Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Interaction learning for dynamic movement primitives used in cooperative robotic tasks

Tomas Kulvicius [a,*], Martin Biehl [c], Mohamad Javad Aein [a], Minija Tamosiunaite [a,b], Florentin Wörgötter [a]

[a] *Georg-August-Universität Göttingen, Bernstein Center for Computational Neuroscience, Department for Computational Neuroscience, III Physikalisches Institut - Biophysik, Friedrich-Hund Platz 1, D-37077 Göttingen, Germany*

[b] *Department of Informatics, Vytautas Magnus University, Vileikos g. 8, Kaunas, Lithuania*

[c] *Adaptive Systems Research Group, The University of Hertfordshire, School of Computer Science College Lane, Hatfield, Hertfordshire AL10 9AB, United Kingdom*

HIGHLIGHTS

- We present a tightly-coupled robotics systems based on Dynamic Movement Primitives.
- We provide an analytical stability analysis for an equilibration of coupled system.
- We introduce sensory feedback with a predictive learning for the agent interaction.
- We show that such a mechanism allows us to learn an adaptive, sensor-driven interaction.
- We demonstrate that agents learn to cooperate when adding adaptive sensor control.

ARTICLE INFO

Article history:
Received 4 April 2013
Received in revised form
7 July 2013
Accepted 15 July 2013
Available online xxxx

Keywords:
Interactive agents
Obstacle avoidance
Hebbian learning

ABSTRACT

Since several years dynamic movement primitives (DMPs) are more and more getting into the center of interest for flexible movement control in robotics. In this study we introduce sensory feedback together with a predictive learning mechanism which allows tightly coupled dual-agent systems to learn an adaptive, sensor-driven interaction based on DMPs. The coupled conventional (no-sensors, no learning) DMP-system automatically equilibrates and can still be solved analytically allowing us to derive conditions for stability. When adding adaptive sensor control we can show that both agents learn to cooperate. Simulations as well as real-robot experiments are shown. Interestingly, all these mechanisms are entirely based on low level interactions without any planning or cognitive component.

1. Introduction

Novel trajectory generation methods such as Dynamic Movement Primitives (DMPs [1,2]) or Gaussian Mixture Models (GMMs [3–5]) can generalize over different start and end points of the movement trajectory and they can efficiently emulate different trajectory shapes also allowing us to combine them in a dynamic way [6,7]. Such methods also allow an on-line alteration of the trajectory, if need be. For example, it is clearly useful to alter the trajectory of an agent as soon as an obstacle (a path disturbance) is sensed. Such problems have been addressed by using sensory feedback and applied in a variety of different applications, like obstacle avoidance [8–15], grasping and object manipulation [16,17], locomotion and crawling [18,19], drumming [20], Ball-in-a-Cup game [21].

So far DMPs and GMMs have mainly been used for uncoupled agent systems. In this study, we analyze tightly coupled dual agent systems where each agent has its own path plan defined by a DMP. Note that in simulations we couple agents by a stiff virtual spring whereas in real robot scenario agents are coupled by a rigid object, i.e., a trace. In a coupled system the problem exists that both agents might not cooperate. This leads to the situations that agents will first have to equilibrate with respect to each other. Only on top of this any sensor influence – for example for obstacle avoidance – and/or learning can take place. As shown here analytically both agents will indeed equilibrate into a shared fixed point representing the two new trajectories. This leads to the situation that sensor reactions and learning can operate in a stable way also in the dual agent system. Specifically, we will show that learning can be employed to create a system, where both agents in the end "help each other". Probably one interesting aspect of this approach is that, due to the intrinsic attractor properties of DMPs, these systems do not need any conventional active control-components (impedance control, servoing, etc.), while still performing remarkably well.

* Corresponding author. Tel.: +49 551 39 107 63.
E-mail address: tomas@physik3.gwdg.de (T. Kulvicius).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.07.009

In the following we will describe our framework for interactive DMPs also introducing the learning method. Mathematical derivations are given and a detailed analysis of signals and learning statistics is performed using simulations. In the end we then show experiments with a real robot-arm. Finally, in the discussion section, we will relate and compare our method to other existing approaches.

2. Methods

2.1. Dynamic Movement Primitives (DMPs)

To describe the movement trajectory of an agent we use the method for generating movement sequences proposed in [7] which is a modification of the original dynamic movement primitives (DMPs, [1,22,23,2]). Here we use modified DMPs since they have faster convergence at the end-point compared to the original DMP formulation and allow smooth joining of movement sequences with non-zero velocities at the joining point [7]. Similar to the original approach, modified DMPs are based on differential equations and consist of two dynamic systems: the transformation system and the canonical system. The transformation system is described as follows:

$$\dot{z} = \alpha(\beta(r - y) - z) + f, \tag{1}$$

$$\dot{y} = z, \tag{2}$$

$$\dot{r} = \begin{cases} (g - s)/T, & \text{if } t \le T \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where α and β are time constants (in this study we used $\alpha = 0.75$, $\beta = \alpha/4$), $\dot{z}$, $\dot{y}$ and y correspond to acceleration, velocity and position, respectively. Here r defines a piecewise-linear goal function where s and g are the known start and goal states (start/end-point) and T is the duration of the movement. Initially we set $y_0 = r_0 = s$, $\dot{y}_0 = 0$ and $\dot{z}_0 = 0$.

The canonical system is described by a sigmoidal decay function:

$$\dot{\xi} = -\frac{\alpha_\xi \exp(\alpha_\xi(T - t))}{(1 + \exp(\alpha_\xi(T - t)))^2}, \tag{4}$$

where α_ξ is a time constant and defines the steepness of the sigmoidal function (in this study we used $\alpha_\xi = 1.0$) centered at time moment T. Initially we set $\xi_0 = 1$. The nonlinear function f is given by

$$f = \alpha_w \frac{\sum_{i=1}^{n} \psi_i \omega_i \xi}{\sum_i \psi_i}, \tag{5}$$

with

$$\psi_i = \exp\left(\frac{-(\frac{t}{T} - c_i)^2}{2\sigma_i^2}\right), \tag{6}$$

where ψ_i denote Gaussian kernels, c_i and σ_i is the center and width of the ith kernel, respectively. Kernels are placed evenly along the trajectory in time and spaced between 0 and 1, where 0 denotes the beginning of the movement trajectory and 1 the end. The shape of the movement trajectory is defined by weights ω_i and in our study they were generated manually but in general case they can be obtained by imitation learning [1,2]. Here we use α_w as a general scaling factor for all learned weights and in this study we set it to 1. Note that here DMPs are time dependent. This can be changed to phase-based DMPs without problems as shown in one of our older studies [7]. However, this is not relevant for the current investigations. In simulations and a real robot experiment we used a sampling rate of 200 Hz.

2.2. Interactive DMPs

We model the two agent system as two point particles coupled by a spring. Here we treat agents with equal mass. Each agent is subject to a primary force generated by a dynamic movement primitive, which can be viewed as the control signal. We denote the ith coordinate ($i = 1, 2, 3$ correspond to X, Y and Z-coordinates, respectively) of the jth particle ($j = 1, 2$ correspond to agent P and Q, respectively) as $y_{i,j}$ and the corresponding velocities as $z_{i,j}$. Assuming that the particles have mass m Newton's equation of motion is

$$m\dot{z}_{i,j} = F^S_{i,j} + F^D_{i,j}, \tag{7}$$

where $F^S_{i,j}$ are the forces acting due to the spring coupling and $F^D_{i,j}$ the forces from the DMP. The spring forces can be written as

$$F^S_{i,1} = k\, direction\, (d - offset) = -F^S_{i,2}, \tag{8}$$

where $direction = (y_{i,1} - y_{i,2})/offset$ and $offset = \sqrt{\sum_l (y_{l,1} - y_{l,2})^2}$. Here d denotes the spring length when relaxed and k is the spring constant. In this study we used $d = 50$ cm and $k = 0.95$ N/cm. Note that in our simulations we omitted masses by setting them to $m = 1$ for both robots. In general, robots would adapt to any masses due to learning. Also, in simulations we chose k in order to obtain a relatively hard stiffness for coupling, whereas in the real robot scenario (as shown later) there is no need to tune k.

