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

Karlsson, Martin; Bagge Carlson, Fredrik; Robertsson, Anders; Johansson, Rolf

Published in:
IFAC-PapersOnLine

DOI:
10.1016/j.ifacol.2017.08.383

2017

Document Version:
Peer reviewed version (aka post-print)

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).

⋆ The authors are members of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University. The research leading to these results has received funding from the European Community's Framework Programme Horizon 2020 – under grant agreement No 644938 – SARAFun.

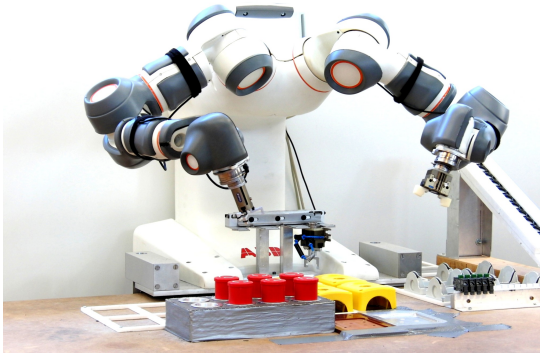


Fig. 1. The ABB YuMi robot prototype used in the experiments, (ABB Robotics, 2016b).

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

$$\tau^2\ddot{y} = \alpha_z(\beta_z(g-y) - \tau\dot{y}) + f(x) \tag{1}$$

Here, τ is a time constant, α_z, β_z and α_x are positive parameters, and x is a scalar phase parameter that evolves as

$$\tau\dot{x} = -\alpha_x x \tag{2}$$

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

$$\tau\dot{y} = z \tag{3}$$

$$\tau\dot{z} = \alpha_z(\beta_z(g-y) - z) + f(x) \tag{4}$$

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

$$f(x) = \frac{\sum_{i=1}^{N_b}\Psi_i(x)w_i}{\sum_{i=1}^{N_b}\Psi_i(x)}x\cdot(g-y_0), \tag{5}$$

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

$$\Psi_i(x) = \exp\left(-\frac{1}{2\sigma_i^2}(x-c_i)^2\right) \tag{6}$$

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; σ_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_{demo}, a corresponding DMP could be formed. The goal point g would then be given by the end position of y_{demo}, whereas τ 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

$$w_i = \frac{\boldsymbol{s}^T\boldsymbol{\Gamma}_i\boldsymbol{f}_{target}}{\boldsymbol{s}^T\boldsymbol{\Gamma}_i\boldsymbol{s}}, \qquad \boldsymbol{s} = \begin{pmatrix} x^1(g-y^1_{demo}) \\ x^2(g-y^1_{demo}) \\ \vdots \\ x^N(g-y^1_{demo}) \end{pmatrix}$$

$$\boldsymbol{\Gamma_i} = \mathrm{diag}(\Psi_i^1, \Psi_i^2 \cdots \Psi_i^N), \quad \boldsymbol{f}_{target} = \begin{pmatrix} f^1_{target} \\ f^2_{target} \\ \vdots \\ f^N_{target} \end{pmatrix} \tag{7}$$

$$f_{target} = \tau^2\ddot{y}_{demo} - a_z(b_z(g-y_{demo}) - \tau\dot{y}_{demo})$$

Here, N is the number of samples in the demonstrated trajectory.

2.2 Related work on DMP perturbation recovery

We here consider the case where a disturbance is introduced, such that the actual trajectory, denoted y_a, evolves differently from y, where y evolves according to Eqs. (1)-(6). Without any coupling terms, the time evolution of Eqs. (2)-(5) would be unaffected by a perturbation. This behavior is undesired, since it is then likely that the actual trajectory y_a deviates significantly from the intended trajectory even after the cause of the perturbation has vanished. This is more thoroughly described in (Ijspeert et al., 2002) and (Ijspeert et al., 2013). To mitigate this problem, the solution described below has been suggested in (Ijspeert et al., 2013).

