchronized, the velocities have to be as well. We therefore use (for a position-based error learning in simulated setting)

\[
e(t) = K_s k_1 \left[ \frac{x_d - x_2(t)}{2 \log(q_d(t) + q_2(t))} \right] + K_s k_2 \left[ \frac{x_1(t) - x_2(t)}{\omega_1(t) - \omega_2(t)} \right],
\]

where \( K_s \) comprises the elements of both \( K_p \) and \( K_o \) on its diagonal. The terms \( k_1, k_2 \) are positive constants and represent the gains of the separate position and velocity components. The terms \( k_1, k_2 \) originate from the notion of a simple controller in feedback error learning. We defined the values empirically. The error as defined in (6) is now used instead of (5) to update the coupling term. Note that an additional acceleration term is not used because the coupling of the DMPs, as shown in (3), is at the first derivative of the DMP, leaving only one level in the second-order DMPs we use. The DP-DMP we use is explained in Section IV.

C. Iterative Learning Control

Iterative learning control (ILC) provides an effective way to minimize feedback error. It uses feedback error to improve the performance in the next repetition of the same behavior. We use it to update the coupling term \( u \) after every attempt at a task, i.e., an epoch. We propose to apply the current-iteration ILC, which is given by the formula [15]

\[
u_{j+1}(k) = Q(u_j(k) + L e_j(k + 1)) + C e_{j+1}(k),
\]

where \( u \) is the coupling term (or control signal), \( k \) denotes the \( k \)-th time sample, \( j \) denotes the learning iteration, and \( Q \) and \( L \) are the learning parameters. The value of \( Q \) determines the robustness, with \( Q = 1 \) the most accurate but the least robust implementation. We used \( Q = 0.99 \). \( L \) is the gain of the error of the previous cycle and is often the same as the proportional gain of the feedback controller. ILC is distinguished from simple feedback control by the prediction of the error \( e(k+1,j) \) in the \( j+1 \) iteration, which serves to anticipate the error caused by the action taken at the \( k \)-th time step. ILC modifies the control input in the next iteration based on the control input and feedback error in the previous iteration.

III. DP-DMP COUPLING AT THE JOINT LEVEL

While demonstrations may be in task space, transfer of motion to the robot can also include joint space trajectories, acquired by kinesthetic guiding. By maintaining the movement representation in the joint space, we preserve the information about the robot configuration, which is specifically important if the robot is redundant with respect to the task. Previously modulating the DMP trajectory using external (virtual) force feedback considered only task space trajectories [14]. In the following we show how it can be applied to joint trajectories.

To map the task-space feedback into the joint space, we need to separately map both the position and the velocity feedback. In the case of modifying only the motion of the second arm with respect to the first, we obtain the following error terms:

- position-based error learning

\[
e(t) = K_s k_1 J^T_s \left[ \frac{x_d - x_2(t)}{2 \log(q_d(t) + q_2(t))} \right] + K_s k_2 J^T_s \left[ \frac{x_1(t) - x_2(t)}{\omega_1(t) - \omega_2(t)} \right],
\]

- force-based error learning

\[
e(t) = K_f k_1 J^T_s \left[ \frac{F_d - F_2(t)}{M_d - M_2(t)} \right] + K_s k_2 J^T_s \left[ \frac{x_1(t) - x_2(t)}{\omega_1(t) - \omega_2(t)} \right].
\]

Here \( J^T_s \) denotes the pseudoinverse of the task Jacobian of the second arm. The second term in the above equation is based on the fact that the relative motion of the two arms holding an object should be equal to zero.

IV. UNIFIED REPRESENTATION FOR DISCRETE AND PERIODIC MOVEMENTS

To show an extended applicability of the modulation approach, we apply it to the DMP formulation that allows us to encode a periodic motion and its initial discrete transient part, i.e., a DP-DMP. The formulation of DP-DMPs is based on the original work described in [5], with a changed canonical system to allow for temporal scaling, a basic DMP property which was not possible with the original formulation.

