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

Deliverable Number:	D4.1.1
Deliverable Title:	Cooperative Tasks (I)
Type (Internal, Restricted, Public):	PU
Authors:	Aleš Ude, Tamim Asfour, David Schiebener, Bojan Nemec, Denis Forte, Christian Böge and Rüdiger Dillmann
Contributing Partners:	JSI, KIT

Contractual Date of Delivery to the EC:	31-01-2012
Actual Date of Delivery to the EC:	06-02-2012

Contents

Chapter 1

Executive Summary

The core focus of WP4.1 is to study cooperative tasks among multiple agents. We distinguish between tightly-coupled cooperative manipulation tasks, in which a direct, physical interaction between multiple agents must take place in order to achieve the task goal, and loosely-coupled interaction tasks, where multiple robots are engaged in scene interpretation and reasoning. In loosely-coupled interaction each agent is assigned to independently perform one or more subtasks that together lead to the attainment of the goal of the overall task.

The Xperience project is concerned both with learning at the level of motor primitives and with the application of the learned motor primitives to accumulate new knowledge, which is then used to bootstrap further learning. In D4.1.1 we describe our work on tightly- and loosely-coupled interaction in this context. We focused on the following topics:

In Section 2.1 we explain how a library of movement primitives can be used for tightly-coupled interaction, where the initial motor knowledge was acquired by coaching [FGMU11], [6]. The key here is to enable real-time generation of movement primitives that take into account the external perturbations caused by contact with another agent.

In Section 2.2 we present our approach to tightly-coupled physical interaction where a human and a humanoid robot collaborate to carry a heavy object.

In Section 2.3 we study the problem of object learning by dual-arm manipulation. Here a humanoid robot learns an object representation by repeatedly pushing it with one or the other hand. The pushing behavior was again learned by coaching. Using both hands makes it easier for the robot to acquire object views from different viewing directions while keeping the object in the center of its workspace.

The above three problems are quite different. The first two belong to the class of tightly-coupled interactive tasks. In this context we investigated how to generate motor primitives from the accumulated training data and how to parameterize the learned primitives in real-time with respect to the incoming sensory feedback. On the other hand, the problem described in Chapter 2.3 belongs to the class of loosely-coupled interactive tasks. With such tasks we normally end up with a decision problem rather than feedback control problem. The goal is to increase the utility of cooperative actions as opposed to actions performed by a single agent. In our practical experiment each arm of a humanoid robot was considered as an independent agent.

To enable faster acquisition of sensorimotor knowledge we focused on coaching to acquire the initial motor skills. In our current work coaching mainly takes the form of kinesthetic guiding, where a human coach guides a robot through a number of example trajectories (e. g. when learning reaching and grasping movements) or through a number of postures (e. g. when learning pushing movements). This is a master-slave architecture with the human coach as the leader and the robot as the follower. In all cases we combined the acquired motor information with the simultaneously acquired sensing data relevant for the task. For example, in the case of pushing the training data contained not only joint configurations of the robot arm but also the locations of the object to be pushed. The locations were acquired by the robot's active vision system simultaneously with the arm configurations. Such sensing data enables the robot to

generalize the training data to new situations that arise at execution time in a task-specific manner. In the current version the human coach manually specifies which sensing data is relevant for the task. In the future we plan to explore how to automatically select the appropriate sensing data to index into the library of the accumulated sensorimotor knowledge.

The performed research is described in the following three chapters and in two paper attached to this deliverable. It is primarily the result of our work in WP4.1. It is based on representations developed in WP2 (learning sensorimotor representations) and provides data for WP3 (generative mechanisms) as well as results for the demonstration workpackage WP5.

Chapter 2

Description of Results

2.1 Tightly-Coupled Interaction: On-line Generalization of Movement Primitives

Scenarios that involve tightly-coupled human-robot interaction require that a robot generates new movements on-line as new situations arise. This is only possible if the robot can generalize the previously acquired sensorimotor knowledge in real-time. Here we propose a methodology that enables the robot

1. to learn movement primitives from a number of example trajectories, and
2. to select and adapt the appropriate primitive in real-time (within the robot's sensory feedback loop) and with respect to the current task configuration.

