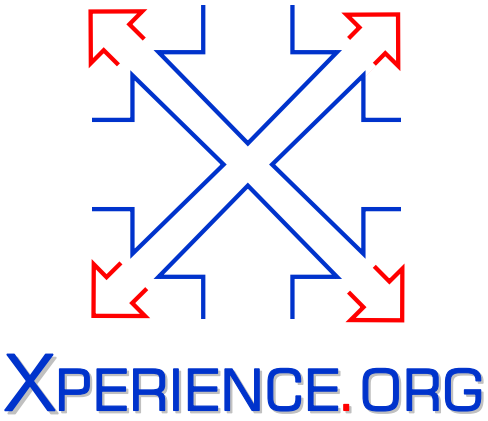

| | |
|---|---|
| Project Acronym: | Xperience |
| Project Type: | IP |
| Project Title: | Robots Bootstrapped through Learning from Experience |
| Contract Number: | 270273 |
| Starting Date: | 01-01-2011 |
| Ending Date: | 31-12-2015 |




| | |
|---|---|
| Deliverable Number: | D2.2.3 |
| Deliverable Title: | Transfer of Motor Actions: Technical report or scientific publication on how to use the developed representations of motor actions within the architecture and in the final demonstration. |
| Type (Internal, Restricted, Public): | PU |
| Authors: | Aleš Ude, Tamim Asfour, Andrej Gams, Bojan Nemec, Mirko Wächter, Martin Do |
| Contributing Partners: | JSI, KIT |

| | |
|---|---|
| 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 General Objective of WP2.2: Motor Actions

*From the proposal*: WP2.2 is primarily concerned with how to learn and obtain complex action sequences and how to organize and structure the acquired data to 1) generate advanced motor behaviours, e.g. by blending and sequencing of behaviours in the available data sets, and 2) interact with higher-level cognitive processes that mainly use discrete representations, thus providing the bridge for planning to access representations at the sensorimotor level and vice versa.

Methods like imitation learning, reinforcement learning and other exploratory approaches, which have been shown to be successful at the acquisition of motor knowledge, will be considered for implementation.

## 1.2 Contents of the Deliverable

**Transfer report on the implementation of WP2.2**
Here we describe how the developed motor learning algorithms were transferred to the main demonstration platform of the Xperience project, i. e. the humanoid robot ARMAR-III. This work is based on the DMP framework [6] with many extensions that were developed in the course of the project [2, 3, 4, 1].

**Scientific contribution**
The results and publications generated in WP2.2 in Year 4 of the Xperience project are described. These works provide more information about the representations and learning techniques that were developed to drive the Xperience demonstration platforms and include 1. Learning and adaptation of motor actions, 2. Integration of force-based exploration with coaching, and 3. Hierarchical segmentation of manipulation action sequences. Earlier work in Xperience in this area includes the integration of periodic and discrete DMPs [2] (reported in D2.2.1), representations for dual-arm cooperative DMPs [7, 5, 4] (reported in D4.1.2), and new learning approaches for statistical generalization [3] (reported in D4.1.1).

# Chapter 2

# Transfer to the Demonstration Scenarios

## 2.1 Implementation of Dynamic Movement Primitives

Two different computer environments were used for the implementation, namely Modular Controller Architecture framework (MCA) and ArmarX.

**Implementation in Modular Controller Architecture**
Transfer of the abilities to adapt dynamic motion primitives (DMPs) based on external feedback, and the learning of the said feedback to be included in the non-changing environment was implemented on the ARMAR-III robot. The transfer also included the implementation of the motion primitives themselves, for both discrete and periodic implementations, combined with the means to autonomously determine the basic frequency of a periodic input signal, based on a pool of adaptive frequency oscillators or on a single adaptive frequency oscillator and an adaptive Fourier series.

As Modular Controller Architecture framework (MCA) is an open source modular network transparent and real-time capable C/C++ framework for controlling robots and other kinds of hardware, all methods and classes within are implemented by C++ modules with standardized interfaces for easy integration of software components.

The transfer itself consisted of adapting the source code in the control algorithms, forming a scenario as a part of the Robot Interface which is built on top of the MCA. Because the implemented transfer consisted of different, not directly related tasks, there were several scenarios. One comprised the transfer of the motion from a user to the robot through imitation of the periodic signals. The underlying algorithms detected the basic frequency and isolated a single period of motion for the representation in DMPs. Another scenario allowed the adaptation of the DMPs, in this case discrete DMPs, to the measured external force signal, and learn this signal to be used in subsequent executions in order to achieve the desired behavior.

It is important to note that by using the scenario abstraction level, the implementation comprises several tasks and skills in a single, concluded module. The user can thus re-run or adapt the scenario without explicit knowledge of the underlying programming code and does not even have to be aware in which way the information is communicated to and from the relevant component of the robot.

The implementation of the underlying code in C++ is in a class, i.e. *CInteractiveDMPScenario class*, which runs in a loop in the the background, essentially numerically integrating the differential equations of the DMPs, combined with the feedback of external signals. The following list includes the implemented C++ classes and the skills they comprise

- DMPstructure
  Implementation of discrete and periodic DMPs, generalization using gaussian Process Regression
- CPeriodicDMP (Periodic DMP scenario)

Implementation of the periodic DMPs, extraction of the basic frequency of periodic signals, adaptation of the motion to achieve desired forces of non-rigid contact with the environment (wiping).

- CInteractiveDMP
  Adaptation of DMPs to external force signals, i.e., measured forces, allowing for adaptation to the environment, other robots and even humans, iterative learning control.

**Relation to demonstration Scenarios**

*Scenario 1 (prepare a salad)*: The following tasks will apply skills transferred to the ARMAR robot in the context of Scenario 1:

- Moving of the salad bowl: Interactive DMPs for bimanual tasks

*Scenario 2 (rearrange the room and setup the table)*: The following tasks will apply skills transferred to the ARMAR robot in the context of Scenario 2:

- Positioning the table in the room by pushing/carrying it, jointly together with a human (or a robot): Interactive DMPs for bimanual tasks
- Wipe the table: Periodic DMPs and adaptation to external force signals
- Two robots should carry a narrow carpet table runner to put it on the table: Interactive DMPs for bimanual tasks

For the demonstration of the capabilities of the algorithm we implemented two tasks. In the first we implemented table wiping on two different robotic platforms as shown in Figure 2.1.

In the second demonstration we executed a bimanual task, where each arm was controlled separately, without a central controller. The arms synchronized their motions to achieve a common height and distance from the robot, allowing for the manipulation of a larger object. Figure 2.2 shows the final positions after separate epochs.


Figure 2.1: Table wiping using the KUKA robotic platform at JSI on the left and using the ARMAR humanoid robot on the right.

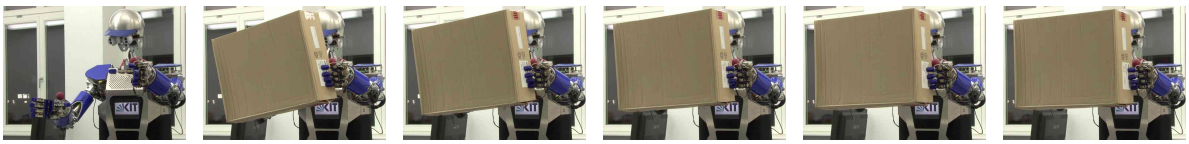


Figure 2.2: Images showing the final position of the box in the bimanual manipulation task with the ARMAR-III robot. The first image shows the position of the arms after the execution of the original, uncoupled trajectories. Note that they are at different heights. The following images show the end positions of the box, held with the arms, after each epoch. The final image shows the position after the learning. Note that the arms have reached almost the desired, parallel position and therefore the box is not tilted anymore.

## 2.2 Implementation of Chair Pushing

**Implementation in ArmarX**

The aim of this scenario is to push an office chair with wheels under the table at the specified location, as shown in Figure 2.3. In this scenario we consider the possibility that the robot can not approach the chair from the back side, which is the preferable pose for pushing. The chair and the table pose as well as an eventual obstacle pose were detected using ARMAR foveal vision and therefore expressed relative to the robot base coordinate system. In order to simplify the visual recognition, we put color markers on the chair, table and the obstacle. These will be removed later as soon as more advanced visual algorithms are ready to use. We also assume that the robot possess environment (kitchen) map, relative to the current robot position. The algorithms for performing chair pushing scenarios are based on methods for representing of robot actions with DMP-s (see D2.1.1, D2.1.2, D2.1.3, D4.1.1, D4.1.2 and D4.1.3.). All algorithms were implemented in C++ using the new programming environment ARMAR-X.

The robot development environment ArmarX provides an infrastructure for developing a customized robot framework that allows to realize distributed robot software components. The framework comprises distributed communication functionality. Separate distributed components, i.e., customizable building blocks, can be implemented using either Python, C++, C# or Java (see [10]). By using ArmarX customizable building blocks for high level robot control, the backbone of the robots software architecture was implemented.

We customized several blocks for the chair-pushing task. Each block contains a state of the robot and the environment and is subject to defined inputs. Inputs and outputs of the states define also their relations.

Separate states, that is the backbone of the robot's software architecture, are first created and later connected in Statechart Editor. Statechart Editor automatically creates a template in C++ for the underlying program code, which describes the behavior of the state and as such also the robot.

**Relation to demonstration scenarios**

*Scenario 2 (rearrange the room and setup the table)*: The following tasks will apply skills transferred to the ARMAR robot in the context of Scenario 2:

- Positioning the table in the room by pushing/carrying it, jointly together with a human (or a robot): ChairPushing Scenario

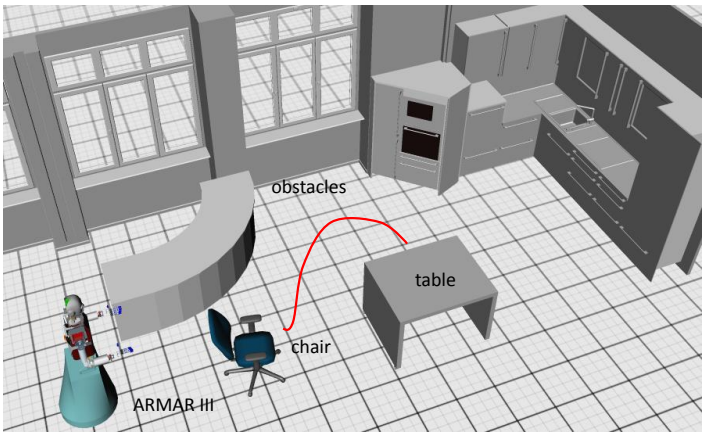



Figure 2.3: Chair-grasping scenario environment scheme. Red line denotes the collision-free chair pushing trajectory.

The basic task is subdivided in the following subtask, implemented by the following states in Statechart Editor:

- Chair localization (*LocalizeChair.**)
- Compute target robot pose (*ComputeTargetRobotPose.**)
- Approach the chair (*ApproachChair.**)
- Grasp the chair (*GraspChair.**)
- Pushing the chair to the final pose (*PushChair.**)

After the localisation of the chair, the local map is fitted to the global kitchen map, which enables to localize static obstacles such as kitchen walls and kitchen appliances. Next, the reachability maps, discretized representations of the workspace which are enriched by quality information related to reachability, manipulability for different end-effectors [9], are calculated for the desired grasp (see Figure 2.4). Based on these maps, the optimal target position for grasping is calculated and a DMP based trajectory is generated. During the chair approach phase, robot arms are kept in the initial position.

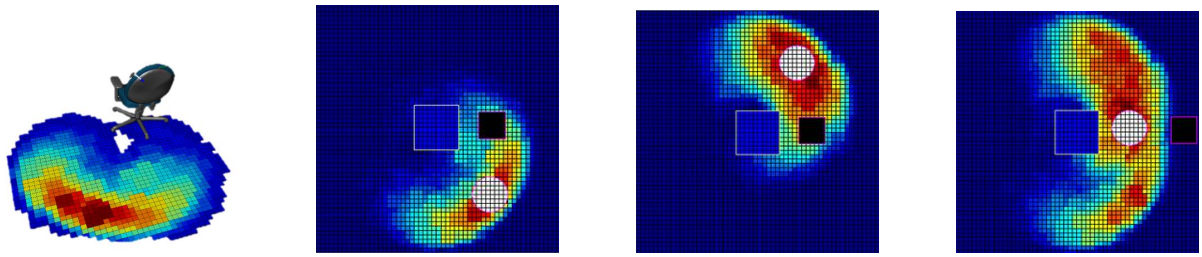


Figure 2.4: a) Reachability map for grasping the chair with the right rand. Dark red regions denote maximal manipulability index and dark blue minimum manipulability index. Optimal position is the position with maximal manipulability without violating collision constraints, not shown here. b) Blue square denote chair location, black square obstacle position. White circle shows the robot. Only grasp with left hand is feasible in this case. c) Only grasp with right hand is feasible. d) The robot can grasp the chair with both hands.

An instance of chair grasping with visual servoing is show in Figure 2.5.

Before the chair pushing phase, the robot first localizes target position at the table using foveal vision. At this point, collision free trajectory is generated and encoded as DMPs. During the chair pushing with single arm, the robot is treated as a kinematically redundant mechanism with 13 DOF composed of the robot base (3 DOF), robot torso (2 DOF) and the corresponding arm (7 DOF). Weighted damped pseudo-inverse was used to resolve the redundancy and singularity. The corresponding weighing matrix was set to $[100 * I_3, 10 * I_2, I_7]$, where $I_n$ denotes the identity matrix of dimension $n \times n$. Since the one handed grasp is not firm enough, the actual chair trajectory deviates from the desired one. Therefore, it needs to be controlled. We implemented control as schematically outlined in Figure 2.2. Results of the experimental evaluation of the tracking the chair position during the pushing is shown in Figure 2.7.

In the case of grasping the chair with both hands, we need to control only the robot base during the platform. Due to the limited workspace of both arms, the null space is very close to zero and we can treat the robot as non-redundant keeping both arms in fixed position.

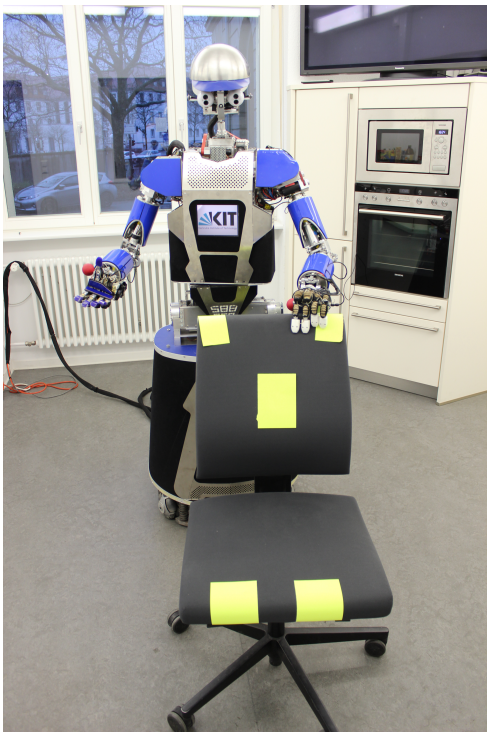


Figure 2.5: ARMAR III grasps chair using visual servoing. Yellow spots are markers which are used in early implementations, but will be removed in a later stage as soon as more advanced visual recognition and localization approaches are ready for integration in the final demonstration

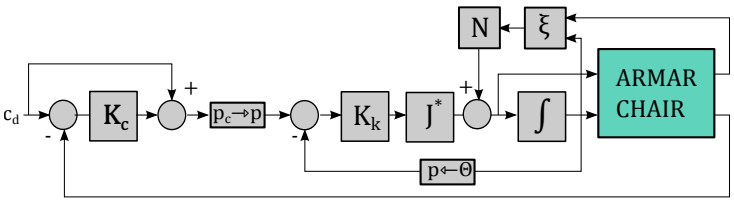



Figure 2.6: Block scheme of the chair control and kinematic control of the ARMAR III. $K_c$ and $K_k$ are the gains of the chair position controller and the kinematic control. $J^*$ denotes the robot Jacobian, $N$ denotes mapping to the Jacobian null space, $\zeta$ are the null space commanded velocities, set to 0 in all of our experiments. $p_c \rightarrow p$ denotes kinematic mapping from the chair coordinates to the robot hand coordinates and $p \leftarrow \theta$ denotes mapping from joint coordinates to the Cartesian coordinates

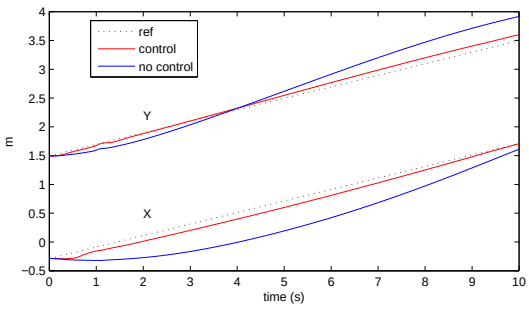



Figure 2.7: Results of the chair pushing along the straight line with and without the chair controller

# Chapter 3

# Scientific Results

In this section we report on scientific work in Year 4 that supports the activities targeting the final demonstration of the Xperience project.

## 3.1 Learning and Adaptation of Motor Actions

In the first years of the project we extended the framework of dynamic movement primitives [6] with many new features. We continued this work also in the fourth year of the project. In particular, we studied how to improve the adaptation speed and robustness of iterative learning mechanisms applied to robots controlled with force feedback [NPUed, GvdKV+14]. The work reported in this section is an extension of the work on bootstrapping from Semantic Event Chains similarities („wipe“and „stir“) to define the motion pattern for „stir“starting from „wipe“, which was reported in Deliverable D3.1.2 and in [12].

In the attached two papers we propose a new learning controller that combines the ideas of standard iterative learning control (ILC), repetitive control (RC), and on-line coaching [8] to adapt the initially available motion into a new movement with respect to the constraints of the task. The structure of the proposed algorithm is closely related to the DMP framework. It is general and can be used with both types of movements that can be represented by DMPs, i. e. discrete and periodic movements. The discrete variant is called Recursive Regression Iterative Learning Control (RRILC) and the periodic variant Recursive Regression Repetitive Control (RRRC).

The performance of the proposed algorithms was evaluated and compared with standard ILC and RC schemes, both in simulation and with real robots. We performed two experiments, one showing the performance of the algorithm on a discrete task and the other on a periodic task, respectively.

RRILC and ILC were evaluated on task of force-based surface following (see Fig. 3.1 and 3.2. It was evident in our experiments that RRILC outmatches the ILC, especially in the first cycle. Since unlike

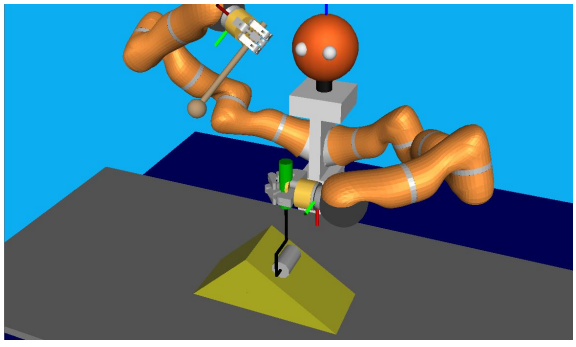


Figure 3.1: Simulated environment for surface following

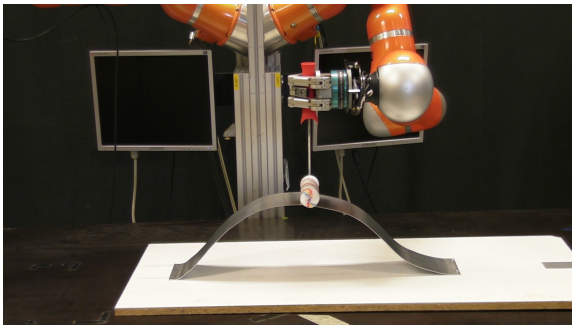


Figure 3.2: Real environment for surface following

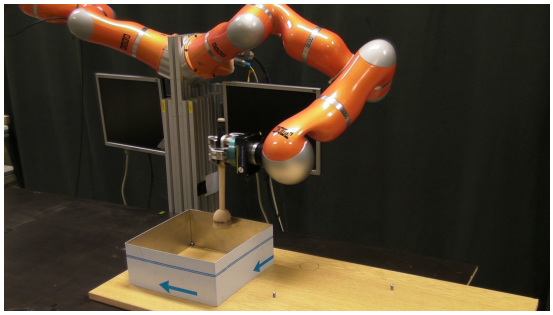


Figure 3.3: Experimental setup for stirring in square pot.

ILC, RRILC generates the feed forward compensation signal already in the first cycle, RRILC converged considerably faster. Similar results were obtained for the periodic task of stirring (see Fig. 3.3). The RRRC approach established full contact with the pot wall already in the second stirring cycle and proceeded with almost unchanged response for the following 6 cycles. On the other hand, RC took 6 stirring cycles to establish full contact with the pot wall. In the last cycle, learned compensation terms were almost identical for both algorithms.

## 3.2 Integration of Force-Based Exploration with Coaching

In Deliverable D4.1.3 and in reference [8] we describe a new coaching mechanism that allow us to alter the previously learned motor actions. Here we describe how coaching can be integrated with explorative learning mechanisms to modify preexisting robot behaviors, such as for example wiping [GPNU14].

We considered the integration of two types of on-line motion adaptation schemes. The first is the adaptation of trajectories to the external environment in order to achieve desired forces of contact throughout the complete trajectory. The second is adapting the trajectories to the interventions of an instructor, who modifies the trajectories with physical contact or with predefined gestures. Thus, he acts as a tutor, coaching the robot the desired behavior. The common point of these two approaches, besides force feedback, is also the use of a uniform trajectory representation, i.e. the dynamic movement primitives (DMPs). The learning properties of DMPs can be exploited to simultaneously realize adaptation to external forces and response to coaching gestures. The combination creates an intuitive and user-friendly interface for learning and modifying robotic trajectories with the potential for creating complex interaction trajectories with a simple user demonstration and a little interactive tutoring.