The following coupling terms were introduced,

$$\dot{e} = \alpha_e(y_a - y_c - e) \tag{8}$$

$$C_t = k_t e \tag{9}$$

$$\tau_a = 1 + k_c e^2 \tag{10}$$

Here, α_e, k_t and k_c are constant parameters. The parameter τ_a was used to determine the evolution rate of the entire dynamical system. Further, the term C_t was added to Eq. (4) so that the coupled version of y, denoted y_c, fulfilled the following.

$$\tau_a\dot{z} = \alpha_z(\beta_z(g-y_c) - z) + f(x) + C_t \tag{11}$$

$$\tau_a\dot{y_c} = z \tag{12}$$

A PD controller, given by

$$\ddot{y}_r = K_p(y_c - y_a) + K_v(\dot{y_c} - \dot{y_a}) \tag{13}$$

was used to drive y_a to y. Here, $\ddot{y}_r$ 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_u 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_u, 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 $\ddot{y}_r = 100\,\mathrm{m/s^2}$. In Figs. 2 and 3, two example scenarios are 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 $\ddot{y}_r$ 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.

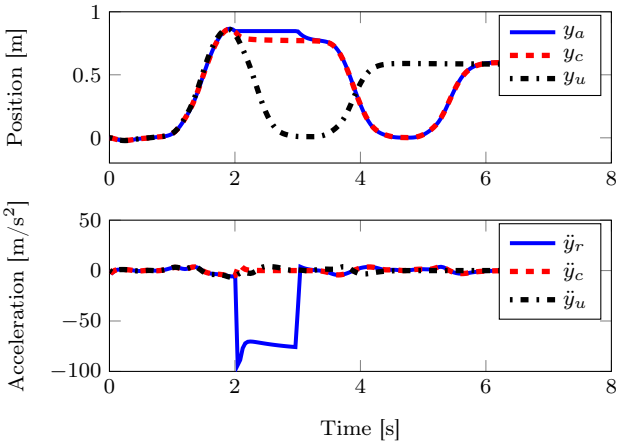


Fig. 2. Simulated trajectories, where y_a was subjected to a stopping perturbation from 2 s to 3 s, using the approach in (Ijspeert et al., 2013). When y_a was stopped, the evolution of y_c slowed down, and when y_a was released, it was driven to y_c and then behaved like a delayed version of y_u. This behavior was desired. However, a prohibitively large accelereration $\ddot{y}_r$ was generated.

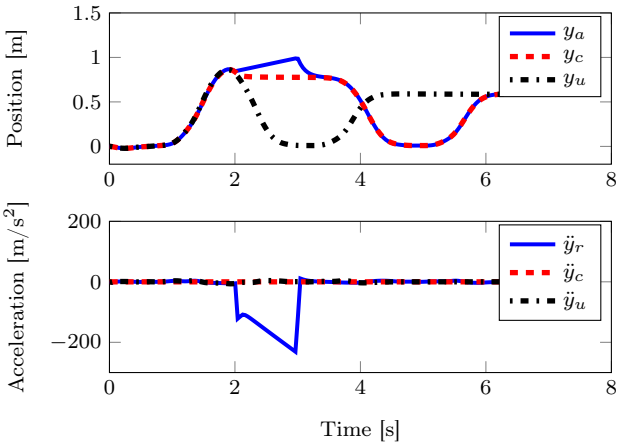


Fig. 3. Similar to Fig. 2, except that y_a was moved away from the nominal path between 2 s to 3 s. Again, a prohibitively large accelereration $\ddot{y}_r$ was generated.

3. PROBLEM FORMULATION

In this paper, we address the question of whether perturbations of DMPs could be recovered from, while fulfilling the following requirements. Only moderate control signals must be used. The benefits of the DMP framework described in (Ijspeert et al., 2013), *i.e.*, scalability in time and space as well as guaranteed convergence to the goal g, must be preserved. Further, in the absence of significant perturbations, the behavior of y_a should resemble that of the original DMP framework described in Sec. 2.1.

4. METHOD

Our proposed method extends that in (Ijspeert et al., 2013) as follows. The PD controller in Eq. (13) was augmented with feedforward control, as shown in Eq. (14). Further, the PD controller gains were moderate, to get a practically realizable control signal. Additionally, the time constant τ was introduced as a factor in the expression for the adaptive time parameter τ_a, see Eqs. (10) and (15). Our method is detailed below.