We developed a system where a human instructor teaches the robot new movements by physically guiding it through a number of example trajectories (see Fig. 2.1). We applied the following control law to enable kinesthetic guiding on a torque-controlled robot

$$\tau_c = \mathbf{h} + \mathbf{J}^T \mathbf{f}, \tag{2.1}$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{h}$ the term combining centrifugal, Coriolis and gravity forces, $\mathbf{f}$ the vector of external forces acting on the manipulator's end-effector, and τ_c the commanded torques. With this control law, a force-controlled robot can accurately follow the motion of the coach. In many practical cases we can omit the second term from Eq. (2.1). Note that we assumed that the dynamics of the robot is known. While it is possible to acquire inverse kinematics or inverse dynamics models by learning [9], this was not necessary on the system (Kuka LightWeight Robot arm) we used in our practical experiments.

Based on our previous work [12] we proposed a new approach that enables interactive learning and execution of motor primitives. In [12] we proposed a statistical approach for the generalization of motor primitives where raw trajectories were used as input for generalization. However, the resulting computational processes were too expensive to allow for on-line modification and switching of movement primitives. In our newly developed approach we therefore first reduce the dimensionality of the input data, which we achieve by encoding the training trajectories as dynamic movement primitives (DMPs) [7, 8]. Based on DMP representation we developed a methodology for statistical generalization using

Figure 2.1: Coaching via kinesthetic guiding

Gaussian process regression [11]. The parameters describing the task are employed as query points into the trajectory database. We showed on real-world tasks that the proposed methodology can be integrated into a sensory feedback loop, where the generalization algorithm is applied in real-time to adapt the robot motion to the perceived changes of the task configuration. Technical details about this work can be found in the paper attached to this deliverable [FGMU11].

In practical experiments we showed how a robot can modify or even switch the motor primitives for reaching and grasping in order to account for perturbations caused by contact with a human. To be able to react to external perturbations and thus ensure the safety of the person collaborating with the robot, the robot needs to monitor the external forces while executing the desired motion. We applied the following control law to track a trajectory in the joint space

$$\tau_c \quad = \quad \mathbf{H}(\ddot{\mathbf{q}}_d + \mathbf{K}_v\dot{\mathbf{e}} + \mathbf{K}_p\mathbf{e}) + \mathbf{h}, \tag{2.2}$$

where $\mathbf{H}$ is the inertia matrix, $\ddot{\mathbf{q}}_d$ are the desired joint accelerations, $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$ is the joint space tracking error, $\mathbf{K}_v$ and $\mathbf{K}_p$ are the corresponding positive definite gain matrices, and the rest of the parameters are defined as above. The stiffness of the robot is specified by an appropriate selection of gain matrices $\mathbf{K}_v$ and $\mathbf{K}_p$. When the external forces perturbing the motion become too large, the robot is not able to follow the desired trajectory any more and starts deviating from the planned movement. Once this happens the initial movement primitive is discarded and a new dynamic system optimal for the new task configuration is generated. Since the calculations can be done in real-time, the robot is able to start using the newly calculated DMP immediately. In the same way we also enabled the switching between movement primitives, which we demonstrated in a practical experiment. In this experiment the robot learned how to grasp an object form two different sides. The selection of the appropriate movement for grasping from left or right side was done by taking into account the current location of the end-effector. If the end-effector moves from one side of the object to another, the robot can immediately switch to the different movement primitive appropriate for grasping from the opposite side.

Besides in the attached paper [FGMU11], this work was also presented at ICRA Workshop on Autonomous Grasping [5] and at Humanoids 2011 [6]. A video showing our experimental results has been submitted with deliverable D5.2.1.

2.2 Tightly-Coupled Interaction: Cooperative Carrying

In this work we study cooperative manipulation tasks, in which two agents (two arms, robot-robot, human-robot) collaborate to carry big and/or heavy objects. Apart from new methods for motion and grasp planning and control, we also aim at developing new approaches for cooperative manipulation tasks and intuitive physical human-robot interaction and evaluating them on humanoid robots. Our focus is on learning cooperative interaction motion primitives during physical interaction and endowing robots with the strategies for human motion prediction in physical human-robot interaction tasks. Such prediction strategies will enable the robot to work proactively with the human. Our hypothesis stems from the observation that, in human-human collaborative tasks, each human constantly predicts the other's motion. Based on motion prediction of the other person, the human can decide whether to lead him or to follow him. Models for prediction of human motion during such tasks will be acquired from both 1) cooperative interaction motion primitives from human motion capture data and 2) sensorimotor experience consisting of proprioceptive, force and tactile information during cooperative task execution.

Figure 2.2: Coaching of the humanoid robot ARMAR-III via kinesthetic guiding

Once the robot has acquired a model of human's motion, it can start behaving as a leader and proactively perform the next action based on its prediction. If the robot is not able to correctly predict human's motion, it will comply reactively with the human. This strategy will allow for a continuous and dynamic adjustment of the leader/follower role of the robot when acting as partner in cooperative tasks.

