Two-Degree-of-Freedom Control for Trajectory Tracking and Perturbation Recovery during Execution of Dynamical Movement Primitives

Martin Karlsson ∗ Fredrik Bagge Carlson ∗ Anders Robertsson ∗ Rolf Johansson ∗

Department of Automatic Control, Lund University, PO Box 118, SE–221 00 Lund, Sweden (e-mail: martin.karlsson@control.lth.se).

Abstract: Modeling of robot motion as dynamical movement primitives (DMPs) has become an important framework within robot learning and control. The ability of DMPs to adapt online with respect to the surroundings, e.g., to moving targets, has been used and developed by several researchers. In this work, a method for handling perturbations during execution of DMPs on robots was developed. Two-degree-of-freedom control was introduced in the DMP context, for reference trajectory tracking and perturbation recovery. Benefits compared to the state of the art were demonstrated. The functionality of the method was verified in simulations and in real-world experiments.

1. INTRODUCTION

Industrial robots have mostly operated in structured, predictable, environments through sequential execution of predefined motion trajectories. This implies high cost for engineering work, consisting of robot programming and careful work-space preparation. It also limits the range of tasks that are suitable for robots. Improving their ability to operate in unstructured environments with unforeseen events is therefore an important field of research.

This has motivated the development of dynamical movement primitives (DMPs), that are used to model and execute trajectories with an emphasis on online modification. Early forms were presented in (Ijspeert et al., 2002, 2003; Schaal et al., 2000), and a review can be found in (Ijspeert et al., 2013). The framework has been widely used by robot researchers. For instance, the ability to generalize demonstrated trajectories towards new, although static, goal positions has been used in (Niekum et al., 2015). Online modulation with respect to a moving goal has been applied in (Prada et al., 2014) for object handover. A method to modify DMP parameters by demonstration has been presented in (Karlsson et al., 2017). Learning and adaptation based on force/torque measurements has been explored in, e.g., (Abu-Dakka et al., 2015; Pastor et al., 2013). Previous work on DMP perturbation recovery in particular is elaborated on in Sec. 2.2.

In the standard form, without temporal coupling, a DMP would continue its time evolution regardless of any significant perturbation, as discussed in (Ijspeert et al., 2013). Therefore, its behavior after the perturbation would likely be undesirable and not intuitive.

The work described in this paper addressed perturbation recovery for DMPs, and a method was developed where a two-degree-of-freedom controller was integrated with the DMP framework, see, e.g., (Åström and Wittenmark, 2013) for an introduction to the two-degree-of-freedom controller structure. The feedforward part of the controller promoted tracking of the DMP trajectory in the absence of significant perturbations, thus mitigating unnecessarily slow trajectory evolution due to temporal coupling acting on small tracking errors. The feedback part suppressed significant errors. The functionality of this method was verified in simulations, as well as in experiments in a real-time robot application. The robot used for experimental evaluation is shown in Fig. 1.

A code example is available on (Karlsson, 2017), to allow exploration of the system proposed. The system was also integrated in the Julia DMP package on (Bagge Carlson, 2016), originally based on (Ijspeert et al., 2013).
2. PRELIMINARIES

2.1 Dynamical movement primitives

A review of the DMP concept for robotics has been presented in (Ijspeert et al., 2013), and here follows a condensed description of the fundamentals. A trajectory, \( y \), is modeled by the system

\[
\tau^2 \ddot{y} = \alpha_z (\beta_z (g - y) - \tau \dot{y}) + f(x) 
\]  

(1)

Here, \( \tau \) is a time constant, \( \alpha_z, \beta_z \) and \( \alpha_x \) are positive parameters, and \( x \) is a scalar phase parameter that evolves as

\[
\tau \dot{x} = -\alpha_x x
\]  

(2)

Equation (1) is commonly written in the following equivalent form.

\[
\tau \dot{y} = z
\]  

(3)

\[
\tau \dot{z} = \alpha_z (\beta_z (g - y) - z) + f(x)
\]  

(4)

In Eqs. (1) and (4), \( f(x) \) is given by

\[
f(x) = \sum_{i=1}^{N_b} \Psi_i(x)w_i = \sum_{i=1}^{N_b} \Psi_i(x) \cdot (g - y_0),
\]  

(5)

where the basis functions, \( \Psi_i(x) \), are determined as

\[
\Psi_i(x) = \exp \left( -\frac{1}{2 \sigma_i^2} (x - c_i)^2 \right)
\]  