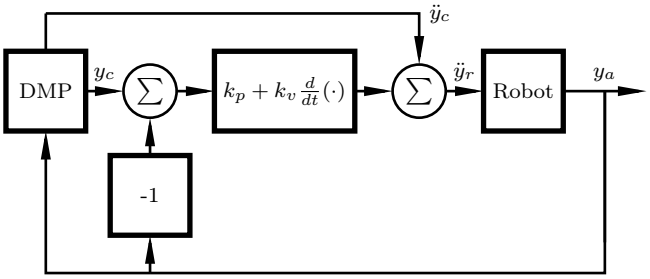


Fig. 4. Schematic overview of the control structure described in Sec. 4. The block denoted 'Robot' includes the internal controller of the robot.

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

$$\ddot{y}_r = k_p(y_c - y_a) + k_v(\dot{y}_c - \dot{y}_a) + \ddot{y}_c, \tag{14}$$

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 τ, as follows.

$$\tau_a = \tau(1 + k_c e^2) \tag{15}$$

The coupling term C_t was omitted in this present method. This is elaborated on in Sec. 9.

Since τ_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.

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

$$= \frac{\dot{z}\tau_a - 2\tau k_c z e\dot{e}}{\tau_a^2}, \tag{17}$$

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 $\dot{y}_a$ was complemented with a low-pass filter, to mitigate amplification of measurement noise.

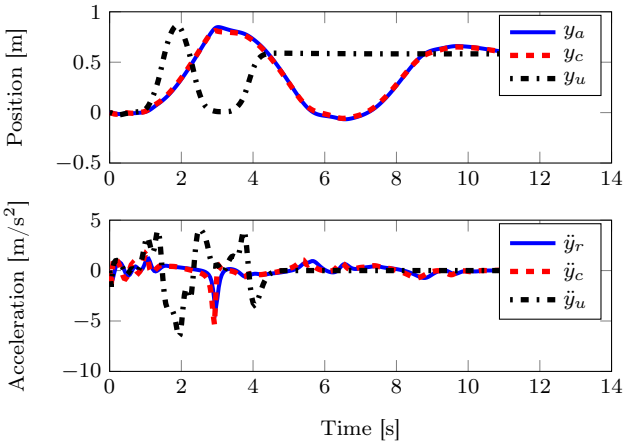


Fig. 5. Simulation with the control structure in (Ijspeert et al., 2013), except that the gains were lower ($K_p = 25$ and $K_v = 10$). The reference acceleration was of reasonable magnitude, but the coupled and real systems were slowed down due to small tracking errors combined with temporal coupling.

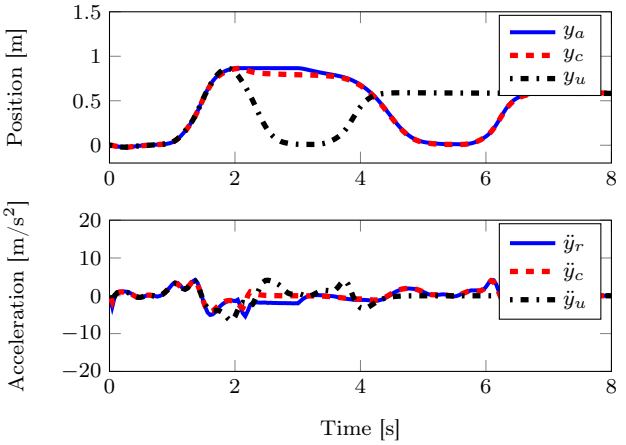


Fig. 6. Similar to Fig. 2, but with modeled noise and delay, and using the controller presented in this paper. The behavior was satisfactory both regarding position and acceleration.