As explained above, the position of the agent is defined by a DMP and we denote its force $F^D_{i,j}$ by

$$F^D_{i,j} = m(\alpha\left(\beta\left(r_{i,j} - y_{i,j}\right) - z_{i,j}\right) + f_{i,j} + u^A + u^I). \tag{9}$$

Here u^A is a reactive term which is used for obstacle avoidance and in this study is fixed, whereas u^I is an interactive term and is learnt. Definitions of these terms are given below.

Also we have for the accelerations and velocities:

$$\dot{z}_{i,j} = \frac{1}{m}(F^S_{i,j} + F^D_{i,j}), \tag{10}$$

$$\dot{y}_{i,j} = z_{i,j}. \tag{11}$$

3. Implementation

3.1. Task definition

In the interaction learning scenario, as explained above, we have two identical agents (P and Q) which are physically coupled via the linear spring. Initially the agents are going to follow their planned path, so in case the agents have different paths or the path gets changed due to obstacle avoidance forces between agents will increase due to the coupling. The goal is to learn to interact in a way that the forces between agents are minimized. For example, if agent P is going to avoid the obstacle, then agent Q has to learn interacting and helping agent P by moving to the same direction as shown in Fig. 2B.

3.2. Definition of the sensor inputs

We consider two types of sensor inputs. (1) Two avoidance sensors (touch and vision, with fixed characteristics used for obstacle avoidance) and (2) two interaction-relevant sensors (displacement and force used-for interaction learning).

Obstacle avoidance is implemented in the conventional way (potential field approach [8,9,12]) and is used to create realistic situations for interaction learning. We use two types of sensors which generate a compound avoidance signal: touch and vision. Normally the agent should be able to use the gradually rising vision signal alone to avoid the obstacle. The binary touch signal is used as an emergence fall-back. For example, if the robot–robot interaction massively pushes one robot into an obstacle, then this situation can be recovered by the reaction to touch. Specifically we define:

(1a) The touch sensor obtains value 1 if the agents hits the obstacle and 0 if there is no collision:

$$A_T(t) = \begin{cases} 1, & \text{if } \delta(t) < \Theta_T \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

where δ is the minimal distance from the agent's center point to the obstacle, and Θ_T is the threshold for collision detection.

(1b) The visual sensor is triggered whenever the obstacle appears within the vision field (defined by a threshold) of the agent and is described by

$$A_V(t) = \begin{cases} \delta(t)/\Theta_V, & \text{if } \delta(t) < \Theta_V \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where Θ_V defines the radius of the visual field. In this study we used $\Theta_T = 5$ cm and $\Theta_V = 20$ cm.

In general, we use a filter to smooth sensor signals and obtain as final input:

$$u^A(t) = au^A(t - \Delta t) + (1 - a)(\Gamma_T A_T(t) + \Gamma_V A_V(t)), \tag{14}$$

with $a = 0.98$ the filter parameter. Here $\Gamma_{T,V}$ are weights which define the strength and direction of the obstacle avoidance reaction where positive weights were used to generate leftward/upward movements and negative weights for rightward/downward movements. Values for $\Gamma_{T,V}$ can change for different experiment and are, thus, given below. In this study we used $\Delta t = 0.005$ s.

(2a) The displacement sensor D is defined by

$$D(t) = \begin{cases} \eta(t)\,(y_a(t) - y_p(t)), & \text{if } |y_a(t) - y_p(t)| < \Theta_D \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

where $\eta(t) = 1$, if $u^A(t) < \epsilon$ and $\eta(t) = 0$, otherwise. In this study we used $\epsilon = 10^{-4}$. Thus, η acts as an inhibition term where an agent, that encounters an obstacle will not react to any push or pull produced by the other agent, as the need to avoid the obstacle is fundamental. This way the agents obtain their roles (leader and follower) naturally depending on the situation and there is no need to define them in advance. Here y_a is the actual trajectory of the agent defined by Eq. (11) and y_p is the planned trajectory obtained without spring force (i.e., in Eq. (10) we set $F^S = 0$). Note that here we use the threshold Θ_D for the displacement sensor in order to compensate for tracking errors, since in the real robot applications the actual and the planned paths will never match exactly. In this study we used $\Theta_D = 1$ cm. Also, the displacement signal does not influence the trajectories of the robots. It is only used for learning.

(2b) The force sensor signal F is defined by the force F^S of the spring model (see Eq. (8)):

$$F(t) = \eta(t)F^S(t). \tag{16}$$

This signal is filtered with

$$u^I(t) = au^I(t - \Delta t) + (1 - a)\rho(t)F(t), \tag{17}$$

with $a = 0.98$ and ρ a weight, which will be changed by the learning rule described next. Note that in real robot experiment (as shown later) F^S is obtained from a force sensor of the robot. Also, please keep in mind that the agents are independent and have their respective sensors.

3.3. Learning rule

For learning we make use of the physical fact that the position signal follows the acceleration-dependent signal. Hence force (acceleration-dependent) is predictive for a displacement (position-dependent) that will arise later. We can use the displacement signal to learn a predictive reaction in response to the (earlier occurring) force signal. Thus, for learning the sensor signal D (displacement) is paired with the sensor signal F (force) to grow weight ρ [24].

For this, we use a correlation based learning rule (Hebbian type [25]).

$$\dot{\rho} = \mu DF, \tag{18}$$

where μ is the learning rate. Learning stops as soon as $D = 0$, i.e., as soon as the displacement sensor is not triggered anymore and the predictive response has fully taken over.

4. Results

This section will start with an analytical stability analysis of the coupled agent system. This will be followed by simulation results, first for 1D cases and then for a 3D case. Finally we will show a real robot experiment.

4.1. Stability analysis

Such coupled DMP-based systems are quite interesting as coupling leads to mutual influence of one DMP onto the other. The question arises, thus, under which conditions these systems are stable and how they converge along the trajectory. In Appendix A we provide analytical details addressing this question. They are based on the, quite conventional, approximation of each agent as a point-source (end point of the robot's end-effector) and a mutual compliance that is modeled by a spring. Robot experiments shown here confirm that these approximations are justified for systems that have limited intrinsic dynamics (kinematically stiff conventional robots). As solutions are complex, here we will just summarize the result. We discuss the only case that $m_1 = m_2 = m$. Generalization to different masses are straightforward as only their relation m_1/m_2 is relevant.

Conventionally (uncoupled) DMPs are used with parameters α and $\beta = \alpha/4$ as this leads to critical damping along the DMP-trajectory and, thus, to optimally fast convergence [1]. Fixed point analysis (see Appendix A) of the coupled system reveals that for these parameters the system will oscillate along the direction of the spring. One has to choose $\alpha > \sqrt{8k}/\sqrt{m}$ and $\beta_c = (m\alpha^2 - 8k)/(4m\alpha)$ to achieve critical damping along the spring, but will receive now a slightly suboptimal (over-damped) behavior along the DMP trajectories. From analysis we also found that depending on the parameters there exist either one stable fixed point or two fixed points, one stable and the other unstable. The stable one represents the desired solution (agents approach goal-points). This analysis is valid for all start and goal positions and without any constraints on the system.

