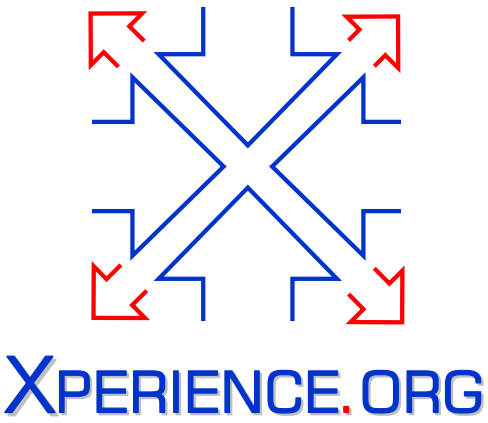
Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015



Deliverable Number:	D4.1.3
Deliverable Title:	Transfer of cooperative tasks: Report or scientific publication on implementation of cooperative task mechanisms within the architecture and in the final demonstration
Type (Internal, Restricted, Public):	PU
Authors:	Tamim Asfour, Aleš Ude, Martin Do, Tadej Petrič, Andrej Gams
Contributing Partners:	KIT, JSI

Contractual Date of Delivery to the EC:	31-01-2015
Actual Date of Delivery to the EC:	31-01-2015

Contents

Chapter 1

Summary

1.1 Objective of WP4.1: Cooperative Tasks

The objective of WP4.1 as stated in the Description of Work is to study tightly coupled physical interaction (excluding explicit language) for multiple agents to accomplish a cooperative task. The goal is to study different types of cooperation, in which multiple robots are engaged in task execution to fulfill the goal. We distinguish between:

- Tightly coupled cooperative manipulation tasks, in which a direct physical interaction between multiple agents (two arms, two robots, human-robot) must take place to achieve the task goal. Examples are cooperative table/box pushing, cooperative lifting, cooperative tool use, etc.
- Loosely coupled cooperative tasks, in which multiple robots are engaged in scene interpretation and reasoning about the role of each agent involved in the execution of the task.

The main issues that will be addressed are: a) How do agents that participate in a tightly coupled cooperation synchronize their actions? b) How can agents maximize the amount of mutual information for scene interpretation, c) How do agents recognize tasks that require cooperation, and d) How will the interpretation of the felt, seen and heard lead to the recognition of intention of "other" and finally to the recognition of plans.

1.2 Summary of the Results

In the area of tightly coupled cooperative manipulation tasks, the coaching interface, which is needed to implement some actions from the demonstration scenario, e. g. wiping, was transferred to the main demonstration platform ARMAR-III. These methods are described in Chapter 2. In addition, methods needed for synchronisation of robot-robot and robot-human collaborative manipulation actions were transferred to ARMAR-III. To accomplish this task we had to modify the underlying action representation based on DMPs, which were developed in WP 2.2. Therefore this work was reported in Deliverable D2.2.3 and is not repeated here.

The results and publications generated in WP4.1 in Year 4 of the Xperience project are described. These works provide more information about the cooperative behaviors that were developed for the Xperience demonstration platform ARMAR-III. Work in Year 4 includes 1. Altering robot behaviors based on human in the loop coaching gestures, and 2. Synchronization of dual-arm humanoid robot movement primitives. A lot of the important work in the area of tightly coupled cooperation was reported last year in deliverable D4.1.2 and in references [5, 4, 3]. Work on loosely coupled cooperative task perception and manipulation was also reported in this previous deliverable and in references [9, 10].

Chapter 2

Transfer to the Demonstration Platform

2.1 Transfer report for tightly coupled interaction

Among the targeted scenarios in WP4.1 are cooperative table/box pushing, cooperative lifting, cooperative tool use, etc. We addressed these issues by extending the framework of dynamic movement primitives with coupling terms, which enables the synchronization of DMPs in multi-agent behaviors. Since motor action representations including DMPs are also studied in WP2.2, we reported on the transfer of DMPs, including cooperative DMPs, in D2.2.3. In this report we focus on the implementation of coaching through tightly coupled visual and force-based interaction.

Coaching abilities to adapt dynamic motion primitives (DMPs) based on external feedback, which were initially implemented at JSI, were transferred to the main Xperience demonstration platform, i. e. ARMAR-III humanoid robot. The experimental setup is shown in Figure 2.1.

Transfer of the abilities to create and adapt motor behaviors based on human coaching gestures were implemented on the ARMAR-III robot. The technical implementation includes the periodic motor primitives with coaching abilities combined with the means to autonomously determine the basic frequency of a periodic input signal, based on a single adaptive frequency oscillator and an adaptive Fourier series [6].

The learning and coaching behavior is implemented in the ArmarX framework. This framework allows the implementation of distributed robot software components and the control of complex robot systems in a real-time manner. To support the development of novel robot programs ArmarX provides customizable building blocks for high level robot control e.g. robot actions are implemented as states which enhances the reusability and the integration in more complex scenarios. Following this methodology, novel actions needed for the learning and coaching of DMPs are implemented as states as well. With existing states the entire learning and coaching behavior is realized in the form of state chart.

The task for this scenario was wiping the table. It consists of three parts:

- Learning new behavior for wiping the table by using the movement imitation algorithm. Initially, given the color of a wiping tool, the robot localizes and grasps the tool using a visual servoing approach. The human coach demonstrates a wiping action using a similar wiping tool which is tracked by the robot in task space. The initial human movement, i. e. swipe from lower left to the upper right corner, which defines the working space is followed by periodic motion pattern. The learning of this motor behavior starts immediately after, allowing the robot to learn new motor behaviors. The underlying algorithms detected the basic frequency and learns the motion for the representation in the periodic DMPs. Note that here the robot is moving in a free space, without any contacts with the environment, i. e. table.
- Adapting robot motion behavior to the environmental constrains, for example to the table surface. Here the measured external force signal is used to properly alter the motion behavior. The previously

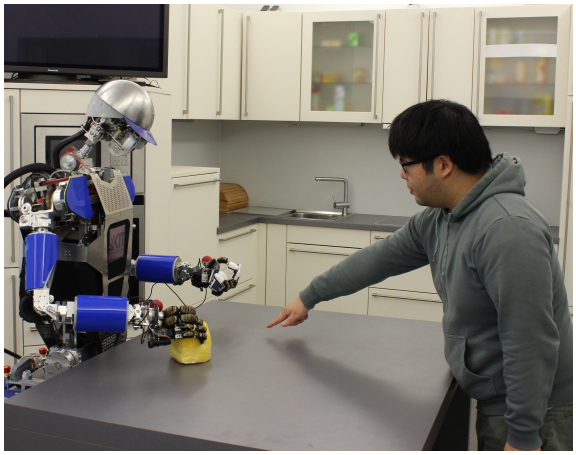


Figure 2.1: Experimental setup, where a human coach is modifying the robot's motion. The human coaching gesture is captured using either visual or hepatic feedback.

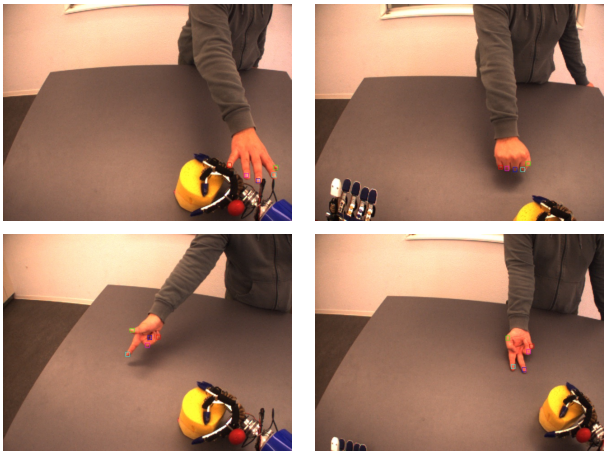


Figure 2.2: Different hand gestures were used for coaching. Top left shows the fist, top right shows the open hand and bottom plots shows the 1-finger and 2-finger pointing gestures respectively.

learned wiping DMP is augmented with the movement towards the table surface needed in order maintain constant force between the robot and the table.

- Altering robot motion based on human coaching using either hand gestures or force interaction. To determine human intentions in terms of coaching visual as well as haptic information is used. Embedded in a task transition control mechanism the robot is able to switch between force or vision-based coaching depending on the forces applied on the robot's TCP and the current configuration of the human hand. To detect whether forces are applied on the robot's TCP measurements from the force torque sensor in the robots wrist are evaluated. The recognition of the hand gesture of the human coach is based on the fingertip positions which are detected and tracked using a tracking algorithm which has been introduced in [1]. Based on the fingertip configuration, the robot can distinguish between varieties of different hand gestures. Four different hand gestures, as shown in Figure 2.2, were used for coaching: fist, open hand, and 1-finger and 2-finger pointing. With the 1-finger pointing gesture the coach can move the complete motion behavior to a different position in space. On the opposite, by using either fist or 2-finger pointing hand, the coach can alter only parts of motion behavior. Here the fist is used for pushing the motion trajectory towards the hand, and 2-finger pointing gesture is used to push trajectory away from the hand. The direction of altering the behavior is determine based on the vector that connects the hand and the robots end effector. The open hand denotes that the coach approaches the robot with the intention of force-based coaching. To facilitate the coaching for the human the frequency of the reproduced wiping movement decreases when the coaching hand approaches the robot's TCP.

Note that this scenario includes the implementation of several tasks and skills in a single, complete module. The user can thus re-run or adapt the scenario without explicitly knowing all the details of the underlying programming code.

The implementation of the underlying code in C++ is in a class which runs in a loop in the background. It is essentially integrating the equations of the periodic DMPs combined with the frequency and feedback adaptation to the external signals or coaching gestures. The following list includes the implemented C++ classes and the skills they comprise

- pDMPstructure

  Implementation of periodic DMPs with coaching algorithm for adaptation of the motion based on human coaching gestures and to achieve desired forces of non-rigid contact with the environment.

- AFSstructure

  Implementation of Adaptive Fourier series for extraction of the basic frequency of periodic signals.

- MotionControlGroup

ArmarX states which implement skills for the velocity- and position-based control of the robot's TCP. These states are needed to control the humanoid's arm for the imitation and reproduction of the demonstrated wiping movement.

- ForceAdaptation

  A state which implements the force-based mechanism needed for the adaptation of a learned wiping movement towards the surface to be wiped. Once the robot constantly applies a desired force on the surface, the robot's trajectories are stored and used for the augmentation of the corresponding DMP.

- GraspObjectGroup

  A collection of states implemented in ArmarX which are needed for the grasping of objects.

- VisualServoTowardsTargetPose

  Visual servoing skill which is needed for the robust grasping of the sponge. Based on visual information the robot's hand is accurately positioned at a desired object-specific grasping pose.

- ColorMarkerObjectTracking

  This skill implements a thread which is used for the capturing of the demonstrated wiping movement. It exploits that the wiping tool features a specific color and that the color is defined within the skill. By means of color segmentation and a particle filter tracking algorithm, the movements of the tool in task space are captured.

- FingerTipTracking

  Implementation of the fingertip tracking method. The method uses a skin color segmentation method for the segmenation of the human hand. Based on the segmented image, an edge map is formed. Using a circular hough transformation fingertip candidates are extracted. For the estimation of the actual fingertip positions, a particle filter framework is used. This estimation is refined using a Mean Shift algorithm

- AdaptiveHoughTransform

  Implementation of a Hough transformation needed for the calculation of fingertip features. It exploits that fingertips are represented by circular edge image features. Based on the depth information of the fingertips the radius of the Hough transformation is adapted for an appropriate scaling.

- ParticleFilterFingerTracking

  Implementation of the particle filter algorithm for the fingertip tracking. It transform the tracking of five fingertips into a contour tracking problem where the nodes of the contours denote the fingertips.

- FingerARModel

  Using image sequences as training data an AR2 model is learned for the prediction of the fingertip positions based on the movements of the fingertips in the past frames. This model is used to increase the robustness of the tracking method.

- GestureRecognizer

  Compares the determined fingertip configuration with manually defined patterns which are labeled with a certain gesture.

- ForceCoaching

  A state which implements the force-based mechanism needed for the coaching of a learned wiping movement. Once the robot experiences forces at the its wrist, the wiping movement is modified according to the forces exerted by the coach.

- VisionCoaching

  A state which implements the vision-based mechanism needed for the coaching based on the hand gesture of the human coach. Based on the recognized gesture using the stereo vision system, the human coach can modify the learned wiping movement by attracting or repelling the TCP or shifting the entire movement towards a new target location.

In addition to the above transferred and implemented capabilities on ARMAR-III, we integrated the developed capabilities in the first years of the project in the new software framework ArmarX. Among others, these are capabilities of force-based guiding of the robot and cooperative carrying of big objects which were described in D4.1.1.

2.2 Transfer report for loosely coupled interaction

The work on the transfer of loosely coupled interaction concentrated on the integration of the methods for cooperative perception and scene interpretation in the ArmarX framework to allow their use in the final demonstration. In particular, we focused on implementation of a MemoryX architecture in ArmarX which allow making the scene interpretation of one agent available to another agent for cooperative task execution.