The structure of a DP-DMP follows the standard DMP formulation as given by Eq. (1) – (2). To encode the initial discrete movement and subsequent periodic movement, the forcing term \( f \) is defined as a function of two phase variables \( \phi \) and \( r \), i.e., \( f(\phi, r) \). The evolution of the two-dimensional phase variable \( (r, \phi) \) is governed by the following canonical system

\[
\dot{\phi} = \Omega,
\]

\[
\dot{r} = \Omega \eta (\mu^\alpha - r^\alpha) r^\beta.
\]

In contrast to [5], we added parameter \( \Omega \) also to Eq. (11), enabling temporal scaling. \( \Omega \) is defined as \( \Omega = 2\pi/p \), where \( p \) is equal to the period of rhythmic movement in seconds. The parameters \( \alpha, \beta, \) and \( \eta \) are positive constants with which the speed of convergence of \( r \) to \( \mu \) can be adjusted. We used \( \eta = 6, \alpha = 1/6, \beta = 0.001 \) and \( \mu = 1 \) as constants. To determine the initial phase \( (r_{init}, \phi_{init}) \) from where the integration of (10) – (11) should be started, we first define
the phase at which the discrete movement transitions into the periodic movement. In our experiments this transition was defined to occur at the phase \( (\mu_1, 0) \). The initial phase was then calculated by back-integrating Eq. (10) – (11) for the duration \( T_d \) of the discrete part of the movement. This duration can be obtained from the training data.

Note that both the transformation system (1) – (2) and the canonical system (10) – (11) are only indirectly dependent on time, which is essential to ensure many favourable properties of DMPs, e.g. scale and temporal invariance and easy modulation of control parameters.

Using the above phase, \( f \) can be defined as [5]

\[
f(\phi, r) = \frac{\sum_{j=1}^{N} u_j \psi_j(r, \phi) + \sum_{i=1}^{M} w_i \xi_j(r, \phi)}{\sum_{j=1}^{N} \psi_j(r, \phi) + \sum_{i=1}^{M} \xi_j(r, \phi)} \tag{12}
\]

where \( \psi_j \) and \( \xi_j \) are the forcing terms specifying the discrete and periodic parts of movement, respectively:

\[
\psi_j(r, \phi) = b(r) \exp \left( -l_j (\cos(\phi - c_j) - 1) \right), \tag{13}
\]

\[
\xi_j(r, \phi) = a(r) \exp \left( -h_j \left[ \left( \frac{r \cos(\phi)}{r \sin(\phi)} \right) - q_j \right]^2 \right). \tag{14}
\]

The centres of periodic forcing terms \( c_j = (j-1)2\pi/N + \pi/N \) are uniformly distributed with constant widths \( l_j = 1.25N, j = 1, \ldots, N \). The centers of discrete forcing terms are taken as \( q_j = [r_j \cos(\phi_j), r_j \sin(\phi_j)]^T \), where the phase centers \( (r_j, \phi_j) \) are determined so that approximately the same number of integration steps is needed to integrate (10) – (11) from \((r_{j-1}, \phi_{j-1})\) to \((r_j, \phi_j)\), for all \( j \). The first center point \((r_1, \phi_1)\) is equal to the initial phase of the movement, \((r_{\text{init}}, \phi_{\text{init}})\), while \((r_M, \phi_M)\) is equal to the phase where discrete movement transitions into the periodic movement, \((\mu_1, 0)\). The widths of the kernels are determined as \( h_j = 0.5/(\|q_{j+1} - q_j\|^2), j = 1, \ldots, M - 1, h_M = h_{M-1} \).

Kernel functions \( a \) and \( b \) implement the transition from discrete to periodic movement. We defined them using a tricube kernel

\[
b(r) = \begin{cases} 1, & r < \mu_1 \\ \left(1 - \left(\frac{r - \mu_1}{\mu_2 - \mu_1}\right)^3\right), & \mu_1 \leq r \leq \mu_2 \tag{15} \\ 0, & r > \mu_2 \end{cases}
\]

\[
a(r) = \begin{cases} 0, & r < \mu_1 \\ \left(1 - \left(\frac{h_2 - r}{h_2 - \mu_1}\right)^3\right), & \mu_1 \leq r \leq \mu_2 \tag{16} \\ 1, & r > \mu_2 \end{cases}
\]

Note that in [5] Gaussian kernels were used to define \( a \) and \( b \). The advantage of our definition is that the above kernel functions really become equal to zero and not just tend to zero as Gaussian kernels do. In our experiments we used \( \mu_1 = 1.2\mu \) and \( \mu_2 = 1.4\mu \) as constants.