Based on our previous work on complaint interaction based on impedance control [10], combining visual and force information for physical interaction tasks [16] and visual servoing [15], we conducted experiments on tightly coupled cooperative tasks on humanoid robot ARMAR-III [1]. In particular, we investigated and implemented

- Kinesthetic guiding of the humanoid robot ARMAR-III based on interaction forces applied by the human. Based on applied forces, whole body motions of the robot were generated exploiting the kinematics redundancy of the robot (see Fig. 2.2).

  A video showing guiding of ARMAR-III has been submitted with deliverable D5.3.1.

- Master-slave architecture with the human as the "leader" and the robot as the "follower" in the context of object carrying task (see Fig. 2.3). In the experiments, we relied on proprioceptive and force information, which is measured by 6D force-torque sensors mounted in the writs of the robot. The forces resulting from the interaction between the object and the robot were used to infer the direction of motion of the human (leader) and mapped to motion commands of the robot.

  A video showing the implemented cooperative human-robot carrying of a long big object has been submitted with deliverable D5.3.1.

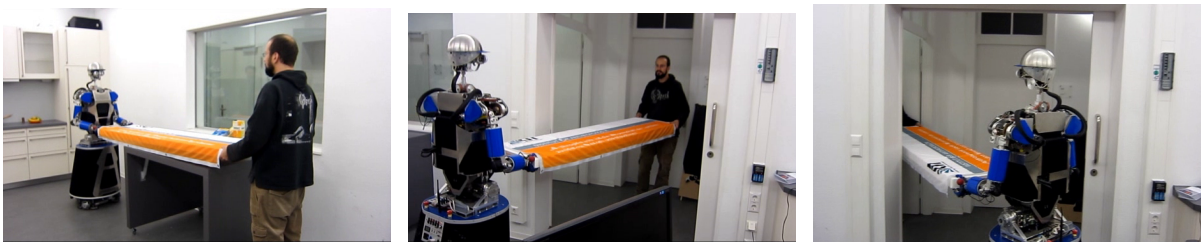

Figure 2.3: Cooperative carrying of big objects: motion of the robot is generated based on the 6D force-torque information measured in the wrists.

2.3 Loosely Coupled Interaction: Bimanual Object Learning

This work demonstrates how cooperation between two independent agents (here each of the humanoid robot's arms is assumed to be an independent agent) can improve explorative learning. It was conducted as part of the research that aims at showing how objects can be defined and learned if the concept of object manipulability is taken into account [3]. Based on the concept of object manipulability, we can define objects as physical entities that are manipulable by a robot and whose features move consistently when the robot moves the object. The current status of our research on this issue can be summarized as follows: the robot starts generating initial object hypotheses from the extracted 3-D points, which are obtained through stereo vision, by detecting regular surfaces, e. g. planes and cylinders, amongst them. The hypotheses are then statistically verified, corrected and extended by pushing them repeatedly. In this way we can segment objects from the background without having any prior information about their appearance. Object representations are learned using bag-of-features histograms of the SIFT descriptors (other descriptors are also possible) of the points belonging to the object, as well as color histograms of the area spanned by those points. We have shown experimentally that the objects learned this way can later be recognized, and that the segmentation by pushing can serve as a powerful methodology for recognition in complex scenes. The underlying visual processes are described in the attached paper [SUM+11]. Pushing actions for each arm were learned by kinesthetic guiding and are described in [14] (paper attached to deliverable D2.3.1). In the following we explain how to increase the robustness of the learning process by utilizing both arms of a humanoid robot for pushing.

We tested the concept described above on a torque-controlled humanoid robot [2]. However, one problem we often encountered in our experiments was that the learning process can get stuck when the object is pushed to a certain location from where the robot cannot push it to a new location (using the available pushing skill). This problem can be resolved either by

1. training additional types of pushing movements to increase the versatility of the available pushing behavior,
2. walking to a different location from where the root can push the object using the currently available pushing skill, or
3. regularly switching the pushing arm based on the suitability of each arm to push the object at its current location.