4.2. Interaction learning without obstacle avoidance

First of all we will present interaction learning between two agents without obstacle avoidance when the agents initially have different paths (paths were generated by setting DMP weights manually). In this case we are going to learn the interaction only for the Y-coordinate (1D case). Simulation results of such an experiment are shown in Fig. 1. Signal development during the learning process is given in panels A1–A3 (we show signals only for the agent Q, because signals of agent P are identical with an inverted sign). The first few seconds show the control case (before learning, 0–5 s). The first five learning trials follow (5–30 s). Note that here one trial lasts 5 s, which given a sampling rate of 200 Hz corresponds to 10^3 time steps). One can see that already during the first learning trial (5–10 s) forces are significantly reduced compared

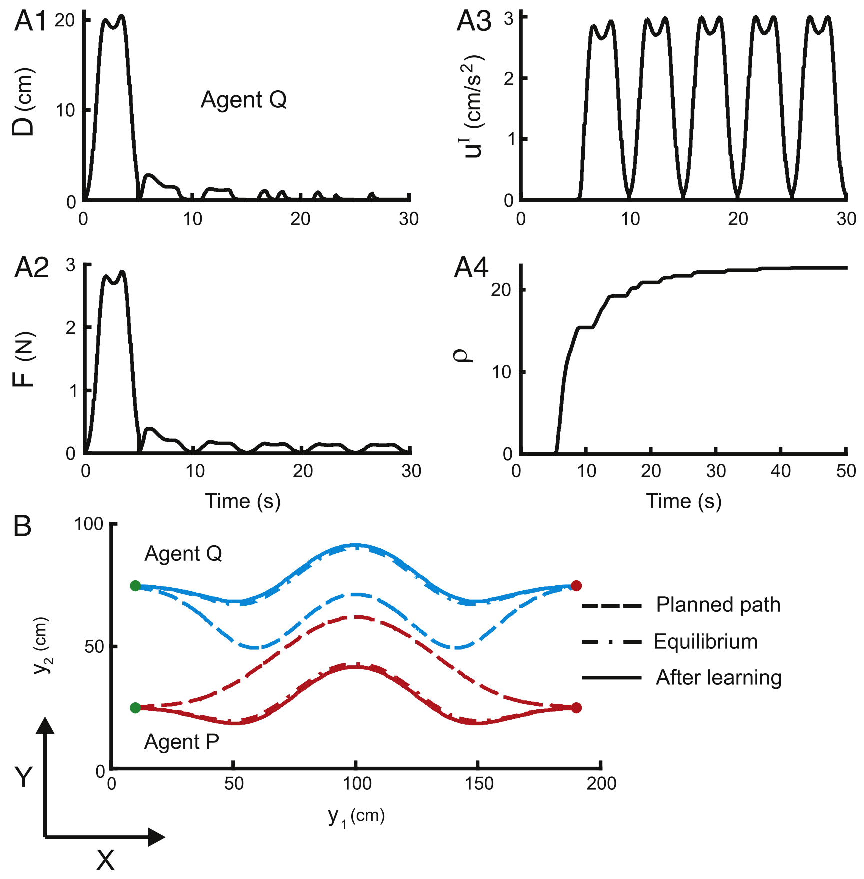

Fig. 1. Simulation results from interaction learning without obstacles. In panels **A1–A4** signals only for the agent Q are shown. **(A1)** Displacement signal D; **(A2)** predictive force signal F; **(A3)** output signal u^l; **(A4)** weight ρ; **(B)** trajectories. Green and red dots correspond to the start and end points, respectively. The learning rate was $\mu = 0.04$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

to the control trial (0–5 s). The displacement signal (panel A1) was fully avoided and the weights (panel A4) stabilized after eight learning trials. Resulting trajectories are presented in panel B. We observe that the agents converge to the "natural" equilibrium, which is the one that would also be obtained by purely passive equilibration, i.e., terms u^A and u^l are equal to zero all the time. The main gain from employing learning, however, is that this way the forces between both agents will be minimized, which is not the case for passive equilibration (in real robot scenarios, if the forces are high then the robots might lose or damage the object by which they are coupled).

4.3. Interaction learning with obstacle avoidance

In the next experiment we have a scenario where the agents initially have the same planned trajectories but they will encounter obstacles along their paths. So in this case agents have to learn to help each other to avoid obstacles by moving to the same direction. As in the previous experiment we are going to learn interaction only for the Y-coordinate.

Results for interaction learning with obstacle avoidance are presented in Fig. 2. As in the previous case signals for the control case (before learning, 0–5 s) and the first three learning trials (5–20 s) are shown in panels A1–A3. Here we can see that before learning agents will be pushed into the obstacles (see touch signals A_T, inset of panel A1) whereas – as learning proceeds – agents are learning to help each other and the obstacles are not touched anymore. Weight development is shown in panel A4. In this case it took one trial longer for agent Q than for agent P (17 and 16 trials, respectively) until weights finally stabilized since agent Q was pushed stronger away by agent P due to its obstacle avoidance reaction. Resulting trajectories are shown in panel B where we can see that the roles of agents interchange depending on their sensor inputs, i.e., the agent which is going to avoid the obstacle becomes the leader and the other agent is the follower (which learns to follow the leader). For comparison we also show the case when the roles of the agents were predefined at the beginning (P = master and Q = slave). Resulting trajectories are presented in panel C where we can see that in this case only agent Q was learning to help the other agent. As a consequence agent Q was never able to avoid the obstacle (see also touch signal in the inset). These results clearly demonstrate that presented non-rigid Leader/Follower mechanism is advantageous compared to a fixed Master/Slave architecture.

4.4. Statistical evaluation

We evaluated our model statistically in a 3D scenario with obstacles (see Fig. 3C1) where the agents initially had different paths (most general case). In this case we were learning weights for the interaction for all three dimensions (X, Y and Z-coordinates). We analyzed the robustness of the learning and the influence of the learning rate μ by the number of learning trials (experiences) needed to learn interacting. Signals and resulting trajectories (see the supplementary video, Appendix B) from a single simulated experiment are shown in Fig. 3A–C, where in panels A1–A3 we show signals only for agent P (signals for agent Q look similar). Again, we show the control case (before learning, 5–10 s) and the first four learning trials. As in the previous examples, we see that the amplitude of the displacement signals for all three dimensions is