Our experimental results have shown that the proposed approach is applicable for learning of complete trajectories through force-based exploration as well as for adapting just parts of trajectories using physical human-robot interaction and visual pointing gestures for coaching, which are reported in [8]. Thus, the proposed approach provides a user-friendly and intuitive framework for learning of periodic motions in contact with the environment, where the user can teach the desired trajectories, alter their parts, and the robot finds and maintains the desired contact with the surfaces, all in one system. Additional details of the proposed approach are described in the attached paper [GPNU14].

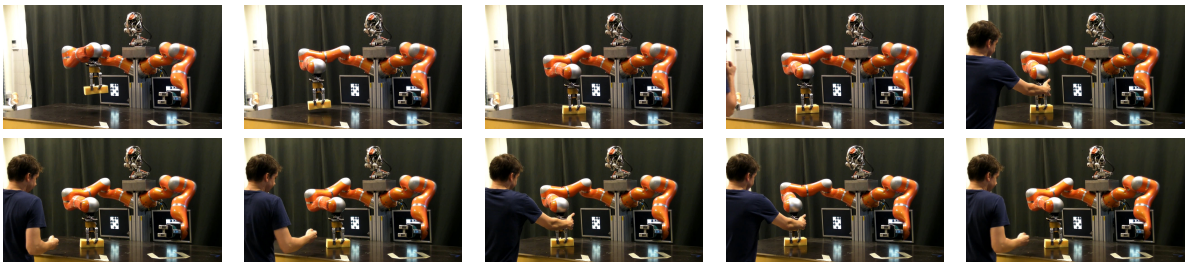


Figure 3.4: Sequence of images showing modification of robot motion though force-based exploration and on-line modification of robotic motion using force feedback while the robot is maintaining the contact with the table.

## 3.3 Hierarchical Segmentation of Manipulation Action Sequences

We continued our work on automatic segmentation of manipulation sequences described in D2.2.2 and reference [11]. We extended our previous approach, which used object relation changes only, to a hierarchical segmentation approach of human demonstration, which takes into account object relation changes and motion characteristics. While the previous approach provided good results for segmentation of a semantic level, it was not able to segment demonstrations with unobservable effects or changes of motion characteristics. Therefore, we developed a hierarchical two level approach. On the first level, semantic segments are extracted based on object relation changes where contact/non-contact information is used to determine meaningful segments. On the second level, the obtained semantic segments are further analyzed and subdivided into more granular segments. To this end, we developed a method for assessing the characteristics of a motion within a semantic segment based on the motion acceleration profile. To find new segments, the motion is recursively divided by searching iteratively with a sliding window for the best split. For each position of the sliding window the trajectory left and right of the center of the window is evaluated. A novel assessment function describes the motion characteristic of the trajectory segments by incorporating the dynamics of the motion. A good split means that the difference between the assessment values is high, i.e. the difference in motion dynamics is high. On the next recursion level, the segments left and right of the found split are analyzed again until a minimum segment length is reached or the difference between the assessment function values is under a certain threshold. Based on this sub-segmentation it is possible to find segments that differ in dynamics of the motion and thus to allow distinguishing different actions, e.g. e.g. shaking a bottle or tossing a bottle, within a semantic segment.

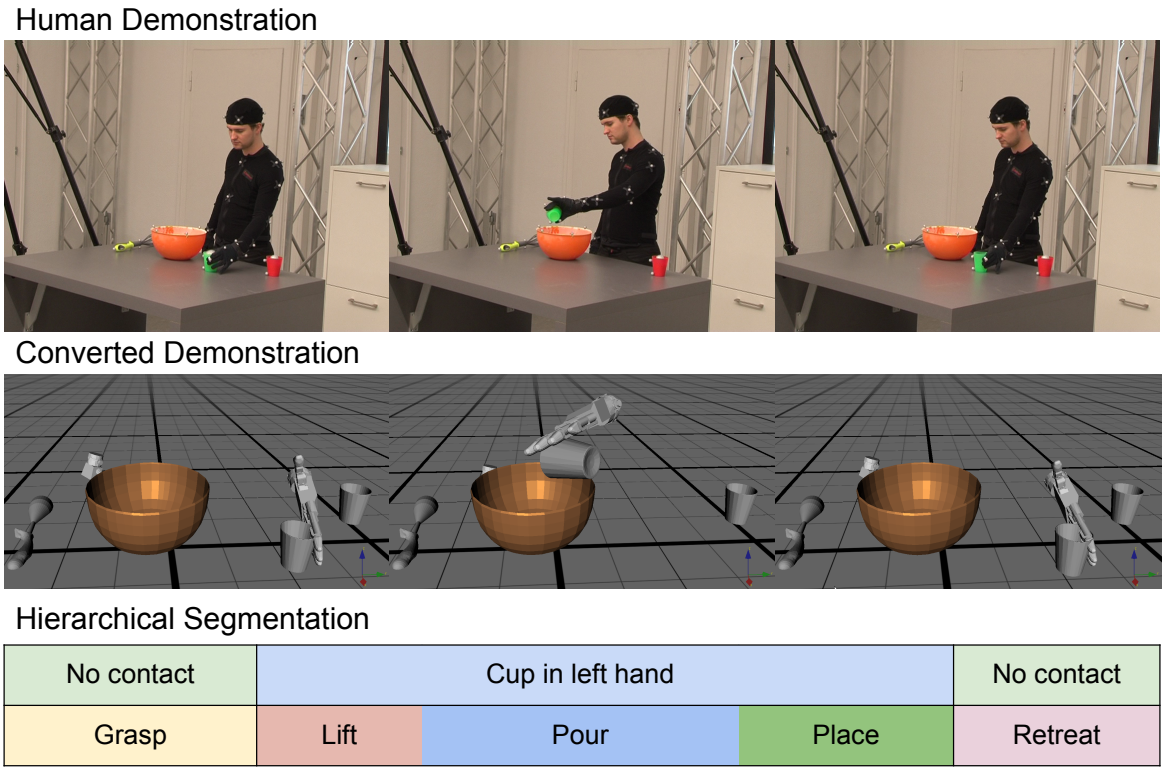



Figure 3.5: A human demonstration of a complex task (top) is being observed with a marker-based motion capture system. This marker-trajectories are converted into 6D object pose trajectories (middle), which serve as input for the proposed segmentation and recognition algorithms. The result of the segmentation and recognition (bottom) contains segments with distinct object-relations on the first level and sub-segments with distinct motion characteristics on the second level. The sub-segments in this figure are labeled manually to illustrate the meaning of the sub-segments.

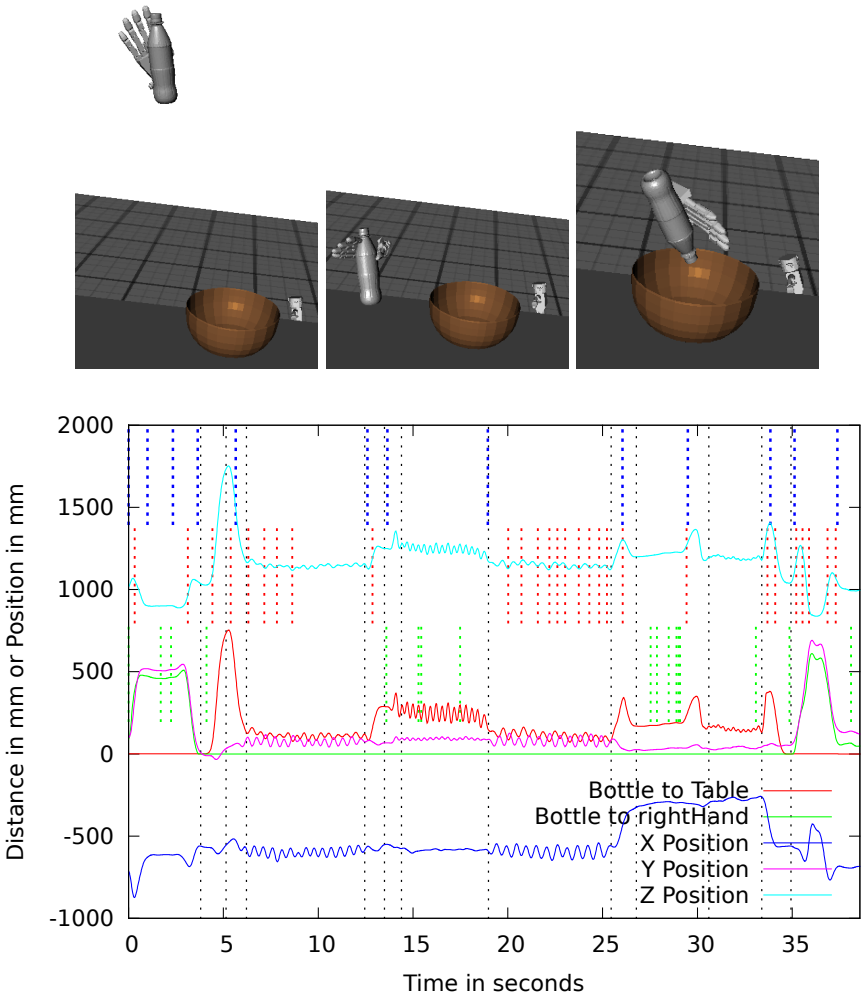



Figure 3.6: Right: Comparison of manual segmentation (black lines), the proposed approach (blue lines, PCA (red lines) and Zero-Velocity-Crossing (green lines) of an action sequence shown by the key frames (vertical lines). The action sequence contains grasping, placing, shaking, tossing, pouring, inspecting and dripping off a bottle. The vertical lines are only partly drawn for better clarity. Left: Snapshots of the segmented action sequence for visualization.

In Figure 3.5 the whole process from demonstration to segmentation is depicted. The results of the approach applied on an action sequence compared with a manual segmentation and other automatic segmentation algorithms are shown in Fig. 3.6. The results of the novel approach exceed the results of segmentation methods based on Principal Component Analysis and Zero-Velocity-Crossing as these detect many false positive and miss more segments than in the case of our approach.

For more details the reader it referred to the attached paper [WDAed].

# References


[1] M. Do, J. Schill, J. Ernesti, and T. Asfour. Learning how to wipe: A case study of structural bootstrapping from sensorimotor experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1858–1864, 2014.

[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] D. Forte, A. Gams, J. Morimoto, and A. Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):740–757, 2012.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] N. Vahrenkamp and T. Asfour. Representing the robot's workspace through constrained manipulability analysis. *Autonomous Robots*, 38(1):17–30, 2015.

[10] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour. The ArmarX Framework - Supporting high level robot programming through state disclosure. *Information Technology*, 57(1), 2015.

[11] M. Wächter, S. Schulz, T. Asfour, E. Aksoy, F. Wörgötter, and R. Dillmann. Action Sequence Reproduction based on Automatic Segmentation and Object-Action Complexes. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 189–195, Atlanta, USA, October 2013.

[12] F. Wörgötter, C. Geib, M. Tamosiunaite, E. E. Aksoy, J. Piater, H. Xiong, A. Ude, B. Nemec, D. Kraft, N. Krüger, M. Wächter, and T. Asfour. Structural bootstrapping – a novel concept for the fast acquisition of action-knowledge. *IEEE Transaction of Autonomous Mental Development*, 2014 (submitted).

# Attached Papers


[GPNU14] A. Gams, T. Petrič, B. Nemec, and A. Ude. Learning and adaptation of periodic motion primitives based on force feedback and human coaching interaction. In *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 166–171, Madrid, Spain, 2014.

[GvdKV+14] A. Gams, J. van den Kieboom, M. Vespignani, G. Luc, A. Ude, and A. Ijspeert. Rich periodic motor skills on humanoid robots: Riding the pedal racer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2326–2332, Hong Kong, 2014.

[NPUed] B. Nemec, T. Petrič, and A. Ude. Improved iterative adaptation scheme with recursive estimation of compensation signals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015 (submitted).

[WDAed] M. Wächter, M. Do, and T. Asfour. Hierarchical segmentation of manipulation actions based on object relations and motion characteristics. In *International Conference on Advanced Robotics (ICAR)*, Istanbul, Turkey, 2015 (submitted).

2014 14th IEEE-RAS International Conference on
Humanoid Robots (Humanoids)
November 18-20, 2014. Madrid, Spain

# Learning and Adaptation of Periodic Motion Primitives Based on Force Feedback and Human Coaching Interaction

Andrej Gams, Tadej Petrič, Bojan Nemec and Aleš Ude

*Abstract*— **Dynamic movement primitives (DMP) allow efficient learning and control of complex robot behaviors for both periodic and discrete point-to-point movements either in joint or Cartesian space. They also allow efficient modulation by changing of parameters. In this paper we introduce and evaluate the means of adapting periodic DMP trajectories with respect to force feedback. We simultaneously consider two aspects: 1) adaptation of whole trajectories to comply with the constraints set by the environment; and 2) partially modifying the trajectories during the execution based on human intervention to improve the task performance. The latter can either be force-based, i. e. through physical contact, or through predefined gestures. By intervening when necessary the human acts as a tutor, instructing the robot how to modify the trajectory and bypassing the need to learn new trajectories by autonomous exploration. We introduce the approach in the context of wiping a surface, where the robot first has to acquire and maintain contact, and where later the human tutor modifies the originally learned trajectory in order to achieve the desired robot behavior. We present simulation and real world results of wiping a surface with a Kuka 7 degree-of-freedom LWR robot.**

## I. INTRODUCTION

In this paper we consider two problems of on-line motion adaptation. The first is the adaptation of trajectories to the external environment in order to achieve desired forces of contact throughout the complete trajectory. The second is adapting the trajectories to the interventions of an instructor, who modifies the trajectories with physical contact or with predefined gestures. Thus, he acts as a tutor, coaching the robot into desired behavior. The common point of these two approaches, besides force feedback, is also the use of a unified trajectory representation, i. e. the dynamic movement primitives (DMPs). The learning properties of DMPs can be exploited to realize both adaptation to external forces and response to coaching gestures. The combination creates an intuitive and user-friendly interface to learning and modifying robotic trajectories with the potential of creating complex interaction trajectories with a simple demonstration and a little tutoring.

Dynamic movement primitives, introduced by Ijspeert et al. [1], are the means of encoding a trajectory in the form of a linear second order differential equation with an added nonlinear forcing term, which changes the simple second order attractor dynamics to the desired behavior. DMPs have many favorable properties for encoding of robotic trajectories, e. g. they contain open parameters that can be used for learning without affecting the overall convergence of the system. They can easily be temporally modulated without requiring an explicit time recalculation, which can be effectively used for synchronization to external signals and devices [2]. They are robust against perturbations and can be spatially modulated to adapt to different requirements [3]. Other trajectory representations include splines [4], mixture models [5] and different dynamical systems, which can also be adapted based on force feedback [6].

Modulation of DMPs with respect to force feedback has been studied for both discrete point-to-point movements, as well as for periodic applications. Discrete trajectories were adapted to force feedback by Pastor et al. [7], who recorded the forces during an execution and used these recordings as the referential signal for a controller, plugged into the acceleration of the DMP. The accelerations were thus altered so that the same force behavior was achieved when the conditions of the task would change. On a very similar basis Kulvicius et al. [8] modified the accelerations of a DMP based on virtual forces defined from the proximity to obstacles. Additionally, the optimal gains of the trajectory adaptation were learned. On the other hand, Gams et al. [9] used force feedback at both velocity and acceleration levels of a DMP to adapt to force feedback. In several discrete iterations the approach also learned a feed-forward term to minimize the feedback signal by means of iterative learning control (ILC). Thus any desired force trajectory (profile) was made achievable if physically feasible. Using ILC requires several repetitions of the same task with resetting of the initial conditions. The latter cannot be achieved in periodic tasks, where the conditions at the start of the period change until steady-state behavior has been reached. Consequently the use of repetitive control (RC) in a feed-forward term was proposed for periodic movements instead of ILC [10].

Both ILC or RC can learn a coupling term, which is fed into the DMP at either velocity level or at both velocity and acceleration levels as a feed-forward component to reduce the feedback error signals [9], [10]. On the other hand, DMPs themselves can be modified to cancel out feedback error signals. This was presented in a wiping task [11], where the feedback force of contact with the environment was used to alter the referential (target) trajectory of DMP learning with a velocity-resolved admittance approach [12]. The wiping motion was expanded to use transient initial motions in [13] and for structural bootstrapping from sensorimotor experience [14], but the force adaptation algorithm was not altered in these works. The force adaptation algorithm from [11] provides the basis for comparison in this paper.

*This work was supported by EU Seventh Framework Programme grant 270273, Xperience

All authors are with Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stean Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia. name.surname@ijs.si

We modify the approach by excluding the change of the reference trajectory itself and by introducing means to online modify desired trajectories using coaching gestures.

Once learned, a periodic DMP trajectory can be modulated by changing the frequency, amplitude, and/or center of oscillation. However, in order to alter just a part of the trajectory, for example to go more left only at a certain point, one would have to re-learn the entire trajectory. To modify only parts of learned trajectories, we propose to again use the learning of DMPs, by externally changing the error of learning, similar as proposed in [15]. This way DMPs act as an efficient coaching system. The novelty in this paper is twofold: 1) we use physical contact, i. e. force feedback instead of visual coaching gestures, and 2) it is combined with learning of complete periodic trajectories for contact with the environment.

Coaching of robot trajectories has been previously applied to various tasks in different settings. For example, voice commands of a human coach were used as a reward function in the learning algorithm by Gruebler et al. [16]. Verbal instructions, applicable also for non-experts, were used to modify movements obtained by human demonstration [17]. Physical contact was also used, for example Lee & Ott [18] used kinesthetic teaching with iterative updates to modify the behavior of a humanoid robot.

The paper is organized as follows. In Section II we provide a recap of dynamic movement primitives and of learning the forcing term, i. e. the weights of the DMP. In Section III we then show how the learning of the DMP weights can be used when the reference is changed. Section IV shows how we can alter complete trajectories without external algorithms of changing the reference, also showing results. Section V expands on this notion by changing only parts of trajectories, culminating in an efficient force adaptation and coaching algorithm. A discussion which highlights the benefits and differences to other approaches concludes the paper.

## II. Periodic Dynamic Movement Primitives

We provide a brief recap of the periodic notation of dynamic movement primitives (DMP), with the formulation based on [19]. Only the basics are provided as DMPs have been thoroughly discussed [1]. We also provide the equations for learning of the DMPs, i. e. the weights of DMPs, as this is the basis for both force learning and coaching.

The following applies for a single degree of freedom (DOF), i. e. one of the external task-space coordinates. It is denoted by $y$, while $z$ stands for the (scaled) velocity. Note that DMPs can be applied to joint space coordinates as well.

Periodic DMPs are defined as a nonlinear system of differential equations

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

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

The nonlinear part of (1), $f(\phi)$, is comprised of a linear combination of radial basis functions $\Gamma_i(\phi)$, defined by

$$f(\phi) = \frac{\sum_{i=1}^{N} w_i \Gamma_i(\phi)}{\sum_{i=1}^{N} \Gamma_i(\phi)} r, \quad (3)$$

$$\Gamma_i(\phi) = \exp\left(h_i\left(\cos\left(\phi - c_i\right)-1\right)\right). \quad (4)$$

The variable $r$ is the amplitude control parameter, while $h_i > 0$ are the widths of the kernels and $c_i$ equally spaced along the phase $\phi$ from 0 to $2\pi$ in $N$ steps. The phase variable $\phi$ bypasses explicit dependency on time and is assumed to increase with constant rate, where the parameter $\Omega$ denotes the frequency

$$\dot{\phi} = \Omega. \quad (5)$$

The frequency does not have to remain constant, but the phase has to be evaluated on-line, for example with adaptive frequency oscillators, as in [3] and [2].

The parameters $\alpha_z,\ \beta_z,\ > 0$ and $\alpha_z = 4\beta_z$ make the system (1) – (2) converge to the oscillations given by $f(x)$ around the goal $g$ in a critically damped manner. To realize multiple DOFs we use separate sets of (1) – (2), and a single canonical system given by (5) to synchronize them through the common phase.

The linear part of (1) – (2) defines convergence to the goal $g$. It is only the weights $w_i$, $i = 1, ..., N$, where $N$ is the number of kernel functions, given in the vector $\mathbf{w}$, that define the actual shape of the encoded periodic trajectory. Only one period of motion is encoded and it repeats as the phase resets at $2\pi$. To encode a trajectory as a DMP, we have to learn the weight vector. The latter is accomplished using incremental locally weighted regression, where the target data for fitting is

$$f_{targ} = \frac{1}{\Omega^2}\ddot{y}_{\mathrm{ref}} - \alpha_z\left(\beta_z\left(g - y_{\mathrm{ref}}\right) - \frac{1}{\Omega}\dot{y}_{\mathrm{ref}}\right), \quad (6)$$

obtained by matching $y$ from (1) – (2) to $y_{ref}$, $z$ to $\dot{y}_{ref}/\Omega$, and $\dot{z}$ to $\ddot{y}_{ref}/\Omega$. This means that basically we learn the accelerations needed to force the otherwise critically damped spring-mass system given by the linear part of (1) – (2) to follow the desired trajectory.

Given $f_{targ}$, $w_i$ is updated incrementally for each time-step $j$

$$w_{i,j+1} = w_{i,j} + \Psi_i P_{i,j+1} r e_j \quad (7)$$

$$P_{i,j+1} = \frac{1}{\lambda}\left(P_{i,j} - \frac{P_{i,j}^2 r^2}{\frac{\lambda}{\Psi_i} + P_{i,j} r^2}\right) \quad (8)$$

$$e_j = f_{\mathrm{targ},j} - w_{i,j} r \quad (9)$$

$P_i$, in general, is the inverse covariance of $w_i$ [20]. The recursion is started with $w_i = 0$ and $P_i = 1$. $r$ is the amplitude gain. $\lambda$ provides the forgetting factor. If $\lambda < 1$, then the incremental regression gives more weight to recent data, meaning that it tends to forget older ones.

## III. Adaptation of DMPs by changing the reference

In this section we provide and modify the algorithm from [11], which serves as comparison for two possibilities of modifying DMP trajectories using incremental locally weighted regression, which is at the core of the DMP learning.