MemoryX comprises all memory related components of the ArmarX framework. These components include basic building blocks for memory structures which can be either held in the system's memory or made persistent in a database. A memory architecture comprising a working memory (WM) and a long-term memory (LTM) is realized using these building blocks (see Figure **??**. Both memory types are organized in individually addressable segments containing arbitrary types or classes which are accessible within the distributed application. The WM is updated via an updater interface either by perceptual processes or by prior knowledge. Prior knowledge is stored in a non-relational database and allows enriching entities with known data (such as models or features). Besides directly addressing the WM, the working memory observer allows generating events based on changes of the memory content. The LTM offers an inference interface which allows attaching learning and inference processes.

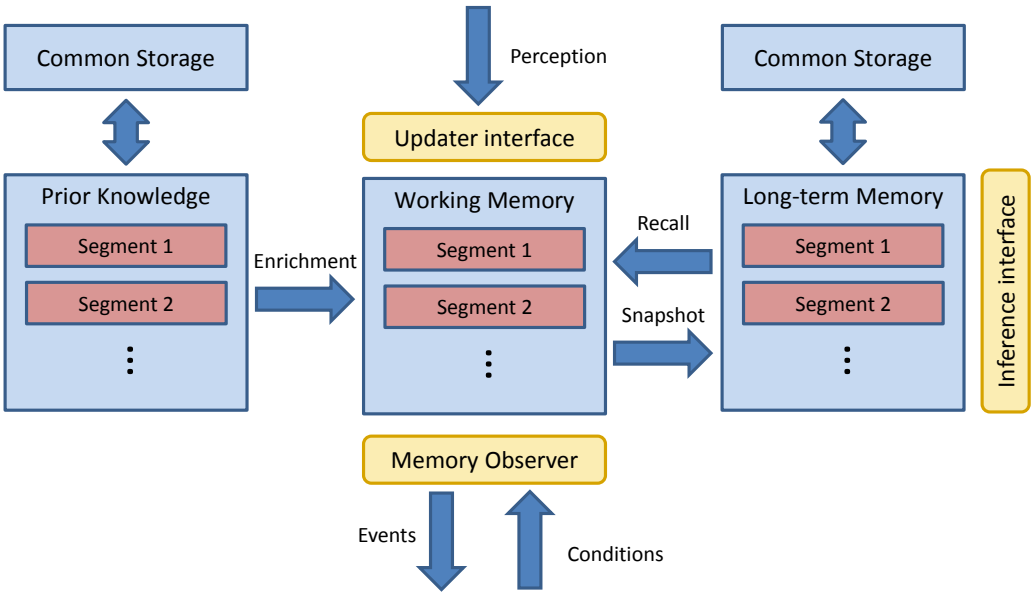


Figure 2.3: ArmarX offers the MemoryX architecture consisting of working memory, long-term memory, and a prior knowledge component. All memories are accessible within the distributed application. Appropriate interfaces allow attaching processes to the memory for updating and inference.

The perception components of the ArmarX provide facilities for including camera based image processing in the distributed robot application. VisionX allows implementing image providers and image processors as illustrated in Figure **??**. The image provider abstracts from imaging hardware and provides a data stream via shared memory or over Ethernet. Different image processors can be implemented that fall into the classes of object perception, human perception, and scene perception. Processing results are written to the working memory of MemoryX via the updater interface.

In cooperative tasks, each agent has its own environmental model represented in MemoryX, where spatial segments of MemoryX contain the agents knowledge about the environment including environmental

models, objects, grasping information, etc. Resulting object objects and their locations from perception are stored and made available in the spatial memory segments of the working memory.

We implemented and tested several perceptual components in ArmarX and MemoryX, which needed for loosely coupled interaction tasks. Among others, these are the following components:

- Gaze selection mechanism addresses three key problems of resource-aware visual perception: environmental model fusion, saliency calculation, and gaze selection (see [9, 10]).
- Object segmentation by physical interaction [7] and visual collision detection during grasping [8]; see also D2.1.3.
- Visual Servoing for single and bimanual manipulation tasks.

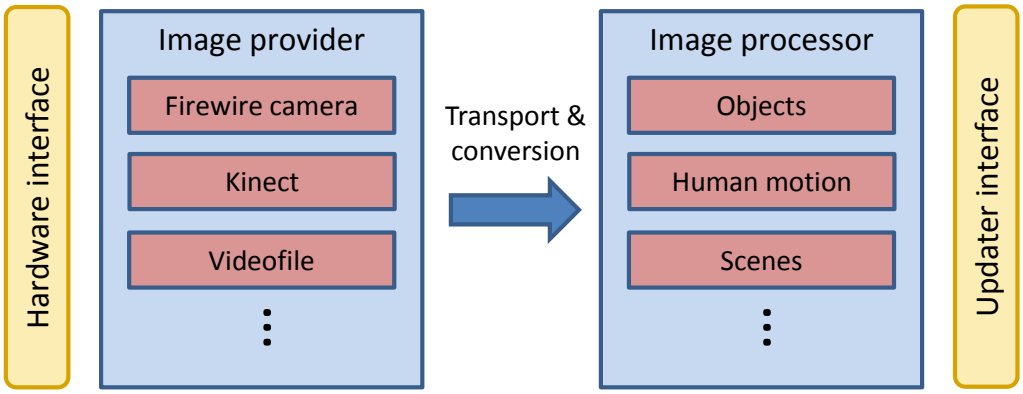


Figure 2.4: Image processing in VisionX. The image provider abstracts from hardware and streams data via shared memory or Ethernet to an image processor. Processing results are written to the working memory.

Chapter 3

Scientific Results

Here we describe the technical results of our research that were predominantly achieved in Year 4. Earlier work on cooperative tasks, which was carried out in WP4.1 in the first three years of the project, is described in Deliverable D4.1.1 and D4.1.2.

3.1 Altering Robot Behaviors Based on Human in the Loop Coaching Gestures

The creation and adaptation of motor behaviors is an important capability for autonomous robots. We developed a new approach for altering existing robot behaviors online, where a human coach interactively changes the robot motion to achieve the desired outcome [PGv+14]. Using hand gestures, the human coach can specify the desired modifications to the previously acquired behavior.

We were inspired by the efficiency of human-to-human skill transfer when developing a more effective approach to modify the existing robot behaviors. Rather than learning how to program robots, people can bring their own knowledge from interacting with each other directly into the robot domain. Our coaching system is based on periodic Dynamic Movement Primitives (DMPs) combined with an adaptive frequency oscillator, which can extract the phase and the frequency from an arbitrary periodic signal [6]. The primary goal of movement modeling with dynamical systems is to exploit the coupling phenomena to generate more complex behaviors. To preserve a natural posture while performing the task, the movement is encoded in the robots joint space.

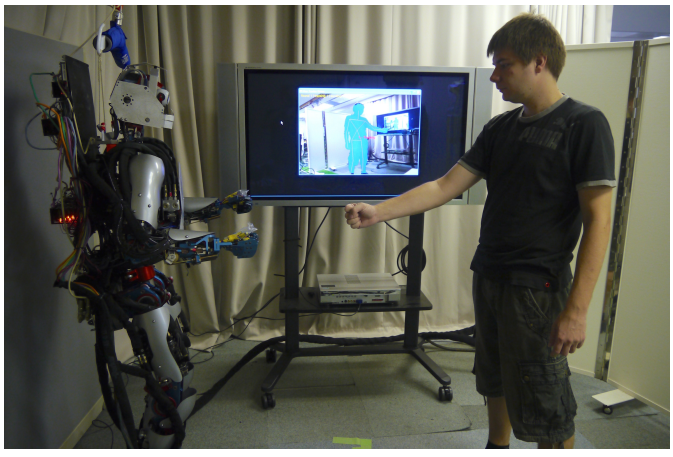


Figure 3.1: Experimental setup, where a human coach is modifying the robot's motion. The human coaching gesture is captured using Microsoft Kinect sensor.

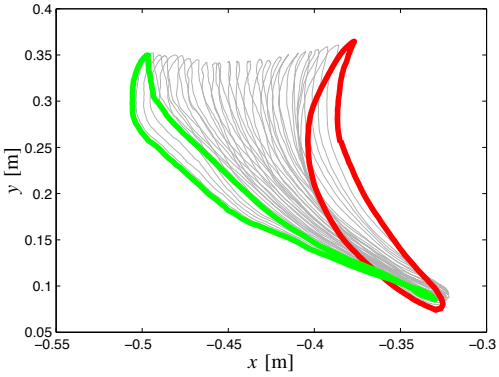


Figure 3.2: Task space motion of the robot's end-effector, where human coach was modifying the motion pattern. The initial trajectory is in red and the final trajectory is in green. The time evolution of the trajectory modification is indicated with grey line.

To make coaching as intuitive as possible, we developed an interface where the human coach can modify the trajectory by either pushing it away from him using pointing gestures with his right hand or attracting it towards him with his left hand. The coaching gestures are mapped to the robot joint space via robot Jacobian and used to create a virtual force field to affect the movement. The DMP modification method is based on a recursive least-squares technique for updating the weights of periodic DMPs. A recursive least squares technique was used to modify the existing movement with respect to the virtual force field. It is similar to what occurs during standard DMP learning, with the updating term changed from the difference between the desired human and the existing robot movement to the update term generated by the virtual force field, which is defined by the coaching gesture.

The proposed approach was evaluated on a simulated three degrees of freedom planar robot and on a real humanoid robot, where human coaching gestures were captured by an RGB-D sensor as shown in Fig. 3.1. Fig. 3.2 shows the task space motion of the robot's end-effector in the $x - y$ plane. We can see a successful modification of the motion based on the human coaching gestures. Although our focus was on rhythmic movements, the developed approach is also applicable to discrete (point-to-point) movements.

3.2 Synchronization of Dual-Arm Humanoid Robot Movement Primitives

In previous years we extended the framework of dynamic movement primitives with force/torque feedback, which is essential to enable the execution of bimanual and tightly coupled cooperative tasks. This work is described in [4, 3] and deliverable D4.1.2. Here and in the attached paper [GUMed] we use this previously developed mechanism to enable fast synchronization of dual-arm dynamic movement primitives (DMPs). Our approach is based on ideas from iterative learning control (ILC) and feedback error learning and integrates with a combined representation for discrete transient motion and periodic behavior in a single system [2].

Before the onset of the periodic movement pattern, rhythmic movements are often started in a non-periodic way. A practical example is walking, where the first step is different from the following steps. Indeed, the initial motion could be treated as separate motion and a combining mechanism would merge it with periodic motion, but treating the whole discrete-periodic process in a uniform system simplifies the structure of the control system for such discrete-periodic tasks.

To treat the discrete-periodic motion in a uniform system, represented by a dynamic movement primitive, a common canonical system is required. For the basis of the canonical system we augmented the system initially proposed by [2], to fully incorporate the modulation capabilities in speed and frequency of separate discrete or periodic motion, respectively. The canonical system is based on the location in the phase plane for the initial discrete behavior, and the location on the limit cycle for the subsequent periodic behavior.

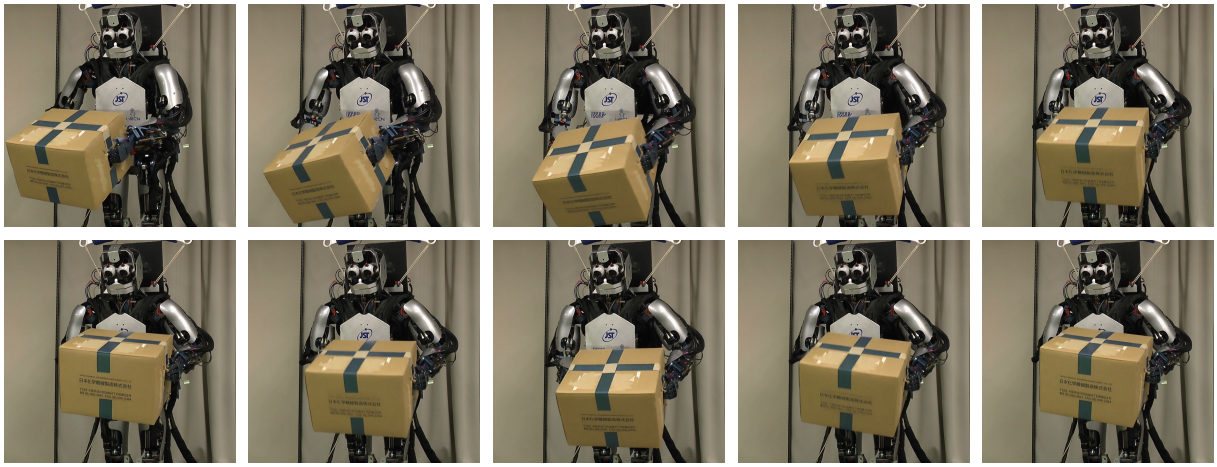


Figure 3.3: Dual arm box manipulation after the adaptation of the motion of the arms. The movement consists of initial point-to-point movement (first three images), followed by periodic box shaking.