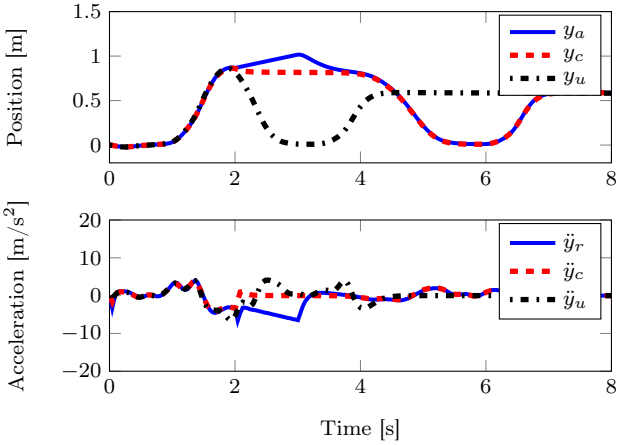


Fig. 7. Similar to Fig. 6, except that y_a was moved away from the nominal path between 2 s to 3 s. Again, the behavior was satisfactory both regarding position and acceleration.

5. SIMULATIONS

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 τ_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.

Fig. 5 shows the result from a simulation where the controller detailed in (Ijspeert et al., 2013) was used, except that the gains were lowered to moderate values. The conditions were ideal, and no perturbation was present. This resulted in reasonable control signals. However, small control errors in combination with the temporal coupling slowed down the evolution of the coupled system as well as the actual movement.

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 (Blomdell 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-

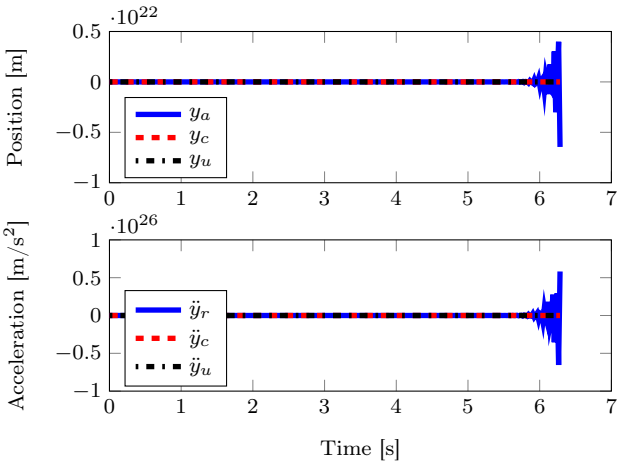


Fig. 8. Using the control system in (Ijspeert et al., 2013), subject to the simulated noise and time delay, resulted in unstable behavior.

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

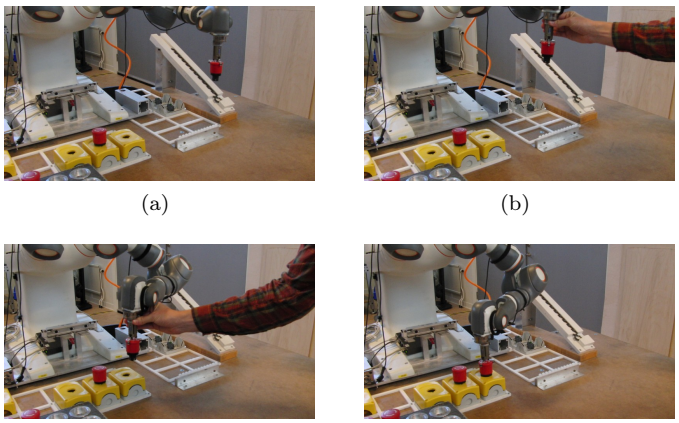
trl. Similar to the simulations, the control system ran at 250 Hz, and the delay between process and controller was 3 sample periods, corresponding to 12 ms.

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,



(a) (b) (c) (d)

Fig. 10. First scenario. In (a), the robot started to execute a DMP for placing the stop button in the rightmost yellow case. A human perturbed the motion twice. The first perturbation (b) was formed by moving the end-effector away from its path, and then releasing it. The second perturbation (c) lasted for a longer time, and consisted of unstructured movement. The robot recovered from both perturbations, and managed to place the stop button in the case (d). Data from one trial are shown in Fig. 12.

and then releasing it. The second perturbation consisted of a longer, unstructured, movement later along the trajectory.