In this work we chose the third option, which required the robot to learn pushing with both arms. This allows the robot to keep the object in a central area in front of it. To this end, the object is always pushed towards a central point, but with sufficiently long pushes that its position oscillates around that point rather than converging at it. There are many possible criteria that can be used to select the appropriate arm for pushing in order to achieve such behavior, but it turned out that checking the side where the object is currently located with respect to the robot body is sufficiently effective. That means that if the object is to the right of the target position, which is defined to lie directly in front of the robot, the right arm is used and vice versa. Although it is not completely trivial to calculate this information because both the trunk and the head of the robot are allowed to move, we could nevertheless calculate it using active 3-D vision. A suitable calibration process has been suggested in [13]. The developed bimanual pushing strategy was sufficient to reliably push the object several times without any human intervention, which provided the robot with a sufficient number of different object views to learn a view-idenpendent representation of satisfactory quality for reliable object recognition.

In this way we provided an important new explorative behavior for the Xperience project. Note that the developed system requires that many different robot behaviors are synchronized and work in unison towards the same goal: the head and eyes need to move to keep the object in the center of the robot's viewfield; based on information acquired by active vision a decision process must decide which arm to use for pushing; the previously learned pushing behavior of the selected arm needs to be initialized based on information acquired by active vision and executed in the current configuration of the external world; the motor control system must take into account the possibility of collision with the environment; example images need to be acquired at the appropriate time and the object features need to be segmented from the background. It is thus clear that longterm (and even more so lifelong) explorative learning requires an ample set of robust sensorimotor skills.

Even though our current implementation of object learning already includes many different behaviors, some are still missing. For example, we have not yet used touch information to detect the moment when the pusher hits the object, which would allow us to determine the appropriate length of the pushing movements. Accumulating new sensorimotor behaviors and improving the available ones need to be a constant topic of Xperience. Our current system could obviously be improved by including additional behaviors such as the above option 1 (learn additional pushing strategies) and 2 (move to a different

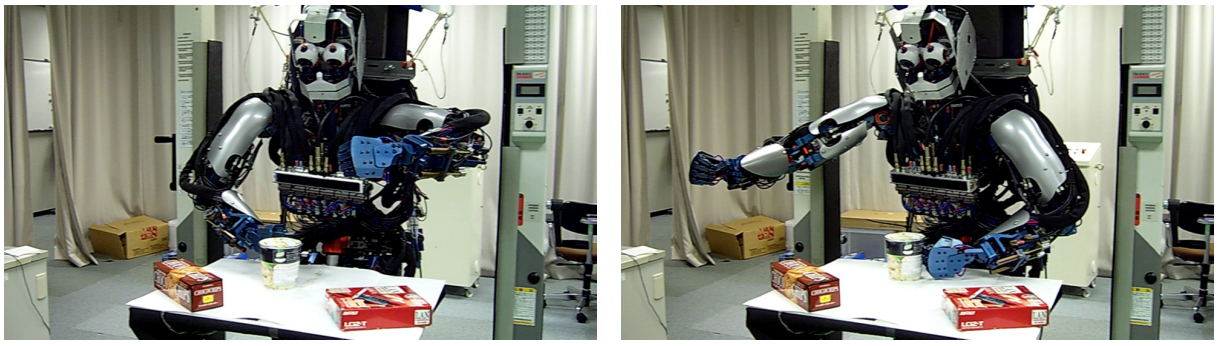

Figure 2.4: Pushing the object to be learned with two hands. The appropriate hand is selected based on the current position and orientation of the object relative to the robot base coordinate system.

location by walking). Moreover, we can envision a learning process in which the pushes are planned so that they induce object motion that maximizes the amount of information that can be gained from the next object view, which the robot acquires after the push. This type of planning process was studied for example in [4] in the context of walking. Our work in the next year of the project will explore these possibilities.

A video showing the implemented bimanual object learning process has been submitted with deliverable D5.3.1.

References

[1] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In *Proc. IEEE-RAS International Conference on Humanoid Robots*, pages 169–175, Genoa, Italy, 2006.

[2] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen. CB: a humanoid research platform for exploring neuroscience. *Advanced Robotics*, 21(10):1097–1114, 2007.

[3] J. Feldman. What is a visual object? *Trends in Cognitive Sciences*, 7(6):252–256, 2003.

[4] T. Foissotte, O. Stasse, P.-B. Wieber, A. Escande, and A. Kheddar. Autonomous 3D object modeling by a humanoid using an optimization-driven next-best-view formulation. *International Journal of Humanoid Robotics*, 7(3):407–428, 2010.

[5] D. Forte and A. Ude. Responding to perturbations by switching grasp strategies. In *ICRA Workshop on Autonomous Grasping - From Multi-Sensing to Symbolic Representations for Robotic Manipulation Tasks*, Shanghai, China, 2011.

[6] D. Forte, A. Ude, and A. Gams. Real-time generalization and integration of different movement primitives. In *Proc. IEEE-RAS International Conference on Humanoid Robots*, pages 590–595, Bled, Slovenia, 2011.