Just as the motion itself, the adaptation of motion also has to account for the different nature of the transient and periodic parts. To adapt the transient (discrete) part, ILC can be used just as in [4, 3]. Repetitive control (RC) is a method similar to iterative learning control, but for periodic motions. However, even though ILC and RC have distinct differences, their essential features are nearly equivalent, and ILC has been applied to processes with periodic inputs as no-reset ILC. We therefore applied ILC for both parts of the discrete-periodic-DMP. One attempt at a trajectory, i. e. one epoch, now consists of one instance of the discrete part, and of ν periods of the periodic part.

To improve on the adaptation we modified the error of adaptation with elements of feedback error learning. By including the difference in the velocity terms, we accelerated the adaptation, leading to the reduction of forces between two agents that apply this approach and thereby reduced the already low number of iterations needed to achieve satisfactory results.

The application in our research was to synchronize the motion of two robot arms in the context of dual arm manipulation. We considered the task of lifting an unknown object with two arms in such a way that the relative object position and orientation between the two arms remain constant. Even though the task was set in the task space, we have shown that the adaptation can be applied in the joint space. Figure 3.3 shows the implementation on a full-sized Sarcos CBi humanoid robot.

References

[1] M. Do, T. Asfour, and R. Dillmann. Particle filter-based fingertip tracking with circular Hough transform features. In *IAPR International Conference on Machine Vision Applications (MVA)*, Nara, Japan, 2011.

[2] J. Ernesti, L. Righetti, M. Do, T. Asfour, and S. Schaal. Encoding of periodic and their transient motions by a single dynamic movement primitive. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 57–64, Osaka, Japan, 2012.

[3] A. Gams, B. Nemec, A. Ijspeert, and A. Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics*, 30(4):816–830, 2014.

[4] A. Gams, B. Nemec, L. Žlajpah, A. Ijspeert, T. Asfour, and A. Ude. Modulation of motor primitives using force feedback: Interaction with the environment and bimanual tasks. In *IEEE/RSJ International Conference on Intelligent Systems and Robots*, pages 5629–5635, Tokyo, Japan, 2013.

[5] T. Kulvicius, M. Biehl, M. J. Aein, M. Tamosiunaite, and F. Wörgötter. Interaction learning for dynamic movement primitives used in cooperative robotic tasks. *Robotics and Autonomous Systems*, 61(12):1450–1459, 2013.

[6] T. Petrič, A. Gams, A. J. Ijspeert, and L. Žlajpah. On-line frequency adaptation and movement imitation for rhythmic robotic tasks. *The International Journal of Robotics Research*, 30(14):1775–1788, 2011.

[7] D. Schiebener, A. Ude, and T. Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014.

[8] D. Schiebener, N. Vahrenkamp, and T. Asfour. Visual collision detection for corrective movements during grasping on a humanoid robot. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Madrid, 2014.

[9] K. Welke, P. Kaiser, A. Kozlov, N. Adermann, T. Asfour, and M. Steedman. Grounded spatial symbols for task planning based on experience. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 484–491, Atlanta, Georgia, 2013.

[10] K. Welke, D. Schiebener, T. Asfour, and R. Dillmann. Gaze selection during manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 652–659, Karlsruhe, Germany, May 2013.

Attached Papers

[GUMed] A. Gams, A. Ude, and J. Morimoto. Efficient synchronization of dual-arm humanoid robot movement primitives. In *IEEE/RSJ International Conference on Intelligent Systems and Robots (IROS)*, Hamburg, Germany, 2015 (submitted).

[PGv+14] T. Petrič, A. Gams, L. Žlajpah, A. Ude, and J. Morimoto. Online approach for altering robot behaviors based on human in the loop coaching gestures. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1790–1795, Hong Kong, 2014.

Efficient Synchronization of Dual-Arm Humanoid Robot Movement Primitives

Andrej Gams[1], Aleš Ude[2] and Jun Morimoto[3]

Abstract— **Direct replication of human-demonstrated motion on a robotic platform usually does not achieve the same result as the demonstrated motion due to different kinematic and dynamic properties of the human and the robot. Nevertheless, it can serve as a first approximation for learning of new motor skills. In this paper we propose a new approach for the synchronization of dual-arm dynamic movement primitives (DMPs), which is based on ideas from iterative learning control (ILC) and feedback error learning. To simultaneously represent discrete and periodic movements, the proposed approach utilizes a DMP formulation, which encodes an initial discrete motion, followed by a periodic behavior, all in a single system. We successfully applied the developed method to an example dual arm manipulation task, which was implemented on a real humanoid robot. Our results show that by incorporating the elements of feedback error learning into current-iteration ILC, efficient synchronization of dual-arm humanoid robot movements can be achieved.**

I. INTRODUCTION

Robot programming by demonstration has been extensively used for the generation of robotic motion [1] including full body motion [**?**]. Neurobiological findings motivated also the use of action primitives [2]. Amongst the various methods of encoding demonstrated motion using primitives, dynamic movement primitives (DMP) have emerged for both discrete point-to-point motion as well as for periodic motion [3]. Our study is based on a computational model that combines discrete and rhythmic movement primitives in a unified representation, where discrete movement can smoothly transition to a rhythmic behavior. We showed how the initially trained behaviour can be modified to account for external feedback generated by the constraints of the task.

Combining discrete and periodic motion has been studied from the perspective of common neural control systems in humans [4], [5]. In robotics, motor primitives and central pattern generators were used for combined rhythmic-discrete motion represntation by Degallier et al. [6], who applied their framework to infant crawling and drumming on a humanoid robot. As un underlying structure they used nonlinear oscillators, which allowed the transition between different types of behaviors. Oscillators were also used in a control structure by Ajallooeian et al. [7], who showed how nonlinear phase oscillators can be morphed to an arbitrary limit cycle, including an initial transient motion. A combined representation for discrete transient motion and periodic behavior in a single system was developed by Ernesti et al. [8], who applied DMPs. These authors modified the standard DMP representation so that the canonical system starts in the phase plane away from the limit cycle and later converges to the limit cycle. The location in the phase plane provides the canonical system for the initial discrete behavior, while location on the limit cycle for the subsequent periodic behavior. In this paper we name this kind of a DMP a discrete-periodic dynamic motion primitive (DP-DMP).

Despite the rich and favorable properties of DMPs for robotic control, direct replication of the demonstrated movement on the robot usually does not produce the desired task behavior. However, direct demonstration using DMPs can be modified by external feedback [9], [10], or can serve as an initial approximation for learning of the correct signal, e. g. using reinforcement learning [11], [12]. As DMPs can also be coupled to external signals, e.g., Gams et al. [13] proposed using iterative learning control (ILC) to learn task-appropriate coupling signals. The nature of ILC makes it applicable to discrete processes [14] and repetitive control (RC) was proposed for periodic DMPs [15]. Periodic DMPs were also modified using feedback error learning [16]. The authors learned the feed-forward accelerations provided to an inverse dynamics model in order to minimize the feedback control effort of an additional stabilizing control law. DMPs have also been extended to Interaction Primitives [17], incorporating tools for synchronizing, adapting, and correlating motor primitives between cooperating agents. Interaction forces were learned from human demonstrations in [18].

In this paper we build on DP-DMP formulation and include the means to adapt to external feedback and learn coupling terms which exclude feedback errors. The main contributions of the paper are

- A modification of a unified representation based on dynamic systems to simultaneously encode rhythmic and periodic components of the movement.
- The application of current-iteration iterative learning control to modify the initially trained discrete and periodic movements that smoothly transition from one to another.
- Improving the convergence and performance of the

*This work was supported by EU Seventh Framework Programme grant 270273, Xperience. It was also supported by MEXT KAKENHI Grant Number 23120004; by MIC-SCOPE; by JSPS and MIZS: Japan-Slovenia research Cooperative Program; by JST-SICP; and by SRPBS, MEXT. A. Ude would like to thank NICT for its support within the JAPAN TRUST International Research Cooperation Program.

[1]Andrej Gams is with Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia andrej.gams@ijs.si

[2]Aleš Ude is with Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia and with Department of Brain Robot Interface (BRI), ATR Computational Neuroscience Labs, Kyoto, Japan ales.ude@ijs.si

[3]Jun Morimoto is with Department of Brain Robot Interface (BRI) ATR Computational Neuroscience Labs, Kyoto, Japan xmorimo@atr.jp

iterative learning control algorithm by incorporating the principles of feedback-error-learning method, where the transformed feedback signal is used to train the underlying motor representation.
- The modification of the trajectories in joint space as opposite to the previous applications of modifications in task space.
- Application of the developed approach for the synchronization of dual arm movements.

This paper is organized as follows. In Section II we first provide the structure of the periodic DMP with a transient part, including the modifications of the original contribution of Ernesti et al. [8]. The introduction of both iterative learning control and elements of feedback learning is explained in Section III. Application of the DMP modulation in joint space is provided in Section IV. Section V discusses simulated and real-world results, including an assessment of the learning when more complex feedback error is added. Discussion and conclusions follow.

II. Unified representation for discrete and periodic movements

This section provides a recap of the DMP formulation that allows encoding a periodic motion and its initial discrete transient part, i. e. a DP-DMP, and explains our extensions.

Standard dynamic movement primitives (DMPs) are based on a damped spring model [3]

$$\dot{z} = \Omega\left(\alpha_z\left(\beta_z\left(g-y\right)-z\right)+\mathrm{f}\right), \quad (1)$$

$$\dot{y} = \Omega z, \quad (2)$$

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

In this work we make use of the forcing term proposed in [8], which results in a representation that simultaneously encodes discrete and periodic movements. For this purpose, the forcing term f is defined as a function of two phase variables ϕ and r, i. e. $\mathrm{f}(\phi, r)$. The evolution of the two-dimensional phase variable (r, ϕ) is governed by the following canonical system

$$\dot{\phi} = \Omega, \quad (3)$$

$$\dot{r} = \Omega\eta\left(\mu^{\alpha} - r^{\alpha}\right) r^{\beta}. \quad (4)$$

In contrast to [8], we added parameter Ω also to Eq. (4), which is important to enable one of the basic properties of DMPs, i. e. temporal scaling. Ω is defined as $\Omega = 2\pi/p$, where p is equal to the period of rhythmic movement in seconds. The parameters α, β, and η are positive constants with which the speed of convergence of r to μ can be adjusted. We used $\eta = 6$, $\alpha = 1/6$, $\beta = 0.001$ and $\mu = 1$ as constants. To determine the initial phase $(r_{\mathrm{init}}, \phi_{\mathrm{init}})$ from where the integration of (3) – (4) is started, we first define the phase at which the discrete movement transitions into the periodic movement. In our experiments this transition was defined to occur at the phase $(\mu_1, 0)$. The initial phase was then calculated by back-integrating Eq. (3) – (4) for the duration T of the discrete part of the movement. This duration can be obtained from the training data.

Note that both the transformation system (1) – (2) and the canonical system (3) – (4) are only indirectly dependent on time, which is essential to ensure many favourable properties of DMPs, e. g. scale and temporal invariance and easy modulation of control parameters for online trajectory modification.

Using the above phase, f can be defined as [8]

$$\mathrm{f}(\phi, r) = \frac{\sum_{j=1}^{N} v_j \psi_j(r,\phi) + \sum_{i=1}^{M} w_i \xi_i(r,\phi)}{\sum_{j=1}^{N} \psi_j(r,\phi) + \sum_{i=1}^{M} \xi_i(r,\phi)}, \quad (5)$$

where ψ_j and ξ_j are the forcing terms specifying the discrete and periodic parts of movement, respectively:

$$\psi_j(r,\phi) = \mathrm{b}(r)\exp\left(l_j(\cos(\phi - c_j) - 1)\right), \quad (6)$$

$$\xi_j(r,\phi) = \mathrm{a}(r)\exp\left(-h_j\left\|\begin{bmatrix} r\cos(\phi) \\ r\sin(\phi)\end{bmatrix} - \mathbf{q}_j\right\|^2\right). \quad (7)$$

The centres of periodic forcing terms $c_j = (j-1)2\pi/N + \pi/N$ are uniformly distributed with constant widths $l_j = 1.25N, j = 1, \ldots, N$. The centers of discrete forcing terms are taken at $\mathbf{q}_j = [r_j\cos(\phi_j), r_j\sin(\phi_j)]^{\mathrm{T}}$, where the phase centers (r_j, ϕ_j) are determined so that approximately the same number of integration steps is needed to integrate (3) – (4) from (r_{j-1}, ϕ_{j-1}) to (r_j, ϕ_j), for all j. The first center point (r_1, ϕ_1) is equal to the initial phase of the movement, $(r_1, \phi_1) = (r_{\mathrm{init}}, \phi_{\mathrm{init}})$, while (r_M, ϕ_M) is equal to the phase where discrete movement transitions into the periodic movement, $(r_M, \phi_M) = (\mu_1, 0)$. The widths of the kernels are determined as $h_j = 0.5/(\|\mathbf{q}_{j+1} - \mathbf{q}_j\|^2), j = 1, \ldots, M-1, h_M = h_{M-1}$.