### A. Learning of Coupling Terms in DP-DMPs

In the case of DP-DMP, the learning needs to apply both to the discrete and periodic part. However, ILC was originally developed for discrete processes only. Repetitive control (RC) was proposed instead of ILC to modify periodic DMPs [16], but this method cannot be applied to the initial discrete part. However, as stated in [23], ILC and RC have distinct differences, but their essential features are nearly equivalent, and ILC has been applied to processes with periodic inputs as no-reset ILC [24]. We therefore apply no-rest ILC to adapt both parts of the DP-DMP. An attempt to apply RC to the periodic part and ILC to the discrete part of motion would result in a discontinuity at the transition point from discrete to periodic movement because different learning algorithms result in different learning processes even if the underlying cost function is the same. One attempt at a trajectory, i.e. one epoch, now consists of one instance of the discrete part, followed by \( \nu \) periods of the periodic part.

A DP-DMPs can be coupled to an external signal as in Eq. (3). We define \( u \) as follows

\[
u(\phi, r) = \frac{\sum_{j=1}^{N} \tilde{v}_j \psi_j(r, \phi) + \sum_{i=1}^{M} \tilde{w}_j \xi_j(r, \phi)}{\sum_{j=1}^{N} \psi_j(r, \phi) + \sum_{i=1}^{M} \xi_j(r, \phi)}. \tag{17}
\]

Thus \( u \) is defined using the same kernel functions (13 – 14) as the forcing term (12). It contains adjustable parameters \([\tilde{v}, \tilde{w}]^T\), \( \tilde{v} = [\tilde{v}_1, \ldots, \tilde{v}_N], \tilde{w} = [\tilde{w}_1, \ldots, \tilde{w}_M] \), which are mapped to the control signal \( u = [u_0(1), \ldots, u_0(T)] \) during the execution. Similarly, mapping from the phase dependent control signal \( u \) to \([\tilde{v}, \tilde{w}]^T\) is accomplished with regression [4]. Thus learning the feedforward signal \( u \) involves its adaptation using formula (7), followed by regression to calculate the parameters \([\tilde{v}, \tilde{w}]^T\).

### V. RESULTS

#### A. Experimental Setup

In our experiments we considered dual arm manipulation, where the trajectories of the arms were encoded as DP-DMPs in joint space. Our example task was to synchronize the movement of both arms in 3D space so that the robot could perform the pre-defined object manipulation motion. The trajectory of the right arm was predefined, while the joint trajectories of the left arm had to adapt using our approach so that the two arms moved in synchrony, keeping the box held by both arms at constant relative position between the two arms. In the experiments we used a full-size humanoid robot Sarcos CB-i [25]. The setup and the resulting motion after learning is presented in the image sequence of Fig. 5. The accompanying video shows that the robot was able to hold the box between the two arms.

#### B. Dual Arm Manipulation

As described above, the joint trajectories of the arms are synchronized to minimize the feedback received in the task space. The final goal was that the arms move with constant distance for the complete discrete-periodic trajectory. One epoch of motion consists of a discrete part, followed by 4
periods of the periodic part. The robot then returns to the original position, which is also depicted in the plots.

Figure 1 shows the trajectories in $z$ (up-down) direction of the task space. The other two task-directions were already synchronized so that the robot could manipulate the object. We can see in the top plot the discrete-periodic nature of the motion, where four periods of periodic motion were executed in every epoch. The top plot shows the trajectory of the right arm in blue. The red and the green trajectories show the trajectory of the left arm – the one that adapts its motion. Two trajectories are shown for different feedback errors. Results show a clear acceleration of adaptation when using (6) instead of (5) for iterative learning control of the coupling term in (7). The bottom plot shows the relative error of motion in $z$ direction at the end of each epoch. Exponential convergence can be observed. We can see that the proposed extension makes adaptation considerably faster, converging in roughly 4 epochs, as opposed to roughly 10 epochs with the previous method. Acceleration is observable also in the velocities, as shown in Fig 2.

C. Adaptation in Joint Space

The actual adaptation of trajectories was implemented in the joint space, as proposed by (8). The adaptation of the joint space trajectories for the same experiment is shown in Fig. 3. These joint trajectories correspond to task-space trajectories shown in Figs. 1 and 2. In Fig. 3 the blue line shows the original trajectories of the first 4 joints of the robot’s left arm – the top three approximating the shoulder joint and the last one the elbow. The green lines show the adapted trajectories of motion for the same joints. Wrist joints were not modulated.