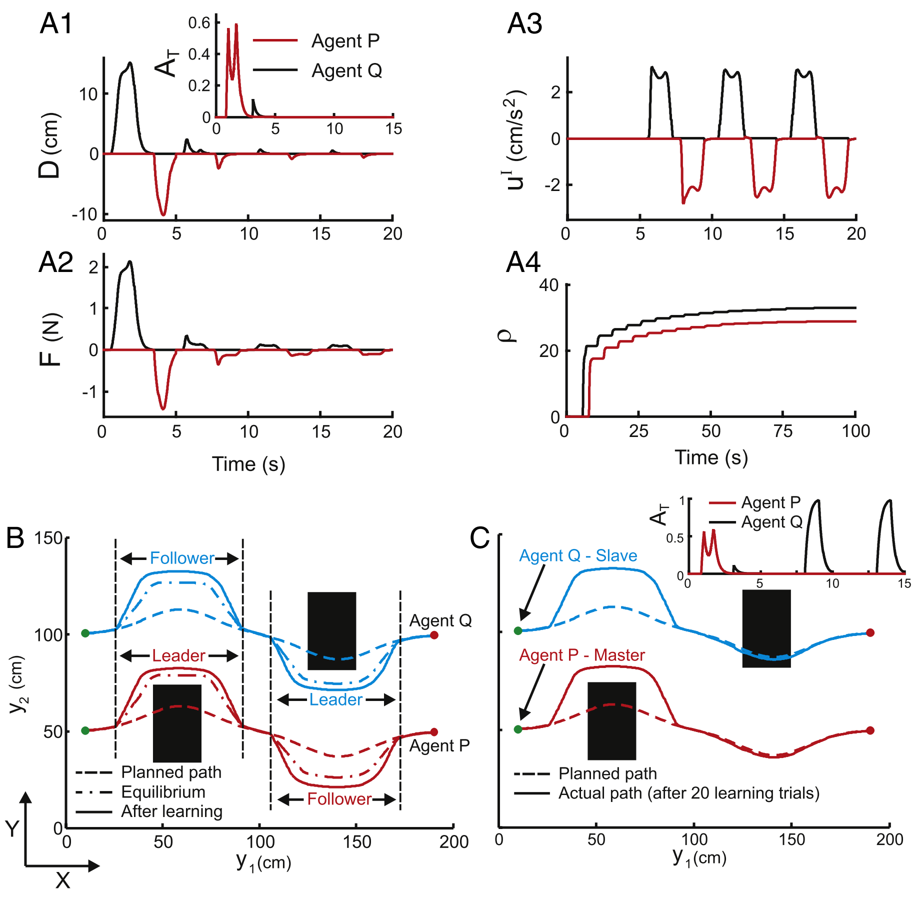

Fig. 2. **(A, B)** Simulation results from interaction learning with obstacles without predefined roles of the agents and **C** with initially predefined roles (P = master and Q = slave). Note that in simulations we let the robots go through obstacles, whereas in the real robot experiment this would not be the case and the behavior would much depend on the obstacle avoidance module, e.g., if the reflex is strong enough, then the robot would avoid an obstacle without toggling it. **(A1)** Displacement signals D; **(A2)** predictive force signals F; **(A3)** output signals u^I; **(A4)** weights ρ; **(B, C)** trajectories. The learning rate was $\mu = 0.4$. Weights for obstacle avoidance were tuned experimentally and were as follows: $\Gamma^P_{T,y_2} = 2$, $\Gamma^P_{V,y_2} = 5$, $\Gamma^Q_{T,y_2} = -2$, $\Gamma^Q_{V,y_2} = -5$ and $\Gamma^{P/Q}_{T/V,y_1} = 0$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

decreasing as learning proceeds. In this case, for agent P it took longer compared to agent Q (17 and 9 trials, respectively) until weights finally stabilized due to differences in trajectories and configuration of the obstacles.

To gather statistics we changed start and end positions of the coupled agents in every learning trial. We considered learning as finished and successful if (1) the displacement signal was not triggered and (2) the weights were not changing anymore within five consecutive trials. We found that weights converged and learning was successful in all 100 experiments. The influence of the learning rate is shown in Fig. 3D where we can observe, as expected, that higher learning rates lead to fewer learning experiences needed to learn interacting. Results also demonstrate that depending on the differences in trajectories and configuration of the obstacles the number of learning experiences between agents can vary. Note that in general too high learning rate will potentially lead to the one-shot learning where the learnt reaction might be not optimal for that particular case, i.e., overlearning.

4.5. Robot experiments

Finally, we performed two robot experiments (similar to the simulations) with KUKA light-weight arms [26]. In the first experiment we demonstrate the human–robot interaction, whereas in the second experiment we let two robots interact with each other. The results of these experiments are presented below.

4.5.1. Human–robot interaction

Here we let a human and a robot interact carrying a tray with bottles. Hence, here we do not consider two equilibrating DMPs (two robots), but only one. The goal is to avoid the red bar (left) not hitting it with the tray when moving along a curved trajectory (see Fig. 4B, T0). As in the simulations the robot has to learn to move in the same direction by reacting to the force sensor thereby helping the human avoiding the obstacle. Signals and resulting trajectories are shown in Fig. 4 and are similar to the experiments presented above. Note that here learning was applied only for the Y-coordinate. In this case learning stopped and weights stabilized after three learning trials (T1–T3, see the supplementary video, Appendix B).

4.5.2. Robot–robot interaction

In the second experiment, we let two KUKA robot-arms interact with each other where both robots were learning in this case. The experimental setup is shown in Fig. 5A. Robots were coupled by a wooden (painted in red) bar. The goal, as in the previous experiments, was to avoid obstacles (a box and a bottle) without hitting them when moving along a planned path. The positions of the obstacles were fixed and the paths for obstacle avoidance was predefined (as the planned trajectories) such that the robots were able to avoid obstacles without touching them when being decoupled. As paths are initially independent of each other, this, however, leads to the situation that prior to learning obstacles will be

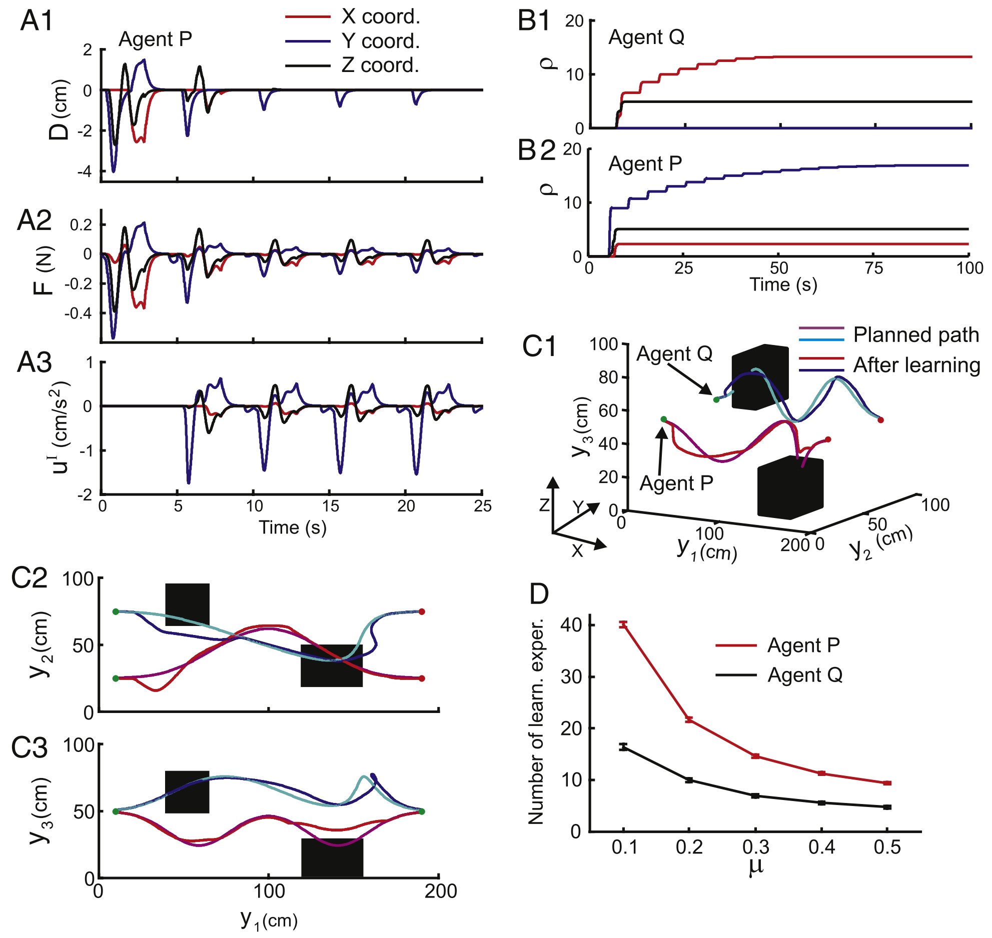

Fig. 3. **(A–C)** Simulation results from interaction learning with obstacles in a 3D scenario. In panels **A1–A3** signals only for agent P are shown. **(A1)** Displacement signals D; **(A2)**; predictive force signals F; **(A3)** output signals u^I; **(B1, B2)** weights ρ; **(C1–C3)** trajectories. The learning rate was $\mu = 0.2$. Weights for obstacle avoidance were tuned experimentally and were as follows: $\Gamma^P_{T,y_3} = 2$, $\Gamma^P_{V,y_3} = 2.5$, $\Gamma^Q_{T,y_2} = -2$, $\Gamma^Q_{V,y_2} = -2.5$, $\Gamma^P_{T/V,y_{1/2}} = 0$ and $\Gamma^Q_{T/V,y_{1/3}} = 0$. **(D)** Statistics for the 3D simulation as shown in panels A–C obtained from 100 experiments. In this case we changed start and end positions of the coupled agents in every trial (from a uniform distribution with interval $[-5; 5]$ cm for all, X, Y and Z positions), whereas the position of the obstacles was always the same. The average number of learning experiences needed to learn to interact is plotted vs. the learning rate μ. Error bars show confidence intervals of the mean (95%).