Kernel functions a and b implement the transition from discrete to period movement. We defined them using a tricube kernel

$$\mathrm{b}(r) = \begin{cases} 1, & r < \mu_1 \\ \left(1 - \left(\frac{r-\mu_1}{\mu_2-\mu_1}\right)^3\right)^3, & \mu_1 \le r \le \mu_2 \\ 0, & r > \mu_2 \end{cases} \quad (8)$$

$$\mathrm{a}(r) = \begin{cases} 0, & r < \mu_1 \\ \left(1 - \left(\frac{\mu_2-r}{\mu_2-\mu_1}\right)^3\right)^3, & \mu_1 \le r \le \mu_2 \\ 1, & r > \mu_2 \end{cases} \quad (9)$$

Note that in [8] Gaussian kernels were used to define a and b. The advantage of our definition is that the above kernel functions really become equal to zero and not just tend to

zero as Gaussian kernels do. In our experiments we used $\mu_1 = 1.2\mu$ and $\mu_2 = 1.4\mu$ as constants.

III. Introducing feedback error learning into iterative learning control

The application we had in mind in our research was to synchronize the motion of two robot arms in the context of dual arm manipulation. We considered the task of lifting an unknown object with two arms in such a way that the relative object position and orientation between the two arms remain constant. This happens if the robot holds the object so that the force acting on the arms is constant. If the movement of the two arms is specified by two DMPs given by Eq. (1) – (4), synchronous arm behavior can be achieved by modifying one of the two DMPs so that the forces and torques acting on both arms are constant, i. e. $\mathbf{F}(t) = \mathbf{F}_d$ and $\mathbf{M}(t) = \mathbf{M}_d$.

Lets consider now dual arm manipulation, where the initial arm movements are specified in task coordinates. The application to joint coordinates is presented in the next section. The sensory feedback is provided by measuring gravity-compensated forces and torques acting on both arms. If the task is to modify the movement of the second arm so that it lifts the object together with the first arm, then the feedback signal can be defined as

$$\mathbf{e}(t) = \begin{bmatrix} \mathbf{F}_d - \mathbf{F}_2(t) \\ \mathbf{M}_d - \mathbf{M}_2(t) \end{bmatrix}, \tag{10}$$

where $\mathbf{F}_d$, $\mathbf{M}_d$ are the desired forces and torques and $\mathbf{F}_2(t)$ and $\mathbf{M}_2(t)$ the actual forces and torques acting on the second arm. The basic idea for adaptation is taken from the feedback error learning approach [19], where the robot motion is modified in such a way that the feedback signal tends to zero as the learning proceeds.

Iterative learning control (ILC) provides an effective way to minimize feedback error. In iterative learning control information about feedback error can be used to improve the performance in the next repetition of the same behavior. We propose to apply the current-iteration ILC, which is given by the formula [14]

$$u_{j+1}(k) = \underbrace{Q(u_j(k) + Le_j(k+1))}_{\text{feedforward term}} + \underbrace{Ce_{j+1}(k)}_{\text{feedback term}}, \tag{11}$$

where u is the control signal, k denotes the k-th time sample, j denotes the learning iteration, and Q and L are the learning parameters. ILC is distinguished from simple feedback control by the prediction of the error $e(k+1, j)$, which serves to anticipate the error caused by the action taken at the k-th time step. ILC modifies the control input in the next iteration based on the control input and feedback error in the previous iteration.

In the case of DP-DMP, the learning needs to apply both to the discrete and the periodic parts. However, ILC was originally developed for discrete processes only. Repetitive control (RC) was proposed instead of ILC to modify periodic DMPs [15], but this method cannot be applied to the initial discrete part. However, as stated in [20], ILC and RC have distinct differences, but their essential features are nearly equivalent, and ILC has been applied to processes with periodic inputs as no-reset ILC [21]. We therefore apply ILC for both parts of the DP-DMP. One attempt at a trajectory, i. e. one epoch, now consists of one instance of the discrete part, and of ν periods of the periodic part.

In the context of DP-DMPs, we add the control signal u to Eq. (2)

$$\dot{y} = \Omega z + u, \tag{12}$$

where u is written as

$$u(\phi, r) = \frac{\sum_{j=1}^{N} \tilde{v}_j \psi_j(r, \phi) + \sum_{i=1}^{M} \tilde{w}_j \xi_j(r, \phi)}{\sum_{j=1}^{N} \psi_j(r, \phi) + \sum_{i=1}^{M} \xi_j(r, \phi)}. \tag{13}$$

u, defined using the same kernel functions (6) and (7) as the forcing term (5), is thus encoded by kernel function parameters $[\tilde{\mathbf{v}}, \tilde{\mathbf{w}}]^{\mathrm{T}}$, $\tilde{\mathrm{v}} = [\tilde{v}_1, \ldots, \tilde{v}_N]$, $\tilde{\mathrm{w}} = [\tilde{w}_1, \ldots, \tilde{w}_M]$, which are mapped to the control signal $\mathbf{u}_j = [u_j(1), \ldots, u_j(T)]$ during the execution. Similarly, mapping from the phase dependent control signal $\mathbf{u}$ to $[\tilde{\mathbf{v}}, \tilde{\mathbf{w}}]^{\mathrm{T}}$ is accomplished with regression [3], [22]. Thus learning the feedforward signal $\mathbf{u}$ involves its adaptation using formula (11), followed by regression to calculate the parameters $[\tilde{\mathbf{v}}, \tilde{\mathbf{w}}]^{\mathrm{T}}$.

IV. DP-DMP coupling at the joint level

While demonstrations may be in task space, transfer of motion to the robot can also include joint space trajectories, acquired by kinesthetic guiding. By maintaining the movement representation in the joint space, we preserve the information about the robot configuration, which is specifically important for a redundant robot. Previous publications for modulating the DMP trajectory using external force feedback considered task space trajectories [13]. In the following we show how it can be applied to joint trajectories.

Ideally, one would use the full inverse dynamic model to calculate the motor command error given the task space error. Since accurate inverse dynamics models are difficult to obtain, feedback error learning approach [19] uses the output of a simple controller consisting of proportional, differentiation, and acceleration feedback to iteratively improve the motor command. In our work, the proportional part is provided by (10), but the performance can be improved by adding a differentiation term. If the Cartesian space velocities of both arms are available, the force feedback signal (10) can be enhanced by the velocity feedback and mapped into the joint space

$$\begin{aligned} \mathbf{e}(t) \;=\; & K_1 \mathbf{J}_2^{\mathrm{T}} \begin{bmatrix} \mathbf{F}_d - \mathbf{F}_2(t) \\ \mathbf{M}_d - \mathbf{M}_2(t) \end{bmatrix} + \\ & K_2 \mathbf{J}_2^{+} \begin{bmatrix} \dot{\mathbf{x}}_1(t) - \dot{\mathbf{x}}_2(t) \\ \boldsymbol{\omega}_1(t) - \boldsymbol{\omega}_2(t) \end{bmatrix}. \end{aligned} \tag{14}$$

Here $\mathbf{J}_2^{+}$ denotes the pseudoinverse of the task Jacobian of the second arm. The second term in the above equation is based on the fact that the relative motion of the two arms holding an object should be equal to zero. Feedback signal (14) ensures faster convergence to the desired behavior than the simpler feedback signal (10) multiplied by the transpose of the task Jacobian. Since the control signal u is given at

the velocity level, it does not make sense to add also the acceleration feedback term.

V. RESULTS

A. Experimental setup

In our experiments we considered dual arm manipulation, where the trajectories of the arms were encoded as DP-DMPs in joint space. The task was to synchronize the movement of both arms in 3D space so that the robot could perform the pre-defined object manipulation motion. The trajectory of the right arm was predefined, while the joint trajectories of the left arm had to adapt using our approach so that the two arms moved in synchrony, keeping the box held by both arms at constant relative position between the two arms. In the experiments we used a full-size humanoid robot Sarcos CB-i [23]. The setup and the resulting motion after learning is presented in the image sequence of Fig. 4. The accompanying video shows that the robot was able to hold the box between the two arms.

B. Dual arm manipulation

As described above, the joint trajectories of the arms are synchronized to minimize the feedback received in the task space. The final goal was that the arms move with constant distance for the complete discrete-periodic trajectory. One epoch of motion consists of a discrete part, followed by 3 periods of the periodic part. The robot then returns to the original position, which is also depicted in the plots.

Figure 1 shows the results of the trajectories in z (up-down) direction of the task space. The other two task-directions were already synchronized so that the robot could manipulate the object. We can see in the top plot the discrete-periodic nature of the motion, where three periods of periodic motion were executed in every epoch. The periodic part is, due to 3 periods per repetition, adapted within 10 epochs. The discrete motion, which is quite fast, takes slightly longer. The bottom plot shows the error of learning over time. Exponential convergence can be observed.

Figure 2 shows the adaptation of the joint space trajectories for the same experiment. The blue line shows the original trajectories of the first 4 joints of the robot's left arm – the top three approximating the shoulder joint and the last one the elbow. The red lines show the adapted trajectories of motion for the same joints. Wrist joints were not modulated.

It is important to note that even though the feedback is in task space and the task space trajectories match in two dimensions, the adaptation of the DMP occurs in joint space and joint-space trajectories do not match in all dimensions. This is clearly seen in Fig. 3, where the trajectories of the left and the right arm joints are shown. Since we used 4 DOFs for the robot's arm and only 3 are needed to control the task space position trajectory, the robot arm is redundant for the task and the joint trajectories of the left arm are not symmetrical to the joint trajectories of the right arm, which is clearly visible when comparing shoulder abduction-adduction (SAA) and elbow joints (EB).

C. Effect of Feedback Error Learning

Changing the feedback to include the velocities as in (14), as an element of feedback error learning, increases the speed of learning. Faster adaptation is specifically clearly observable in the first few epochs. Figure 5 shows the task space end-effector velocities for the same experiment. The green line shows the right arm velocities, the blue line the velocities when using (10) and the red line when using (14). While some overshoot appears when using the velocity as in feedback term (14), overshoot occurs also when velocity is not used, where it also takes longer to fade-out.

The difference in positions is shown in Fig. 6. The difference is clearly visible in the first epoch, while both approaches converge to the final trajectories roughly at the same time.

D. Modulation of execution frequency

We improved the original formulation of the canonical system to allow for the modulation of the execution frequency of the periodic part by modifying (4). Without this change DP-DMPs do not include one of the basic DMP properties – modulation of execution (playback) speed with only one parameter. Fig. 7 shows the results of reducing the parameter Ω from 4π to 2π. In the top plot we can see that when using the original canonical system the execution is not stable, given in red. The blue trajectory shows the behavior of the system with $\Omega = 4\pi$. The bottom plot shows the behavior of the system when using the modified canonical system. Note that the adaptation of the trajectories is not affected.

VI. DISCUSSION AND CONCLUSION

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

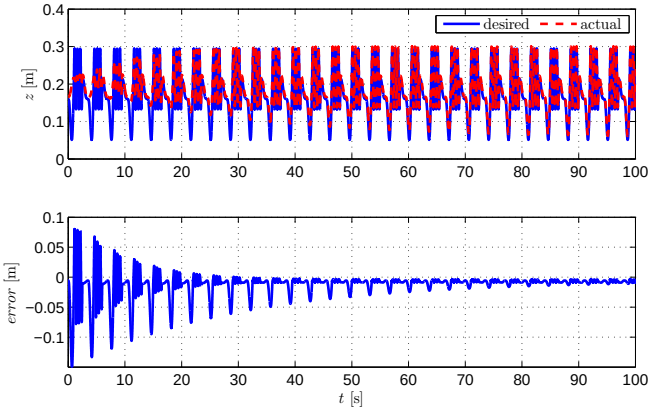


Fig. 1. Results of adaptation of one robotic arm to the other. The top plot shows the trajectories of both robots in z direction of the task. Only one robot adapted the trajectories. In the bottom plot we can see the error of adaptation over time. We can see that the error is reduced very fast and eventually also completely fades away.