(6)

Here, \( N_b \) is the number of basis functions, \( w_i \) is the weight for basis function \( i \), \( y_0 \) is the starting point of the trajectory \( y \), and \( g \) denotes the goal state; \( \sigma_i \) and \( c_i \) are the width and center of each basis function, respectively. Based on the dynamical system in Eqs. (3) and (4), a robot trajectory could be generated. Vice versa, given a demonstrated trajectory, \( y_{\text{demo}} \), a corresponding DMP could be formed. The goal point \( g \) would then be given by the end position of \( y_{\text{demo}} \), whereas \( \tau \) could be set to get a desired time scale. Further, the weights could be determined by, e.g., locally weighted linear regression, see (Atkeson et al., 1997; Schaal and Atkeson, 1998), with the solution

\[
w_i = \frac{s^T \Gamma_i f_{\text{target}}}{s^T \Gamma_i s}, \quad s = \begin{pmatrix} x^1 (g - y_{1\text{demo}}) \\ x^2 (g - y_{2\text{demo}}) \\ \vdots \\ x^N (g - y_{N\text{demo}}) \end{pmatrix}
\]  

(7)

\[
\Gamma_i = \text{diag} (\Psi_i^1, \Psi_i^2 \ldots \Psi_i^N), \quad f_{\text{target}} = \begin{pmatrix} f_{1\text{target}} \\ f_{2\text{target}} \\ \vdots \\ f_{N\text{target}} \end{pmatrix}
\]  

The following coupling terms were introduced, \( \tau \dot{y} = \alpha_z (\beta_z (g - y) - z) + f(x) \) (11)

\[
\tau = 1 + k_e e^2
\]  

(10)

Here, \( \alpha_z, k_i \) and \( k_e \) are constant parameters. The parameter \( \tau \) was used to determine the evolution rate of the entire dynamical system. Further, the term \( C_i \) was added to Eq. (4) so that the coupled version of \( y \), denoted \( y_c \), fulfills the following.

\[
\tau \dot{y} = \alpha_z (\beta_z (g - y_c) - z) + f(x) + C_i
\]  

(12)

A PD controller, given by

\[
\dot{y} = K_p (y_c - y_a) + K_v (y_c - y_a)
\]  

(13)

was used to drive \( y_a \) to \( y_c \). Here, \( y_c \) denotes the reference acceleration, while \( K_p \) and \( K_v \) are control gains.

This approach from previous research has taken several important parts of disturbance recovery into account, and it should be emphasized that it forms the foundation of this presented work. In this section, however, some aspects are considered where there is room for improvement.

Denote by \( y_a \) an unperturbed trajectory generated by an uncoupled DMP, as described in Sec. 2.1. It is desirable that, in the absence of significant perturbations, \( y_a \) should follow \( y_c \) closely. If this would not be achieved, in addition to the deviation itself, \( y_a \) and \( y_c \) would be slowed down, compared to \( y_a \), due to the temporal coupling in Eq. (10). This phenomenon is visualized in Fig. 5. In (Ijspeert et al., 2013), very high controller gains for Eq. (13) were suggested, which would have mitigated the issue under ideal conditions and unlimited magnitude of the control signals. Specifically, \( K_p = 1000 \) and \( K_v = 125 \) were chosen. However, even for moderate perturbations, this would imply control signals too large to be realized practically. For instance, a position error in Cartesian space of 1 dm would yield \( \ddot{y} = 100 \text{ m/s}^2 \). In Figs. 2 and 3, two example scenarios were displayed; one where the actual movement was stopped, and one where it was moved away from the nominal path. The method described in (Ijspeert et al., 2013) was used for recovery, with prohibitively large values of \( \ddot{y} \), as a consequence. Moreover, this control system is sensitive to noise and has a dangerously low delay margin of 12 ms.

Feedforward control has been used in the DMP context previously, but then only for low-level joint control, with motor torque commands as control signals, see (Pastor et al., 2009; Park et al., 2008). This control structure was also applied in the internal controller used in the implementation in this present paper, see Sec. 6. This inner control design should not be confused with the feedforward control described in Sec. 4, which operated outside the internal robot controller, and was used to determine the reference acceleration for the robot.
practically realizable control signal. Additionally, the time constant \( \tau \) was introduced as a factor in the expression for the adaptive time parameter \( \tau_a \), see Eqs. (10) and (15). Our method is detailed below.