hit, when robots are coupled. Resulting signals are shown in panels C and D where we show one control case (equilibration, 0–20 s) followed by four learning trials and one post-learning trial. Note that here we have a full 3D case. In this case for the agent Q, learning stopped and weights stabilized after three trials and for agent P after four trials. Trajectories for planned, equilibrated and post-learning paths are shown in panels B1 and B2 (see the supplementary video, Appendix B).

5. Discussion

In this study we presented a combination of sensory driven interaction learning with dynamic movement primitives in a dual tightly-coupled agent system. Of importance for motivating the here-used learning mechanism is the following. Learning does not require manual tuning of parameters for sensors in order to produce appropriate behavior, but let the system find the right parameters by itself. Another advantage of using learning is that learning makes the system easily transferable to different agents/robots with different sensor–motor embodiments. The third reason which motivates the here-used learning is that the agent can adapt their interaction to new situations, which might occur due to environmental changes or changes in agents behavior. In the following we will compare our method to other existing approaches.

Previously, many different learning techniques, such as local/global regression techniques [1,2,27–29]) or reinforcement learning methods [30–34], were successfully applied to the DMP framework in order to learn movement trajectory and/or goal. Different from these approaches, we do not learn DMP weights ω, which encode the shape of the trajectory, but instead weight ρ of the interactive term u^I by which the trajectory is locally influenced. It is, of course, possible to combine ω- and ρ-learning. In addition to this, our approach can also be applied to other trajectory-shaping methods (like GMMs [3–5]). In this case the reactive term will have to be added to velocity, instead of acceleration as done here.

As already mentioned above, so far sensor-driven DMPs and GMMs have mainly been used in uncoupled systems. Different from this in the current study we were concerned with constructing sensor-driven *interactive* dynamic movement primitives in order to use them for cooperative tasks. We were interested in understanding the passive (equilibration) DMP-properties of coupled agents as well as their potential for conjoint adaptation. Conventionally, there are two architectures existing for introducing coupling: (1) master/slave and (2) non-master/slave [35]. In the first case one needs in advance to explicitly define master and slave, which is often a drawback of this architecture. The position of the slave (force-controlled) is defined by the position of the master (position controlled). The second architecture is a centralized approach where some reference frame is used to control both agents

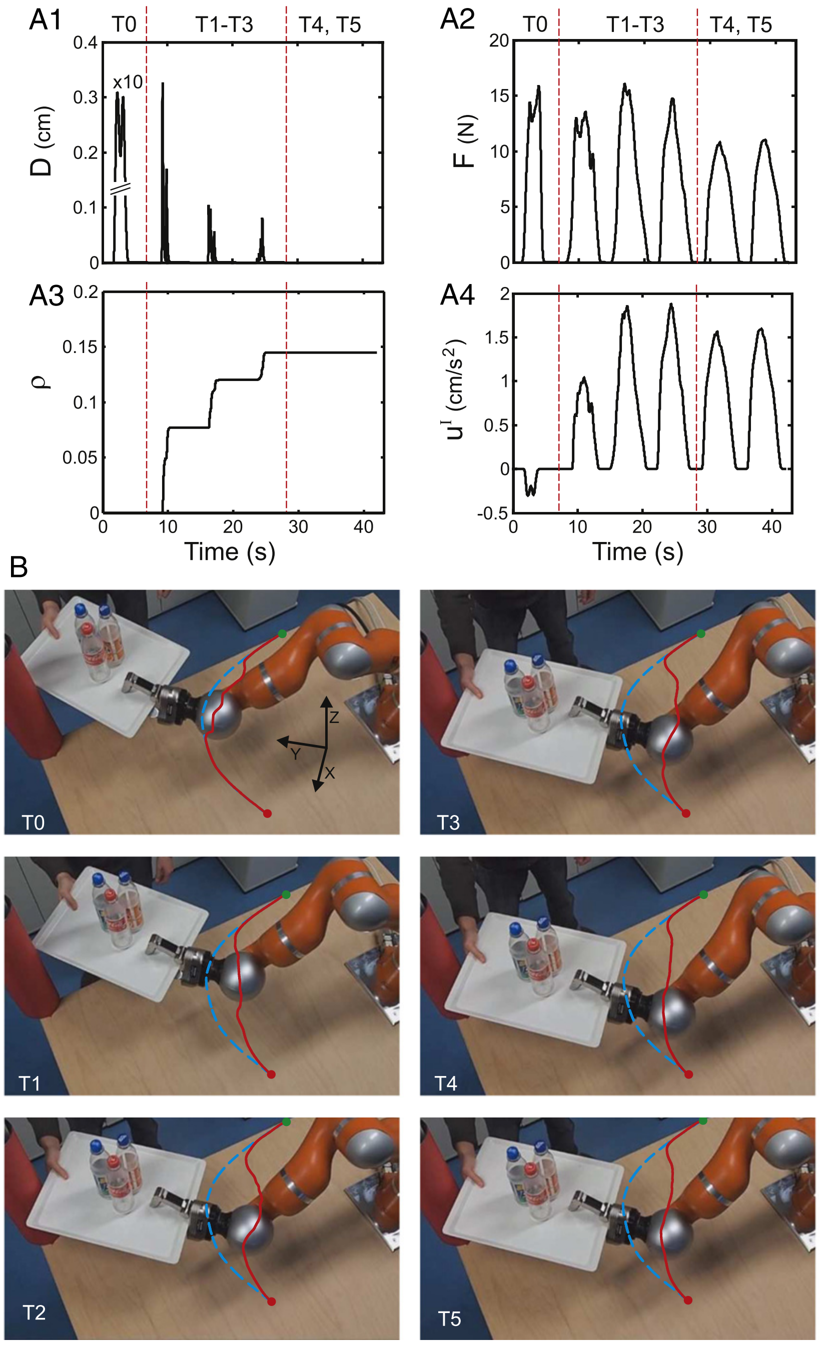

Fig. 4. Results from human–robot interaction learning. **(A)** Signal development from six trials (separated by dashed lines). Trial T0 is a control case-path-persistence behavior (no learning). **(A1)** Displacement signal D; **(A2)** predictive force signal F; **(A3)** weight ρ; **(A4)** output signal u^l. The learning rate was $\mu = 0.04$. **(B)** Trajectories for control case (T0), learning process (T1–T3) and post-learning (T4, T5). Dashed and solid lines represent planned and actual paths, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

at the same time. Such architectures usually are realized by using position/force [35–38], impedance control or variable impedance control with virtual stiffness [39,40]. Our approach is similar to the impedance control, however, instead of setting/adapting stiffness parameters, we learn (in a model-free way) the appropriate reaction in order to minimize external forces between agents and help to cooperate with each other.