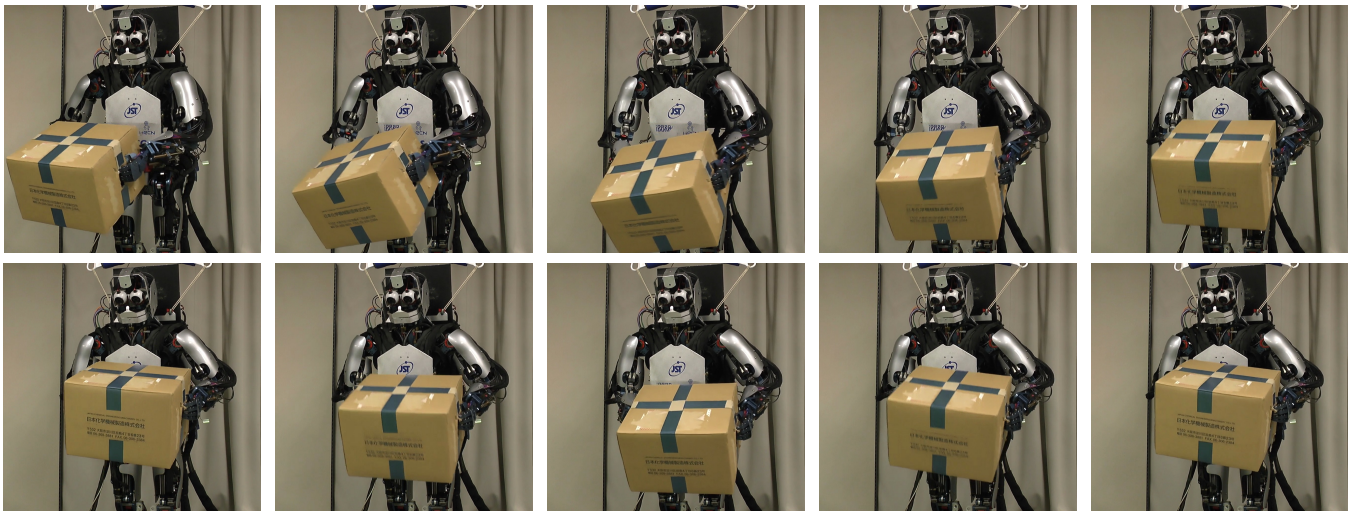


Fig. 4. Dual arm box manipulation. The movement consists of initial point-to-point movement (first three images), followed by periodic box shaking.

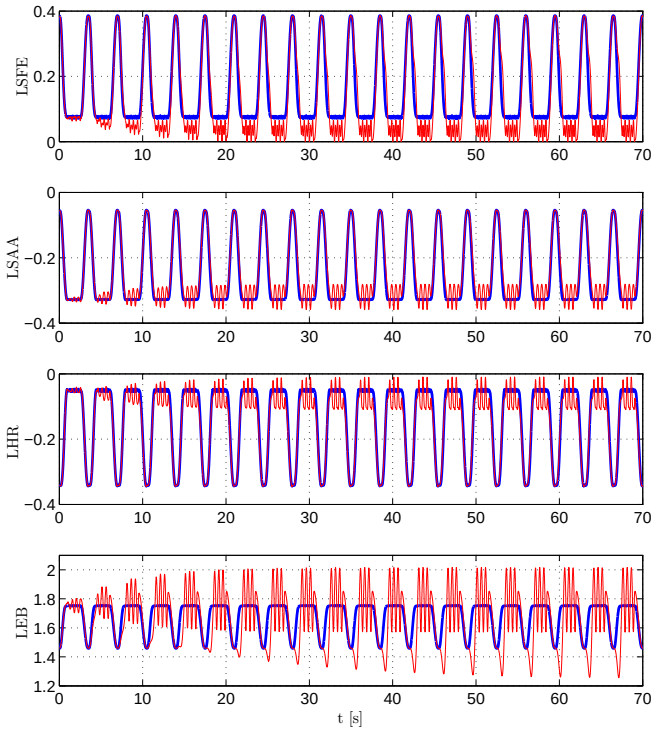


Fig. 2. Adaptation of the robotic arm in the joint space, where the modulation actually takes place. The blue lines show the original joint trajectories and the red lines the modulated trajectories for left shoulder flexion extension (LSFE), abduction-adduction (LSAA), left humerus rotation (LHR) and left elbow (LEB).

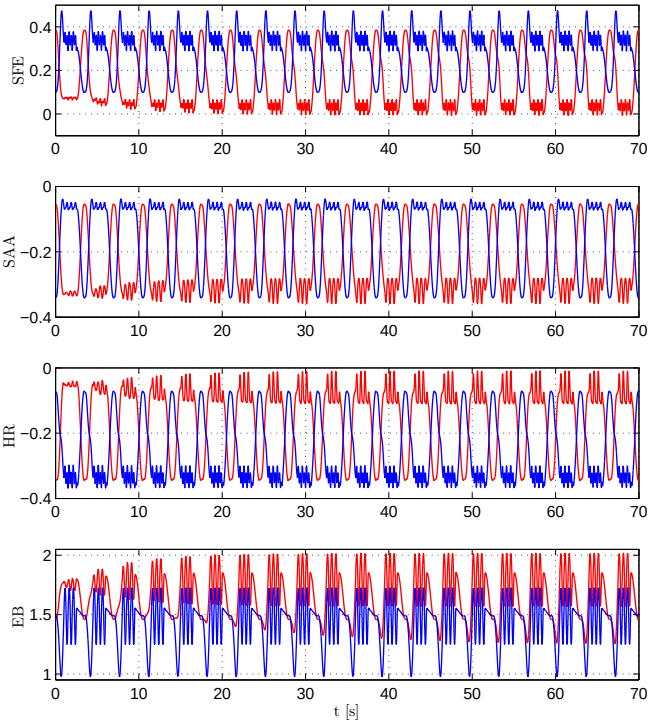


Fig. 3. Adaptation of the left arm joint trajectories in red and of the right arm trajectories in blue in all four plots.

in a uniform system simplifies the structure of the control system for such tasks.

As stated in [5], rhythmic and discrete movements are not the same from the neural point of view. This difference comes into play also when adapting trajectories as proposed in our paper. For example, the convergence of coupling term learning by ILC is different for each part. Furthermore, the iterative nature of the algorithm has led to the notion of repetitive control and no-reset ILC [21].

The blending terms a and b, which are used to encode the initial movements by DP-DMPs, are later also used to separate the discrete and periodic part of modulation terms u. The encoding of the modulation term as a combination of kernel functions for both parts completely accounts for the phase evolution, but also ensures smooth transition between the discrete and periodic part. With the introduction of tricube kernel instead of the Gaussian kernel for the blending terms, we improved on the previous implementation by

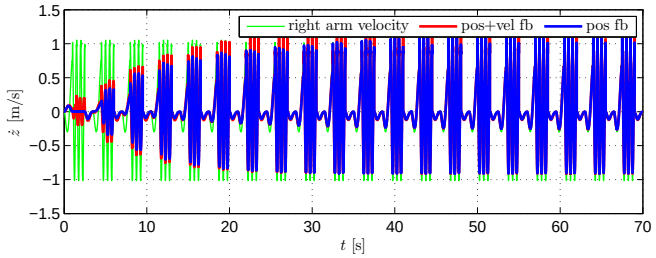


Fig. 5. The effect of using velocity-modified feedback for the ILC on the velocities. The green line shows the right arm end effector velocities. The red line shows the task space velocity adaptation when using the velocity modifed feedback as in (14), while the blue when using only position feedback (10).

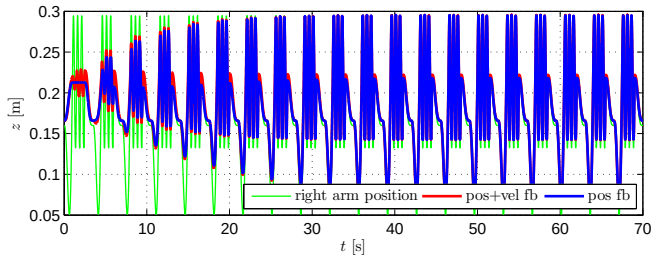


Fig. 6. The effect of using velocity-modified feedback for the ILC on the positions. The green line shows the right arm end effector velocities. The red line shows the task space velocity adaptation when using the velocity modifed feedback as in (14), while the blue when using only position feedback (10).

achieving a complete separation at the given transition point so that the effect of one part did not linger into the other.

In the paper we proposed a modified feedback within the current-iteration iterative learning control framework, taking into account the ideas developed in feedback error learning [19]. As evident from our results, this allows for faster adaptation, even though the final outcome is very similar. Something would be wrong with one of the methods if this wasn't the case. The introduction of elements of feedback error learning thus expands the realm of possible applications, but also more closely account for the actual conditions of the experiment. For the arms to move in synchrony, both position and velocity should be synchronized.

The developed approach maintains a small set of tuning parameters, as effectively K_1, K_2 in the feedback term and Q, L and C have to be specified. While the former are task-specific, the latter follow a well established ILC theory [14], where decreasing the Q (from 1) value will increase robustness but also the steady-state error. Therefore, L needs to be calculated in order to maintain the stability [14], while C is again a task-specific feedback gain.

Implementation of the algorithm in joint space – as proposed in the paper – allows for maintaining the demonstrated joint space posture, which comes specifically practical for redundant robots such as humanoids.

In summary, we have developed a modified approach for encoding combined discrete-periodic dynamic movement primitives. Current-iteration iterative learning control has

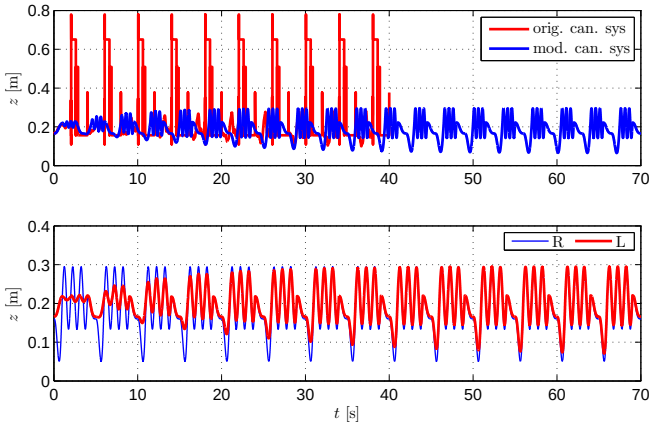


Fig. 7. Top: The results of changing the frequency of the execution to from $\Omega = 4\pi$ to $\Omega = 2\pi$ when using the original canonical system from [8], shown in red. The blue trajectory shows the behavior of the system at $\Omega = 4\pi$. Bottom: Results of changing $\Omega = 4\pi$ to $\Omega = 2\pi$ when using our proposed canonical system. We can also see that trajectory adaptation is not affected.

been applied to modulate DP-DMPs at the velocity level. By introducing the elements of feedback error learning in the learning framework, we have accelerated the convergence of the approach, thereby reducing the low number of iterations needed to achieve satisfactory results even further. In our experiments we demonstrated that the proposed approach is applicable to real-world problems such as dual arm manipulation.

In the future we intend to expand the proposed methodology to take into account variations in the frequency of motion during learning. In addition, we will focus on how to include the learning of varying coupling terms. In the context of dual-arm manipulation, the learning of varying coupling terms will enable the manipulation of boxes of varying size.

REFERENCES

[1] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[2] D. Kulić, D. Kragic, and V. Krüger, "Learning action primitives," in *Visual Analysis of Humans - Looking at People.*, 2011, pp. 333–353.

[3] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[4] A. d. Rugy and D. Sternad, "Interaction Between Discrete and Rhythmic Movements: Reaction Time and Phase of Discrete Movement Initiation During Oscillatory Movements," *Brain Research*, vol. 994, no. 2, pp. 160–174, 2003.

[5] S. Schaal, D. Sternad, R. Osu, and M. Kawato, "Rhythmic Movement Is Not Discrete," *Nature Neuroscience*, vol. 7, no. 10, pp. 1137–1144, 2004.

[6] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert, "Toward simple control for complex, autonomous robotic applications: Combining discrete and rhythmic motor primitives," *Autonomous Robots*, vol. 31, no. 2-3, pp. 155–181, 2011.

[7] M. Ajallooeian, J. van den Kieboom, A. Mukovskiy, M. A. Giese, and A. J. Ijspeert, "A general family of morphed nonlinear phase oscillators with arbitrary limit cycle shape," *Physica D: Nonlinear Phenomena*, vol. 263, pp. 41–56, 2013.

[8] J. Ernesti, L. Righetti, M. Do, T. Asfour, and S. Schaal, "Encoding of periodic and their transient motions by a single dynamic movement primitive," in *2012 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Osaka, Japan, 2012, pp. 57–64.

[9] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, 2011, pp. 365–371.

[10] T. Kulvicius, M. Biehl, M. J. Aein, M. Tamosiunaite, and F. Wörgötter, "Interaction learning for dynamic movement primitives used in cooperative robotic tasks," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1450–1459, 2013.

[11] J. Kober, D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[12] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn. Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, September 2013.

[13] A. Gams, B. Nemec, A. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[14] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[15] A. Gams, J. van den Kieboom, M. Vespignani, L. Guyot, A. Ude, and A. Ijspeert, "Rich periodic motor skills on humanoid robots: Riding the pedal racer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014, pp. 2326–2332.

[16] N. Gopalan, M. Deisenroth, and J. Peters, "Feedback error learning for rhythmic motor primitives," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013, pp. 1317–1322.

[17] H. Ben Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2831–2837.

[18] V. Koropouli, D. Lee, and S. Hirche, "Learning interaction control policies by demonstration," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 344–349.

[19] M. Kawato, "Feedback-errror-learning neural network for supervised motor learning," in *Advanced Neural Computers*, R. Eckmiller, Ed. North-Holland: Elsevier, 1990, pp. 365–372.

[20] Y. Wang, F. Gao, and F. J. Doyle III, "Survey on iterative learning control, repetitive control, and run-to-run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[21] L. Sison and E. Chong, "No-reset iterative learning control," in *35th IEEE Conference on Decision and Control*, Kobe, Japan, 1996, pp. 3062–3063.