In order for \( y_a \) to follow \( y_c \), we applied the control law below.

\[
\ddot{y}_c = k_p(y_c - y_a) + k_v(\dot{y}_c - \dot{y}_a) + \ddot{y}_a \tag{14}
\]

Here, \( \ddot{y}_c \) was obtained by feedforwarding the acceleration of \( y_c \). This allowed the controller to act also for zero position- and velocity error. In turn, the trajectory tracking worked also for moderate controller gains; \( k_p = 25 \) and \( k_v = 10 \) are used throughout this paper. With these gains, the closed control loop had a double pole in -5 rad/s. Since the real parts were negative, the system was asymptotically stable, and since the imaginary parts were 0, it was critically damped. The delay margin was 130 ms, which was an improvement compared to 12 ms for the previous method, described in Sec. 2.2. A schematic overview of the control system is shown in Fig. 4.

Further, Eq. (10) was modified in order to include the nominal time constant \( \tau \), as follows.

\[
\tau_a = \tau (1 + k_ze^2) \tag{15}
\]

The coupling term \( C_I \) was omitted in this present method. This is elaborated on in Sec. 9.

Since \( \tau_a \) was not constant over time, determining \( \ddot{y}_c \) was more involved than determining \( \ddot{y} \) by differentiating Eq. (3). One option would be to approximate \( \ddot{y}_c \) by discrete-time differentiation of \( \dot{y}_c \). However, this would introduce difficulties due to noise amplification. Instead we determined the instantaneous acceleration as follows.

\[
\ddot{y}_c = \frac{d}{dt}(\dot{y}_c) = \frac{d}{dt} \left( \frac{z}{\tau_a} \right) = \frac{\dot{z} \tau_a - z \dot{\tau}_a}{\tau_a^2} = \frac{\dot{z} \tau_a - 2\tau k_ze \dot{e}}{\tau_a^2} \tag{16}
\]

where \( \dot{z} \) and \( \dot{e} \) are given by Eqs. (11) and (8), respectively. It is noteworthy that the computation of \( \ddot{y}_c \) did not require any first- or second-order time-derivative of any measured signal, which would have required prior filtering to mitigate amplification of high-frequency noise. Similarly, \( \dot{y}_c \) was determined by Eqs. (11) and (12). In contrast, the computation of \( y_a \) was complemented with a low-pass filter, to mitigate amplification of measurement noise.
Two different perturbations were considered in the following simulations; one where \( y_a \) was stopped, and one where it was moved. The perturbations took place from time 2 s to 3 s. The systems were sampled at 250 Hz. The same DMP, yielding the same \( y_u \), was used in each trial. The adaptive time parameter \( \tau_a \) was determined according to Eq. (15) in all simulations, to get comparable time scales. First, the controller detailed in (Ijspeert et al., 2013) was applied. Except for the perturbations themselves, the conditions were assumed to be ideal, i.e., no delay and no noise were present. The results are shown in Figs. 2 and 3. Despite ideal conditions, prohibitively large accelerations were generated by the controller in both cases.

Thereafter, the controller proposed in this paper, described in Sec. 4, was used. In order to verify robustness under realistic conditions, noise and time delay were introduced. Position measurement noise, and velocity process noise, were modeled as zero mean Gaussian white noise, with standard deviations of 1 mm and 1 mm/s, respectively. Further, an additional configuration dependent forward kinematics error was modeled as a slowly varying position measurement error with standard deviation 1 mm. The time delay between the process and the controller was \( L = 12 \) ms. This delay was suitable to simulate since it corresponds both to the delay margin of the method suggested in (Ijspeert et al., 2013), and to the actual delay in the implementation presented in this paper, see Sec. 6. (It is, however, a coincidence that these two have the same value. Nevertheless, this shows that a 12 ms delay margin is not necessarily enough.) The results are shown in Figs. 6 and 7. For comparison, the method in (Ijspeert et al., 2013), with the large gains, was also evaluated under these conditions, although without any perturbation except for the noise. Because of the time delay, this system was unstable, as shown in Fig. 8.