[7] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 958–963, Lausanne, Switzerland, 2002.

[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1398–1403, Washington, DC, 2002.

[9] D. Nguyen-Tuong and J. Peters. Model learning for robot control: A survey. *Cognitive Processing*, pages 319–340, 2011.

[10] M. Prats, S. Wieland, T. Asfour, A. P. del Pobil, and R. Dillmann. Compliant interaction in household environments by the Armar-III humanoid robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots*, pages 475–480, Daejeon, Korea, 2008.

[11] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Massachusetts Institute of Technology, 2006.

[12] A. Ude, A. Gams, T. Asfour, and J. Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, pages 800–815, 2010.

[13] A. Ude and E. Oztop. Active 3-D vision on a humanoid head. In *Proc. 14th Int. Conf. Advanced Robotics*, Munich, Germany, 2009.

[14] A. Ude, D. Schiebener, H. Sugimoto, and J. Morimoto. Integrating visual processing and manipulation for autonomous learning of object representations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Saint Paul, MN, 2012 (to appear).

[15] N. Vahrenkamp, C. Böge, K. Welke, T. Asfour, J. Walter, and R. Dillmann. Visual servoing for dual arm motions on a humanoid robot. In *Proc. IEEE-RAS International Conference on Humanoid Robots*, pages 208–214, Paris, France, 2009.

[16] S. Wieland, D. Gonzalez-Aguirre, N. Vahrenkamp, T. Asfour, and R. Dillmann. Combining force and visual feedback for physical interaction tasks in humanoid robots. In *Proc. IEEE-RAS International Conference on Humanoid Robots*, pages 439–446, Paris, France, 2009.

Attached Papers

[FGMU11] Denis Forte, Andrej Gams, Jun Morimoto, and Aleš Ude. On-line motion synthesis and adaptation using a trajectory database. Submitted to *Robotics and Autonomous Systems*, 2011.

[SUM+11] David Schiebener, Aleš Ude, Jun Morimoto, Tamim Asfour, and Rüdiger Dillmann. Segmentation and learning of unknown objects through physical interaction. In *Proc. 2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 500–506, Bled, Slovenia, 2011.

On-line Motion Synthesis and Adaptation Using a Trajectory Database

Denis Forte[a,*], Andrej Gams[a], Jun Morimoto[b], Aleš Ude[a,b]

[a] *Jožef Stefan Institute, Department of Automatics, Biocybernetics, and Robotics, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[b] *ATR Computational Neuroscience Laboratories, Department of Brain Robot Interface, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan*

Abstract

Autonomous robots cannot be programmed in advance for all possible situations. Instead, they should be able to generalize the previously acquired knowledge to operate in new situations as they arise. A possible solution to the problem of generalization is to apply statistical methods that can generate useful robot responses in situations for which the robot has not been specifically instructed how to respond. In this paper we propose a methodology for the statistical generalization of the available sensorimotor knowledge in real-time. Example trajectories are generalized by applying Gaussian process regression, using the parameters describing a task as query points into the trajectory database. We show on real-world tasks that the proposed methodology can be integrated into a sensory feedback loop, where the generalization algorithm is applied in real-time to adapt robot motion to the perceived changes of the external world.

1. Introduction

In this paper we investigate the problem of real-time, goal-directed trajectory generation using a database of example movements. This problem has received a considerable amount of attention in recent years [4, 7, 12, 22]. It has often been studied as part of programming by demonstration systems [5, 3]. Our primary interest is in real-time synthesis of new trajectories using local methods. For the purpose of this paper, real-time is defined to be the frequency comparable to a typical camera stream, i. e. 30 Hz. The initial acquisition of training movements has been performed by sequentially guiding the robot through a set of example trajectories, but automatic approaches to segmentation from a long sequence of example movements is possible [11].

It has been shown by Ude et al. [22] that it is possible to generalize the movements collected in an example database to new situations by utilizing the

*Corresponding author

goal of an action as a query point into the database. In [22] movement generalization was implemented by employing a combination of locally weighted regression [2] and Gaussian process regression [16], where raw trajectory data was used as input for generalization. The proposed approach was applied to generalize various behaviors including reaching, throwing, and drumming. While this approach can take into account external perceptual feedback to generalize example movements to different situations, its computational cost is prohibitive for use in a real-time feedback loop. The goal of this paper is to provide an approach that is efficient enough to be applied in such a loop.