Our approach creates an alternating (depending on the environment) leader/follower architecture. Thus, we do not have to explicitly define the agents' roles beforehand, but they obtain them depending on their sensory information. This way the one agent that first encounters the obstacle becomes the leader, whereas the other agent becomes the follower and learns to help the leader. Only in the real robot scenario presented above, we do indeed have a predefined master/slave architecture. However, here we wanted to show the example of the interaction learning where the robot learns to interact and help human. So here, only one (the robot) of the two agents was learning.

Another aspect of traditional master/slave architectures is that there is an important constraint which must be satisfied: the distance between the master and slave should be equal to the length of the object. In our approach we do not need to define this constraint. Agents act autonomously but this can also lead to situations where the object will be lost or deformed at the start of

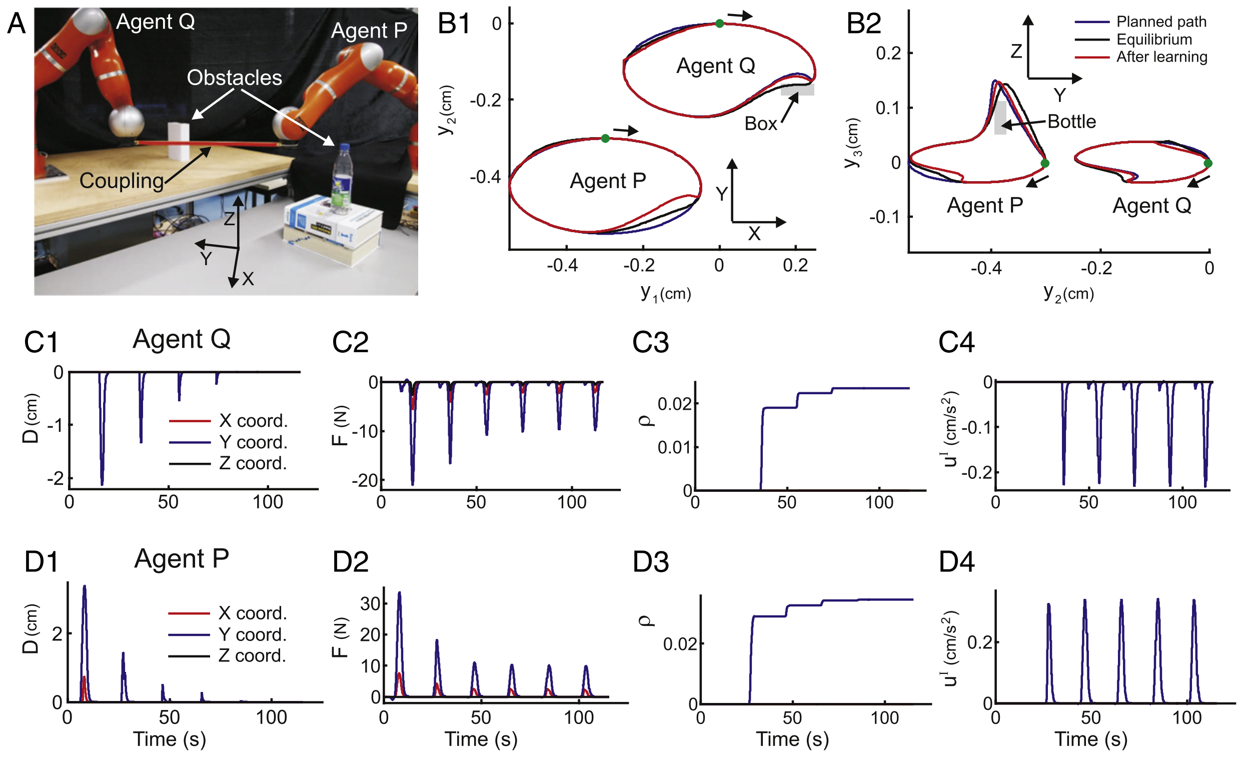

Fig. 5. Results from robot–robot interaction learning. **(A)** Experimental setup. **(B)** Trajectories for planned, equilibrated (before learning) and post-learning paths. Note that here robots started and ended at the same point marked by a green dot, whereas arrows show the direction of the movement. **(C1, D1)** Displacement signal D; **(C2, D2)** predictive force signal F; **(C3, D3)** weight ρ; **(C4, D4)** output signal u^l. The learning rate was $\mu = 0.001$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

learning. The trade-off between autonomy and potential damage cannot be resolved up front for all situations. In general, however, a "floating" master/slave architecture, such as the one present here, appears advantageous if the roles and/or the environmental constraints are not known in advance, i.e., for truly autonomous robots in a dynamic environment.

Also, different from the above mentioned tightly-coupled agent interaction approaches we use learning in order to acquire cooperate behavior instead of pre-programming it. There is a recent study by Gribovskaya et al. [41], which more closely relates to our method, where adaptive impedance control is employed for the agent interaction. Here the impedance parameters are adapted by using an iterative algorithm based on an error function. However, in this case robot-leader and robot-follower are again predefined in advance (master/slave architecture).

The way we couple DMPs with sensory inputs in order to produce an on-line reaction is similar to the approaches presented in [8,21,9,16,29,12,17]. Here, we show that such coupling can also be applied for dual-agent systems in order to solve cooperative tasks. Different from our approach, in [8,9,16,12] there is no learning applied and the reaction is generated by manually defined potential fields with fixed parameters which makes them incapable to adapt their behavior to new situations. [21,29,17] as in our case use learning, however, different from our approach, learning acts on DMP weights [21,29] and not on sensory terms as in our case. Also, learning in those approaches is performed in several phases, for example, first of all DMP weights are learnt to obtain basic behavior and only afterward they are modified by sensory feedback [21].

As explained above, the trajectory planning and on-line modification in our case was done in a task-space taking into account collision avoidance only for an end-effector of the manipulator. However, due to the interaction trajectories of the end-effectors might be altered in such a way that it will lead to link-collisions. To prevent such situations one can augment the system by adding potential fields on the manipulator links [42] or use an inverse-kinematics model in which the null-space is constrained to avoid link-collisions [43,8].

In summary, in this study we stressed the importance of combining sensory information with dynamic movement primitives and learning in a dual tightly-coupled agent system where the behavior and cooperation of agents is purely based on low level sensory information without any advanced planning. We believe that the here arising attractive properties, like fast adaptation, mutual equilibration, and cooperative interaction, can be very helpful for designing reactive, DMP-based motor control for co-operative tasks. It should also be easier to introduce planning as well as other (higher) cognitive traits into such systems as their sensory-reactions and low-level learning makes them already "well-behaved" from the beginning.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007–2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience. This research was also supported by BMBF (Federal Ministry of Education and Research), BCCN D2 (01GQ1005A).

Authors gratefully thank Christian Tetzlaff for helping with the stability analysis.

Appendix A. Stability analysis

To investigate how the coupled agents converge toward the goal we look at the fixed points of the dynamical system in Eqs. (10) and (11). We set $f_{i,j}[t] = 0$ and $r_{i,j}[t] = g_{i,j}$ because we are interested in the behavior as $t \rightarrow \infty$. Then without loss of generality we choose coordinates such that $g_{1,1} = g_{1,2} = g_x$, $g_{3,1} = g_{3,2} = g_z$ and $g_{2,1} = a/2 = -g_{2,2}$, for some constant $a \geq 0$. In other words, we align the y-axis with the spring in the goal position. Note that all parameters (i.e., α, β, m, k, d, a) are assumed positive.