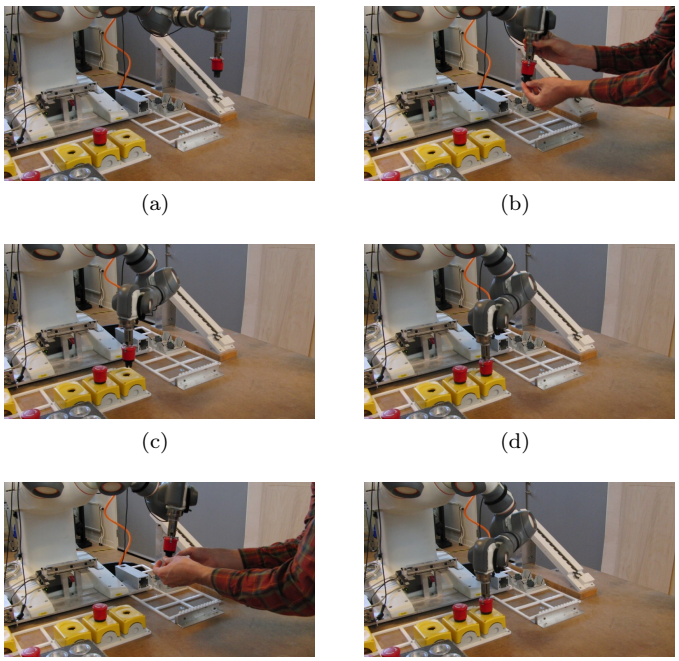
In the second scenario, a human co-worker realized that the stop buttons in the current batch were missing rubber gaskets, and acted to modify the robot trajectory, allowing the co-worker to attach the gasket on the stop button manually. During execution of the DMP, the end-effector was stopped and lifted 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.



(a) (b) (c) (d) (e) (f)

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 τ_a was also modified, to include the nominal time constant τ as a factor. These changes resulted in the following benefits, compared to the previous method. The feedforward part allowed the controller to act also for insignificant position- and velocity error, 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 τ_a was not the main focus of this paper, but it was necessary since it allowed the actual trajectory to converge

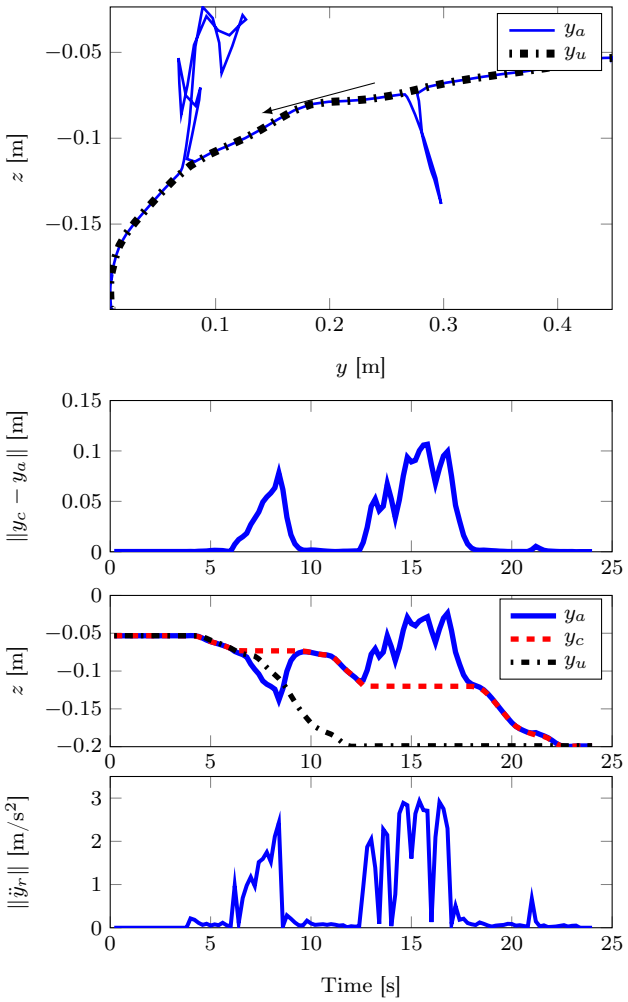


Fig. 12. Experimental data from a trial of the first scenario. The first (from above) plot shows the path of the end-effector in the Cartesian base frame of the robot, projected on the yz-plane. The arrow indicates the movement direction, which started in the upper right and finished in the lower left 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_u. The reference acceleration was of reasonable magnitude, as shown in the fourth plot.