The approach described in [22] uses Dynamic Movement Primitives (DMPs) [10, 9] as the basic representation for the encoding of robot movements. DMPs have many useful properties such as a built-in ability to react to perturbations without introducing discontinuities in the resulting robot motion. As an autonomous representation, they are not directly dependent on time, which makes it easy to stop the execution of movement without extensive bookkeeping of time evolution [18]. DMPs can also be extended to include capabilities such as obstacle avoidance [14] and avoidance of joint limits. All these adaptations can be done in real-time, which enables the robot to react to external sensory feedback. In this paper we expand on such built-in abilities by providing a methodology for real-time generation of DMPs based on a trajectory database. Thus we provide means for on-the-fly, task-specific adaptation of motion.

One DMP can encode one specific robot trajectory. In case of point-to-point (discrete) movements, the trajectory of each robot degree of freedom y (given either in joint or in task space) 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. α_x, α_z, β_z and τ need to be specified in such a way that the system converges to the unique equilibrium point $(z, y, x) = (0, g, 0)$. The nonlinear term f contains free parameters that enable the robot to follow any smooth point-to-point trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{k=1}^{N} w_k \Psi_k(x)}{\sum_{k=1}^{N} \Psi_k(x)} x, \;\; \Psi_k(x) = \exp\left(-h_k \left(x - c_k\right)^2\right). \quad (4)$$

Here c_k are the centers of radial basis functions distributed along the trajectory and $h_k > 0$. Weights w_k are estimated so that the DMP encodes the desired trajectory. For robots with many degrees of freedom, each degree of freedom is represented by its own equation system (1) - (2), but with a common phase (3). We used the algorithm described in [22] to determine the placement, width, and number of radial basis functions Ψ_k.

Recently, Gribovskaya et al. [7] proposed an alternative approach based on dynamic systems that can encode a complete class of movements. In their

approach, a class of movements is represented by a more general nonlinear system of differential equations

$$\dot{\mathbf{y}} = \mathbf{h}(\mathbf{y}). \tag{5}$$

Note that the phase information is not available in this approach, which is problematic in case of intersecting velocity fields, where the same robot configuration $\mathbf{y}$ is associated with more than one velocity $\dot{\mathbf{y}}$. Nevertheless, unlike DMPs, this representation is not limited to one specific movement but can represent more complete velocity fields that can encode a class of movements. The nonlinear function $\mathbf{h}$ can be estimated using Gaussian mixture regression [7], which results in a large-scale global optimization problem. Once this optimization problem is solved, the on-line robot control can be realized by integrating Eq. (5), which enables the robot to switch to a different movement (within the learned class of movements) in real-time. The stability analysis of the resulting system of differential equations is also given in [7].

Since DMPs have not been designed to represent whole classes of movements, a standard DMP cannot switch to a different type of movement in case of perturbations. It can only react to a certain perturbation by "pulling" the robot back to the desired trajectory in a generic fashion. While the recently proposed approach described in [22] does allow for the generation of DMPs that are adapted to a given configuration of the external world, this approach is computationally too expensive to be applied within the robot feedback loop. However, it has the advantage that the generalized DMP is computed by local regression methods, which makes it possible to apply local, linear optimization as compared to the global, nonlinear optimization approach described in [7]. Since the approach from [22] does not attempt to represent a whole class of movements within one differential equation, the selection of basis functions for regression is less critical than in [7]. In this paper we propose a new approach to movement generalization using Gaussian process regression, which not only preserves the advantages of the local approach, but also allows for real-time computation of DMPs, thus making it suitable for its application within a real-time sensory feedback loop.

2. Approximation of a class of movements with Gaussian process regression

Lets assume that we have a set of robot movements $\mathbf{M}_i,\ i = 1, \ldots, NumEx$, which all result in a successful execution of a given task in different situations. As example we consider a set of reaching movements towards different targets in 3-D space. We denote the parameters characterizing the task by $\mathbf{q}_i \in \mathbb{R}^m,\ i = 1, \ldots, NumEx$, m being the dimensionality of these parameters, which we also call query points. Every movement $\mathbf{M}_i$ is encoded by a sequence of trajectory points $\left\{\mathbf{y}_{ij}, \dot{\mathbf{y}}_{ij}, \ddot{\mathbf{y}}_{ij} \in \mathbb{R}^{dof}\right\}$, measured at times $t_{ij},\ j = 1, \ldots, n_i,\ t_{i1} = 0$. Here n_i denotes the number of samples on trajectory $\mathbf{M}_i$, while dof denotes the number of degrees of freedom encoded by the example trajectories. We have

experimented both with end-effector trajectories (in this case $\mathbf{y}_{ij}$ are points in the Cartesian space) and with robot joint trajectories (in this case $\mathbf{y}_{ij}$ are the joint angles stemming from the active degrees of freedom). The problem is to compute a trajectory for any given query point $\mathbf{q}$. For example, in case of reaching, a query point is given by the desired target position and we need to compute the associated reaching trajectory $\mathbf{M}$. Example movements $\mathbf{M}_i$ can be acquired either by kinesthetic guiding [8] or by imitation [21].