[22] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[23] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "CB: A humanoid research platform for exploring neuroscience," *Advanced Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.

2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Online approach for altering robot behaviors based on human in the loop coaching gestures

Tadej Petrič[1,2], Andrej Gams[1,3], Leon Žlajpah[1], Aleš Ude[1,2], and Jun Morimoto[2]

Abstract— **The creation and adaptation of motor behaviors is an important capability for autonomous robots. In this paper we propose an approach for altering existing robot behaviors online, where a human coach interactively changes the robot motion to achieve the desired outcome. Using hand gestures, the human coach can specify the desired modifications to the previously acquired behavior. To preserve a natural posture while performing the task, the movement is encoded in the robot's joint space using periodic dynamic movement primitives. The coaching gestures are mapped to the robot joint space via robot Jacobian and used to create a virtual force field affecting the movement. A recursive least squares technique is used to modify the existing movement with respect to the virtual force field. The proposed approach was evaluated on a simulated three degrees of freedom planar robot and on a real humanoid robot, where human coaching gestures were captured by an RGB-D sensor. Although our focus was on rhythmic movements, the developed approach is also applicable to discrete (point-to-point) movements.**

I. INTRODUCTION

The interaction between a pupil and a teacher when learning or improving existing skills usually involves natural communication such as speech or gestures. Based on the instructions of a coach, humans can quickly learn and modify their motion patterns to achieve the desired behavior. The development of an effecting coaching system for humanoid robots is, however, a difficult task. In practice, modifying robot behaviors remains the task of experts in robotics.

Robotics researchers developed various robot coaching methods in the past decade. For example, Nakatani et al. [1] used the coach's qualitative evaluations of the robot performance to improve balancing and walking. In [2], supervised learning was combined with voice commands of a human coach, where the voice commands were used as a reward function in the learning algorithm. In [3], coaching was used on a mobile platform with the emphasis on learning high level task representations rather than motor skills. An approach that uses qualitative, verbal instructions to modify movements obtained by human demonstration was proposed in [4]. The developed system was suitable also for non-expert users. Kinesthetic teaching with iterative updates to modify a humanoid behavior was proposed in [5]. An area closely related to robot coaching is learning by demonstration, where a variety of different methods were proposed [6], [7], [8], [9], [10], [11]. However, most of the learning by demonstration methods do not address the problem of easily modifying an existing behavior to acquire a new desired outcome.

We were inspired by the efficiency of human-to-human skill transfer when developing a more effective approach to modify the existing robot behaviors. Rather than learning how to program robots, people can bring their own knowledge from interacting with each other directly into the robot domain [4]. Ideally, the human-robot interaction should focus on approaches that are intuitive for a human coach. In this paper we propose an approach for modifying existing robot behaviors based on online guidance provided by the human coach. The guidance is provided in the form of pointing gestures, i. e. the coach indicates to the robot where and how it should modify its motion. Such an interface is intuitive for humans as movement shaping through physical guidance and other means of communication is common in human motor learning [12]. It allows also non-experts to teach and alter the existing robot skills in order to obtain new desired outcomes.

A motor representation used to encode robot movements in an online coaching system must have the ability to generate smooth movements even when its parameters change online. This is important to supply an immediate feedback to the coach. Such a capability is provided by dynamic movement primitives (DMPs) [13], [14], which are defined by a set of critically damped second order linear differential equations, supplemented with a nonlinear forcing term. In this paper we focus on periodic movements [15], but the approach is fully applicable to discrete (point-to-point) movements as well. Periodic DMPs are often combined with adaptive oscillators [16]. Adaptive oscillators generate a stable limit cycle and provide the phase signal to the DMP. We assume that the initial motion pattern has been defined somehow, e. g. by kinesthetic guiding. To avoid losing postural information when using redundant robots like humanoids, the demonstrated motion pattern is encoded in the joint-space.

We developed a new DMP adaptation algorithm that can be used to modify existing motor behaviors encoded by DMPs based on human coaching gestures. The coaching gestures are specified by pointing towards the part of the movement that needs to be changed. The pointing gesture defines the direction and magnitude of change. To demonstrate the applicability of the proposed coaching approach, we implemented it both in simulation and on a real humanoid robot, where coaching gestures were obtained by Microsoft Kinect RBG-D sensor and a body tracker. The paper is organized as follows. In Section II we provide a short review of periodic DMPs. We then describe the newly developed

[1]Humanoid and Cognitive Robotics Lab & Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute (JSI), Ljubljana, Slovenia. tadej.petric@ijs.si

[2]Dept. of Brain Robot Interface (BRI), ATR Computational Neuroscience Laboratories, Kyoto, Japan

[3]Biorobotics Laboratory, Institute of Bioengineering, Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland

coaching algorithm. In Section III we analyze the properties of the proposed algorithm in simulation and in Section IV we evaluate it on a real humanoid robot. Conclusions, summary and prospective future work are explained in Section V.

II. COACHING SYSTEM

The basic framework of our coaching system consists of periodic Dynamic Movement Primitives (DMPs) combined with an adaptive frequency oscillator [16], which can extract the phase and the frequency from an arbitrary periodic signal. This framework is also called a two-layered system for movement imitation [15]. In our previous work [15], [16], we proposed a learning algorithm that can be used to extract the basic frequency from the demonstrated periodic movement, learn the waveform of one period, and reconstruct the desired waveform at an arbitrary frequency. To learn a new control policy based on the human coaching gestures, this two-layered imitation system is embedded into the proposed control framework for coaching.

A. Dynamic movement primitives combined with adaptive frequency oscillators

The first layer of the imitation system is based on adaptive frequency oscillators combined with the adaptive Fourier series. The details and the properties of the learning approach are given in [16]. In summary, an adaptive frequency oscillator is defined by a set of second order differential equations

$$\dot{\phi} = \Omega - K \cdot e \cdot \sin(\phi), \tag{1}$$

$$\dot{\Omega} = -K \cdot e \cdot \sin(\phi), \tag{2}$$

where Ω is the extracted frequency, ϕ is the phase, K is the coupling constant and e is the difference between the actual and the estimated input signal. Here we denote the input signal by v and the estimated signal by $\hat{v}$. The input signal v is the signal on which the motion pattern is synchronized. Note that once the error signal e becomes zero we obtain $\dot{\Omega} = 0$ and $\dot{\phi} = \Omega$. The estimated input signal $\hat{v}$ is represented as

$$\hat{v} = \sum_{c=0}^{m} (\alpha_c \cos(c\phi) + \beta_c \sin(c\phi)). \tag{3}$$

Here m is the size of the Fourier series. Our learning algorithm simultaneously estimates the frequency Ω and the input signal $\hat{v}$, i e. the weights α_c and β_c. See [16] for details.

We augment this first layer by anchoring the dynamic movement primitives to the phase signal ϕ of the adaptive oscillator as in [15], [16]. This makes it possible to synchronize an arbitrary trajectory to an arbitrary periodic signal congruent with the desired behavior. The basic equations of dynamic movement primitives are summarized from [14], [13], [15]. For a single degree of freedom denoted by y, which can either be one of the internal joint-space coordinates or one of the external task-space coordinates, the following system of linear differential equations with constant coefficients, augmented by a nonlinear forceing term f, has been applied to derive DMPs

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

$$\dot{y} = \Omega z, \tag{5}$$

where α_z and β_z are the positive constants, which guarantee that the system monotonically converges to the desired trajectory, g is the center of oscillation, and f is the nonlinear part that determines the shape of the trajectory. It is given by

$$f(\phi) = \frac{\sum_{i=1}^{N} w_i \psi_i(\phi)}{\sum_{i=1}^{N} \psi_i(\phi)} r, \tag{6}$$

where r is the parameter that can be used to modulate the amplitude of the movement and ψ are the Gaussian like kernel functions given by

$$\psi_i(\phi) = \exp(h(\cos(\phi - c_i) - 1)). \tag{7}$$

Here, h is the width and c_i is the distribution on one period. If not stated otherwise, in the following we used c_i, $i = 1, \ldots, 25$, and they were equally spread between 0 and 2π.

By applying the locally weighted regression the system can learn the shape of the trajectory on-line. The equations for incremental learning are summarized from [15], where the equations (4) and (5) were rewritten as one second order differential equation

$$f_d = \frac{\ddot{y}_d}{\Omega^2} - \alpha_z \left(\beta_z(g - y_d) - \frac{\dot{y}_d}{\Omega} \right). \tag{8}$$

Here the triplet of y_d, $\dot{y}_d$ and $\ddot{y}_d$ denotes the desired position, the velocity and the acceleration. To update the weights w_i of the kernel function ψ_i, we use the recursive least-squares method with the forgetting factor λ. In our experiments, the forgetting factor was set to $\lambda = 0.9995$. With the given target (8), the recursive algorithm updates the weights w_i using the following rule

$$P_i(t+1) = \frac{1}{\lambda} \left(P_i(t) - \frac{P_i(t)^2 r^2}{\frac{\lambda}{\psi_i(\phi(t))} + P_i(t) r^2} \right), \tag{9}$$

$$w_i(t+1) = w_i(t) + \psi_i(\phi(t)) P_i(t+1) r e_r(t), \tag{10}$$

$$e_r(t) = f_d(t) - w_i(t) r. \tag{11}$$

If not stated otherwise, we use $w_i(0) = 0$ and $P_i(0) = 1$, where $i = 1, ..., 25$.

In general DMPs provide a comprehensive framework for generating smooth kinematic control policies. Other important properties are: time invariance, online modulations including using a repulsive force to influence the course of the trajectory, framework for the trajectory learning, and smooth behavior in case of sudden change in the trajectory. Even though the DMP framework already posseses methods for amplitude, phase and frequency modulation, these modulations are insufficient to modify the behavior within a general coaching system. To modify the behavior online with a human in the loop, we propose an algorithm that can update

the weights of the DMP based on the coaching gestures. The goal is to provide means to generate arbitrary modifications to the available movement patterns and successfully perform the desired task. The coaching system can also be used for building a library of motion patterns, which can later be used by movement generalization methods [17].

B. Coaching with Potential Fields

The primary goal of movement modeling with dynamical systems is to exploit the coupling phenomena to generate more complex behaviors [14]. We showed in the previous section how two dynamical systems can be connected together for imitation learning of periodic movements. In this section we discuss how to modify a robot trajectory online based on the input of a human coach. An ability to modulate movement trajectories online based on the human input is a very important capability for robots that interact with humans in natural environments. The proposed algorithm can modify the robot's motion online based on the human in the loop coaching gestures and is therefore an important step towards providing such a capability.

There are different ways for adding a coupling term to modify motion patterns. For example, it can be added to the transformation system or to the canonical system, or even to both [14]. For 3-D Cartesian space movements, Hoffmann et al. [18] showed that obstacle avoidance can be achieved by adding a coupling term $\boldsymbol{C}_y$ to Eq. (4)

$$\dot{\boldsymbol{z}} = \Omega\left(\alpha_z\left(\beta_z(\boldsymbol{g}-\mathbf{y})-\mathbf{z}\right)+\boldsymbol{C}_y+\boldsymbol{f}\right), \tag{12}$$

where $\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{g}$, $\boldsymbol{C}_y$, and $\boldsymbol{f}$ are in this case three dimensional values. In [18] this equation was used to drive the robot's behavior and ensure obstacle avoidance. In our case we intend to modify the behavior permanently, therefore we use this term as input to the recursive least-squares method for updating the weights of the canonical system. Since in this case the reference trajectory is simply the output of the DMP (there is no training signal y_d, $\dot{y}_d$ and $\ddot{y}_d$), $e_r(t)$ as defined in Eq. (11) would be equal to zero if the DMP equations had not changed. However, since the differential equation (4) was changed to (12), there is an additional coupling term $\boldsymbol{C}_y$, which was not accounted for during training. Thus Eq. (11) transforms into

$$e_r(t) = C_{y,j}(t), \tag{13}$$

where $C_{y,j}(t)$ is the coupling term for the degree of freedom denoted by j.

A proper definition of the coupling term $\boldsymbol{C}_y$ is crucial and of course task dependent. To enable coaching by human gestures, we modified the obstacle avoidance coupling term $\boldsymbol{C}_y$ from [18] as follows

$$\boldsymbol{C}_y = \gamma\ \mathrm{s}(||\boldsymbol{o}-\boldsymbol{x}||)\ \exp(-\beta\phi)\ \boldsymbol{d}, \tag{14}$$

Here $\boldsymbol{x}$ is the Cartesian position of the end-effector, $\boldsymbol{o}$ is the center position of the perturbation potential field (defined by hand position), $\boldsymbol{d}$ is the perturbation direction (defined by the pointing gesture), γ and β are the scaling factors, ϕ is given by

$$\phi = \arccos\left(\frac{(\boldsymbol{o}-\boldsymbol{x})^T\dot{\boldsymbol{x}}}{||(\boldsymbol{o}-\boldsymbol{x})||\ ||\dot{\boldsymbol{x}}||}\right). \tag{15}$$

$\mathrm{s}(r)$ is defined as

$$\mathrm{s}(r) = \frac{1}{1+e^{\eta(r-r_m)}}, \tag{16}$$

where η is the scaling factor and r_m the distance at which the perturbation field should start affecting the robot's motion. A one degree of freedom example for the coupling term is shown in Fig. 1.