The algorithm in [11] provides the means to adapt to external contact by adapting the referential trajectory that a DMP encodes in the weight vector $\mathbf{w}$. The error of learning $e_j$ in (9), updated incrementally in every time step, is calculated using (6), where the referential trajectory $y_{ref}$ (along with $\dot{y}_{ref}$ and $\ddot{y}_{ref}$) in (6) is provided by (13). Omitting the details given in [11], the algorithm uses a velocity-resolved approach. For one degree of freedom denoted by $y$, it calculates the resolved velocity $\dot{y}_r$ with

$$\dot{y}_r = k_p(F_{y0} - F_y). \tag{10}$$

Here parameter $k_p$ is the force gain, $F_y$ the measured and $F_{y0}$ the desired force in this degree of freedom. The new referential trajectory $y_{ref}$ is integrated using this velocity and the starting position $y_0$ by

$$y_{ref} = y_0 + \int \dot{y}_r dt. \tag{11}$$

Instead of a simple proportional law in (10), we can also use a proportional-derivative law to increase the damping, resulting in

$$\dot{y}_r = k_p(F_{y0} - F_y) + k_d \frac{d}{dt}(F_{y0} - F_y). \tag{12}$$

Given that the measured force signal is very noisy, this might result in an even noisier resolved velocity. Inserting (12) into (11) yields

$$y_{ref} = y_0 + \int k_p(F_{y0} - F_y)dt + k_d(F_{y0} - F_y). \tag{13}$$

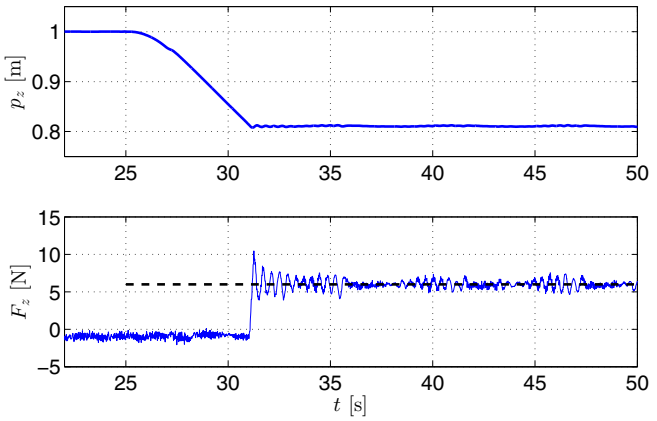



Fig. 1. The top plot shows real world results of adaptation of motion in $p_z$ direction (downwards). The resulting forces with the referential force of contact given at 6 N (dashed line) is shown in the bottom plot. As the robot was performing left-right wiping motion, some oscillations due to the contact are visible in the force measurement. Fig. 2 left shows the complete 3-D trajectory for this experiment.

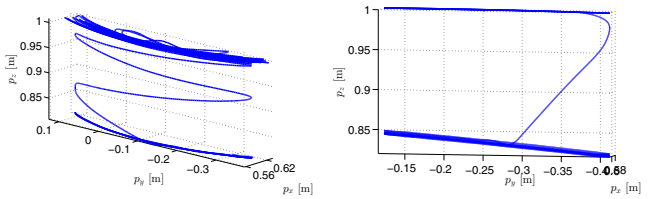


Fig. 2. The complete 3D trajectories for 2 experiments where the adaptation of the trajectory in $p_z$ direction was implemented using the velocity-resolved approach. We can see in the left plot the adaptation to a flat surface. The results also depict initial learning of motion. The force results are presented in Fig. 1. The right plot shows the adaptation to an inclined surface. The wiping motion was not learned, but re-used from the experiment in the left plot. The forces, which show a clear hysteresis, are depicted in Fig. 3.

This has the advantage that there is no derivative term and that the system has an additional proportional term directly on the force (and not integrated force), which results in faster responses.

Even with the formulation in (13), the change of the reference $y_{ref}$ is still subject to $k_p$ and $k_d$ and will always have some delay. It is still a feedback controller, where an error has to appear in order to change the reference. Even though a DMP is incrementally encoding the reference, it can only exactly encode (without delay) adaptation to a flat surface, where the output of the integral in (13) provides the exact reference. The results are shown in Fig. 1 and 2, left. The robot adapts its height in order to maintain the desired contact force $F_{y0}$ with the surface. All of our experiments are shown also in the accompanying video.

If the surface is not constant, for example a slope, or even an arbitrary surface, the DMP will only encode what (13) will provide, and this always has a slight delay. This is evident in Fig. 3, where we show the adaptation of a DMP to a sideways tilted flat surface, performed by a real robot. The bottom plot shows that a hysteresis of force is present, depending if the robot is moving up or down the slope. The hysteresis of learned motion is also seen in Fig. 2, right, where we can

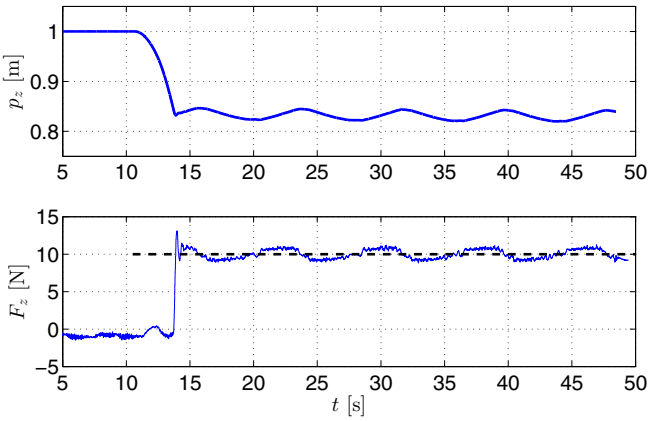



Fig. 3. The trajectory of motion when adapting to a flat but tilted surface in the top plot. The resulting forces show a clear hysteresis resulting from moving up or down the slope in the bottom plot. Fig. 2-right shows the complete 3-D trajectory arising in this experiment.

note that the bottom left-right line has a hysteresis, albeit not a very pronounced one.

## IV. Direct adaptation of trajectories

To exclude the delay of the algorithm in the previous section, we exploit the weight-fitting algorithm of the DMP. Let's assume a given trajectory encoded as a DMP, where the weights $\mathbf{w}$ encode the trajectory. The trajectory is following the demonstrated trajectory with the error of learning in (9) at $e_j \cong 0$, meaning that $\mathbf{w}$ does not change anymore.

By changing (9) into

$$e_j = k_l(F_{y0} - F_y), \tag{14}$$

and using it in (7), the fitting, i. e. the incremental locally weighted regression will update the weights whenever the error of force in (14) will not be zero. In other words, it will adapt the trajectory to fulfill the condition of (14), which is that the actual force of contact $F_y$ is the same as the desired force $F_{y0}$. Parameter $k_l$ is a positive constant. Note that the implementation of adaptation should take care that the values of the inverse covariance $P_i$ do not decrease to $P_i \cong 0$ as this will stop the adaptation, given that the update of weights is multiplied by $P$.

A feedback term can be added to the acceleration level of the DMP in order to account for noise and non-repeating disturbances, changing (1) into

$$\dot{z} = \Omega\left(\alpha_z\left(\beta_z\left(g - y\right) - z\right) + f(\phi) + d(F)\right). \tag{15}$$

The feedback term can be a simple proportional control law with gain $k_{fb} > 0$, for example $d(F) = k_{fb}(F_{y0} - F_y)$.

In this paper we name the trajectory adaptation method based on (14) the *Direct* method. In simulation, where we can model the forces of contact with displacement of the elastic environment with stiffness $k_{\text{env}}$, we can rewrite (14) into $F = k_{\text{env}}(y_0 - y) = k_{\text{env}}\overline{y}$. Any error of forces at end-effector will therefore introduce a position difference $k_l k_{\text{env}}(y_0 - y)$, which will through (7) reflect in $f(\phi)$. From (1) – (2) we can see that through integration of the DMP differential equations, $f(\phi)$ (and consequently $y_0 - y$) is integrated twice, which results in a slight delay.

From a physical standpoint, the linear part of (1) represents accelerations of a spring-mass system, while $f(\phi)$ provides the modification for accelerations that force the system to follow the desired trajectory. In order to exclude the above mentioned delay from position-difference integration, we need to change (14) so that it provides proper accelerations. These cannot be just any accelerations, but accelerations to drive the given DMP spring-mass system, which are calculated according to (6). We therefore write

$$e_j = \frac{1}{\Omega^2}k_2\ddot{\overline{y}} - \alpha_z\left(\beta_z\left(g - k_2\overline{y}\right) - \frac{1}{\Omega}k_2\dot{\overline{y}}\right), \tag{16}$$

where $k_2\overline{y} = k_l(k_{\text{env}}(y_0 - y))$ models the forces. In this paper we name the trajectory adaptation method based on error signal (16) the *Diff* method.

Figure 4 shows simulated results of trajectory adaptation on a tilted flat surface. We can see in the top plot the location of the surface (red) and the trajectory of the robot in three different control settings. The green plot is the result of the velocity-resolved approach, i. e. (13) provides the referential trajectory for (9). The delay of adaptation is clearly seen. The black plot shows the results of using (14), i. e. the *Direct* method. The blue plot shows results of using (16), i. e. the *Diff* method. The plots show that the *Diff* plot converges faster and to a smaller error that the *Direct* method. The errors of adaptation are shown in the bottom plot.

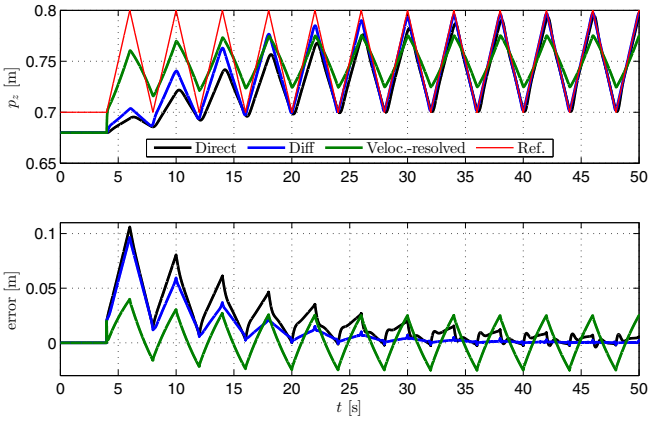



Fig. 4. The results of simulated trajectory adaptation using three different control methods, with a tilted flat surface as a reference. The experiment started with the robot already in contact with the surface. We can see the reference (red) and the three resulting trajectories in the top plot. The errors of adaptation are shown in the bottom plot. See the text for a description of separate lines.

Real-world results of adaptation to a sinusoidal reference force, where we used the *Direct* method and no feedback term in (15) are shown in Fig. 5. The experiment required that the robot, after establishing contact with the table, presses on it with a changing, sinusoidal force. We can see from the results that the trajectory was adapted to the desired shape.

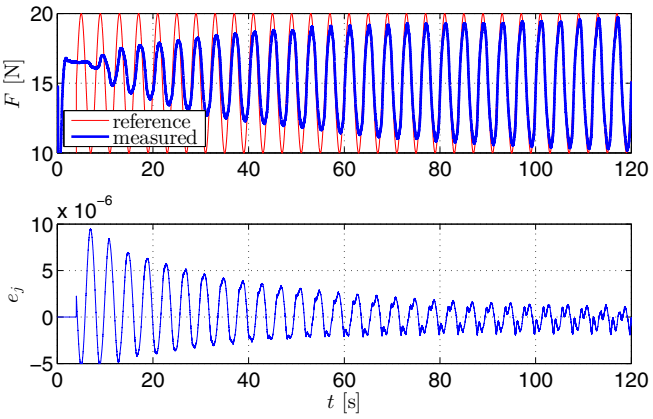



Fig. 5. The results of adapting the robot trajectory using 14, with a sinusoidal referential force. The experiment started with the robot already in contact with the surface. The referential and resulting forces are in the top plot, while the error signal, which is an input to the weight fitting, is in the bottom plot.

## V. Adapting Parts of Trajectories

During trajectory learning the demonstrator repeats several periods of motion and the collected data are given as reference to the incremental locally weighted regression. The trajectory is learned, but it might not exactly do what the demonstrator intended, as is often the case when giving instructions to another person on how to perform something. When the resulting motion is not satisfactory, the demonstrator can coach the other person, specifying how to alter the motion in certain parts, or simply showing the complete motion again.

In order to avoid re-learning of the complete trajectory, we can exploit the same mechanism as in Section IV to change only parts of the trajectory. We again rely on changing (9). If $e_j = 0$, there is no learning and the robot just repeats the trajectory it learned during the demonstration. Again, for a single degree of freedom, we change (9) into

$$e_j = C(\text{input}), \tag{17}$$

making the error a function of the input, where input can be either the force applied to the robot or the demonstrator's pointing gesture, visually illustrating in which direction to change the trajectory. For the case of force input, (17) changes into

$$e_j = k_F F_y, \tag{18}$$

where parameter $k_F$ scales the measured force $F_y$. The measured force in this case should be the force exerted by the human tutor to the robot. If the robot is in contact with an object, for example when wiping the table, one must distinguish between the forces that arise from the contact with the table and as a result of friction, and the forces applied by the human operator. A simple solution is to decouple forces by direction and to include sufficient dead zones.

Fig. 6 shows the robot end-effector trajectory before and after coaching. The initial robot wiping motion is in green. The blue line shows the trajectory of the robot during coaching, i. e. while the human was pushing on it. The measured forces of contact are shown in Fig. 7. Four clear peaks of force show where the human pushed on the robot. The final wiping motion of the robot after coaching is shown in red. The initial motion was performed using a previously learned DMP, the one from Fig. 2, left. The robot found and maintained a contact with a flat surface from the start of the experiment. Coaching was applied in $x$ direction only.

When using pointing gestures, we can use active motion capture markers to first demonstrate a motion and later use the same markers and their relative positions for tutoring. This was the case in our experiment. We defined the following repulsive force field

$$e_j(x) = \begin{cases} 0 & p > 0.1 \\ (0.001/p^2 - 0.1)/40 & p \leq 0.1,\ p_{1z} > p_{2z} \\ (-0.001/p^2 - 0.1)/40 & \text{otherwise} \end{cases} \tag{19}$$

where $p$ stands for the distance between the robot and the closest marker. Index $_{iz}$ is the $z$ axis location of the $i$-th

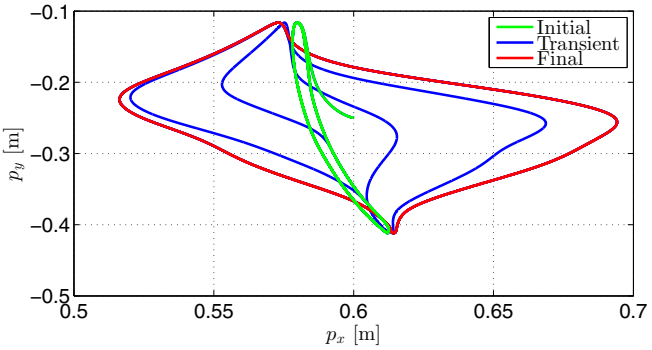



Fig. 6. $X - Y$ plot of the end-effector motion depicts the initial motion in green, the motion during tutoring in blue and the final trajectory in red.

marker. The given force field has no effect on the robot if the closest marker is more than 10 cm away, whereas its effect increases quadratically with proximity, effectively pushing the robot away if $p \approx 0$. The relative location of the markers also defines if the robot is being pushed away or pulled towards the tutor. The given force field was determined empirically. This experiment is only shown in the accompanying video.

Fig. 8 shows the still images of a person coaching the motion of a robot through interaction, i. e. by force.

## VI. Discussion and Conclusions

In this paper we presented an alternative to the velocity-resolved approach applied to DMPs in [11]. The approach utilizes iterative locally weighted regression of DMPs to change the weights of the DMP and therefore the trajectory of the robot based on external input. This external input is not the target trajectory, but the output of a control law, which provides a force reference for the robot. The results show that the approach reduces the error of achieved forces for arbitrary surfaces more than what is achievable with the velocity-resolved approach. The latter is limited with the

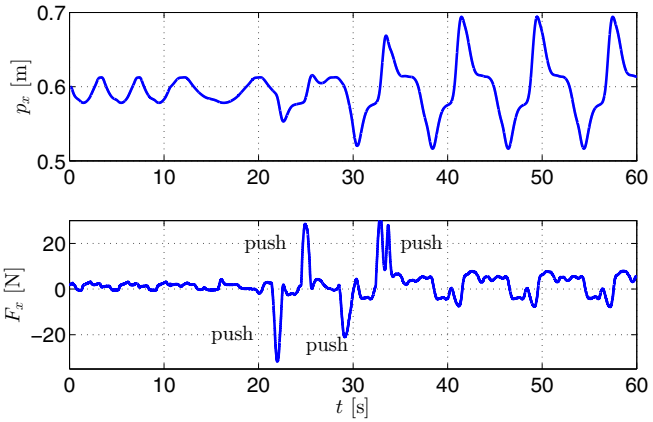



Fig. 7. The top plot depicts the trajectory of motion in $x$ direction of the end effector of the robot. The external forces applied by the tutor, depicted in the bottom plot, modified the motion to achieve the intended result.

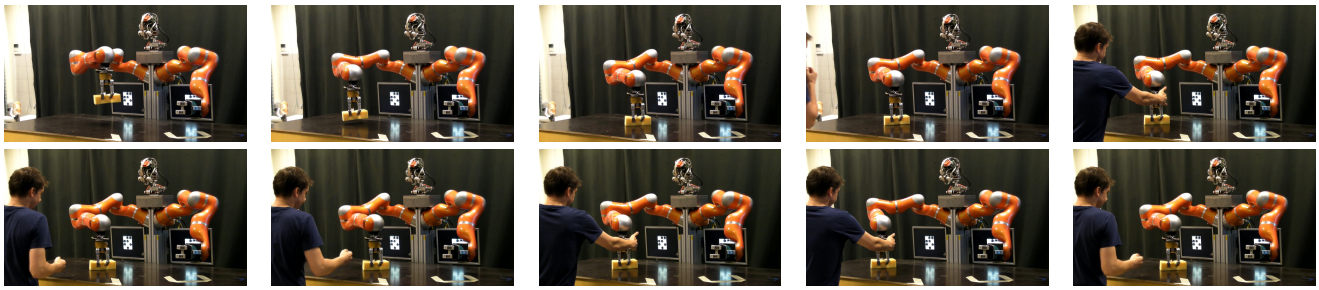


Fig. 8. Sequence of images showing on-line coaching of robotic motion using force feedback while the robot is maintaining contact with the table. The experiment is also presented in the video and its results in Figs. 6 and 7.

bandwidth of the admittance controller, which relies on the integral part and therefore always introduces a delay.

While we have not discussed rotations, these can be modified just as the positions, as was described also in [11]. In the case of wiping, one needs to make sure to distinguish between the forces that arise from the contact, for example wiping, and the forces and torques that should actually change the orientation. For example, when wiping a surface, the friction will produce a force in the opposite direction of wiping. This force should not change the trajectory of motion or the orientation.

Another drawback of the velocity-resolved approach is in the introduction of an external control system, which is out of scope of the well defined and stable DMP framework. This complicates the overall system structure and introduces additional stability criteria, which have to be met.

Our results have shown that the proposed approach is applicable for learning of complete trajectories as well as for adapting just parts of trajectories using physical human-robot interaction and visual pointing gestures for coaching. Thus, the proposed approach provides a user-friendly and intuitive framework for learning of periodic motions in contact with the environment. It is intuitive in the sense that it is first demonstrated and then altered by pushing on it, the more it is pushed, the more it is altered. The user can teach the desired trajectories, alter their parts, and the robot finds and maintains the desired contact with the surfaces, all in one system.

## REFERENCES


[1] 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.

[2] 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.

[3] 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.

[4] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, and G. Cheng, "Synthesizing goal-directed actions from a library of example movements," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, Nov 2007, pp. 115–121.

[5] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, Oct 2011.

[6] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 323–329.

[7] 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.

[8] 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.

[9] A. Gams, B. Nemec, A. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, 2014, to appear.

[10] 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)*, 2014.

[11] A. Gams, M. Do, A. Ude, T. Asfour, and R. Dillmann, "On-line periodic movement and force-profile learning for adaptation to new surfaces," in *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Nashville, TN, 2010, pp. 560–565.

[12] L. Villani and J. De Schutter, "Force control," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008.

[13] 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)*, 2012, pp. 57–64.

[14] M. Do, J. Schill, J. Ernesti, and T. Asfour, "Learn to wipe: A case study of structural bootstrapping from sensorimotor experience," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[15] T. Petrič, A. Gams, L. Žlajpah, A. Ude, and J. Morimoto, "On-line approach for altering robot behaviors based on human in the loop coaching gestures," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[16] 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)*, Oct 2011, pp. 466–471.

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

[18] 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.

[19] 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.

[20] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. MIT Press, 1986.

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

# Rich Periodic Motor Skills on Humanoid Robots: Riding the Pedal Racer

Andrej Gams[1,2], Jesse van den Kieboom[1], Massimo Vespignani[1], Luc Guyot[1], Aleš Ude[2] and Auke Ijspeert[1]

***Abstract*— Just as their discrete counterparts, periodic or rhythmic dynamic motion primitives allow easily modulated and robust motion generation, but for periodic tasks. In this paper we present an approach for modulating periodic dynamic movement primitives based on force feedback, allowing for rich motor behavior and skills. We propose and evaluate the combination of feedback and learned feed-forward terms to fully adapt the motions of a robot in order to achieve a desired force interaction with the environment. For the learning we employ the notion of repetitive control, which can effectively minimize the error of behavior towards a given reference. To demonstrate the approach, we show results of simulated and real world experiments on a compliant humanoid robot COMAN. We show the initial results of utilizing the approach to control a pedal-racer, a demanding balance toy best described as a hybrid between a skateboard and a bicycle.**

## I. INTRODUCTION

Periodic motions, often termed also as rhythmic motions, appear in many biological systems and range from manipulatory tasks to locomotion. As they are periodic, specific control approaches can be applied. The latter is not only observable in engineering approaches, but also in biological systems. The term *central pattern generator* (CPG) describes neural circuits found in both invertebrate and vertebrate animals that can produce rhythmic patterns of neural activity without receiving rhythmic inputs [1]. CPGs represent fundamental building blocks for the locomotor neural circuits in animals.