6. IMPLEMENTATION OF REAL-TIME APPLICATION

The implementation presented here was performed on a prototype of the dual-arm ABB YuMi robot (previously under the name FRIDA), (ABB Robotics, 2016b), with 7 joints per arm, see Fig. 1. The method described in Sec. 4 was implemented in C++, and the linear algebra library Armadillo, see (Sanderson and Curtin, 2016), was used in a large proportion of the program. The research interface ExtCtrl (Blondell et al., 2005, 2010) was used to send references to the low-level robot joint controller in the ABB IRC5 system (ABB Robotics, 2016a). The LabComm protocol (LabComm, 2017) was used to manage the communication between the C++ program and ExtC-
Fig. 8. Using the control system in (Ijspeert et al., 2013), subject to the simulated noise and time delay, resulted in unstable behavior.

Fig. 9. Yellow case (left), stop button (upper right) and gasket (lower right) used in the experiments.

7. EXPERIMENTAL SETUP

The real-time implementation described in Sec. 6 was used for evaluation. The computations took place in joint space, and the robot’s forward kinematics was used for visualization in Cartesian space in the figures presented. The functionality of the method was evaluated in two assembly scenarios. The assembly parts used are shown in Fig. 9.

For both scenarios, a new DMP for placing a stop button into the hole of a corresponding case had been taught to the robot by lead-through programming, based on (Stolt et al., 2015), prior to each trial. This implied some variation among the demonstrated trajectories, even though they were qualitatively similar. Subsequently, the DMP was executed on the robot. During the execution, a human perturbed the movement of the robot by physical contact. A wrist-mounted ATI Mini force/torque sensor was used to measure the contact force, and a proportional acceleration, in the same direction as the force, was added to $\ddot{y}_r$ as a load disturbance.

In the first scenario, the human introduced two perturbations during the DMP execution. The first perturbation was formed by moving the end-effector away from its path, and then releasing it. The second perturbation consisted of a longer, unstructured, movement later along the trajectory.

In the second scenario, a human co-worker realized that the stop buttons in the current batch were missing rubber gaskets, and acted to modify the robot trajectory, allowing the co-worker to attach the gasket on the stop button manually. During execution of the DMP, the end-effector was stopped and lifted to a comfortable height by the co-worker. Thereafter, the gasket was attached, and finally the end-effector was released. For the sake of completeness, the modified trajectory was used to form yet another DMP, which allowed the co-worker to attach the gaskets without perturbing the trajectory of the robot, for the remaining buttons in the batch. To verify this functionality, one such modified DMP was executed at the end of each trial.

The first and second scenario are visualized in Figs. 10 and 11, respectively. To verify repeatability, 50 similar trials were performed for each scenario.

8. EXPERIMENTAL RESULTS

Data from a trial of the first scenario are displayed in Fig. 12. The two disturbances were successfully recovered from as intended. The reference acceleration was of reasonable magnitude. The results from all 50 trials were qualitatively mutually similar.

Data from a trial of the second scenario are displayed in Fig. 13. First, the perturbation was successfully recovered from as intended. The reference acceleration was of reasonable magnitude. When the modified DMP was executed, it behaved like a smooth version of the perturbed original trajectory. Again, the results from all 50 trials were qualitatively mutually similar.
FIG. 11. Second scenario. The robot started its motion towards the rightmost yellow case in (a). The end-effector was stopped and lifted, and the gasket was mounted in (b). The robot was then released, and continued its motion to the case, (c) and (d). The actual trajectory was saved and used to form a modified DMP, and the robot was reset to a configuration similar to that in (a). When executing the modified DMP, the human co-worker could attach the gasket without perturbing the motion of the robot (e). The robot finished the modified DMP in (f). Data from one trial are shown in Fig. 13.

To facilitate understanding of the experimental setup and results, a video is publicly available on (Karlsson, 2016).

9. DISCUSSION

Compared to previous related research, described in (Ijspeert et al., 2013), the method in this paper contained the following extensions. Feedforward control was added to the PD controller, thus forming a two-degree-of-freedom controller. Further, the PD controller gains were reduced to moderate magnitudes. The expression for $\tau_a$ was also modified, to include the nominal time constant $\tau$ as a factor. These changes resulted in the following benefits, compared to the previous method. The feedback part allowed the controller to act also for insignificant position-and velocity errors, thus improving the trajectory tracking. Because of this, the large controller gains used in (Ijspeert et al., 2013), that were used to mitigate significant tracking errors, could be reduced to moderate magnitudes. In turn, using moderate gains instead of very large ones, resulted in control signals that were practically realizable, instead of prohibitively large. It also improved the delay margin significantly. The aspects above form the main contribution of this paper. In contrast, the modification of the expression for $\tau_a$ was not the main focus of this paper, but it was necessary since it allowed the actual trajectory to converge to the trajectory defined by the DMP, with time constant $\tau$. Without this modification, the time parameter $\tau$ would not have affected the trajectory generated by the DMP. Instead, $\tau_a$ would have converged to 1, regardless of $\tau$, which would not have been desirable.