It is important to note that since the adaptation of the DMP occurs in joint space, they in fact do not match from the right arm to the left arm, but the task-space motion is synchronized. This is clearly seen in Fig. 4, where the

\begin{figure}
    \centering
    \includegraphics[width=\textwidth]{fig1.png}
    \caption{The top plot shows the location of the lead, right arm in $z$ direction in blue, and the location of the left arm as it is adapting over time. The green plot shows the modified approach from this paper using (6), while the red plot the original approach of using (5). We can clearly see accelerated adaptation. The bottom plot shows the relative error of motion in $z$ direction at the end of each epoch. Accelerated adaptation is clearly seen.}
\end{figure}

\begin{figure}
    \centering
    \includegraphics[width=\textwidth]{fig2.png}
    \caption{Results of adaptation of velocity when using (6) instead of (5) for the feedback error. Accelerated adaptation is shown with the new method.}
\end{figure}

\begin{figure}
    \centering
    \includegraphics[width=\textwidth]{fig3.png}
    \caption{Adaptation of the robotic arm in the joint space, where the modulation actually takes place. The blue lines show the original joint trajectories and the green lines the adapted trajectories for left shoulder flexion-extension (LSFE), abduction-adduction (LSAA), left humerus rotation (LHR) and left elbow (LEB).}
\end{figure}

D. Modulation of Execution Frequency

We improved the original formulation of the DP-DMP canonical system from [5] to allow for temporal scaling by modifying (11). Without this change DP-DMPs do not include one of the basic DMP properties – modulation of execution (playback) speed with only one parameter. Fig. 6 shows the results of reducing the parameter $\Omega$ from $4\pi$ to $2\pi$. In the top plot we can see that when using the original canonical system the execution is not stable, given in red. The blue trajectory shows the behavior of the system with...
Fig. 5. Dual arm box manipulation. The movement consists of initial point-to-point movement (first three images), followed by periodic box shaking.

Fig. 4. Trajectories of the left arm joints as they adapt, in green. The right arm joint trajectories are depicted in blue in all four plots.

\[ \Omega = 4\pi \]. The bottom plot shows the behavior of the system when using the modified canonical system. Note that the adaptation of the trajectories is not affected.

VI. DISCUSSION AND CONCLUSION

Before the onset of the periodic movement pattern, rhythmic movements are often started in a non-periodic way. A practical example is walking, where the first step is different from the following steps. In running, this condition is even more prominent because the initial motion acceleration might take a few steps. Treating the whole discrete-periodic process in a uniform system simplifies the structure of the control system for such tasks.

As stated in [7], rhythmic and discrete movements are not the same from the neural point of view. This difference comes into play also when adapting trajectories as proposed in our paper. For example, the convergence of coupling term learning by ILC is different for each part. Furthermore, the iterative nature of the algorithm has led to the notion of repetitive control and no-reset ILC [24].

In the paper we proposed a modified feedback within the current-iteration iterative learning control framework, taking the clue from the theory of feedback error learning [21], where position, velocity and acceleration error terms are used in a simple controller to model the behavior of a system. As evident from our results, this allows for much faster adaptation, opening up possibilities for practical implementation of different tasks. Could the same be achieved by only changing the \( L \) and \( Q \) parameters of the ILC? While it is
true that increasing these parameters will increase the adaptation speed, one must take into consideration that higher values will make the system less stable. For a thorough stability analysis of ILC and DMPs see [14]. In our case these parameters can be set lower to ensure stability. Even with significantly increased parameters, which resulted in apparent stability, i.e., the behavior initially seems stable but eventually diverges after some iterations, we could not achieve as fast adaptation as with including the velocity term. On top of that, when including the velocity term as in (6), the behavior remained stable.

The developed approach maintains a small set of tuning parameters, as effectively $k_1$, $k_2$ in the feedback term and $Q$, $L$ and $C$ have to be specified. While the former are task-specific, the latter follow a well established ILC theory [15], where decreasing the $Q$ (from the value of 1) will increase robustness but also the steady-state error. Therefore, $L$ needs to be calculated in order to maintain the stability [15], while $C$ is again a task-specific feedback gain. The proposed implementation of the algorithm in joint space allows for maintaining the demonstrated joint space posture, which is especially practical for redundant robots such as humanoids.

Our example task has demonstrated the applicability of the proposed approach in the real-world, but it only serves as a test for more demanding tasks. Lets imagine applying the approach to walking. The initial motion imitation from a human demonstration could result in the robot tipping over. Hence the movements need to be adapted very fast so that the robot does not break in the attempts. We have shown that adaptation can take as little as 3 or 4 repetitions. The biggest advantage of our system is that it is designed to allow for modulation of trajectories based on some other, more complex feedback variable. An example of such is modifying (8) or (9) to include a posture-stability criterion, which would update the trajectory of walking so that the robot would remain (more) stable. Extensions of the system that include such feedback errors remain a subject of our future work.

**References**