The difference of controlling periodic or discrete motions was also investigated in humans. Schaal et. al. [2] compared fMRI measurements during discrete and rhythmic tasks, which showed that higher cortical centers are used for discrete motions and much less for rhythmic tasks, indicating that rhythmic tasks require separate theoretical treatment. Computational models for rhythmic and discrete models motions were also investigated by Ronsse et al. [3] and Degallier & Ijspeert [4].

Many researchers propose that motor control is based on the combination of motor primitives, i.e., that complex movements are generated by combining a finite set of simpler elementary movements [1], applicable also to locomotion [5]. This notion has been extensively applied to control of robotic tasks; notably with *dynamic movement primitives* (DMPs), first introduced by Ijspeert et al. [6]. DMPs represent one of the approaches for encoding and generating trajectories; other approaches include, for example, Gaussian Mixture Regression [7] and Mixture Models [8], splines and wavelets [9], etc.

While DMPs were originally designed for discrete, finite tasks, they can be effectively applied to periodic tasks [9]–[12]. Just as their discrete counterparts, they use a set of differential equations to compactly represent control policies and at the same time allow adaptation by modifying only a few parameters [12]. The latter can be exploited in several ways, for example for temporal modulation, where the frequency of motion, which determines the behavior of the whole system, is modulated by a single parameter. The frequency of motion can either be set in advance, or can be modified online to autonomously adapt to external, driven systems, for example by the use of adaptive frequency oscillators [13]. Gams et al. [9] have shown how periodic DMPs can be combined with a pool of adaptive oscillators to facilitate a system with combined frequency and waveform learning. On the other hand, Petrič et al. [11] have shown how a single adaptive frequency oscillator and an adaptive Fourier series can be used in combination with periodic DMPs for complex synchronization tasks. Periodic DMPs allow also spatial modulation and generalization from a library of recorded motions [14].

Any kind of periodic interaction task, even simply maintaining contact with a (periodically) moving object, for example the end of a two-person cross-cut saw [15], requires either extremely precise predefined trajectories, or adaptation to specific tasks. Similar applies also to rotating bicycle pedals or periodically applying force on an a surface. If that task is to be executed in an unstructured environment of human daily life, predefining the correct trajectories is practically impossible. Generalization, as one of the possibilities of adapting to the environment, can provide reasonable solutions [14].

The problem we are tackling in this paper is in adapting such periodic trajectories to achieve desired force behavior and rich motors skills, i. e., in the sense of adaptability to different conditions. The novelty of the approach is in enabling online spatial modulation of periodic DMPs by coupling them to the environment using force feedback. Similarly to external limit modulation [9], our approach utilizes coupling at the velocity level of a DMP, but with actual, measured force instead of a virtual one. In order to achieve desired interaction with either the environment or another robot we

Research leading to these results was supported by Sciex-NMSCH project 12.018 and by the EU Seventh Framework Programme grants 611832, WALKMAN, and 270273, Xperience.

[1]Biorobotics Laboratory, Institute of Bioengineering, École Polytechnique Fédérale de Lausanne, EPFL STI IBI BIOROB, Station 14, CH-1015 Lausanne, Switzerland. name.surname@epfl.ch

[2] Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stean Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia. {andrej.gams, ales.ude}@ijs.si

combine feedback with a feed-forward term, which we learn in a few periods of the motion. The learning is based on repetitive control (RC), which can be used to achieve near perfect tracking of a given periodic reference [16], [17]. Our proposed approach allows adaptation of periodic trajectories to the state of the environment in order to achieve desired interaction. We apply the approach to the task of operating the pedal racer, a demanding balance toy best described as a hybrid between a skateboard and a bicycle, with a humanoid robot. The robot and the device are presented in Fig. 8. We test our approach both for operating the device with arms and for riding it. Riding the pedal racer is very complex from the balance point of view. We applied a center of mass controller to maintain the balance and used our proposed approach to modulate the trajectories of motion.

Force feedback was previously used with periodic DMPs in a table wiping task, where complete trajectories were learned using regression techniques [18]. On the other hand, discrete DMPs were previously modulated based on force feedback using iterative learning control (ILC) [19]. While ILC requires initial conditions reset after every repetition, RC can handle the change in the initial conditions – at the start of every period. Furthermore, the approach of learning the correct motion comes more naturally for periodic motions, where the motion is by itself repeated over periods.

Other DMP coupling approaches used acceleration level feedback on discrete DMPs in combination with learning [20], or a feedback controller [21]. A combination of feedback and learned feed-forward terms in rhythmic DMPs to stabilize a two-link robotic arm was presented in [22].

In the rest of this paper we first give a summary on periodic DMPs and present the modulation approach in Section II. We argument and present the Repetitive Learning algorithm applied for the learning in Section III. We evaluate the approach with both simulations and real world experiments in Section IV. In Section V we show how the approach can be used for controlling a pedal-racer. Conclusions and a brief discussion follow in Section VI.

## II. MODULATING PERIODIC MOVEMENT PRIMITIVES

In the following we provide the basic motion representations and the proposed approach to solving the problem of adapting periodic motions with force feedback.

### A. Periodic Motion Primitives

Periodic DMPs have been studied for various tasks [12]. We provide only the basic information, based on the formulation in [14]. For a single degree of freedom (DOF), in our case one of the external task-space coordinates and denoted by $y$, a periodic DMP is defined by the following system of nonlinear differential equations

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

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

$f(\phi)$ is defined as a linear combination of periodic radial basis functions $\Gamma_i(\phi)$

$$f(\phi) = \frac{\sum_{i=1}^{N} w_i \Gamma_i(\phi)}{\sum_{i=1}^{N} \Gamma_i(\phi)} r, \quad (3)$$

$$\Gamma_i(\phi) = \exp\left(h_i\left(\cos\left(\phi - c_i\right)-1\right)\right), \quad (4)$$

where $r$ is the amplitude control paramter, $h_i > 0$ are their widths and $c_i$ are equally spaced between $0$ and $2\pi$ in $N$ steps. The phase variable $\phi$ is introduced to avoid explicit dependency on time. The phase is assumed to increase with constant rate

$$\dot{\phi} = \Omega. \quad (5)$$

The parameter $\Omega$ denotes the frequency. If parameters $\alpha_z,\ \beta_z,\ > 0$ and $\alpha_z = 4\beta_z$, the system (1) – (2) converges in a critically damped manner to the goal $g$.

The weights $w_i$, $i = 1, ..., N$ and $N$ the number of kernel functions, given in the vector $\mathbf{w}$, define the shape of the encoded trajectory. [6] and [14] describe the learning of the weight vector. Multiple DOFs are realized by maintaining separate sets of eqs. (1) – (2), while a single canonical system given by (5) is used to synchronize them by providing a common phase variable.

### B. Coupling of periodic DMPs

An example of coupling to achieve spatial modulation of DMPs is to include a simple virtual repulsive force to avoid moving beyond a given limit [9]. In this paper we propose a similar approach, but by including the real, measured force. This is implemented by modifying (2) into

$$\dot{y} = \Omega\left(z + C\right). \quad (6)$$

Here $C$ represents a coupling term, composed of feedback ($c_{\mathrm{fb}}$) and feed-forward ($c_{\mathrm{ff}}$) terms

$$C = c_{\mathrm{fb}} + c_{\mathrm{ff}}. \quad (7)$$

The feedback term is defined by

$$c_{\mathrm{fb}} = k\left(F_{\mathrm{des}} - F_{\mathrm{act}}\right), \quad (8)$$

where $k$ is a positive constant, $F_{\mathrm{des}}$ is the desired force trajectory and $F_{act}$ is the real, measured force. The desired force can be any force trajectory. For the case of riding the pedal racer (in simulation) we used a measured force profile (also obtained in simulation), as the reference.

A virtual force can also be used, defined for example by a virtual spring between the position of the robot $y_{r1}$ and an arbitrary object $o$

$$F_{\mathrm{act,v}} = k_{\mathrm{env}}(o - y_{r1}). \quad (9)$$

with $k_{\mathrm{env}}$ defining the stiffness of the virtual spring. The position of another robot can be used for an object, coupling two robots. The virtual force between the positions of two robots ($y_{r1}$ and $y_{r2}$), with $d$ the desired distance is then

$$F_{\mathrm{act,v}} = k_{\mathrm{env}}(d - (y_{r1} - y_{r2})). \quad (10)$$

The feedback term acts as a P-controller, with the gain $k_{\text{env}}$ defining the behavior. If it is high, the robot essentially bounces of on impact, while a low $k$ results in slow adaptation of the robot trajectory.

Using the feedback coupling term alone resembles the approach in [21]. It is important to note that we couple the DMP at the velocity level in (2), and not at the acceleration level, i. e., not with providing a coupling term to (1) after $f(\phi)$. This has several advantages. First, this makes feedback modulation highly reactive to the actual force, and low gains can be used. The most important feature of velocity level coupling is in producing lower oscillations in the direction of the force. For details on advantages of velocity-level coupling and stability of coupled DMPs see [19].

If the conditions of the periodic task do not change during the execution, the error of using only a feedback term will not change through the periods of motion, but will continue to repeat. We propose adding a feed-forward term to cancel the error ($e = F_{\text{des}} - F_{\text{act}}$) of tracking the predefined force trajectory. The feed-forward term has to adapt to the specific task by a learning method. We propose using repetitive control (RC), which can achieve near perfect tracking of a given reference if the conditions of the task remain relatively the same [16], [17]. The approach is described in the next section.

To make $c_{\text{ff}}$ also a function of the phase, we encode the feed-forward term in the form of weighted kernel functions $\Gamma_i$, similar to $f(\phi)$ in (4)

$$c_{\text{ff}}(\phi) = \frac{\sum_{i=1}^{N} v_i \Gamma_i(\phi)}{\sum_{i=1}^{N} \Gamma_i(\phi)}. \tag{11}$$

The target for fitting is the output of the repetitive control (see Fig. 1). We use iterative locally weighted regression (ILWR) with a forgetting factor $\lambda = 0.99$ to learn the weights $v_i$. See [9] for details on ILWR. The encoding acts as a filter, canceling out some of the noise of the measurements. Additionally, once we stop learning, the same phase drives both the DMP and the coupling, allowing for temporal modulations.

## III. REPETITIVE CONTROL

Repetitive control is a control method that uses previous experience to design a new control signal and is thus categorized as *learning-type control* [16]. It is mainly used in continuous processes for tracking or rejecting periodic exogenous signals and in most cases, the period of the exogenous signal is known. In this paper we use what is known as a *plug-in type repetitive control*, which incorporates both feed-forward and feedback terms. The structure of such controllers is presented in the top left part of Fig. 1, denoted by the solid border. Figure 1 also shows how RC is combined with a DMP, and how a virtual force can be utilized, shown bottom right and denoted by the dashed border.

Note that the schematic is written for a discrete-time system, where $z^{-1}$ represents a delay of one time sample and $z^{-T}$ of one period. In this paper we also only tackle constant frequencies.

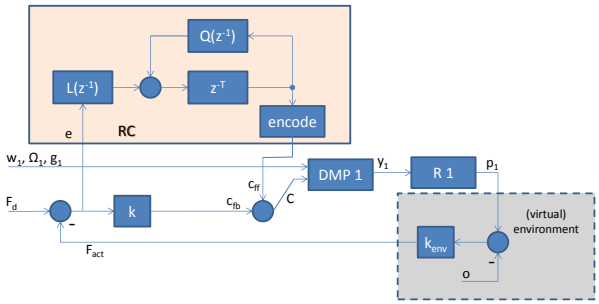



Fig. 1. Schematic of the RC in combination with the DMP and the robot (R1). Contact with the environment provides the force $F_{\text{act}}$. The force can be the real, measured force, or a virtual force calculated with (9).

The feed-forward term is composed of two parts, known as the $Q$ and the $L$ filter, which are defined as [16]

$$L(z^{-1}) = k_{RC}, \tag{12}$$

$$Q(z^{-1}) = \epsilon(\alpha_1 z + \alpha_0 + \alpha_1 z^{-1}), \tag{13}$$

$$2\alpha_1 + \alpha_0 = 1. \tag{14}$$

Here $\alpha_i > 0$, $1 \geq \epsilon > 0$ and $k_{RC} > 0$. Effectively, the feed-forward term uses the error signal from the previous period (see $z^{-T}$ in Fig. 1) and applies it to the control signal. The values of $k_{RC}$, just as the value of $k$ in the feedback term, has to be determined before hand. In our experiments we determined the values empirically.

The robustness of the system is determined through the gain of the $Q$-filter. Stability of RC is a wide and complex topic. As discussed in a survey by Cuiyan et al. [17], different types of RC control require different design and synthesis methods. In general, the selection of control parameters involves a tradeoff between steady state accuracy, robustness and transient response of the system [17]. Furthermore, there is also the phenomenon of apparent convergence, where the system apparently converges, but after some time diverges [23]. Multiplying the output of the $Q$ filter with $\epsilon < 1$ will increase the robustness of the system, but also the steady-state error. In our simulated experiments we used $\epsilon = 0.95$ and in our real-word experiments we used $\epsilon = 0.7$.

The contribution of this paper is in showing how we can combine DMPs with their modulation and disturbance rejection properties, and a learning framework of RC. While RC alone could be applied to the problems at hand, the combination with DMPs allows more robust and adaptive behavior and modifications with a small set of parameters.

## IV. EVALUATION

We evaluated the algorithm in both simulation and in real-world experiments. In the first experiment we simulated a task where the robot has to maintain contact with a periodically moving object (under external actuation) while applying a constant, predefined referential force. The scheme in Fig. 1 was used for simulated experiments and perfect tracking of the robot was assumed ($R1 = 1$, see Fig. 1). We used (9) to simulate the forces.

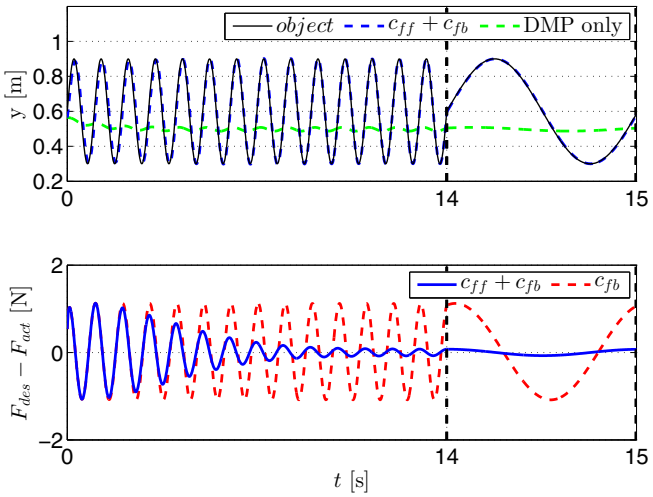



Fig. 2. Simulated results of using the proposed RC learning to reduce the error of force tracking for the task of maintaining contact with a periodically moving object. The trajectory of the object (black), the trajectory with coupling (dashed blue) and the original DMP trajectory (green dashed) in the top plot. The error of force tracking when using the feedback coupling only (red dashed) and when using both feedback and feed-forward couplings (blue). Using both couplings reduces the error about 25 times. The final second of the plot is zoomed in.

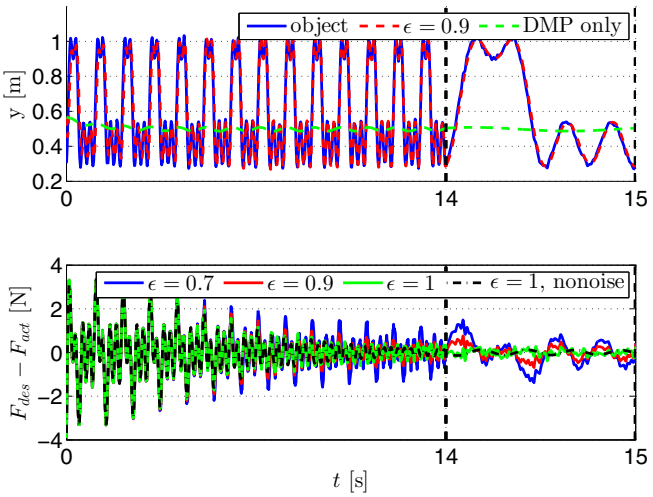



Fig. 3. Simulated results of the effect of $\epsilon$ and noise on using the proposed RC learning to reduce the error of force tracking for the task of maintaining contact with a periodically moving object. The trajectory of the object (black), the trajectory with coupling and $\epsilon = 0.9$ (dashed red) and the original DMP trajectory (green dashed) in the top plot. The error of force tracking with different $\epsilon$ values in the bottom plot. Steady-state error is lowest when there is no noise and highest $\epsilon = 1$, shown in black dash-dot line. The final second of the plot is zoomed in.

Figure 2 shows the results for the task of maintaining desired contact with an object performing a simple sinusoidal trajectory. In the top plot we can see the original, uncoupled DMP trajectory (green dashed), which starts at a randomly chosen position close to the goal ($g = 0.5$) and close to the trajectory of the periodically moving object (black). Also shown is the trajectory of the DMP with both feedback and feed-forward coupling (dashed blue). The initial guess for the motion (green dashed), which was here predefined, but could have been acquired autonomously through generalization, has the correct frequency but too small amplitude. The bottom plot of Fig. 2 shows the error of the desired force for two scenarios: using only the feedback coupling $C = c_{\mathrm{fb}}$ (red dotted); and both feedback and feed-forward coupling $C = c_{\mathrm{fb}}+c_{\mathrm{ff}}$ (blue solid). We can see that the error is reduced approximately 25 times when also the feed-forward term is used.

The steady-state error depends on the value of $\epsilon$ in (13). A higher $\epsilon$, i. e., $\epsilon \approx 1$, reduces the steady state error, but also reduces the robustness of the system. To demonstrate the effect of varying $\epsilon$, Fig. 3 shows the results for the same experiment of maintaining constant contact with a periodically moving object, but with a considerably more complex waveform of object motion. Additionally, the signal of the position of the robot is noisy, with a maximal noise amplitude of 1 cm. Such noise is unrealistically high for the estimation of position through forward kinematics, but could be present if a vision system is used to estimate the position of the object. The top plot of Fig. 3 shows the waveform of the object motion (blue solid), while the bottom plot shows the effect of $\epsilon$ on steady-state error. Steady-state error is lowest when there is no noise (black dash-dot) and $\epsilon = 1$. The results overlap considerably in the transient part, i. e., from the beginning.

Increasing $\epsilon$ decreases the robustness of the learning algorithm, which might cause it to diverge, even after apparent convergence. This is not uncommon [23] and different methods of canceling out this phenomena exist; the easiest in reducing $\epsilon$, reducing the gain of the $L$ filter, see (13), or by cutting the learning after some time. The latter involves some heuristics in determining the time to cut the learning; observing that the error remains below a threshold for more than one period can be used as a criterion. Figure 4 shows the occurrence of apparent convergence, where the system still diverges after some time.

### A. Bimanual Coupling

We tested the approach in a real-world scenario where we used the arms of a humanoid robot to rotate a pedal racer fixed to a solid structure by its axes. Real measured force was used to modulate the trajectories of motion.

We used the 4 DOF arms of the compliant humanoid robot named COMAN [24]. COMAN approximates the size of a 4

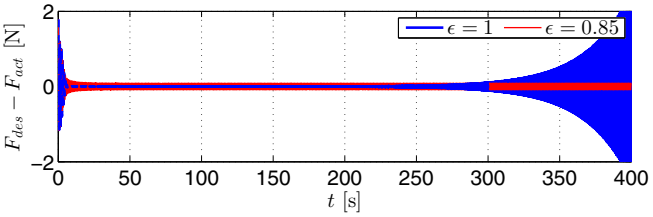



Fig. 4. The phenomena of apparent convergence appears when $\epsilon = 1$ was used, but not for $\epsilon = 0.85$.

Fig. 5. Image sequence showing approximately one period of rotating the vertically mounted pedal racer with the arms of the COMAN robot after the learning, when the contact with the pedals is constantly kept. The attached video shows the complete experiment. The tape on the wheels simulates contact with the ground and keeps them at the same velocity, preventing the device from locking up, which happens if the front and the back wheels do not rotate together.

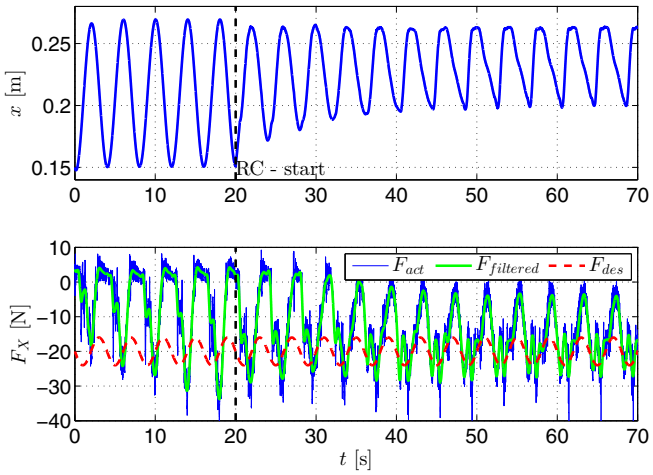



Fig. 6. Results of modifying the arm trajectories for a cooperative bimanual task of rotating the pedal racer with arms. The top plot gives the trajectory of the arm with $x$ denoting the distance from the body to the tip of the arm. The bottom plot gives the estimate of the measured force (blue), filtered force (green) and desired force (red dashed).

year old child, having 945mm from the foot to the center of the neck, and 312mm between the centers of the shoulders. It weights 31.2kg, out of which the legs and the waist module weigh 18.5kg. The complete robot has 23 DOF; each leg has 6: 3 at the hip, 1 at the knee and 2 at the ankle. The trunk is composed of a 3 DOF waist and the body, while each arm has currently 4 DOF, i.e. 3 in the shoulder and 1 in the elbow. Passive compliance based on series elastic actuation (SEA) is added to 14 of the 23 DOF including all flexion/extension DOF of the legs, the flexion/extension of the shoulders and elbows and the shoulder abduction/adduction. The robot is presented in Fig. 8.