to the trajectory defined by the DMP, with time constant τ. Without this modification, the time parameter τ would not have affected the trajectory generated by the DMP. Instead, τ_a would have converged to 1, regardless of τ, 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-

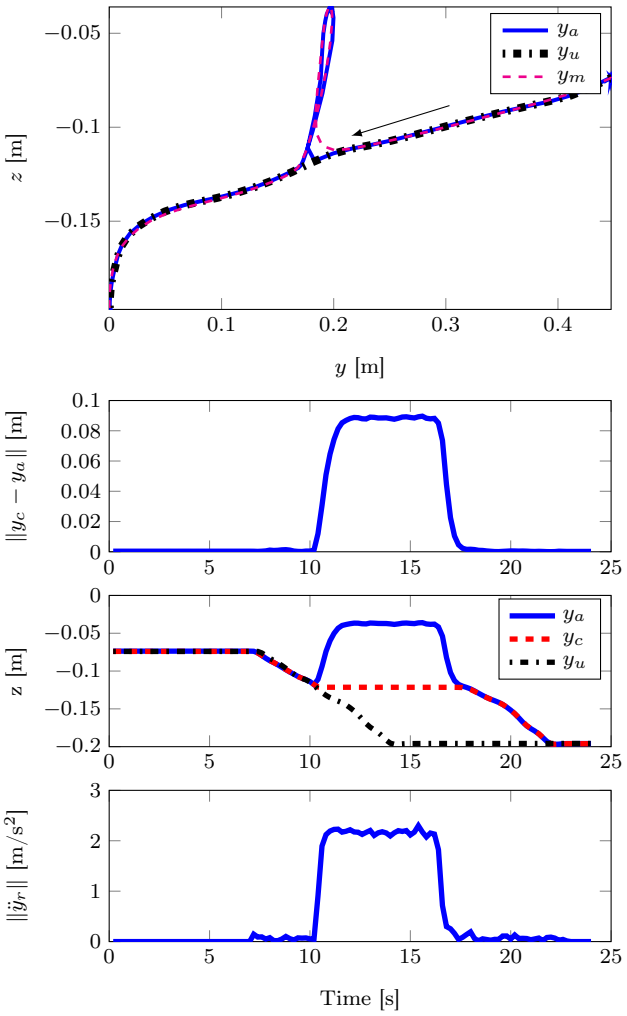


Fig. 13. Experimental data from a trial of the second scenario. The organization of this figure is similar to that of Fig. 12. The perturbation for stopping and lifting the end-effector took place from time 10 s to 17.5 s, and is clearly visible in each plot. This perturbation was recovered from as intended, and the reference acceleration was of reasonable magnitude. The uppermost plot also displays the measured trajectory obtained by executing the modified DMP, denoted y_m. It behaved like a smooth version of the perturbed original trajectory y_a.

tioned motion control signals to explore the surroundings with a force/torque sensor, a detour to allow line-of-sight between a camera and a part of the work-space, or any other unforeseen deviation from the reference trajectory defined by a DMP.

It is necessary to implement saturation on the control signals, in order to prevent too large acceleration and velocity for large perturbations. Such boundaries were implemented, but never reached in the experiments in this work.

The coupling term C_t has been introduced in previous research to drive y_c towards y_a when these were different, see Sec. 2.2. However, whether this effect would be desired, and to what extent, would be context dependent. Further, the effect would be mitigated by the temporal coupling, that would slow down the evolution of y_c 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 straight forward 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_c did not approach y_a 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 Remazeilles at Tecnalia, Donostia, and Diogo Almeida at KTH, Stockholm, are gratefully acknowledged for pointing out some of the previous research.