For convenience we use center of mass coordinates in this section. Let $\vec{y}_j = (y_{1,j}, y_{2,j}, y_{3,j})^T$, $\vec{z}_j = (z_{1,j}, z_{2,j}, z_{3,j})^T$. Then the transformation of the new coordinates is given by

$$\begin{aligned} \vec{y}_1 &= \vec{R} + \vec{r}, \\ \vec{y}_2 &= \vec{R} - \vec{r}, \\ \vec{z}_1 &= \vec{V} + \vec{v}, \\ \vec{z}_2 &= \vec{V} - \vec{v}. \end{aligned} \tag{A.1}$$

Here $\vec{R} = (R_x, R_y, R_z)^T$ and $\vec{V} = (V_x, V_y, V_z)^T$ are the center of mass positions and their velocities respectively, $\vec{r} = (r_x, r_y, r_z)^T$ the relative coordinates, $\vec{v} = (v_x, v_y, v_z)^T$ their respective velocities. Standard linear stability analysis [44] of the transformed dynamical system reveals that depending on parameters, there is either only one stable fixed point or additionally an unstable one. In any case they share all coordinates besides r_y, these are as expected:

$$\begin{aligned} \vec{R}^* &= (g_x, 0, g_z)^T, \quad \vec{V}^* = (0, 0, 0)^T \\ \vec{v}^* &= (0, 0, 0)^T, \quad \vec{r}^* = (0, r_y^*, 0)^T. \end{aligned} \tag{A.2}$$

One fixed point (FP1) always exists and is stable throughout the parameter range.

$$FP1 : r_y^* = \frac{am\alpha\beta + 2dk}{2m\alpha\beta + 4k}. \tag{A.3}$$

This corresponds to both agents being as close to the goal position as the spring allows, e.g., let $d = a$ then $r_y^* = a/2 = g_{2,1}$ such that each agent's position coincides with its goal position. If $am\alpha\beta < 2dk$ there is another, unstable, fixed point (FP2) at

$$FP2 : r_y^* = \frac{am\alpha\beta - 2dk}{2m\alpha\beta + 4k}. \tag{A.4}$$

This corresponds to the situation where the agents switch around ($r_y^* < 0$) and block each other from reaching the goal by compressing the spring in between them. The fixed point FP2 disappears when the pull toward the goal position is stronger than the spring can compensate ($am\alpha\beta > 2dk$).

In order to analyze the asymptotic dynamics toward FP1 we look at the linearization of the dynamical system at FP1. The linearized twelve dimensional first order system can be rewritten as six second order equations of $R_x, R_y, R_z, r_x, r_y, r_z$. Each equation is that of a free damped harmonic oscillator. The equations are

$$\begin{aligned} \ddot{\vec{R}} &= -\alpha\beta\vec{R} - \alpha\dot{\vec{R}}, \\ \ddot{r}_x &= -\frac{\alpha\beta(m\alpha\beta + 2k)}{am\alpha\beta + 2dk} r_x - \alpha\dot{r}_x, \\ \ddot{r}_y &= -\frac{m\alpha\beta + 2k}{m} r_y - \alpha\dot{r}_y, \\ \ddot{r}_z &= -\frac{\alpha\beta(m\alpha\beta + 2k)}{am\alpha\beta + 2dk} r_z - \alpha\dot{r}_z. \end{aligned} \tag{A.5}$$

From here one can follow any text book on mechanics (e.g. [45]) to derive the DMP parameter that leads to critical damping β_c for each coordinate. For R_x, R_y and R_z we find $\beta_c^R = \alpha/4$ which corresponds to the standard value of β for uncoupled DMPs. For r_y we find $\beta_c^y = (m\alpha^2 - 8k)/(4m\alpha)$ which is smaller than β_c^R if $k > 0$. For r_x and r_z the expression is quite complicated

$$\beta_c^{x,z} = \beta_c^y + \frac{1}{8}\sqrt{\frac{am^2\alpha^4 - 16akm\alpha^2 + 32dkm\alpha^2 + 64ak^2}{am^2\alpha^2}}$$

and always larger than β_c^y. Note that for $a = d$, we obtain $\beta_c^{x,z} = \beta_c^R$. The goal in choosing β is that no oscillations occur, which means that if we cannot damp all coordinates critically, we prefer to overdamp them. In all cases above overdamping is achieved if $\beta < \beta_c$; therefore, the smallest β_c, i.e., β_c^y should be chosen for the coupled system. Because we need $\beta > 0$ for a converging DMP, we must require $\alpha > \sqrt{8k/m}$ in this case.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2013.07.009.

References

[1] J.A. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: Proc. 2002 IEEE Int. Conf. Robotics and Automation, 2002, pp. 1398–1403.
[2] S. Schaal, P. Mohajerian, J.A. Ijspeert, Dynamics systems vs. optimal control—a unifying view, Progress in Brain Research 165 (2007) 425–445.
[3] S.M. Khansari-Zadeh, A. Billard, BM: an iterative method to learn stable non-linear dynamical systems with gaussian mixture models, in: Proc. 2010 IEEE Int. Conf. Robotics and Automation, 2010, pp. 2381–2388.
[4] S.M. Khansari-Zadeh, A. Billard, Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming, in: Proc. 2010 IEEE Int. Conf. Intelligent Robots and Systems, 2010, pp. 2676–2683.
[5] S.M. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with gaussian mixture models, IEEE Transactions on Robotics 27 (2011) 943–957.
[6] B. Nemec, A. Ude, Action sequencing using dynamic movement primitives, Robotica 30 (5) (2012) 837–846.
[7] T. Kulvicius, K.J. Ning, M. Tamosiunaite, F. Wörgötter, Joining movement sequences: modified dynamic movement primitives for robotics applications exemplified on handwriting, IEEE Transactions on Robotics 28 (1) (2012) 145–157. http://dx.doi.org/10.1109/TRO.2011.2163863.
[8] D.-H. Park, H. Hoffmann, P. Pastor, S. Schaal, Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields, in: Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots, 2008, pp. 91–98.
[9] H. Hoffmann, P. Pastor, D.-H. Park, S. Schaal, Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance, in: Proc. 2009 IEEE Int. Conf. Robotics and Automation, 2009, pp. 1534–1539.
[10] N. Ratliff, M. Zucker, J.A. Bagnell, S. Srinivasa, Chomp: gradient optimization techniques for efficient motion planning, in: Proc. 2009 IEEE Int. Conf. Robotics and Automation, 2009, pp. 489–494.
[11] S. Calinon, I. Sardellitti, D.G. Caldwell, Learning-based control strategy for safe human–robot interaction exploiting task and robot redundancies, in: Proc. 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2010, pp. 249–254.
[12] A.S. Phung, J. Malzahn, F. Hoffmann, T. Bertram, Get out of the way — obstacle avoidance and learning by demonstration for manipulation, in: Proc. 11th IFAC World Congress, 2011, pp. 11514–11519.
[13] M. Mühlig, M. Gienger, J.J. Steil, Interactive imitation learning of object movement skills, Autonomous Robots 32 (2) (2012) 97–114.
[14] T. Matsubara, S.-H. Hyon, J. Morimoto, Learning parametric dynamic movement primitives from multiple demonstrations, Neural Networks 24 (5) (2011) 493–500.
[15] S.-M. Khansari-Zadeh, A. Billard, A dynamical system approach to realtime obstacle avoidance, Autonomous Robots 32 (2012) 433–454.
[16] O. Krömer, R. Detry, J. Piater, J. Peters, Adapting preshaped grasping movements using vision descriptors, in: Proc. 11th Int. Conf. Simulation of Adaptive Behavior: from Animals to Animats, 2010, pp. 156–166.
[17] P. Pastor, L. Righetti, M. Kalakrishnan, S. Schaal, Online movement adaptation based on previous sensor experiences, in: Proc. 2011 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2011, pp. 365–371.
[18] L. Righetti, J.A. Ijspeert, Programmable central pattern generators: an application to biped locomotion control, in: Proc. 2006 IEEE Int. Conf. Robotics and Automation, 2006, pp. 1585–1590.
[19] S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta, J.A. Ijspeert, A modular bio-inspired architecture for movement generation for the infant-like robot iCub, in: Proc. 2nd IEEE RAS/EMBS Int. Conf. Biomedical Robotics and Biomechatronics, 2008, pp. 795–800.
[20] S. Degallier, C.P. Santos, L. Righetti, J.A. Ijspeert, Movement generation using dynamical systems: a humanoid robot performing a drumming task, in: Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots, 2006, pp. 512–517.
[21] J. Kober, B. Mohler, J. Peters, Learning perceptual coupling for motor primitives, in: Proc. 2008 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2008, pp. 834–839.
[22] J.A. Ijspeert, J. Nakanishi, S. Schaal, Learning attractor landscapes for learning motor primitives, in: Proc. 15th Advances in Neural Information Processing Systems, 2003, pp. 1547–1554.
[23] S. Schaal, J. Peters, J. Nakanishi, J.A. Ijspeert, Learning movement primitives, in: Proc. 2003 Int. Symp. Robotics Research, 2004, pp. 561–572.
[24] B. Porr, F. Wörgötter, Isotropic sequence order learning in a closed loop behavioural system, Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences 361 (2003) 2225–2244.
[25] D.O. Hebb, The Organization of Behavior, Wiley & Sons, New York, 1949.
[26] Kuka Robot Systems. http://www.kuka-robotics.com.
[27] B. Nemec, M. Tamosiunaite, F. Woergoetter, A. Ude, Task adaptation through exploration and action sequencing, in: Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots, 2009, pp. 610–616.