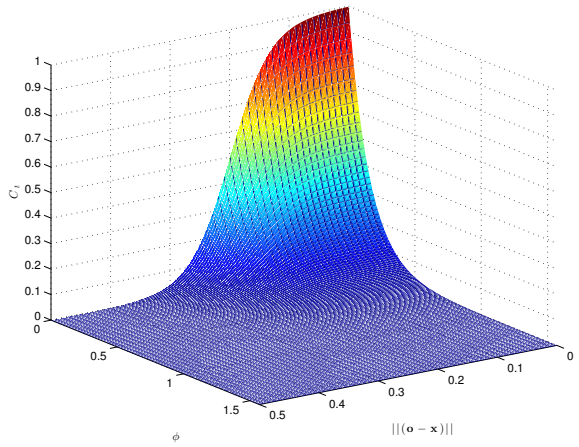


Fig. 1. One degree of freedom example coupling term C_y with parameters $\eta = 30$, $r_m = 0.2$, $\gamma = 1$, and $\beta = -20/\pi$.

To update the trajectories in joint-space while they are perturbed in task space, where the coupling term is denoted by $\boldsymbol{C}_y$, a pseudo inverse of the task Jacobian is used. This essentially maps the task space velocities into the joint space velocities with $\dot{\boldsymbol{q}} = \mathbf{J}\dot{\boldsymbol{x}}$. By applying a similar transformation to $\boldsymbol{C}_y$ we obtain

$$\boldsymbol{C}_q = \mathbf{J}^{\dagger}\boldsymbol{C}_y. \tag{17}$$

where $\boldsymbol{C}_q = [C_{q,1}\ C_{q,2}\ \ldots\ C_{q,k}]^T$ and k is the number of the robot's degrees of freedom. The components of (17) are now inserted into (13), which is used for updating the DMP weights w_i using (9) and (10). In this way we ensure that the joint space trajectories encoded by the DMPs are properly modified according to the coach's instructions.

Keeping the movement representation in the joint space is beneficial because our initial movement trajectories, which are encoded by DMPs, are usually acquired by kinesthetic guiding. By using joint space trajectories we avoid losing information about the selected robot configuration during human guiding on a redundant robot. Hence the DMP representation should remain in the joint space and the behavior should be modified there.

III. SIMULATION RESULTS

To show the properties of the proposed approach, we first applied it to a simulated 3 degrees of freedom planar robot. The robot was simulated using Planar Manipulator Toolbox [19]. The initial joint space trajectory was defined such that it produced a circular motion in the task space. The frequency of motion was constant and it was set to 0.5 Hz. If not stated otherwise, the coaching parameters were $\gamma = 100$, $\eta = 20$, $r_m = 0.35$, and $\beta = -8/\pi$.

Fig. 2 shows simulation results where the coaching point was defined with the parameters $\boldsymbol{o} = [1.8, -0.6]$ $\boldsymbol{d} = [0,1]^T$. The coaching command was inserted at the selected position after 5 seconds. On the right plot we can see the evolution of the task space trajectory. Here the red line shows the initial task space motion and the green line the task space motion after coaching. The grey lines show the evolution of DMP modification in time. It can be seen that the coaching command was only acting at the desired location and therefore it did not affect the rest of the initial trajectory. This can also be seen in the bottom plot left where the scale of the coupling term used in recursive least squares is shown. Here we can see that the coupling term only acts when the end-effector is close to the perturbation point. In addition, we can see that the coupling term is iteratively converging towards zero. The first three plots on the left show the joint trajectories in time. We can see that they are modified only when the end-effector is near to the perturbation point and that they remain smooth.

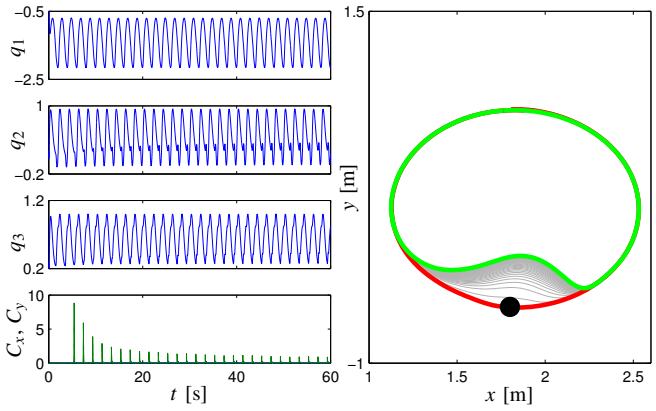


Fig. 2. Simulation results where the circular motion in task space was pushed in. Coaching command parameters were $\boldsymbol{o} = [1.8, -0.6]$ and $\boldsymbol{d} = [0,1]^T$. The coaching point was activated after 5 seconds.

Similar results can also be observed in Fig. 3, where the initial task space trajectory was pushed out. In this case the coaching point parameters were $\boldsymbol{o} = [1.8, 0.8]$ $\boldsymbol{d} = [0,1]^T$. Again the coaching point was inserted after 5 seconds. We can see the same basic performance as in the case of Fig. 2. This two study cases clearly show that we can easily modify the task space behavior at the desired point to achieve the desired course of movement, even though the trajectories are encoded in the joint space. These two case studies show that we can smoothly and iteratively modify the task space behavior in an arbitrary direction.

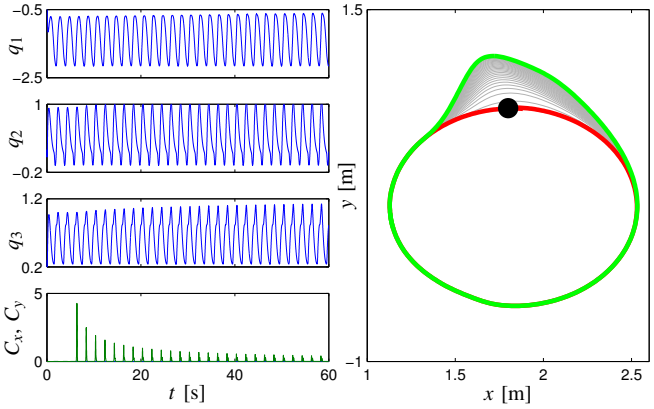


Fig. 3. Simulation results where the circular motion in task space was pushed out. Coaching command parameters were $\boldsymbol{o} = [1.8, 0.8]$ and $\boldsymbol{d} = [0,1]^T$. The coaching command was activated after 5 seconds.

To further support the last statement we show in Fig. 4 an experiment where the coaching command is kept at the constant distance to the end-effector. In other word, the perturbation point moves along the trajectory and the perturbation direction $\boldsymbol{d}$ is in this case focused towards the centre of the circle. The coaching begins after 5 seconds. Here we can see in the right plot that the motion is constantly modified and directed towards the center of the circle. The final task space trajectory is indicated with the green line and the initial trajectory with the red line. In the first three plots left we can also see that as expected, the amplitude of motion is decreasing for all three joints.

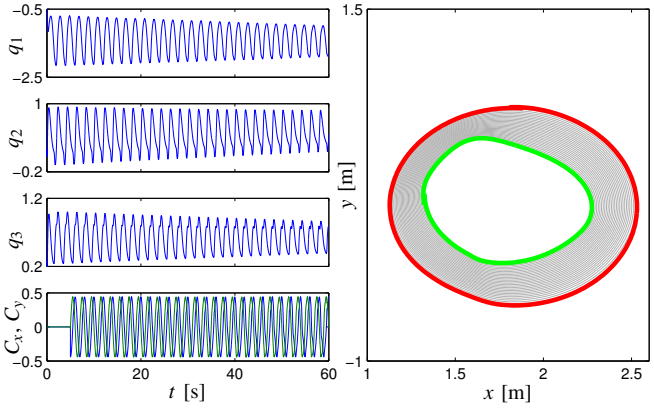


Fig. 4. Simulation results where the circular motion in task space was pushed in all the time, i. e. the direction of coaching command was towards the center of the circle all the time. The coaching began after 5 seconds.

IV. ROBOT EXPERIMENTS

To show the applicability of the proposed approach in real world, we implemented it on the JST-ICORP/SARCOS humanoid robot CBi [20]. We used the Microsoft Kinect sensor and the associated body tracker to capture human coaching gestures [21]. Fig. 5 shows the experimental setup, where the body tracking results can be seen on the display in the background.

To acquire the human coaching gestures in the coordinate system of the robot, we calibrated the Microsoft Kinect sensor to the robot base coordinate system. To obtain the appropriate transformation matrix, we recorded at least four pairs of points in both coordinate systems. For this purpose the human coach placed his hand at the same location as the robot's end-effector and the position of the human hand and the robot's end-effector were measured in the Kinect's and robot base coordinate system, respectively. The transformation matrix was calculated using least-squares fitting of two points set as described in [22].

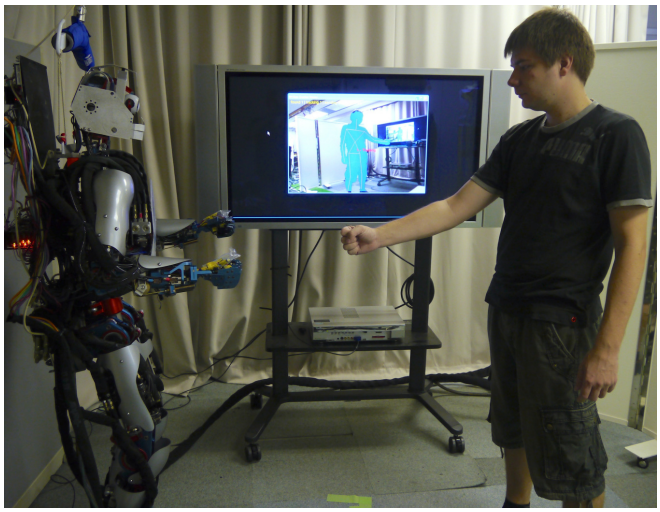


Fig. 5. Experimental setup, where a human coach is modifying the robot's motion. The human coaching gesture is captured using Microsoft Kinect sensor.

To make coaching as intuitive as possible, we developed an interface where the human coach can modify the trajectory by either pushing it away from him using his right hand or attracting it towards him with his left hand. The coaching direction was calculated using the wrist and the elbow location. For the right hand, which pushes the trajectory away form the coach, the direction is given by

$$\boldsymbol{d}_R = \frac{\boldsymbol{x}_{w,R} - \boldsymbol{x}_{e,R}}{||\boldsymbol{x}_{w,R} - \boldsymbol{x}_{e,R}||}, \tag{18}$$

where the $\boldsymbol{x}_{w,R}$ and the $\boldsymbol{x}_{e,R}$ are the Cartesian positions of the right hand wrist and the right hand elbow in the robot's base coordinate system. For attracting the trajectory towards the coach, the direction is given by

$$\boldsymbol{d}_L = \frac{\boldsymbol{x}_{e,L} - \boldsymbol{x}_{w,L}}{||\boldsymbol{x}_{e,L} - \boldsymbol{x}_{w,L}||}. \tag{19}$$

Here $\boldsymbol{x}_{w,L}$ and the $\boldsymbol{x}_{e,L}$ are respectively the Cartesian positions of the left hand wrist and the left hand elbow in the robot's base coordinate system.

Since Microsoft Kinect sensor relies on depth information and our humanoid robot has similar body proportions as a human, the body tracker sometimes becomes confused if the human approaches the robot very closely. For this reason the human coach did not approach the robot too closely in our experiments. Instead, the center of the potential field generated by each hand was moved slightly away from the respective hand. For the right hand, the origin of the potential field defined by the coaching gesture was moved in the direction of the coaching gesture

$$\boldsymbol{o}_R = \boldsymbol{x}_R + \xi_R \boldsymbol{d}_R, \tag{20}$$

where ξ_R is the scalar that defines the distance between the hand and the center of the coaching point in the direction of $\boldsymbol{d}_R$. Similar equation is used also for the left hand which attracts the trajectory towards the hand.

$$\boldsymbol{o}_L = \boldsymbol{x}_L - \xi_L \boldsymbol{d}_L. \tag{21}$$

Here, the effective coaching point is moved in the opposite direction of perturbation $\boldsymbol{d}_L$. With such modifications the effective origins of potential fields are always in front of the human hands in the direction of pointing at the distance defined by ξ_R and ξ_L.

To determine which hand is active, we use the distance between both wrist positions $\boldsymbol{x}_{w,L}$, $\boldsymbol{x}_{w,R}$ and the robot's end-effector position $\boldsymbol{x}$. The active hand is the one which is closer to the robot's hand position.