REFERENCES

ABB Robotics (2016a). ABB IRC5. URL `http://new.abb.com/products/robotics/controllers/irc5`. Accessed: 2016-06-13.

ABB Robotics (2016b). ABB YuMi. URL `http://new.abb.com/products/robotics/yumi`. Accessed: 2016-06-09.

Abu-Dakka, F.J., Nemec, B., Jørgensen, J.A., Savarimuthu, T.R., Krüger, N., and Ude, A. (2015). Adaptation of manipulation skills in physical contact with the environment to reference force profiles. *Autonomous Robots*, 39(2), 199–217.

Åström, K.J. and Wittenmark, B. (2013). *Computer-controlled systems: theory and design*. Courier Corporation, Mineola, New York.

Atkeson, C.G., Moore, A.W., and Schaal, S. (1997). Locally weighted learning for control. In *Lazy Learning*, 75–113. Springer, Dordrecht, Netherlands.

Bagge Carlson, F. (2016). DynamicMovementPrimitives.jl. URL `https://github.com/baggepinnen/DynamicMovementPrimitives.jl`. Accessed: 2017-03-21.

Blomdell, A., Bolmsjö, G., Brogårdh, T., Cederberg, P., Isaksson, M., Johansson, R., Haage, M., Nilsson, K., Olsson, M., Olsson, T., Robertsson, A., and Wang, J. (2005). Extending an industrial robot controller-implementation and applications of a fast open sensor interface. *IEEE Robotics & Automation Magazine*, 12(3), 85–94.

Blomdell, A., Dressler, I., Nilsson, K., and Robertsson, A. (2010). Flexible application development and high-performance motion control based on external sensing and reconfiguration of abb industrial robot controllers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 62–66. May 3-8, Anchorage, Alaska.

Ijspeert, A., Nakanishi, J., and Schaal, S. (2003). Learning control policies for movement imitation and movement recognition. In *Neural Information Processing System (NIPS)*, volume 15, 1547–1554. December 5-10, Stateline, Nevada.

Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25(2), 328–373.

Ijspeert, A.J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, 1398–1403. May 11-15, Washington D.C., USA.

Karlsson, M. (2016). Experimental evaluation of dmp perturbation recovery, Youtube. URL `https://www.youtube.com/watch?v=u8GwsSsL0TI`. Accessed: 2016-10-04.

Karlsson, M. (2017). DMP perturbation, simulation example. URL `https://gitlab.control.lth.se/cont-mkr/dmp_perturbation_sim_example`. Accessed: 2017-03-21.

Karlsson, M., Robertsson, A., and Johansson, R. (2017). Autonomous interpretation of demonstrations for modification of dynamical movement primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*. May 29 - June 3, Singapore.

LabComm (2017). Research tools and software. URL `http://www.control.lth.se/Research/tools.html`. Accessed: 2017-02-13.

Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., and Barto, A.G. (2015). Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2), 131–157.

Park, D.H., Hoffmann, H., Pastor, P., and Schaal, S. (2008). Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoids 2008—8th IEEE-RAS International Conference on Humanoid Robots*, 91–98. December 1-3, Daejeon, Korea.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 763–768. May 12-17, Kobe, Japan.

Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., and Schaal, S. (2013). From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems*, 61(4), 351–361.

Prada, M., Remazeilles, A., Koene, A., and Endo, S. (2014). Implementation and experimental validation of dynamic movement primitives for object handover. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2146–2153. September 14-18, Chicago, Illinois.

Sanderson, C. and Curtin, R. (2016). Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1, 26.

Schaal, S. and Atkeson, C.G. (1998). Constructive incremental learning from only local information. *Neural Computation*, 10(8), 2047–2084.

Schaal, S., Kotosaka, S., and Sternad, D. (2000). Nonlinear dynamical systems as movement primitives. In *IEEE International Conference on Humanoid Robotics*, 1–11. September 7-8, Boston, MA.

Stolt, A., Carlson, F.B., Ardakani, M.G., Lundberg, I., Robertsson, A., and Johansson, R. (2015). Sensorless friction-compensated passive lead-through programming for industrial robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3530–3537. September 28 - October 2, Hamburg, Germany.