[28] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, IEEE Transactions on Robotics 26 (2010) 800–815.
[29] A. Gams, M. Do, A. Ude, T. Asfour, R. Dillmann, On-line periodic movement and force-profile learning for adaptation to new surfaces, in: Proc. 10th IEEE-RAS Int. Conf. Humanoid Robots, 2010, pp. 560–565.
[30] J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients, Neural Networks 21 (4) (2008) 682–697.
[31] E. Theodorou, J. Buchli, S. Schaal, Reinforcement learning of motor skills in high dimensions: a path integral approach, in: Proc. 2010 IEEE Int. Conf. Robotics and Automation, 2010, pp. 2397–2403.
[32] P. Kormushev, S. Calinon, D.G. Caldwell, Robot motor skill coordination with EM-based reinforcement learning, in: Proc. 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2010, pp. 3232–3237.
[33] M. Tamosiunaite, B. Nemec, A. Ude, F. Wörgötter, Learning to pour combining goal and shape learning for dynamic movement primitives, Robotics and Autonomous Systems 59 (11) (2011) 910–922.
[34] J. Kober, J. Peters, Policy search for motor primitives in robotics, Machine Learning 84 (1–2) (2011) 171–203.
[35] M. Uchiyama, P. Dauchez, Symmetric kinematic formulation and non-master/slave coordinated control of two-arm robots, Advanced Robotics 7 (4) (1993) 361–383.
[36] E. Nakano, S. Ozaki, T. Ishida, I. Kato, Cooperational control of the anthropomorphous manipulator 'MELARM', in: Proc. 4th Int. Symp. Industrial Robots, 1974, pp. 251–260.
[37] S. Fujii, S. Kurono, Coordinated computer control of a pair of manipulators, in: Proc. IFTOMM World Congr., 1975, pp. 411–417.
[38] M. Uchiyama, N. Iwasawa, K. Hakomori, Hybrid position/force control for coordination of a two-arm robot, in: Proc. IEEE Int. Conf. Robotics and Automation, vol. 4, 1987, pp. 1242–1247.
[39] F. Caccavale, P. Chiacchio, A. Marino, L. Villani, Six-DOF impedance control of dual-arm cooperative manipulators, IEEE/ASME Transactions on Mechatronics 13 (5) (2008) 576–586.
[40] T. Tsumugiwa, R. Yokogawa, K. Hara, Variable impedance control with virtual stiffness for human–robot cooperative task (human–robot cooperative peg-in-hole task), in: Proc. 41st SICE Ann. Conf., vol. 4, 2002, pp. 2329–2334.
[41] E. Gribovskaya, A. Kheddar, A. Billard, Motion learning and adaptive impedance for robot control during physical interaction with humans, in: Proc. IEEE Int. Conf. Robots and Automation, 2011, pp. 4326–4332.
[42] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research 5 (1) (1986) 90–98.
[43] A.A. Maciejewski, C.A. Klein, Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, International Journal of Robotics Research 4 (3) (1985) 109–117.
[44] S.H. Strogatz, Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, first ed., Westview Press, 2001.
[45] P.A. Tipler, G. Mosca, Physics for Scientists and Engineers: Standard Version, fifth ed., W.H. Freeman, 2003.

Tomas Kulvicius received a Ph.D. degree in Computer Science (2010) from the University of Göttingen, Germany. In his Ph.D. thesis he investigated development of receptive fields in closed loop learning systems. Currently he is a Postdoctoral Researcher at the Department for Computational Neuroscience in the University of Göttingen. His research interests include closed loop behavioral systems, learning algorithms, receptive fields, robotics, dynamic movement primitives, biosignal analysis, biological system modeling.

Martin Biehl received his M.S. degree in Physics from Vienna University of Technology, Austria, in 2009. Currently he pursues a Ph.D. degree at the Adaptive Systems Research Group, The University of Hertfordshire. His research focuses on autonomy, interaction, and information theory of sensor actor systems.

Mohamad Javad Aein received Bachelor of Science degree in Electrical Engineering from Amirkabir University of Technology (AUT), Tehran, Iran, in 2006. He received his Master of Science degree in Control Systems in 2009 at the same university for his work on soft tissue modeling and needle insertion in Robotic surgery. From 2009 to 2011, he worked as research assistant in Advanced Intelligent Systems Research Center (AISRC) in Tehran, Iran, being engaged in numerous modeling and control-related projects. Currently he is doing Ph.D. at the Department of Computational Neuroscience, University of Göttingen, Germany. His research interests include control of robotic manipulators, trajectory planning and behavior-based robotic manipulations.

Minija Tamosiunaite has received a Ph.D. in Informatics in Vytautas Magnus University, Lithuania, in 1997. Currently she works as a senior researcher at the Bernstein Center for Computational Neuroscience, Inst. Physics 3, University of Göttingen. Her research interests include machine learning, biological signal analysis, and application of learning methods in robotics.

Florentin Wörgötter studied biology and mathematics at the University of Düsseldorf, Germany. He received his Ph.D. for work on the visual cortex from the University of Essen, Germany, in 1988. From 1988 to 1990, he was engaged in computational studies with the California Institute of Technology, Pasadena. Between 1990 and 2000, he was a Researcher at the University of Bochum, Germany, where he was investigating the experimental and computational neuroscience of the visual system. From 2000 to 2005, he was a Professor of computational neuroscience with the Psychology Department, University of Stirling, U.K., where his interests strongly turned toward "Learning in Neurons". Since July 2005, he has been the Head of the Computational Neuroscience Department at the Bernstein Center for Computational Neuroscience, Inst. Physics 3, University of Göttingen, Germany. His current research interests include information processing in closed-loop perception–action systems, sensory processing, motor control, and learning/plasticity, which are tested in different robotic implementations. His group has also developed the RunBot, which is a fast and adaptive biped-walking robot.