To show the applicability of the interface for online modification of the initial rhythmic movement using human in the loop coaching gestures, we first provide an example of pulling-in the task space trajectory. The parameters were set to $\gamma = 10$, $\eta = 10$, $r_m = 0.15$ and $\beta = -10/\pi$. Fig. 6 shows the task space motion of the robot's end-effector in the $x-y$ plane. We can see a successful modification of the motion based on the human coaching gestures. In Fig. 7 we show the corresponding joint space trajectories as a function of time. The teaching of the new motion pattern begins after 5 seconds, which is indicated with the first vertical line. We can see that the joint space trajectory was modified successfully to achieve the desired task space motion. In Fig. 7 we can see that at approximately 50 seconds the human coach stopped

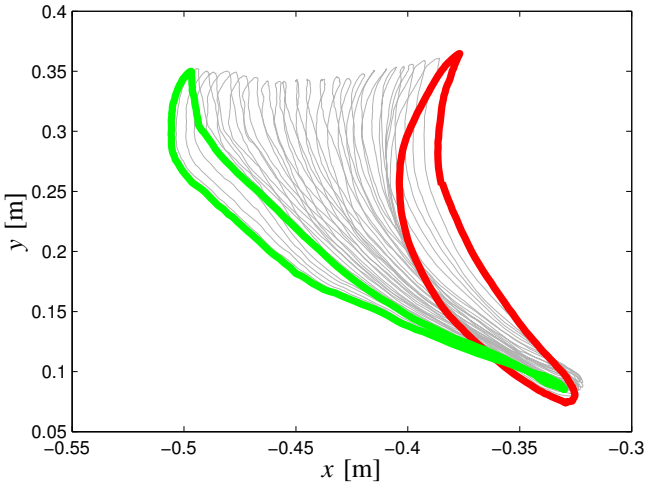


Fig. 6. Task space motion of the robot's end-effector, where human coach was modifying the motion pattern. The initial trajectory is in red and the final trajectory is in green. The time evolution of the trajectory modification is indicated with grey line.

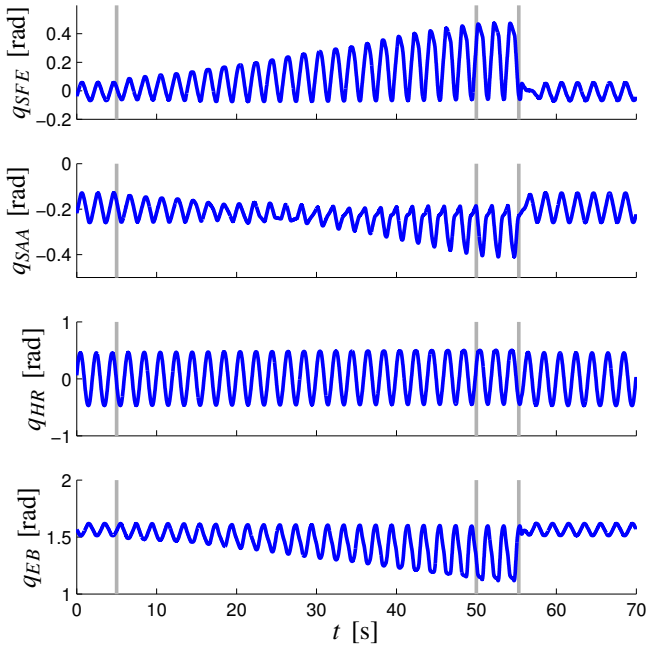


Fig. 7. Joint space motion in time of the robot's right hand, while coaching. Vertical lines indicate the important events described in text.

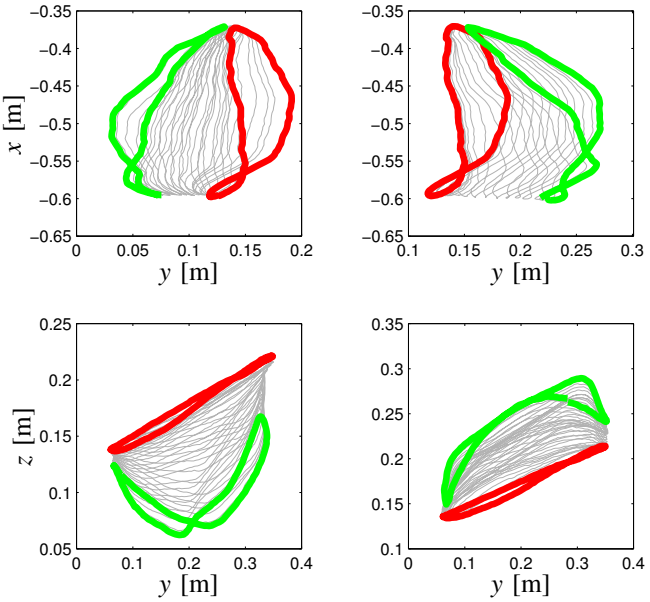


Fig. 9. Four different modifications of the original motion, which are also shown in the supplemental video. The top left graph corresponds to example 1, the top right graph to example 2, the bottom left graph to example 4 and the bottom right graph to example 5 in the supplemental video.

modifying the behavior and at approximately 55 seconds the new motion pattern was switched back to the original motion pattern. At this point the difference between original motion trajectory and the modified motion trajectory is even more evident. The snapshots showing the original and the modified trajectory of the humanoid arm movement are shown in Fig. 8. This experiment is also shown in the supplementary video.

Fig. 9 shows four different modifications of the original motion. In the top row we can see the horizontal pushing and pulling of the motion in the x-y plane and in the bottom row the vertical pushing and pulling in y-z plane. As we can see, the human coach was successful at modifying the movement of the robot in the desired direction using either the pushing or pulling technique, i. e. using either the right or the left hand to define the coaching gesture.

V. CONCLUSION AND FUTURE WORK

In this paper we developed a new coaching methodology that makes use of coaching gestures to modify an existing movement encoded by a periodic DMP. The DMP modification method is based on a recursive least-squares technique for updating the weights of periodic DMPs. With the developed system a human teacher can iteratively modify the previously acquired trajectories. It operates online and can therefore provide an immediate feedback to the coach. We presented simulation case studies where we successfully modified the joint space trajectories to obtain the new desired task space motions. The same method was also applied to the JST-ICORP/SARCOS humanoid robot CBi, where the human coach modified the humanoid robot's behavior to obtain the desired outcome. The proposed approach can easily be extended to discrete DMPs.

The main limitation of the coaching interface was the inability of the body tracker to distinguish between the robot and the human arm when a human teacher was close to the robot. Although the use of Microsoft Kinect sensor is beneficial because it can be used without much preparations, i. e. no markers or other special equipment is necessary, we believe that marker-based systems with more accurate tracking would provide a better and more accurate interface to modify the humanoid robot's movements. With a more reliable tracking of human coaching gestures, we could achieve similar results on the real robot as showed in Fig. 4, which is based on simulated data. The implementation and evaluation of the proposed algorithm with a more accurate body tracking system is an important goal of our future work. On the other hand, it is important that the human interface stays as intuitive as possible.

ACKNOWLEDGMENT

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 agreements no. 270273, Xperience and no. 600716, CoDyCo. It was also supported by MEXT KAKENHI Grant Number 23120004; by JSPS and SRA: Japan-Slovenia research Cooperative Program; by MIC-SCOPE; by JST-SICP; by SRPBS, MEXT; by contract with the Ministry of Internal Affairs and Communications entitled 'Novel and innovative R&D making use of brain structures'.

REFERENCES

[1] M. Nakatani, K. Suzuki, and S. Hashimoto, "Subjective-evaluation oriented teaching scheme for a biped humanoid robot," in *IEEE-*

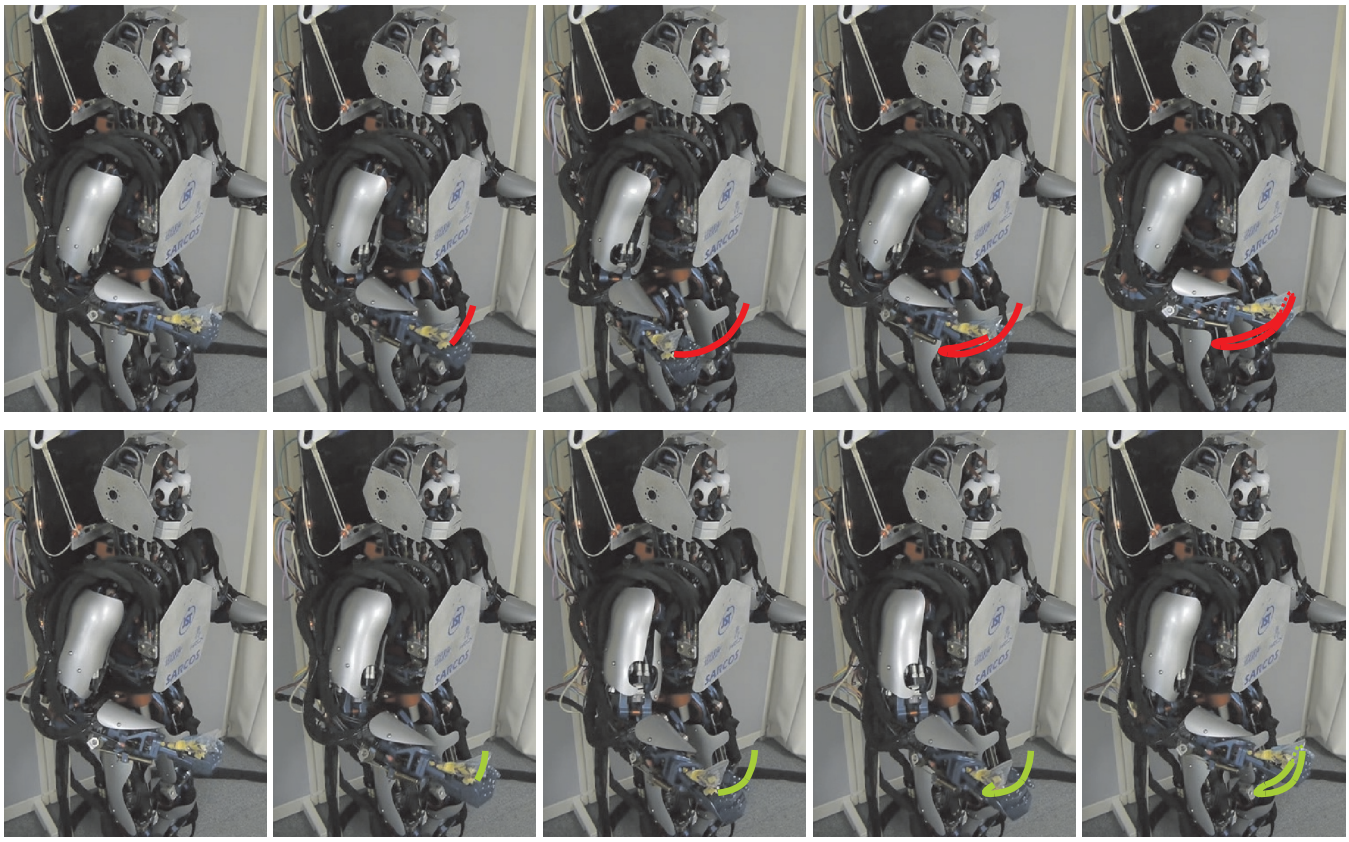


Fig. 8. A sequence of still photos showing the original motion in the top row and the final modified motion in the bottom row. The photos frame rate is 0.4 per second.

RAS International Conference on Humanoid Robots (Humanoids), Karlsuhe, Germany, 2003.

[2] A. Gruebler, V. Berenz, and K. Suzuki, "Coaching robot behavior using continuous physiological affective feedback," in *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Bled, Slovenia, 2011, pp. 466–471.

[3] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003, pp. 241–248.

[4] M. Riley, A. Ude, C. Atkeson, and G. Cheng, "Coaching: An approach to efficiently and intuitively create humanoid robot behaviors," in *2006 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Genoa, Italy, 2006, pp. 567–574.

[5] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2-3, pp. 115–131, 2011.

[6] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[7] A. Billard and K. Dautenhahn, "Experiments in learning by imitation – grounding and use of communication in robotic agents," *Adaptive Behavior*, vol. 7, no. 3-4, pp. 415–438, 1999.

[8] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 93–108, 2004.

[9] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 183–202, 2008.

[10] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer Verlag, 2008.

[11] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[12] R. Schmidt and T. Lee, *Motor Control and Learning: A Behavioral Emphasis*. Champaign, IL: Human Kinetics Publishers Ltd., 2011.

[13] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control – a unifying view," *Progress in Brain Research*, vol. 165, pp. 425–445, 2007.

[14] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[15] A. Gams, A. J. Ijspeert, S. Schaal, and J. Lenarčič, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Autonomous robots*, vol. 27, no. 1, pp. 3–23, 2009.

[16] T. Petrič, A. Gams, A. J. Ijspeert, and L. Žlajpah, "On-line frequency adaptation and movement imitation for rhythmic robotic tasks," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1775–1788, 2011.

[17] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[18] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009, pp. 2587–2592.

[19] L. Žlajpah, "Simulation in robotics," *Mathematics and Computers in Simulation*, vol. 79, no. 4, pp. 879–897, 2008.

[20] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "CB: A humanoid research platform for exploring neuroscience," *Advanced Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.

[21] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[22] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.