The task of the algorithm was to maintain sinusoidal force trajectories on a vertically fixed pedal racer, as presented in Fig. 5. The experiment served to show the applicability of the approach in a real-world cooperative scenario, where the cooperation was between the two robot arms. It also served as a preparation for the task of actually operating the pedal racer by standing on it.

The initial trajectories of motion, encoded by DMPs were accurate for the up-down ($z$) direction, because the arms could not brace against any part of the pedal racer in the $z$ direction. Therefore only friction could be used to produce up-down force on the device. The forward-backward ($x$) motion was a sinusoidal with an estimated amplitude, deliberately set not to maintain contact with the device at all times. The robot was standing straight in front of the pedal racer with no balance control. Therefore, a person had to hold the robot (by the neck) so that it produced forces on the pedal racer and did not tip over when the arms made contact. Joint-torque sensors were used to estimate the force on the end effector. Forces were first estimated for pedaling in the air. These forces were deducted from the estimates during the experiment to estimate the force of contact. The measurements were extremely noisy.

Figure 6 shows the results for the left arm of the robot. The results for the right arm were similar, but in counter-phase. The experiment is presented also in the attached video. The learning was started after 20 s. We can see from the top plot that the arm trajectory changed. In fact, it maintained constant contact with the pedal racer when the learning was stopped. On the bottom plot we see that the measured force curve did change its shape, but not completely to the desired one. This is a direct consequence of three major reasons. The first is that the force measurement was only an estimate, valid for static conditions only. The second is that the robot was held by a person and therefore the periods were not completely repeatable. The third is in setting learning for greater robustness with $\epsilon = 0.7$ in order to cope with the first two reasons. The results show that the trajectories were modified to maintain constant contact with the device and that while not achieving perfect tracking, a general shape of the force profile was still observed.

## V. PEDAL RACER

We tested the proposed approach on the task of operating a pedal racer by standing on it. The task is harder than it seems even for people, and that is also how the device is marketed. The pedal racer moves forward if the pedals are moved in a circular fashion. Contact has to be maintained with both pedals, which imposes kinematical constraints. In one period of motion, the operator has to twice pass through singular configurations of the device by applying forces forward-backward using only friction to brace against the device. The experimental scenario demanded the robot to optimize the initial estimated trajectory of motion over several periods of learning in order to achieve the desired force profile.

We evaluated our approach in simulation using the Webots dynamic simulator [25], and we conducted preliminary experiments on the real robot.

### A. Balance Control

In order to have the robot maintain balance, we implemented a hierarchial control structure with several tasks. We used iterative inverse kinematics algorithms to control the center-of-mass (COM) of the robot and the position/orientation of the two feet. Both were implemented as primary tasks by stacking the COM Jacobian and the Jacobian of one foot with respect to the other in an augmented

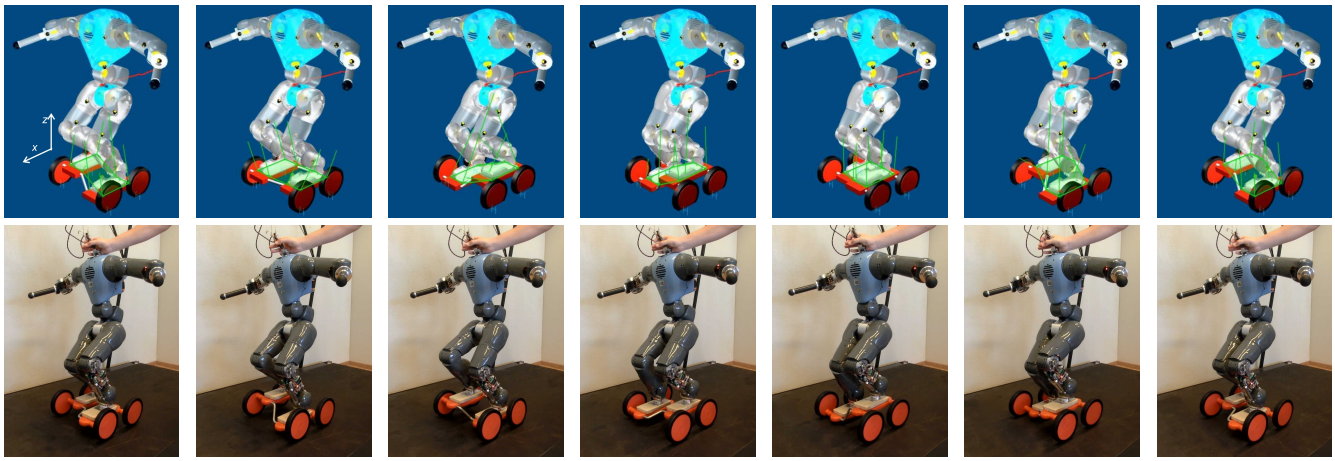


Fig. 8. In the top row an image sequence of one period of virtual COMAN steady-state pedaling on a virtual pedal racer in Webots dynamic simulator, after learning. The bottom row shows the real COMAN robot pedaling on the device. Note that the robot was helped to maintain balance at least once per period.

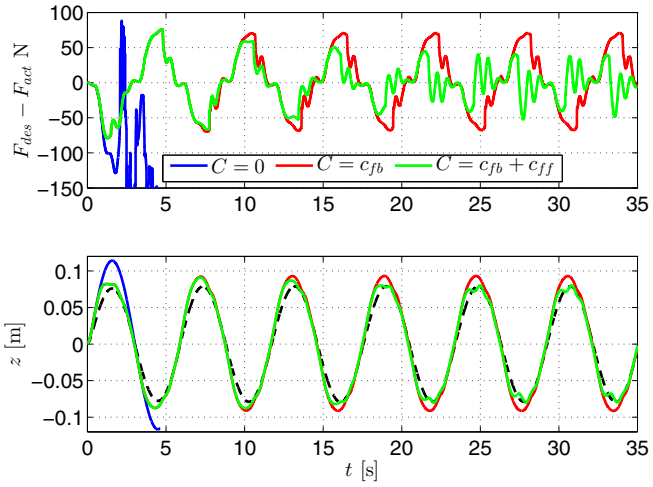



Fig. 7. Simulated results of operating the pedal racer with the robot standing on it. In the top plot the error of the desired force: no coupling (blue), feedback coupling (red), feedback and feed-forward coupling (green). We can see that the error of the force is considerably reduced when both feedback and feed-forward couplings are used. In the bottom plot the relative distance between the feet: first approximation (blue), feedback coupling (red), feedback and feed-forward couplings (green). Ideal trajectory is dashed black.

Jacobian. The control sets the COM to remain in the center between the feet. For details on the implementation of the controller refer to [26].

### B. Results

We conducted several experiments in simulation. The task was designed to maintain a desired periodic force profile on the feet of the robot. Each foot of the robot is equipped with a 6-DOF force-torque sensor.

In the manner of the approach in [21], we used the measured forces of a successful execution as the reference, obtained in simulation by maintaining the balance using a COM controller and providing the exact kinematical motion to the feet of the robot, which was perfectly aligned to the device. Our task was designed to adapt the vertical motion of the robots feet, starting from a reasonable first approximation (off by 4 cm). The approximation alone results in the robot tipping over. The phase relation between the horizontal and vertical movements was preset.

The plots in Fig. 7 show errors of force tracking for three scenarios: using no coupling (blue); using only feedback coupling (red) and; using both feedback and feed-forward coupling (green). The bottom plot shows the relative position of the feet for the same three scenarios and in the same colors. No coupling results in the robot tipping over. Feedback-only coupling results in higher errors and different trajectories than both feedback and feed-forward coupling. The bottom plot shows also the trajectory of referential, i. e., predefined pedaling (black-dotted).

Fig. 8 shows in the top row an image sequence of steady-state simulated pedaling after learning, executed in Webots dynamic simulator. The bottom plot shows the results on a real robot. The robot was pedaling on a treadmill; the frequency of the pedaling was calculated to keep the robot and the pedal racer in place. Both experiments are presented also in the accompanying video. Note that the robot had to be helped to maintain balance at certain times, at least once per period. The reason is in the discrepancy between the real robot and the model used to estimate the position of the center of mass, and in the compliance of the pedal racer, which would visibly bend under load.

### C. Discussion of the results

It is important to note that we designed the task to show the applicability of the proposed approach for a demanding task. While we show the potential of the proposed approach, other, possibly fundamentally different approaches approach might prove equally effective, perhaps more. The issue at hand is in the referential forces. Since the task of riding the pedal racer is demanding – the robot has to be stable, only friction is used to produce forces forward-backward, singular configurations have to be overcome when the feet are the most apart in the up-down direction – we mimicked the approach by Pastor et al. [21] and used a successful execution to acquire

the referential force trajectory. A successful execution is relatively easy to come by in simulation, but achieving the same on the real robot is complex in maintaining the balance of the robot. The latter is crucial for force measurements. It is evident in Fig. 8 that we had to intervene by hand to maintain balance at certain points of the period of motion.

Since the robot is standing on the pedal racer, any instability or perturbation will directly affect the forces. The robot boasts SEAs and is therefore slightly wobbly even when standing on solid ground. Any oscillation of the springs introduces forces which have an effect on the trajectory and also learning, adding to the acceleration of the robot, which results in additional forces, possibly leading to self-excitement of the trajectories. Some of this is evident also in Fig. 7.

## VI. CONCLUSIONS AND FUTURE WORK

While thus far mostly restricted to the kinematic domain, coupling the DMPs with force feedback extends their domain to include dynamic tasks. The combination of periodic DMPs, force-feedback and learned feed-forward coupling allows for generation and execution of previously unattainable motions, resulting in rich and adaptable periodic motor skills of a robot.

We used repetitive control, which can cope with the change of initial conditions of the start of a period, making it suitable for periodic tasks. The plug-in type RC with a feedback term can also cope with noisy and less-than-perfect repetitions through periods of motion. In the paper we only briefly touched the issue of the stability of the algorithm, relying on empirically set values to achieve and present initial results. In the future we will derive explicit stability criteria, just as was derived for discrete tasks in [19].

The complex real-world task of operating the pedal racer shows the potential of the proposed approach. In our scenario the trajectory of required motion to pedal on the device was only estimated and then optimized for the task through several periods of learning. Even though the final, learned trajectory resulted in operating the device, showing that a learning/adaptation approach is needed, it still leaves room for improvement of robotic pedal-racing in future work.

## REFERENCES


[1] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008, robotics and Neuroscience.

[2] S. Schaal, D. Sternad, R. Osu, and M. Kawato, "Rhythmic movement is not discrete," *Nature Neuroscience*, no. 10, pp. 1137–1144, 2004.

[3] R. Ronsse, D. Sternad, and P. Lefèvre, "A computational model for rhythmic and discrete movements in uni- and bimanual coordination," *Neural Computation*, vol. 21, no. 5, pp. 1335 – 13 370, 2009.

[4] S. Degallier and A. Ijspeert, "Modeling discrete and rhythmic movements through motor primitives: a review," *Biological Cybernetics*, vol. 103, no. 4, pp. 319–338, 2010.

[5] N. Hogan and D. Sternad, "Dynamic primitives in the control of locomotion," *Frontiers in Computational Neuroscience*, vol. 7, no. 71, 2013.

[6] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, Washington, DC, 2002, pp. 1398–1403.

[7] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[8] S. Khansari-Zadeh and A. Billard, "Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010, pp. 2676–2683.

[9] 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.

[10] D. Pongas, A. Billard, and S. Schaal, "Rapid Synchronization and Accurate Phase-Locking of Rhythmic Motor Primitives," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 2911–2916.

[11] 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.

[12] 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.

[13] L. Righetti, J. Buchli, and A. J. Ijspeert, "Dynamic hebbian learning in adaptive frequency oscillators," *Physica D*, vol. 216, no. 2, pp. 269–281, 2006.

[14] 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.

[15] L. Peternel, T. Petrič, E. Oztop, and J. Babič, "Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach," *Autonomous Robots*, pp. 1–14, 2013.

[16] 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.

[17] L. Cuiyan, Z. Dongchun, and Z. Xianyi, "A survey of repetitive control," in *2004 IEEE/RSJ International Conference on Robots and Systems (IROS)*, vol. 2, 2004, pp. 1160–1166.

[18] A. Gams, M. Do, A. Ude, T. Asfour, and R. Dillmann, "On-line periodic movement and force-profile learning for adaptation to new surfaces," in *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Nashville, TN, 2010, pp. 560–565.

[19] A. Gams, B. Nemec, L. Žlajpah, M. Waechter, A. Ijspeert, T. Asfour, and A. Ude, "Modulation of motor primitives using force feedback: Interaction with the environment and bimanual tasks," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5629–5635.

[20] 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*, 2013.

[21] 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.

[22] N. Gopalan, M. P. Deisenroth, and J. Peters, "Feedback Error Learning for Rhythmic Motor Primitives," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[23] R. W. Longman and Y.-C. Huang, "The phenomenon of apparent convergence followed by divergence in learning and repetitive control," *Intelligent Automation & Soft Computing*, vol. 8, no. 2, pp. 107–128, 2002.

[24] L. Colasanto, N. Tsagarakis, and D. Caldwell, "A compact model for the compliant humanoid robot coman," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS EMBS International Conference on*, 2012, pp. 688–694.

[25] Webots, "http://www.cyberbotics.com," 2012, commercial Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com

[26] A. Gams, J. van den Kieboom, F. Dzeladini, and A. Ijspeert, "Stable real-time full body motion imitation on the coman humanoid robot," in *Proceedings of the 22nd International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2013.

# Improved Iterative Adaptation Scheme with Recursive Estimation of Compensation Signals

Bojan Nemec[1], Tadej Petrič[1], and Aleš Ude[1]

*Abstract*— **In this paper we address the problem of how to improve the adaptation speed and the robustness of an Iterative Learning Controller (ILC) applied to robot control with force feedback. In contrast to standard ILC, which updates the feedforward control signal based on results from the previous cycle, our proposed controller updates the feedforward signal immediately. This results in significantly faster adaptation and consequently faster reduction of the tracking error. The feedforward control signals is encoded as a linear combination of radial basis functions. The proposed approach was evaluated in simulation and on a Kuka LightWeight Robot Arm where the task was to perform force-based surface following with both discrete and periodic movements.**

## I. Introduction

Iterative learning control (ILC) is often used in robotics due to its simplicity, effectiveness and robustness when dealing with repetitive operations. In industry as well as in home environments, there are many tasks that need to be executed repeatedly. In such cases, humans usually acquire and perfect the skill over several repetitions of the task. The same learning principle can also be adopted in machine motor control theory, where a system follows a similar trajectory repeatedly and updates necessary parameters in order to perfect the skill.

One possible method for improving the skill knowledge is iterative learning control. The general aim of ILC is to improve the behavior of the control system that operates repeatedly by iterative refinement of the feedforward compensation signal [1]. The ILC is also closely related to the Repetitive Control (RC), where the desired task trajectory is a periodic function of time, and without resetting between periods [2]. In contrast to the RC framework, the ILC system is designed to return to the same initial condition before each new execution of the task. The ILC framework has been successfully applied to many practical tasks that arise in manufacturing and robotics [3], [4], humanoid robotics [5], path tracking [6], medical robotics [7], health care robotics [8], and visual servoing [9]. Furthermore, the basic ILC schemes can be extended with methods that enhance their robustness [10], [11] and the adaptability of the overall system [12], [13]. The problem of the minimum variance learning has also been addressed within the ILC framework [14].

[1]Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stean Institute, Ljubljana, Slovenia, bojan.nemec@ijs.si, tadej.petric@ijs.si, ales.ude@ijs.si

In [15] ILC scheme was applied for motion coordination of a bimanual robot task using force feedback. In this case the dynamic motion primitives (DMP) framework was used for the representation of motion trajectories. The feedforward compensation signal was generated by the ILC. DMPs in conjunction with ILC were applied also in [16] to improve a force based assembly skill, where phase stopping algorithm was utilized to enhance the robustness of the overall framework. By replacing the time dependency in the ILC with the phase dependency, a standard ILC assumption of the equal duration of the trials could be removed. Using this substitution, ILC framework was successfully applied also for speed optimization of the demonstrated skills [17].

In this paper we propose a new learning controller that combines the ideas of standard ILC and on-line coaching [18], resulting in improved adaptation speed, robustness and ease of implementation. The structure of the proposed algorithm is closely related to the DMP framework. The proposed algorithm is general and can be used with both types of movements that can be represented by DMPs, i. e. discrete and periodic movements. The performance of the proposed algorithm was evaluated and compared with ILC schemes as proposed by our previous research [16], using both simulation and real robot experiments. Two experiments were performed, one showing the performance of the algorithm on a discrete task and the other on a periodic task, respectively.

The paper is organized as follows. In Section II we briefly outline the dynamic movement primitives for discrete and periodic tasks. In Section III, the classical ILC control scheme and how this scheme evolves into the newly proposed algorithm is presented. In Section IV the experimental results and the effectiveness of the proposed algorithm in comparison to the more standard ILC approaches is shown. Two different cases were considered 1) discrete tasks and 2) periodic tasks, both involving force interaction with the environment.

## II. Encoding trajectories with DMPs

In this section we give a brief overview of the underlying representation used in our approach. It is based on parametric description of trajectories and signals with dynamic movement primitives [19]. Within this framework, trajectories, given either in joint or in task space, are described as an output of the second order differential equation modulated with radial basis functions to get the desired shape. Every

degree of freedom is described by its own dynamic system, but with a common phase which synchronise them together.

For point-to-point movements (also referred to as discrete movements) the trajectory for each degree of freedom $y$ is described by the following system of nonlinear differential equations

$$\tau \dot{z} = \alpha_z(\beta_z(g-y)-z) + f(x), \quad (1)$$

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

$$\tau \dot{x} = -\alpha_x x, \quad (3)$$

where $x$ is the phase variable and $z$ is an auxiliary variable. $\alpha_x$, $\alpha_z$, $\beta_z$ and $\tau$ are defined such that the system converges to the unique equilibrium point $(z, y, x) = (0, g, 0)$. By setting $\alpha_z = 4\beta_z$, the linear dynamic system is critically damped.

The nonlinear term $f$ contains free parameters that are used to modify the dynamics of the second-order differential equation system to approximate any smooth point-to-point trajectory from the initial position $y_0$ to the final configuration denoted as goal $g$. The nonlinear term $f$ is given by

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \quad (4)$$

$$\Psi_i(x) = \exp\left(-h_i\,(x-c_i)^2\right), \quad (5)$$

where the given initial and final velocity are equal to zero. N radial basis functions with width $h_i > 0$ are equally distributed along the trajectory where $c_i$ are the centers. Weights $w_i$ are estimated with regression in such that the DMP encodes the desired trajectory.

For the periodic representation of the trajectories, the nonlinear differential equations take the identical form as for discrete signals given with (1) and (2), except for the phase, which is

$$\dot{x} = \Omega. \quad (6)$$

Here, $\Omega = 1/\tau$ denotes the frequency of an periodic signal. Note that in the case of the periodic representation the phase $x$ is a monotonically increasing function. Also the non-linear function $f$ has the same form as for discrete signals, except that the kernel functions $\psi_j$ are Gaussian periodic functions defined as

$$\psi_j(x) = \exp\left(\frac{1}{2\sigma_j^2}(\cos(x-c_j)-1\right). \quad (7)$$

Parameters $c_j$ and $\sigma_j$ respectively define the center and width of the $j$-th periodic basis function, while $w_i$ are adjustable weights used to define the desired shape of the trajectory.

One of the advantages of DMPs is that they can be modulated both spatially and temporally without changing the overall shape of motion. Ijspeert et al. [19] introduced a slow-down feedback, where the robot is automatically halted on excessive position error. In [16] this principle was used to slow down the trajectory execution on excessive forces and toques. Another benefit of the DMPs is the robustness against the sudden change of the goal variable $g$. Whenever a new goal is specified during the DMP evolution, the resulting trajectory is governed according to the response of the second order system.

## III. Recursive Regression Iterative Learning Controller (RRILC)

The output $y$ of the DMP enables us to follow the demonstrated trajectories. In many practical situation, it is necessary to change the output of the learned DMP to achieve the desired goal (see Section IV for some practical examples). In this paper we consider the situation where we add a control signal $u$ to the output of the DMP $y$, i. e. the positional signal for robot control is defined as

$$y = y_{\mathrm{DMP}} + u. \quad (8)$$

To learn the optimal control signal $u$, the widely used iterative learning control approach can be applied [1]

$$u_{l+1}(k) = Q(u_l(k) + e_l(k+d)), \quad (9)$$

where $l$ is the learning iteration index, $e_l(k)$ is the error signal that describes the difference between the desired and the actual task execution. See Section IV for some practical implementations of $e$. $Q$ and $L$ could be selected as discrete transfer functions that determine the ILC behaviour and the overall stability of the learning system. In many cases, including in our experiments, they are set to constant values. In the cases when the control system time delay is known in advance, a transient response of the system can be improved by setting parameter $d$ to a proper value, otherwise it is often set to 1.

Standard ILC assumptions include: 1) Stable system dynamics, 2) System returns to the same initial conditions at the start of each trial, 3) Each trial has the same length. Stability analysis is usually performed in a lifted time domain system [1]. Parameters may be tuned also using heuristic approach [20] where the choice of $Q$ is a tradeoff between the stability region and steady-state error. $Q$ in the form of low pass filter rejects disturbances and enhances the robustness [20].