To become able to accomplish a task in any situation, the robot needs to learn a function that maps the parameters describing the task $\mathbf{q}$ into the parameters describing the desired trajectory $\mathbf{M}$, i. e.

$$\mathbf{G} : \mathbf{q} \longmapsto \mathbf{M}. \tag{6}$$

In general, $\mathbf{G}$ is not a function. For example, in the case of reaching movements, there are many different ways to reach towards a desired destination. However, we can impose an additional constraint that synthetic reaching trajectories should be similar to the example reaching trajectories. The closer the desired query point $\mathbf{q}$ is to the query point $\mathbf{q}_j$, the more similar the generated trajectory $\mathbf{M}$ should be to the trajectory $\mathbf{M}_j$ associated with query point $\mathbf{q}_j$. With this additional constraint, $\mathbf{G}(\mathbf{q}; \{\mathbf{M}_1, \ldots, \mathbf{M}_{NumEx}\})$ becomes a function that can be learned.

2.1. Converting the example trajectories into dynamic movement primitives

To reduce the amount of data that we need to process for action generalization, we first encode each of the example movements $\mathbf{M}_i$ as a dynamic movement primitive (DMP). Any of the standard methods proposed in the literature can be used for this purpose. Lets denote

$$f_{ijl}^{targ} = \tau_i^2 \ddot{y}_{ijl} - \alpha_z(\beta_z(g_{il} - y_{ijl}) - \tau_i \dot{y}_{ijl}), \tag{7}$$

where $i = 1, \ldots, NumEx,\ j = 1, \ldots, n_i,\ l = 1, \ldots, dof$. The above equation can be derived by rewriting the system of two first-order differential equations (1)-(2) as one second-order differential equation (by writing $z = \tau\dot{y}$, $\dot{z} = \tau\ddot{y}$ in (1)). The parameters $w_{ikl},\ k = 1, \ldots, N$, ($N$ is the number of DMP kernel functions, see (4)) can be estimated by solving the following linear regression problems

$$\mathbf{X}_i \mathbf{w}_{il} = \mathbf{f}_{il}^{targ}, \tag{8}$$

where

$$\mathbf{X}_i = \begin{bmatrix} \frac{\Psi_1(x_{i1})}{\sum_{i=1}^{N} \Psi_i(x_{i1})} x_{i1} & \cdots & \frac{\Psi_N(x_{i1})}{\sum_{k=1}^{N} \Psi_i(x_{i1})} x_{i1} \\ \cdots & \cdots & \cdots \\ \frac{\Psi_1(x_{iT})}{\sum_{k=1}^{N} \Psi_k(x_{iT})} x_{iT} & \cdots & \frac{\Psi_N(x_{iT})}{\sum_{k=1}^{N} \Psi_k(x_{iT})} x_{iT} \end{bmatrix}, \tag{9}$$

and $\mathbf{w}_{il} = [w_{i1l}, \ldots w_{iNl}]^T$, $\mathbf{f}_{il} = [f_{i1l}^{targ}, \ldots, f_{iTl}^{targ}]^T$. The phase parameters $x_{ij} = x(t_{ij})$ are calculated by integrating Eq. (3) with the boundary condition $x_{i1} = x(0) = 1$.

The above process enables us to convert the initial raw trajectory data $\mathbf{M}_i$ into DMPs, i. e. $\mathbf{M}_i \longmapsto (\mathbf{w}_i, \mathbf{g}_i, \tau_i)$, where $\mathbf{w}_i \in \mathbb{R}^{N \times dof}$ are the weights of DMPs for all degrees of freedom, $\mathbf{g}_i \in \mathbb{R}^{dof}$ are the final configurations on the example trajectories, i. e. $\mathbf{g}_i = \mathbf{y}_{in_i}$, and $\tau_i \in \mathbb{R}$ are the time durations of example trajectories, i. e. $\tau_i = t_{in_i}$.