The work presented here focused on the control structure for trajectory tracking and perturbation recovery, rather than on the perturbations themselves. Even though the perturbations in the experiments considered here emerged from physical contact with a human, the control structure would work similarly for any type of perturbation. There are many other possible perturbations, e.g., a pause of the movement until a certain condition is fulfilled, superposi-

FIG. 12. Experimental data from a trial of the first scenario. The first (from above) plot shows the path of the end-effector in the Cartesian base frame of the robot, projected on the $yz$-plane. The arrow indicates the movement direction, which started in the upper right and finished in the lower left of the plot. The two perturbations are clearly visible. The second plot shows the distance between $y_a$ and $y_c$ over time. In the third plot, it can be seen that the evolution of $y_c$ slowed down during each perturbation. Subsequently, $y_a$ recovered, and when it was close to $y_c$, the movement continued as a delayed version of $y_a$. The reference acceleration was of reasonable magnitude, as shown in the fourth plot.
that would slow down the evolution of $y_r$ in Eqs. (11) and (12). Which of these effects that would be dominant in different cases would be difficult to predict intuitively. For these two reasons, the coupling term was not included in the method proposed here, though it would be straightforward to implement. It was, however, included in the simulations where the previous method, described in Sec. 2.2, was evaluated. During the perturbations in Figs. 2 and 3, the effect of the temporal coupling was dominant, as $y_r$ did not approach $y_c$ significantly.

Apart from the perturbations induced by the human, the motion of the robot was affected by process- and measurement noise. After applying the forward kinematics to determine the position of the end-effector, the accuracy was typically $\pm 1$ mm. This implied a limitation of how accurately the robot could follow a given trajectory. Furthermore, some movement might require higher precision than what would be possible to demonstrate using lead-through programming. Then, e.g., teleoperation could be used for demonstration instead.

In the current implementation, the actual trajectory returned to the reference trajectory, approximately where it started to deviate. This might not always be desired. For instance, it might sometimes be more practical to connect further along the reference trajectory, e.g., after avoiding an obstacle. A lower value of $k_c$ would result in such behavior, however, it must then be known what value of $k_c$ that should be used. Further, one could think of scenarios where it would not be desirable to connect to the reference trajectory, e.g., if a human would modify the last part of the trajectory to a new end point. Hence, future work includes development of a method to determine the desired behavior after a perturbation. The method presented in this paper would be useful to execute the desired behavior, once it could be determined. Nevertheless, one can think of various scenarios where the recovery presented here would be desirable, such as those in Sec. 7.

10. CONCLUSIONS

In this work, it was shown how perturbations of DMPs could be recovered from, while preserving the characteristics of the original DMP framework in the absence of significant perturbations. Feedforward control was used to track the reference trajectory generated by a DMP. Feedback control with moderate gains was used to suppress deviations. This design is the first, to the best of our knowledge, that takes the following aspects into account. In the absence of significant disturbances, the position error must be small enough, so that the dynamical system would not slow down unnecessarily due to the temporal coupling. Very large controller gains would result in small errors under ideal conditions, but are not practically realizable. On the other hand, if the gains are moderate and only feedback control is used, too large errors occur.

Feedforward allowed the controller to act even without significant error, which in turn allowed for moderate controller gains. The suggested method was verified in simulations, and a real-time application was implemented and evaluated, with satisfactory results. A video of the experiments is available on (Karlsson, 2016).
ACKNOWLEDGMENTS

The authors would like to thank Björn Olofsson and Fredrik Magnusson at the Department of Automatic Control, Lund University, as well as Maj Stenmark, Mathias Haage and Jacek Malec at Computer Science, Lund University, for valuable discussions throughout this work. Anthony Remazeliass at Tecanal, Donostia, and Diogo Almeida at KTH, Stockholm, are gratefully acknowledged for pointing out some of the previous research.

REFERENCES