ILC in the form (9) 'waits' one cycle before it modifies the control signal. To override this limitation, a feedback control signal is usually added to the plant resulting in

$$\begin{aligned} u_{l+1}(k) &= Ce_{l+1}(k) + Q(u_l(k) + Le_l(k+d)) \\ &= Ce_{l+1}(k) + s_l(k). \end{aligned} \quad (10)$$

This scheme is often referred to as 'current iteration' ILC [1]. The ILC feedforward signal $s_l(k)$ can be computed in advance after the competition of each learning phase $l$. Here we propose to encode the sequence $\{s_l(k)\}$ with radial basis functions, i.e.

$$\sigma_l(x) = \frac{\sum_{i=1}^{N} w_{l,i} \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)}, \quad (11)$$

where $\Psi$ is defined as in (5) in the case of discrete movements and as in (7) in the case of periodic movements. $x$ is the phase (3) used to drive the DMP. In this case, (10) turns into

$$u_{l+1}(k) = Ce_{l+1}(k) + \sigma_l(x(k)) \quad (12)$$

where $\sigma_l(x(k)) \approx s_l(k)$ provides the phase dependent feedforward compensation signal.

The benefits of encoding the feedforward signal using (11) are:

- Since the reference trajectory $y_{DMP}(k)$ is encoded with DMPs (1) – (3), the time dependency is removed form the ILC update and replaced with the phase dependency. Consequently, each learning trial can have different duration and assumption 3) changes to: Each trial has the same length in the phase domain. This enables us to modulate the execution speed according to the interaction with the environment, which enhances the robustness of assembly tasks [16].
- Signal encoded with RBF is close to the original signal passed through the low pass filter. Therefore, with RBF encoding of the ILC feedforward compensation signal we inherit the robustness of the overall system.

Lets rewrite the $s_l(k)$ term from the (10) into the form

$$\begin{aligned} s_l(k) &= Qu_l(k) + QLe_l(k+d) \\ &= Qs_{l-1}(k) + Q(Ce_l(k) + Le_l(k+d)). \end{aligned} \quad (13)$$

In continuous form (11), this expression becomes

$$\sigma_l(x(k)) = Q\sigma_{l-1}(x(k)) + Q(Ce_l(k) + Le_l(k+d)). \quad (14)$$

We can calculate the feedforward signal $\sigma_l(x)$ recursively with a recursive least-squares regression. This mitigates the need for saving discrete samples $s_l(k)$ needed to compute (10). Recursive encoding of $\sigma_l(x)$ also means that it can be used in the current learning cycle, similar to the coaching proposed in [18]. As $\sigma_l$ is now updated at each time sample, the learning cycle $l$ dependency can be deleted and we just write $\sigma(x)$ from now on. Note that at the time sample $k$, the feedback signal $e_l(k+d)$ is not yet available. Therefore, we can estimate the weights $\mathbf{w}_l(k) = [w_1, \ldots, w_N]^{\mathrm{T}}$ at $l$-th learning iteration step and $k$-th time sample only with some delay

$$\begin{aligned} \mathbf{w}_l(k{-}d) &= \mathbf{w}_l(k{-}d{-}1) + \big[Q\sigma(x(k{-}d)) + QCe_l(k{-}d) + \\ &\quad QLe_l(k){-}\mathbf{a}_l(k{-}d)^{\mathrm{T}}\mathbf{w}_l(k{-}d{-}1)\big]\,\mathbf{P}_l(k{-}d)\mathbf{a}_l(k{-}d), \\ \mathbf{P}_l(k{-}d) &= \frac{1}{\lambda}\Big(\mathbf{P}_l(k{-}d{-}1) + \\ &\quad \frac{\mathbf{P}_l(k{-}d{-}1)\mathbf{a}_l(k{-}d)\mathbf{a}_l^T(k{-}d)\mathbf{P}_l(k{-}d{-}1)}{(\lambda + \mathbf{a}_l^T(k{-}d)\mathbf{P}_l(k{-}d{-}1)\mathbf{a}_l(k{-}d))}\Big), \\ \mathbf{a}_l(k{-}d) &= \frac{1}{\sum_{i=1}^{N}\Psi(x_l(k-d))}\begin{bmatrix}\Psi_1(x_l(k{-}d))\\ \vdots \\ \Psi_N(x_l(k{-}d))\end{bmatrix}. \end{aligned} \quad (15)$$

Here $\lambda$ is the forgetting factor and $\mathbf{P}$ the covariance matrix, which is set at the beginning of the recursive estimation to a diagonal matrix with large values. Initial values of weights $\mathbf{w}_1(0)$ are set to 0 at the beginning of learning. The new control signal $u$ is thus given by (12), where the feedforward term $\sigma(x)$ is calculated using (11) and the corresponding weights $\mathbf{w}$ are calculated using (15). As we use recursive regression, we name this approach Recursive Regression Iterative Learning Controller (RRILC). For discrete tasks, the vector $\mathbf{a}_l(k)$ is formed using kernel functions according to (5). For repetitive tasks, kernel functions in vector $\mathbf{a}(k)$ are chosen according to (7). In the continuation of the paper we refer to the version of the algorithm with periodic kernels as Recursive Regression Repetitive Controller (RRRC).

The main difference between the standard ILC and RRILC is that RRILC calculates the control in the current cycle with the recursively updated estimator $\sigma$, thus exploiting the updates calculated in the current learning iteration $l$ immediately.

Since our approach belongs to the class of iterative learning control methods, we can use the standard lifted system framework to analyse the stability of the learning system. The stability analysis can be performed in a standard way (e.g. by applying a lifted system description) by selecting appropriate parameters $Q$ and $L$ [1].

## IV. Simulation and experimental evaluation

In this section we evaluate the performance of the proposed algorithm and compare it to the standard ILC as given by Eq. (10). As some features and benefits of the new algorithm are easier to show in simulation, we evaluate the methods both in simulation and in real experiments. In real experiments we applied a KUKA LWR robot equipped with Barrett hand. The robot was controlled from Matlab/Simulink environment using Fast Research Interface [21] at 100 Hz. Both in simulation and in real world experiments we used an identical setup, where the simulated dynamics of the robot and the environment were updated at 1 KHz. We selected low gains in the position controller of the KUKA robot, which results in compliant behaviour. The arm stiffness was set to $1000, 1000$ and $500$ N/m for translational axes and to $100$ Nm/rd for rotational axes, respectively. Such a high compliance is necessary when a robot is working in a non-structured environment or cooperating with humans.

### A. Force based surface following

In many industrial applications the robot is required to follow a previously unknown surface. Typical operations which involve this problem are polishing, grinding, cleaning, etc. This problem is relevant also for the future generation of home robots while performing task from everyday life such as cleaning the table, polishing furniture, etc [22]. Force control is needed to solve such problems. Clearly, if the robot is able to move around while precisely maintaining the contact force, the surface shape can be directly captured from the robot's motion. In our experiment, one robot was holding a paint roller in the hand. The task was to follow the previously unknown surface at constant speed of 0.15 m/s in $X$ direction while maintaining the constant force of 5 N in $Z$ direction. The shape of the object was a triangle as illustrated in Fig. 1. The search trajectory was defined as a straight line in $X$ direction and encoded with DMPs using (1) and (2). We applied the admittance stiffness force control

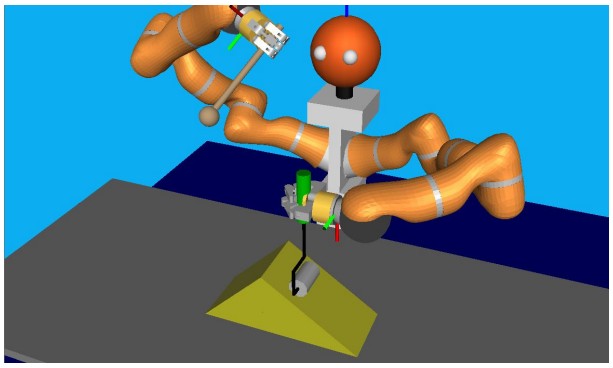


Fig. 1. Simulated environment

[23], combined with ILC, in discrete time form,

$$\begin{aligned}\mathbf{y}_c(k) &= \mathbf{y}_{DMP}(k) + \sum_{j=1}^{k} \mathbf{K}_{fi}\mathbf{e}_{l+1}(k) + \\ &\quad \mathbf{K}_{fp}\mathbf{e}_{l+1}(k) + \mathbf{s}_l(k), \\ \mathbf{e}(k) &= \mathbf{F}_d(k) - \mathbf{R}\mathbf{F}_m(k)), \end{aligned} \tag{16}$$

where $\mathbf{y}_c$ is the control position feed to the robot controller, $\mathbf{y}_{DMP}$ is the position as returned from the DMPs, $\mathbf{R}$ is the current robot rotation matrix, $\mathbf{F}_d$ are the desired forces, $\mathbf{F}_m$ are measured forces in the tool coordinate system and $\mathbf{K}_{fi}$ and $\mathbf{K}_{fp}$ are the force feedback matrices, respectively. ILC feed-forward signal $\mathbf{s}_l(k)$ was calculated as in (13), where error signal (16) was used for learning. $d$ was set to 3.

In RRILC, we applied the same control law as in (16), except that time dependent feedforward signal $\mathbf{s}_l(k)$ was replaced with the phase dependent RRILC generated signal $\sigma(x(k))$, calculated for each robot coordinate according to (11) and (12). The corresponding weights $\mathbf{w}$ were calculated using (15) for each Cartesian coordinate, where the error signal $e_l(k)$ was the corresponding component of the force error (16).

Five learning cycles were performed with ILC and RRILC. The forgetting factor $\lambda$ was set to 1, i.e., no forgetting was applied. After each learning cycle, we reset the covariance matrix to $10\mathbf{I}$. Parameters $\mathbf{K}_{fi}$, $\mathbf{K}_{fp}$, $\mathbf{L}$ and $\mathbf{Q}$ were set to $0.0001$ $\mathbf{I}$, $0.001$ $\mathbf{I}$, $0.001$ $\mathbf{I}$ and $0.99$ $\mathbf{I}$, respectively. Fig. 2 shows compensation signals, forces, positions and orientations of the robot as obtained in simulation. For the sake of clarity, only the results of the first and the fifth (last) learning cycle are shown. It is evident that RRILC outmatches the ILC especially in the first cycle. As it generates the feed forward compensation signal already in the first cycle, it converges considerably faster than the standard ILC. Note that the standard ILC lost the contact with the surface after 4 seconds. It can be easily verified that the controller (16) is not capable of tracking constant environment slope with zero force error in the first cycle, as the compensation term $s_1(k)$ is zero. Note also that the gains $\mathbf{K}_{fi}$ and $\mathbf{K}_{fp}$ were already set to the stability margin of the closed loop system. We can notice also smoother response of RRILC due to the smoothing provided by regression.

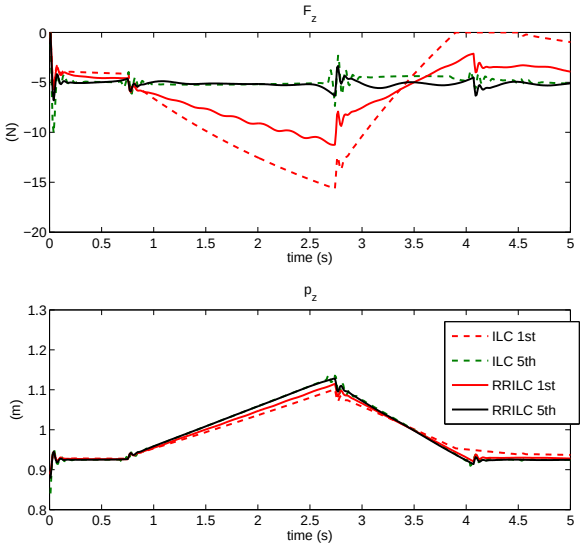



Fig. 2. Comparison of simulation results obtained with ILC and RRILC. Upper graph shows the force tracking in Z direction. Lower graph shows the learned Z coordinate of the surface

The same task was repeated in real environmen. The only difference was that the shape of the unknown object was selected as shown in Fig. 3. The object was made of thin aluminium plate, one part of the plate was firmly attached to the bottom, while the other was not. Therefore, during the surface following the environment stiffness was changing from very stiff at the beginning to very compliant at the end. Again, we compared the performance of the ILC and RRILC algorithm. The results are shown in Fig. 4. Also in the real experiment we can see that the newly proposed RRILC results in faster learning. Similar as in the simulated environment, also here the standard ILC has lost contact with the surface at the end of the first learning cycle. However, after five repetitions, the results were virtually identical with both algorithms, which is the expected behavior.

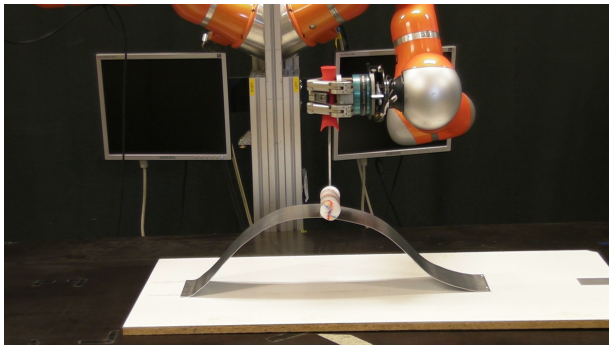


Fig. 3. Experimental setup

### B. *Stirring in a pot*

One of the most common cooking activities is stirring. The purpose is to mix liquid ingredients by moving a spoon (or similar tool) around the pot in a circular motion. In

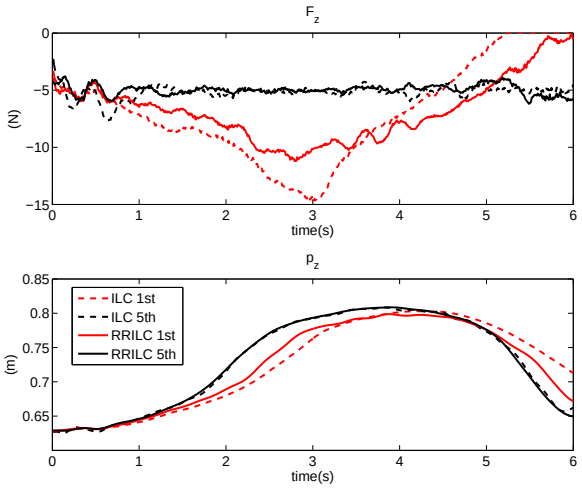



Fig. 4. Comparison of the real world experiments performed with ILC and RRILC. Upper graph shows the force tracking in $Z$ direction. Lower graph shows the learned $Z$ coordinate of the surface.

many cases it is necessary to apply small forces to the pot wall while stirring. Here, we consider a scenario, where the stirring motion was obtained by the demonstration for another pot with smaller diameter and elliptical shape. The desired forces were captured from human demonstration. The center of the pot was displaced in $X$ and $Y$ directions for 2 cm and 3 cm with respect to the center of the demonstrated motion, respectively. The task of the robot was to adapt to the new situation as quickly as possible. For this, we applied RC and RRRC and compared the performance of both algorithm.

The control algorithm in the time discrete form was

$$\begin{aligned} \mathbf{y}_c(k) &= \mathbf{y}_{DMP}(x(k)) + (K_f e_{l+1}(k) + s_l(k))\mathbf{r}(x(k)), \\ e(k) &= \mathbf{r}(x)^{\mathrm{T}}(\mathbf{F}_d(x(k)) - \mathbf{R}\mathbf{F}_m(k)), \\ \mathbf{r}(x) &= [\cos(x), \sin(x), 0]^{\mathrm{T}}, \end{aligned} \tag{17}$$

where $x$ denotes the phase angle calculated by (6), $\mathbf{y}_c$ is the commanded robot position, $\mathbf{y}_{DMP}$ the trained periodic DMP, $\mathbf{R}$ is the current robot rotation matrix, $\mathbf{F}_d$ are the desired forces obtained from user demonstration, $\mathbf{F}_m$ are the measured forces in the robot tool coordinate system, $e_r$ is the radial force error, and $K_f$ is the scalar force feedback gain. Transformation $\mathbf{r}(x)$ transforms the phase angle $x$ into a position on the unit circle at height $z = 0$.

The RC feedforward signal $s(k)$ corresponds to the radial displacement. It was estimated according to (13). $d$ was set to 7. Parameters $K_f$, $L$ and $Q$ were set to 0.001, 0.0005, and 0.99, respectively. For RRRC, the feed forward compensation signal $s(k)$ was replaced with the phase dependent RRRC generated signal $\sigma(x(k))$, calculated according to (11) and (15). The forgetting factor $\lambda$ was experimentally set to 0.992. The duration of the stirring cycle was 2 seconds and 8 stirring cycles were preformed. The results are shown in Fig. 5. It can be seen that the RRRC established full contact with the pot wall already in the second stirring cycle and proceeded with almost unchanged response for the following 6 cycles. On the other hand, RC took 6 stirring cycles to establish full contact with the pot wall. In the last cycle, learned compensation terms were almost identical for both algorithms. Fig. 6 shows the comparison of the learned path for both algorithms, measured at the wrist position. Due to the compliant orientational axes of the robot, this position slightly deviates from the end effector (spoon) position.

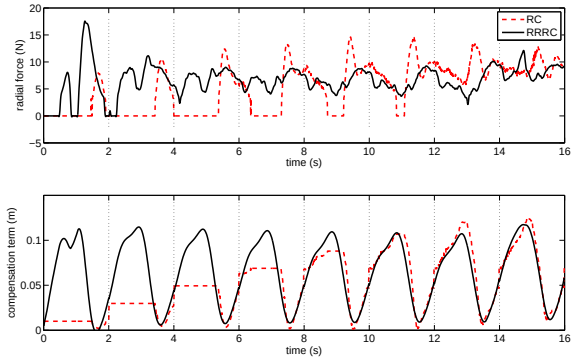



Fig. 5. Comparison of the experimental results for pot stirring obtained with RC and RRRC. Upper graph shows the force tracking in radial direction. Lower graph shows learned compensation term in radial direction. Dotted vertical lines denote stirring cycles.

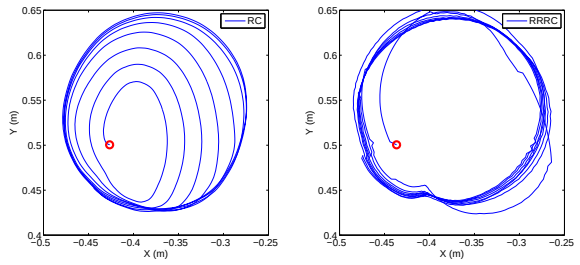



Fig. 6. Comparison of the learned trajectory for pot stirring obtained with RC (left) and RRRC (right).

Finally, we investigated how the motion can be adapted to a pot with non-smooth edges, e.g, to a square pot, as shown in Fig. 7. The reference trajectory and forces were the positions and forces learned in the last stirring cycle from the previous example. The stirring trajectory was slowed down by a factor of 2 by changing the $\tau$ variable in DMPs from 2 to 4. Again, we compared RC and RRRC learning algorithms. Results are outlined in Fig. 8. In this case, the RRRC learned faster, but the final learned offsets caused greater deviations from the desired forces than the RC algorithm. This is even more evident in Fig. 9, which shows the learned path measured at the robot wrist. Although the spoon followed the square pot wall in both cases, the wrist trajectories were substantially different due to the high compliance of the robot wrist. One possible reason for this defect might be excessive smoothing induced by the representation of the feedforward signal with RBFs.

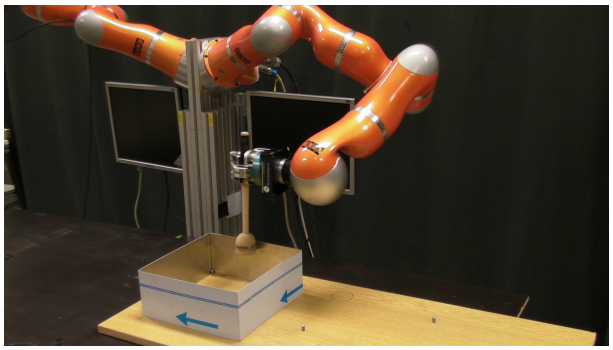


Fig. 7. Experimental setup for stirring in square pot.

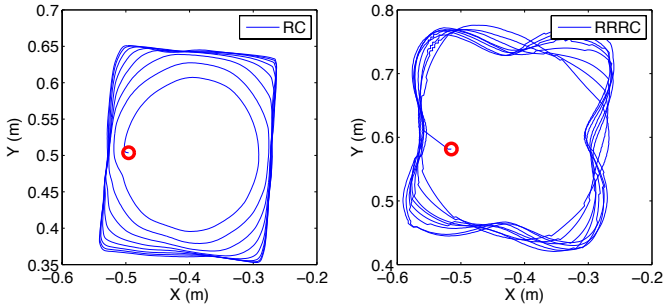



Fig. 9. Comparison of the learned trajectory for stirring in square pot obtained with RC (left) and RRRC (right). The plotted positions correspond to the robot wrist positions

## V. Conclusions

We proposed a new controller based on ILC and RC paradigms, which applies recursive estimation of the feed-forward signal for the minimization of the control error. It enables to compensate pure time delays in the controlled plant and generates smooth control signals. The proposed learning controller was verified both in simulation and in real experiments involving force interaction with the environment. Simulation and experimental results have shown that the main benefit of the new controller is the significantly improved speed of learning. In one experiment out of three we noticed that the standard RC learned better compensation signal than RRRC. However, RRRC converged faster, which opens new possibilities in combining both approaches. RRRC (or RRILC) could be used at the beginning of learning. After a few iteration the system could switch to the standard RC (or ILC) for final refinement of the behavior. We will investigate this possibility in the future.

In this work our aim was to learn the force-based skills with highly compliant robot applying low control gains, which are beneficial for robots working in cooperation with humans and in unstructured environments. In the future we will investigate also the behaviour of the proposed algorithm in torque-based force control schemes.

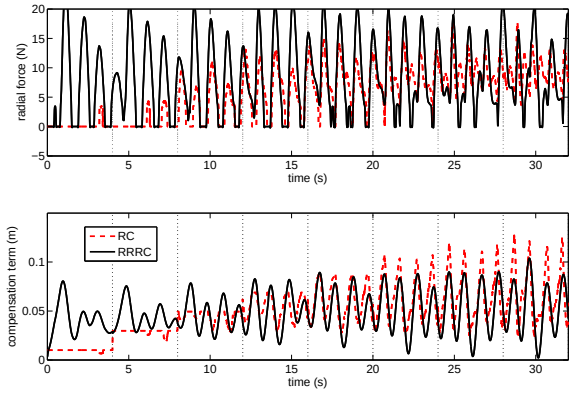



Fig. 8. Comparison of the experimental results for stirring in square pot obtained with RC and RRRC. Upper graph shows the force tracking in radial direction. Lower graph shows learned compensation term in radial direction. Dotted vertical lines denote stirring cycles.

## References


[1] 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.

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

[3] B. Bukkems, D. Kostic, B. de Jager, and M. Steinbuch, "Learning-based identification and iterative learning control of direct-drive robots," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 537–549, 2005.

[4] P. Cano Marchal, O. Sörnmo, B. Olofsson, A. Robertsson, J. Gómez Ortega, and R. Johansson, "Iterative learning control for machining with industrial robots," in *Preprints of the 19th IFAC Congress*, Cape Town, South Africa, 2014.

[5] P. A. Bhounsule and K. Yamane, "Iterative learning control for high-fidelity tracking of fast motions on entertainment humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, Georgia, USA, 2013.

[6] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.

[7] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010, pp. 2074–2081.

[8] C. Freeman, E. Rogers, A. Hughes, J. Burridge, and K. Meadmore, "Iterative learning control in health care: Electrical stimulation and robotic-assisted upper-limb stroke rehabilitation," *IEEE Control Systems Magazine*, vol. 32, no. 1, pp. 18–43, 2012.

[9] P. Jiang, L. Bamforth, Z. Feng, J. E. F. Baruch, and Y.-Q. Chen, "Indirect iterative learning control for a discrete visual servo without a camera-robot model," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 4, pp. 863–876, Aug 2007.

[10] M. Norrlöf and S. Gunnarsson, "Disturbance aspects of iterative learning control," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 1, pp. 87–94, 2001.

[11] S. Gunnarsson and M. Norrlöf, "On the disturbance properties of high order iterative learning control algorithms," *Automatica*, vol. 42, no. 11, pp. 2031–2034, 2006.

[12] M. Norrlöf, "An adaptive iterative learning control algorithm with experiments on an industrial robot," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 245–251, Apr 2002.

[13] A. Tayebi, "Adaptive iterative learning control for robot manipulators," *Automatica*, vol. 40, no. 7, pp. 1195–1203, 2004.

[14] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *40th IEEE Conference on Decision and Control*, Orlando, Florida, 2001, pp. 4463–4468.

[15] 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.

[16] B. Nemec, F. J. Abu-Dakka, B. Ridge, J. A. Jorgensen, T. R. Savarimuthu, J. Jouffroy, H. G. Petersen, N. Krüger, and A. Ude, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *16th International Conference on Advanced Robotics (ICAR)*, Montevideo, Uruguay, 2013.

[17] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, Georgia, USA, 2013, pp. 423–428.

[18] 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)*, Hong Kong, China, 2014, pp. 4770–4776.

[19] 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.

[20] M. Norrlöf and S. Gunnarsson, "Experimental comparison of some classical iterative learning control algorithms," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 636–641, 2002.

[21] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the kuka lightweight robot," in *ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, Anchorage, Alaska, 2010, pp. 15–21.

[22] C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[23] L. Villani and J. De Schutter, "Force control," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, pp. 161–185.

# Hierarchical Segmentation of Manipulation Actions based on Object Relations and Motion Characteristics

Mirko Wächter, Martin Do, Tamim Asfour[1]

***Abstract*— Understanding human actions is a indispensable capability of humanoid robots which have to acquire task knowledge from human observation and adapt this knowledge to new situations to be able to act and interact in the real world in a 24/7 manner.**

**To understand actions, we need to segment the human demonstration into meaningful actions while taking into account preconditions and effects of actions. In this paper, we present an approach for the observation of a human in motion as well as the objects and the environment he/she is interacting with during the demonstration. Human and object motions are captured using a marker-based tracking system, where markers are attached to the human body and objects of interest. Together with 3D object mesh models, a 6D object pose is obtained leading to an accurate representation of the scene and the changes in the world during the demonstration.**

**We propose a hierarchical segmentation approach which not only considers the human motion but also the relevant objects. Therefore, the proposed method segments on a semantic level as well as on a trajectory level. The segmentation results provide the necessary granularity of actions, which allows the association of observed and previously learnt segments as well as the sequencing of these segments to complex actions.**

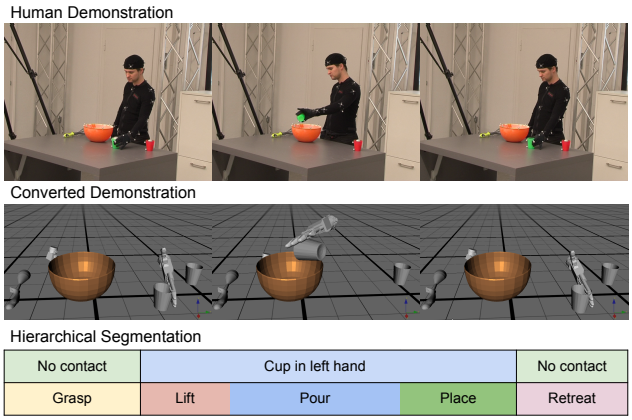



Fig. 1. A human demonstration of a complex task (top) is being recorded with a marker-based motion capture system. This marker-trajectories are converted into 6D object pose trajectories (middle), which serve as input for the proposed segmentation algorithms. The result of the segmentation (bottom) contains segments with distinct object-relations on the top-level and sub-segments with distinct motion characteristics on the bottom-level. The subsegments in this figure are labeled manually to illustrate the meaning of the subsegments.

## I. Introduction

Many research efforts in humanoid robotics have been dedicated to the development of sophisticated systems that can mimic the functionalities of a human. To tap this huge potential, humanoid robots are to be endowed with cognitive abilities for the acquisition of novel motor knowledge and the adaptation of this knowledge to unseen situations in order to account for dynamic changes. An intuitive way to approach this challenge is to acquire motor knowledge through the observation of humans and to transfer this knowledge to robots. In this context, an emerging paradigm is programming by demonstration, which in recent years progressed to the more biological-oriented term of imitation learning. A central concept which provides the basis for numerous imitation learning approaches has been the concept of the motion primitives. Motor primitives are units which incorporate a control policy for the execution of simple, basic motion patterns. Commonly, it is assumed that these motor primitives form the human motion repertoire from which complex movements are generated, adapting and sequencing these primitives in a task-dependent way. To provide data from which a robot can learn these motion primitives, methodologies have to be implemented to allow the automatic segmentation of continuous human motion data.

Therefore, a segmentation procedure which provides a sequence of reliable segmentation points based on which a movement can be divided in meaningful parts is crucial for the further processing of human observations. The found segmentation points should denote changes in the scene which are caused by the enclosed manipulation actions. However, such points are difficult to extract from mere human motion data. Therefore, we consider in this work also motion data from the manipulated objects. In addition, we also wish to determine smaller segments denoting known as well as unknown motormotion primitives depending on the elements the manipulation action is composed of.

To implement a segmentation method which satisfies the demands mentioned above and serves as an automatic tool to enrich a motion library with semantic information and more granular motion elements, in this work, a hierarchical approach is presented. On the higher level a semantic segmentation is performed based on the contact relations between the human end-effectors and the scene and between objects in the scene. To enable the capturing of these relations, a method has been developed which allows the robust and accurate acquisition of action data. On the lower level, the semantic

*The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement Nº 270273 (Xperience).

[1]Mirko Waechter, Martin Do and Tamim Asfour is with Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany {waechter,do,asfour}@kit.edu

segments are further studied in order to identify motion primitives. The proposed segmentation approach constitutes a crucial component in a motion learning framework.
The paper is organized as follows: Section II provides a brief overview of works related to this approach. In Section III, the acquisition of human motion data featuring demonstrations of complex manipulation tasks is described. In Section IV, the proposed action segmentation method procedure is introduced. As an application for the segmentation a recognition method of the found segments is described in Section V. The proposed approach is evaluated in Section VI. The work is summarized and notes on future works are given in Section VII.

## II. Related Work

In order to be able to understand and analyse complex human motor behaviors, demonstrations of these behaviors have to be decomposed into meaningful segments which denote manipulation actions as well as the corresponding action primitives. For this purpose, in the field of robotics, a large number of different approaches have been proposed mainly in the context of imitation learning. In general, segmentation algorithms can be categorized in unsupervised and supervised methods. Unsupervised methods do not require any prior knowledge of the actions which are featured in the behavior to be segmented, and, thus, a temporal segmentation of continuous human movements can be performed in an online manner. According to this methodology, in [1], an approach has been introduced based on the joint velocities. Segments are enclosed between points where the mean squared velocity falls below a predefined threshold. In [2], this method has been extended in a way that this critical threshold is determined based on the scaling of the current mean squared velocity. In addition, tactile feedback has been incorporated in order to detect changes in the contact relations between human and environment while the human is in motion. In [3], a method based on zero velocity crossings is proposed. Using this method, segments are denoted by points where in a sufficient number of joints the movement direction is changing. A probabilistic approach has been proposed by [4] using Principal Component Analysis in order to identify segments represented in a low-dimensional space. The reconstruction error for newly observed movements indicates whether the observation belongs to a current segment or denotes the beginning of a novel one.
In contrast to unsupervised approaches, the segmentation with supervised methods is based on previously known segments mostly represented in a generalized form. Regarding the learning of human actions, a common strategy is to use the same representation for the learning as well as segmentation and recognition of actions and motor primitives. In [5] and [6], segments are identified and represented as states of an Hidden Markov Model (HMM). In [5], the segmentation points are derived by optimizing the cost path using a modified Viterbi algorithm. To gradually verify and to refine the segmentation, prior knowledge is introduced in the form of HMMs representing known segmented movements which are grouped by applying a clustering procedure.
An alternative approach using dynamical systems is proposed in [7]. Linear time invariant dynamical systems are designed and trained to represent specific drawing primitives. For novel observations, parameters for these systems are estimated. Based on the parametric error and the corresponding approximation error segments are identified. In [8], latent force models are used in order to segment human movements. Multiple dynamical systems are used to encode the relations between a latent force space and the joint movements. Smooth trajectories in the latent force space indicate possible segments. However, the segmentation result strongly depends on the dimensionality of the force space. A further approach which uses the Dynamic Movement Primitives (DMP) for segmentation and recognition of basic movements is introduced in [9]. Based on a library of previously trained DMPs representing various basic actions, the observation of an action is encoded as a novel DMP. If this DMP matches an element in the library a segment is found. Otherwise, the novel DMP is used to update the library.
Compared to unsupervised methods, supervised approaches yield more accurate results for movements which are already known to the system. To ensure the robustness of these approaches the prior knowledge should consider segments which can be sufficiently discriminated, and, thus, found segments incorporate a certain complexity. Therefore, unsupervised methods are better suited for a fine-granular decomposition of a manipulation action in motion primitives, especially if the primitives are new to the system. However, finding the segmentation parameters such as thresholds which yield an optimal result are difficult to determine.

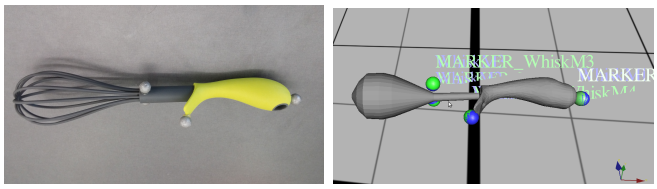



Fig. 2. Left: Image of a real object with attached reflective markers. Right: A visualization of the corresponding 3D mesh model with attached virtual markers(blue/green spheres).

## III. Action Data Acquisition

In this section, we will address how action data can be captured in a reliable way. In order to capture human demonstrations of a complex task which involves a variety of different manipulation actions with a high accuracy and at a high resolution, we employed a marker-based motion capture system [10]. The demonstrations involve not only a human, but also objects, which the human manipulates. To be able to capture both, markers have been attached to the human subject as well as to object of interests located in the current scene (see Fig. 2). For simplicity reasons, only the hands of the human are considered during the capturing process and are treated as rigid bodies similar to the objects. On each object and hand at least three markers are attached, although

more markers increase the robustness in case of occlusions. We will explain later in this section why three markers are needed. All markers are labeled and grouped according to the object they belong. The capture result data format contains Cartesian space trajectories of all markers. In our previous work [11], we used only the grouped marker trajectories for the segmentation. However, the arrangement of the markers do not sufficiently describe the shape of the object, and, thus, the pose of the object can not be inferred based on the mere marker positions. To deal with these shortcomings, we convert the trajectories from a marker representation to a 6D pose representation for each object. The first component of the representation are 3D mesh models. Thus, we create 3D mesh models of each object with a 3D object scanner [12] or a common 3D modeling tool and place virtual markers on the model (see Fig. 2). With this object representation it is possible to calculate the 6D object pose from three or more marker positions. We solve this point set registration problem with an approach from Besl [13]. In Fig. 3 the mapping of observed and virtual markers is visualized.

At least three markers per object are necessary since fewer markers lead to an ambiguous solution. In case of two markers, the object pose could be anything around the axis between the two markers. In case of one marker, the object could be rotated arbitrary around that one marker. To retrieve the 6D pose trajectory for an object, the transformation is calculated for each frame and applied to the base pose of the object, which is usually the identity. This conversion is done for all captured objects, which leads to an accurate representation of the scene during the demonstration. In Fig. 4 two moments of a captured demonstration are shown with the 3D mesh models and their 6D pose. In our applications, we use trajectories with at least three markers for each object in any frame.

## IV. Hierarchical Action Segmentation

For the hierarchical segmentation of captured action data, first, by means of 3D models, the captured 6D trajectories of the objects and the end-effectors are captured based on object contact relations. Subsequently each segment is processed further by applying a segmentation approach based on motion characteristics in order to identify the sequence of motion primitives.

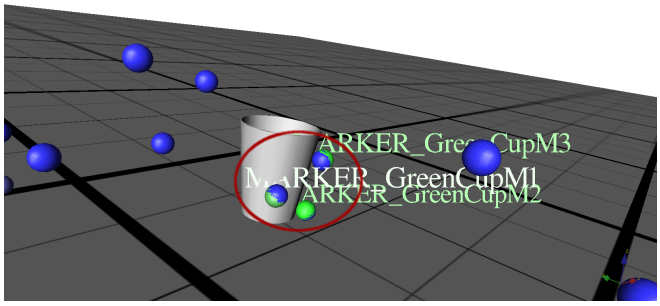


Fig. 3. Mapping of observed markers (blue spheres) attached to the cup to virtual markers on the model. Both, observed and virtual markers, are aligned (green/blue spheres in red circle). The visualization of the cup's 3D mesh shows the 6D pose of the perceived object.

### A. Semantic Segmentation based on Object Contact Relations

Based on the 6D pose trajectories of each object the demonstration is segmented. The segmentation grounds on the spatial relations between the objects. A similar approach was proposed by Aksoy et al. [14], [15], which uses RGB stereo camera images as input and is model-free, but does not provide 6D trajectories of any of the components. They estimate contacts between objects by recognizing overlapping color blobs. In our previous approach [11], we utilize marker distances only for contact detection. As mentioned in Section III, the shape and the pose of the object are not sufficiently represented by the markers alone. A segmentation based on the distances between the objects requires the use of high distance thresholds to detect contact points that are far from the markers. This reduces the robustness of such a method since objects in the demonstrations need to have a relatively large minimum distance between each other.

The introduction of a 3D mesh model as object representation instead of markers allows the use of sophisticated mesh-based collision detection algorithms [16] to accurately calculate the distance between objects and to detect contacts between them.

The demonstration is segmented by detecting key frames and similar to the approach presented in [17]. Key frames appear on relation changes of objects. We use only the relation $contact(A, B)$ (contact between object $A$ and $B$) since other relations like $on$ or $in$ ground on this relation and are not relevant for our segmentation method. For future extension towards planning the incorporation of further relations might be useful. The relation $contact(A, B)$ relies on the closest distance between any part of two objects. For every frame of the demonstration the relations between all objects are calculated and key frames stored whenever the relation between two object changes its status. $contact(A, B)$ returns true if the distance falls below a predefined threshold. To deal with noise on the distance measure, the threshold is increased when a contact has been detected in the last frame.

This results in a sequence of key frames. Additionally, the world state is stored with every key frame. A world state is the set of all relations between all objects. It describes the current status of the scene and can be used for association with known actions (as in [11]).

As stated before, every relation change leads to a key frame. But not all actions correspond to only one relation change. For example, the *pouring* action contains two relation changes (if the liquid can be tracked):

$$\begin{array}{rcl} contact(L,C) & & !contact(L,C) \\ \wedge \quad !contact(L,B) & \rightarrow & \wedge \quad contact(L,B) \end{array}$$

, where $L$ stands for liquid, $C$ for cup and $B$ for bowl.

Hence, key frames need to be merged into groups of key frames that belong semantically together. For most actions, these key frames appear within of a small delay between each other, e.g. *dropping* an object onto another. A simple

way to cope with this, is using the temporal displacement of two key frames to merge them.
State changes are always instantaneous, although e.g. *pouring* might seem to take time. However, the change of the contact relation does not take time. If the *pouring* would take noticably long, it would result in two key frames: At the first key frame the liquid get in contact with the target container and on the second the liquid loses contact with the source container.

### B. Segmentation based on Motion Characteristic

Up to this point, we segmented the human demonstrations in semantic segments, that have observable changes in the world state. Though, some actions have unobservable effects, even for a human. For example, the effect of shaking two transparent liquids in a bottle cannot be observed visually. In the case of observation with technical means the amount of unobservable effects is considerably higher with current state of the art methods. The previously described method can only detects moments when two objects make contact with each other. Thus, the aforementioned effect and therefore the state change cannot be detected. Therefore, we extend the segmentation from the previous section with a subsegmentation that extracts segments based on the trajectory shape. The goal of the subsegmentation is to split up the semantic segments into subsegments that contain motions with different motion characteristics and therefore potentially represents a different motion primitive. Hence, to capture the characteristic of a motion our approach uses as a basis the dynamics of the motion, i.e. the acceleration values of the trajectory.

There are two fundamentally different ways to segment motion data. One is to find key frames, that meet a specific criteria, and the other one is to search for meaningful segments. Our approach lies in between. The approach searches for key frames that maximizes the difference of the trajectory around it. It does not fit to the pure key frame search since the key frame itself is unimportant. However, it does not fit to the segment search either because it does not consider the complete segments. In short, our approach segments the trajectory in most distinctive parts.
To find the key frames, the trajectory is analyzed recursively. On every recursion level, the given trajectory segment is

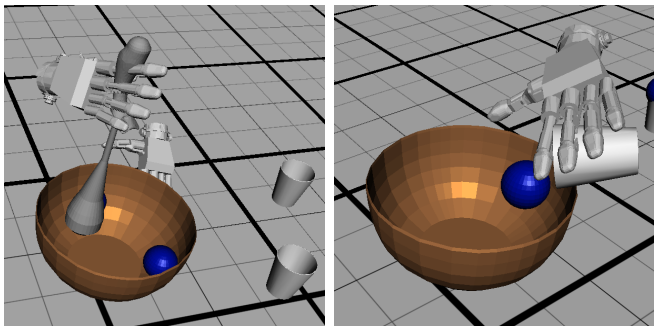


Fig. 4. Simulation of the observed demonstration while mixing with a whisk (left) and pouring from a cup (right).

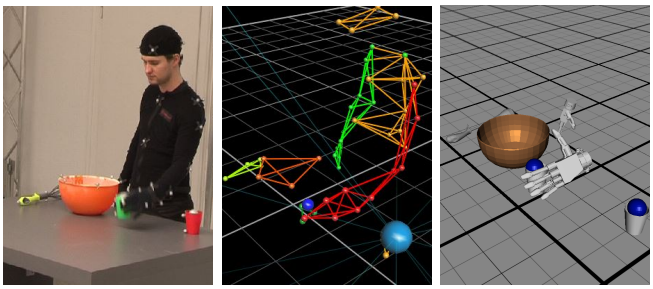


Fig. 5. All stages of the action sequence demonstration capture processing: (left) The human demonstrator with attached markers on the objects; (middle) marker group representation; (right) 3D mesh models with applied 6D object pose.

searched sequentially with a predefined step size for the key frame, that divides the trajectory best. Subsequently, the segments left and right of this key frame candidate are analyzed again in the same manner until the segment size falls below a threshold or no sufficiently good split has been found. This is also described in Algo. 1. The quality of a frame as a key frame is determined as follows:

$$d_{d,tc}(t) = \left(\ddot{f}_d(t) - \ddot{\bar{f}}_{d,t_c})^2 - (\ddot{f}_d(t+1) - \ddot{\bar{f}}_{d,t_c}\right)^2 \quad (1)$$

$$s_{l,d}(t_c) = \frac{\sum_{t=t_c-\frac{w}{2}}^{t_c-1} g_{t_c}(t) \cdot d_{d,tc}(t)}{\sum_{t=t_c-\frac{w}{2}}^{t_c-1} g_{t_c}(t)} \left(\frac{\hat{U}_l}{\hat{U}_r}\right)^2 \quad (2)$$

$$s_{r,d}(t_c) = \frac{\sum_{t=t_c}^{t_c+\frac{w}{2}-1} g_{t_c}(t) \cdot d_{d,tc}(t)}{\sum_{t=t_c}^{t_c+\frac{w}{2}-1} g_{t_c}(t)} \left(\frac{\hat{U}_r}{\hat{U}_l}\right)^2, \quad (3)$$

where $d$ is the dimension of the trajectory, $s_{l,d}(t_c)$ is the score left of the key frame candidate, $t_c$ is the timestamp of the key frame candidate, $w$ is the window size left and right of the key frame candidate, that is analyzed, $\hat{U}_l$ and $\hat{U}_r$ is the peak-to-peak amplitude left respectively right of the key frame candidate and $\ddot{f}_d(t)$ is the second derivation of the perturbation term in dimension $d$ at timestamp $t$. (2) calculates the score of the segment left of the key frame by calculating basically the length of the function. To produce a different value for high perturbation accelerations than for low values, the perturbation acceleration is squared. To normalize the values, the mean $\ddot{\bar{f}}_{t_c}$ of the perturbation function around the key frame candidate is subtracted. Further, the difference is weighted with the Gaussian function

$$g_{t_c}(t) = \frac{1}{\frac{w}{2}\sqrt{2\pi}} e^{\frac{-(t-t_c)^2}{2\frac{w}{2}^2}} \quad (4)$$

centered at the key frame candidate. Subsequently, the sum is normalized with the sum of all weights. To also consider the amplitude of the perturbation acceleration, the score is multiplied with the squared relation of the peak-to-peak

distances left and right of the key frame candidates.
The score $s_d$ of a key frame candidate is then:

$$s_d = \begin{cases} s_{l,d}/s_{r,d} & s_{l,d} > s_{r,d} \\ s_{r,d}/s_{l,d} & s_{l,d} \leq s_{r,d} \end{cases}. \quad (5)$$

Until now the scores for each dimension are normalized to their amplitudes. But the amplitudes of one dimension can be small compared to another dimension. To the end that motions in a dimensions with overall low amplitudes are not as important as another dimension with high amplitudes, the scores for each dimensions are aligned with the maximal peak-to-peak distance $\hat{U}_d$ of all dimensions:

$$\hat{s}_d = s_d \cdot \sqrt[3]{\frac{\hat{U}_d}{\max_d \hat{U}_d}} \quad (6)$$

The best $\hat{s}_d$ of all frames is selected as a key frame, if the value does not violate a quality-threshold $\lambda$ or a minimum segment size $s_{min}$ to avoid oversegmentation.

**Algorithm 1** Motion Characteristic Segmentation Algorithm

```
function FINDKEYFRAMES(kf,t_l,t_r,s_min)
    for t_c := t_l + s_min to t_r − s_min ; t_c += 0.01 do
        for d := 0 to dimensions do
            s_n ← CALCSCORE(t_c, d)
            if s_best < s_n then
                s_best ← s_n
                t_best ← t_c
            end if
        end for
    end for
    if s_best > λ then
        kf.INSERT(t_best,s_best)
        FINDKEYFRAMES(kf,t_l,t_best,s_min)
        FINDKEYFRAMES(kf,t_best,t_r,s_min)
    end if
end function
```

## V. Recognition

In order to facilitate future segmentation tasks and to enrich novel motion segments with additional information, feature vectors which represent the motion segments are extracted, labeled, and stored. Based on a collection of these feature vectors, a novel motion segment can be recognized. In the context of imitation learning, we are not only interested in labeling a new observation. To allow the replacement of motor knowledge and the transfer of task-relevant information between actions, we are rather interested in identifying alternative actions representations which match the observation. For this purpose, a lower-dimensional feature vector $\boldsymbol{\phi}_m$ is proposed which provides a generalized representation of a motion segment $m$. To describe $m$ invariant of the position and the direction of the movement, we only consider the acceleration profile $\dot{v}_m$ of the motion segment. $\dot{v}_m$ is normalized by dividing by the total duration $T_m$ of $m$. Furthermore, for a robust comparison between different motion segments describing the same primitive action, we have to diminish the influence of possible offsets and outliers within the motion on the recognition result. For this purpose, $\boldsymbol{\phi}_m \in \mathbb{R}^{3\times K}$ is represented in the form of a sequence of $K$ principal components which approximate the normalized acceleration profile. Thus, to obtain $\boldsymbol{\phi}_m$, we sequentially apply a PCA algorithm. According to this methodology, a projection matrix $P$ is computed for series of frames $V_k = \{\dot{v}_m(t_s), \ldots, \dot{v}_m(t_s + T_m)\}$. For a subsequent frame $\dot{v}_m(t_s + T_m + 1)$, the reconstruction error $e_k$ is evaluated which is calculated as follows:

$$e_k = \frac{1}{T_m} \sum_{t=t_s}^{t_s+T_M+1} \|\dot{v}_m(t) - (P\dot{v}_m(t)P^T)\|. \quad (7)$$

If $e_k < e_m$ where $e_m$ is a predefined error threshold, $\dot{v}_m(t_s + T_m+1)$ is added to $V_k$. In the case of $e_k \geq e_m$, $\dot{v}_m(t_s+T_m+1)$ denotes the first frame of new series $V_{k+1}$ and the first principal component which corresponds to the first column of $P$ is added to $\boldsymbol{\phi}_m$. To compare two motion segments $m_1$ and $m_2$ represented as $\boldsymbol{\phi}_{m_1} \in \mathbb{R}^{3\times K_1}$ and $\boldsymbol{\phi}_{m_2} \in \mathbb{R}^{3\times K_2}$ where $K_1 < K_2$, we determine a score $s_m$ which is calculated using the dot product between the principal components in $\boldsymbol{\phi}_{m_1}$ and $\boldsymbol{\phi}_{m_2}$. Thus, $s_m$ can be be computed as follows:

$$s_m = \frac{1}{K_1} \sum_{i=1}^{K_1} s_m(i) \quad (8)$$

$$s_m(i) = \underset{n\in I_i, j\in I_i\cap I_{K_2}}{\operatorname{argmax}} \frac{1}{j-n}(\boldsymbol{\phi}_{m_1}(i) \cdot \boldsymbol{\phi}_{m_2}(j)) \quad (9)$$

where $I_{K_2} = \{1, \ldots, K_2\}$ and $I_i$ is a set which contains all the indices of principal components in $\boldsymbol{\phi}_2$ which have been found to be similar to one of the element in $\{\boldsymbol{\phi}_{m_1}(1), \ldots, \boldsymbol{\phi}_{m_1}(i-1)\}$. Besides the similarity of the principal components, the score $s_m$ also considers the order of the principal components which form the subsequence in $\boldsymbol{\phi}_{m_2}$ and presumably corresponds to $\boldsymbol{\phi}_{m_2}$.

## VI. Experiments

In this section, the experimental setup for data acquisition is described and the results of conversion of marker positions into 6D object poses is discussed, the segmentation based on object contact relations, the subsegmentation based on shape characteristics, and the recognition of segments with other labeled segments.

### A. Experimental Setup

The marked-based motion capture system consists of 10 cameras for the observation of the scene, in which all objects are rigid common household objects that had at least three markers in an asymmetric arrangement attached to them. Regarding the human agent, only markers at the hand were considered where the hands were treated as rigid bodies as well. The motion capture system contains a marker relation models of all objects allowing the automatic labeling of the markers (see Fig. 5, middle). For every object, a 3D mesh model was either created with a 3D scanner or by hand and extended with virtual marker positions (see Fig. 2).

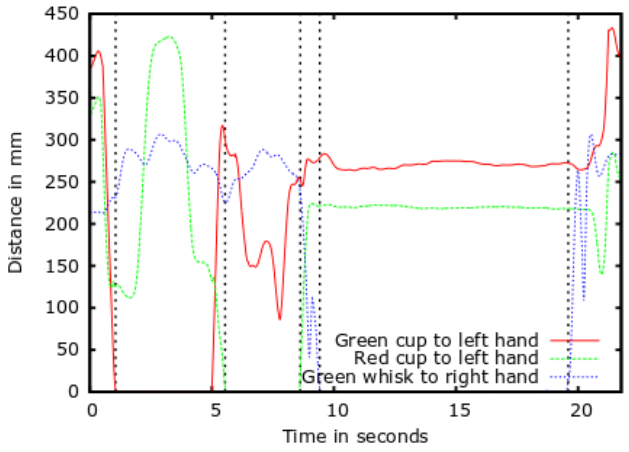



Fig. 6. Segmentation based on object contact relations: When two objects get in contact with each other (contact in this case is approximated as $distance < 5mm$) or lose contact, a new key frame is inserted with the current world state attached to it. The dotted vertical lines depict the detected segments. Only distances between objects that get in contact during the complete demonstration are shown.

### B. Experiments

To test our approach we recorded different scenarios of action sequences: *preparing batter*, *wiping a table* and *shaking and pouring a bottle's content into a bowl*.

*Preparing batter* contains two cups, which are grasped and its content poured into a bowl and then placed again on the table. Afterwards the liquids are mixed with a whisk, which also has to be grasped and placed on the table. This scenario was chosen because it contains several objects and typical actions for a household robot. In Fig. 6 the semantic segmentation of one trial of this scenario is depicted. The distances between objects that do not matter for this segmentation task are omitted in this diagram for better clarity of the figure.

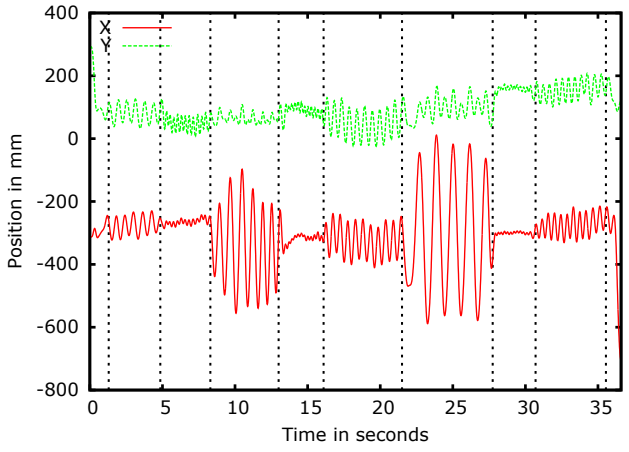



Fig. 7. Subsegmentation of one semantic segment (sponge touches hand and table) into different subsegments, i.e. different wiping styles. The z-dimension is omitted in the figure. The dotted vertical lines depict the segmentation points. Whenever the wiping styles changes, a key frame was inserted.

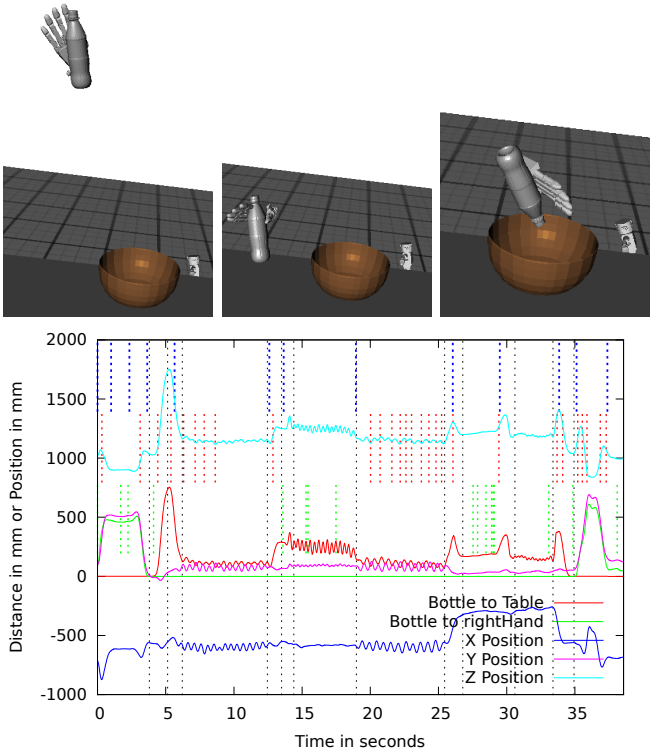



Fig. 8. Bottom: Comparison of manual segmentation (black lines), PCA (red lines), Zero-Velocity-Crossing (green lines) and the proposed approach (blue lines) of an action sequence shown by the key frames (vertical lines). The action sequence contains grasping, placing, shaking, tossing, pouring, inspecting and dripping off of a bottle. Top: Snapshots of the action sequence for visualization. The vertical lines are only partly drawn for better clarity.

Whenever the left or right hand grasps an object or puts down an object a new key frame is inserted. In the *table wiping* scenario the human demonstrator grasps a sponge from a table and wipes this table with several different wiping styles like intensive wiping of a spot or wiping of a big area with circles. The results of the motion characteristic segmentation are demonstrated in Fig. 7.

In the third scenario, a bottle is being grasped, tossed, inspected, shaken, poured and dripped off. In Fig. 8 the results of the hierarchical segmentation for this scenario in comparison with manual segmentation, Principal Component Analysis (PCA) and Zero-Velocity-Crossings (ZVC) are shown. The two types of input values are both depicted in the figure. The distance between objects serves as input for the semantic segmentation and the position values are the input for the other methods. The blue vertical lines contain key frames from both levels of the hierarchical segmentation. It can be seen that the proposed algorithm provides significantly better values than the other two approaches. Further, most of the key frames of our approach are close to the manual extracted key frames.

The difficulties in these scenarios lie in the unobservable effects of some of the actions, e.g. *pouring* and *mixing* since the liquids are not tracked. Therefore, the actions cannot be detected with the semantic segmentation based on object-

relation changes and a detection on motion characteristic level is required.

### C. Evaluation and Discussion

*1) Data acquisition:* Evaluating the precision of the object tracking is difficult since there is no ground truth data. However, in general, the algorithm will produce exact solutions if all input data is not contaminated with noise or error. The precision depends on the following components:

- Positional precision of the markers. With the used capture system, the deviation lies around one millimeter if there is no missperception.
- The position of the virtual markers. This is in practice the part for introducing an error, since the marker are placed by hand on the object in a 3D modeling tool.
- The 3D model of the object. Errors in the modeling of the object affect the contact detection of the objects.
- Markers on non-rigid bodies like hands do not have constant relative positions to the other markers on the object if the object is transformed and therefore impede the calculation of the 6D pose. The marker registration algorithm tries to minimize the error if there are more than three markers. However, an error will persist. In our application, we treat the hands as rigid bodies for simplicity as the error is small enough to deal with.

*2) Semantic segmentation:* Compared to our previous work [11], the segmentation results showed great improvements. The segmentation in the previous approach has been done solely on marker positions, which did not represent the object shape sufficiently and therefore contact detection was not reliable. A high threshold was needed for an approximated contact detection, which easily lead to false-positives. This deficiency has been alleviated by incorporating the region around every key frame. With the new model-based approach this is not needed anymore and the contact detection threshold can be chosen as 1/15 of the old threshold (150 mm to 10 mm).

In general, the new segmentation relies strongly on the precision of the 6D trajectory and the model associated with it. An inaccurate model can lead to false or missed contact detection and therefore to false segmentations. The precision is from our experience sufficient to detect all contacts and segment the trajectory into their semantic parts.

In Fig. 6, the result of an action sequence segmentation is depicted. An action sequence of pouring with two different cups and a subsequent mixing action was performed and captured. The contact between the objects can be extracted from the distance-curve and, thus, the key frames are inserted. For the sake of clarity, several distance curves between the objects have been omitted. In certain cases, a lost contact does not mean that a new actions starts. For example, during the wiping action, the sponge occasionally loses contact with the table. Based on the assumption that actions have minimum duration (we set 500 ms as a threshold), this situation is avoided to a certain degree by merging key frames that follow closely after another.

*3) Subsegmentation based on Motion Characteristic:* The subsegmentation tackles the problem of unobservable effects of actions. Additionally, different styles of a periodic action can be detected (e.g. wiping in lines or intensive wiping on one spot). In Fig. 7, a segmented action of wiping is shown, which is then subsegmented in different wiping styles. Each time the wiping pattern changes, a new key frame is inserted. In Fig. 8, the further inspection and segmentation of a semantic segment which supposedly represents a pouring action indicate these segments comprise more actions without observable effects, and, thus, can be further divided into subsegments. In our experiments, all segments have been detected, though a smooth transition between two actions can be problematic and no key frame might be found.

*4) Recognition:* As depicted in Fig. 9, the proposed feature representation from Section V has been applied to compare various segments which have been found in complex wiping and shaking/pouring tasks. As depicted in Fig. 9, the left figure shows that the proposed method is sufficiently discriminative. It can be observed that a higher similarity has been determined for motion segments which feature the same action. An interesting outcome of the proposed method is that motion segments representing periodic and discrete actions such as a specific wiping motion and the approach/retreat movement can be clearly distinguished. The same applies for the center figure where motion segments of two different wiping demonstrations have been compared to each other. By comparing the wiping demonstration to the shaking/pouring demonstration, except for *Tossing-Retreat* movements, almost no correspondences have been found. However, the trajectory-shape of the motion pairs are similar and the found correspondences suggest that information encoded within the *Tossing* movement can potentially be used to enrich a motion segments which represents a *Retreat* action.

## VII. Conclusion

In this work, a hierarchical segmentation approach has been presented which allows the reliable determination of semantic key points in continuous human movements. These points denote clear transitions between different manipulation actions where the human changes the scene. To detect these changes, a robust method has been implemented which enables the robust and accurate pose estimation of objects and the human hands, which lead to the detection of contact relations between these entities based on mere marker-based motion capture data. Some effects of actions are not observable by technical means and are therefore missed by the semantic segmentation. To this end, we proposed a new method for subsegmenting the semantic segments based on motion characteristics, which is a completely different metric and therefore supplements the semantic segmentation well. As an application the segmentation is used together with a new feature matrix for motion description to recognize the parts of an action sequence. In our evaluation, we showed

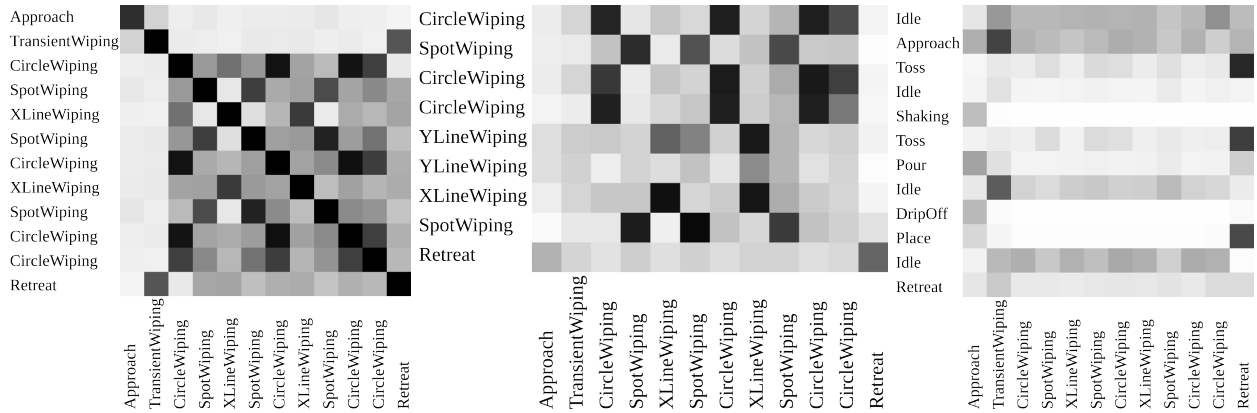



Fig. 9. Left: Comparison of a segmented observation of a wiping demonstration with itself. A black square indicates a perfect similarity where a white square denotes that the compared segments do not match. Center: Comparison of segmented observations of two different wiping demonstrations. Right: Comparison of segmented observations of a wiping and shaking/pouring demonstration.

that the proposed approach allows the identification of meaningful segments in unknown complex human demonstrations without over-segmentation nor the negligence of important key points. Due to the hierarchical approach, the found segmentation points are enriched with additional information which can be useful for the organization, the sequencing, and the reproduction of learned actions and motion primitives. Based on the feature matrix, in future works, we wish to identify similar motor primitives in order to allow the enrichment of unknown primitives with object and action affordances and semantic information, and, thus, to enhance the robotic learning of actions from human observation. Furthermore, we are interested in how primitives can be replaced in order to accomplish the same task. This knowledge might enable a planner to find new plans for a future task.

## REFERENCES


[1] M. Pomplun and M. J. Mataric, "Evaluation metrics and results of human arm movement imitation," in *Proceedings, First IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 2000.

[2] J. Lieberman and C. Breazeal, "Improvements on action parsing and action interpolation for learning through demonstration," in *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, vol. 1. IEEE, 2004, pp. 342–365.

[3] A. Fod, M. J. Matarić, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous robots*, vol. 12, no. 1, pp. 39–54, 2002.

[4] J. Barbic, A. Safonova, J. Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *GI '04: Proceedings of Graphics Interface 2004*. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 185–194.

[5] D. Kulic and Y. Nakamura, "Scaffolding on-line segmentation of fully body human motion patterns," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 2008, pp. 2860–2866.

[6] 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. 2, pp. 183–202, December 2008.

[7] D. Del Vecchio, R. M. Murray, and P. Perona, "Decomposition of human motion into dynamics-based primitives with application to drawing tasks," *Automatica*, vol. 39, no. 12, pp. 2085–2098, Dec. 2003.

[8] M. Alvarez, J. R. Peters, N. D. Lawrence, and B. Schölkopf, "Switched latent force models for movement segmentation," in *Advances in neural information processing systems*, 2010, pp. 55–63.

[9] F. Meier, E. Theodorou, and S. Schaal, "Movement segmentation and recognition for imitation learning," in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[10] J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive control of avatars animated with human motion data," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 491–500.

[11] M. Wächter, S. Schulz, T. Asfour, E. Aksoy, F. Wörgötter, and R. Dillmann, "Action sequence reproduction based on automatic segmentation and object-action complexes," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 189–195.

[12] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models web database: An object model database for object recognition, localizatino and manipulation in service robotics," in *The International Journal of Robotics Research*, 2012.

[13] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[14] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," in *ICRA*, 2010, pp. 398–405.

[15] E. E. Aksoy, A. Abramov, J. Dörr, N. Kejun, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *The International Journal of Robotics Research (IJRR)*, 2011.

[16] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Technical Report TR99-018, Department of Computer Science, University of North Carolina, Tech. Rep., 1999.

[17] M. Mühlig, M. Gienger, and J. J. Steil, "Human-robot interaction for learning and adaptation of object movements," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, 2010, pp. 4901–4907.