2.2. Trajectory generalization using Gaussian process regression

The conversion of raw example trajectories into DMPs results in a significant data reduction. For example, a four second trajectory sampled at 500 Hz contains 2000 data points, which can typically be reduced to a DMP defined by a few tens of radial basis functions. In this section we propose to synthesize new movements directly from the estimated DMP parameters. In this case function (6) becomes

$$\mathbf{G}\left(\{\mathbf{w}_i, \mathbf{g}_i, \tau_i; \mathbf{q}_i\}_{i=1}^{NumEx}\right) : \mathbf{q} \longmapsto (\mathbf{w}, \mathbf{g}, \tau)\,. \tag{10}$$

Gaussian Process Regression (GPR) can be applied to estimate function (10). Gaussian processes are based on Bayesian probability modeling [16]. The resulting models have an interesting and useful feature that, besides output values, they also predict confidence in these values. GPR exhibits good generalization performance and the predictive distribution can be used to measure the uncertainty of the estimated function. It has been demonstrated that this technique outperforms other regression methods on problems such as estimating inverse dynamics of a seven degrees of freedom robot arm [13].

Technically, a Gaussian process is defined as

$$g(\mathbf{q}) \sim \mathcal{GP}\left(m(\mathbf{q}), k(\mathbf{q}, \mathbf{q}')\right), \tag{11}$$

where $m(\mathbf{q}) = \mathbb{E}[g(\mathbf{q})]$ is the mean function and $k(\mathbf{q}, \mathbf{q}') = \mathbb{E}[(g(\mathbf{q}) - m(\mathbf{q}))(g(\mathbf{q}') - m(\mathbf{q}'))]$ the covariance function of the process. Lets assume that we have – as when estimating function (10) – a set of noisy observations $\{(\mathbf{q}_i, y_i) | i = 1, \ldots, NumEx\}$, $y_i = g(\mathbf{q}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Subtracting the mean from the training data, we can further assume that $m(\mathbf{q}) = 0$. Given a set of query points $g(\mathbf{q}^*)$, the joint distribution of all outputs is estimated by

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I} & \mathbf{K}(\mathbf{Q}, \mathbf{Q}^*) \\ \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}) & \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}^*) \end{bmatrix}\right), \tag{12}$$

where $\mathbf{Q}$, $\mathbf{Q}^*$, $\mathbf{y}$, $\mathbf{y}^*$ respectively combine all inputs and outputs and $\mathbf{K}(\cdot, \cdot)$ are the associated joint covariance matrices calculated according to Eq. (11). It can be shown [16] that the expected value $\bar{\mathbf{y}}^*$ associated with the new query points $\mathbf{q}^*$ is given by

$$\bar{\mathbf{y}}^* = \mathbb{E}[\mathbf{y}^* | \mathbf{Q}, \mathbf{y}, \mathbf{Q}^*] = \mathbf{K}(\mathbf{Q}^*, \mathbf{Q})[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}, \tag{13}$$

with the following estimate for the covariance of the prediction

$$\begin{aligned} \mathrm{cov}(\mathbf{y}^*) = \; & \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}^*) - \\ & \mathbf{K}(\mathbf{Q}^*, \mathbf{Q})[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}(\mathbf{Q}, \mathbf{Q}^*). \end{aligned}$$

One commonly used covariance function is

$$k(\mathbf{q}, \mathbf{q}') = \sigma_f^2 \sum_{i=1}^{m} \exp\left(-\frac{1}{2}\frac{(q_i - q_i')^2}{l_i^2}\right), \tag{14}$$

which results in a Bayesian regression model with an infinite number of basis functions. m denotes the dimension of the query point space. See [16] for more details.

With GPR new estimates are calculated using equation (13). The most computationally expensive part is the calculation of $[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}]^{-1}$, but since this matrix depends only on the training data, the necessary calculations can be done off-line using for example the Cholesky decomposition. The dimension of this matrix is equal to the number of data points. In our case, this is equal to the number of example movements $NumEx$, which is typically not too large (at most a few hundred). We thus need to invert a matrix of dimension $NumEx \times NumEx$.

Note that by writing

$$\mathbf{z} = [\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y}, \tag{15}$$

equation (13) and the estimated parameter $\bar{y}^*$ associated with the query $\mathbf{Q}^* = \mathbf{q}^*$ can be written as

$$\bar{y}^* = \sum_{i=1}^{NumEx} k(\mathbf{q}^*, \mathbf{q}_i) z_i, \tag{16}$$

where in our experiments $\bar{y}^*$ stands for $\bar{\tau}^*$, $\bar{g}_l^*$, and $\bar{w}_{kl}^*$. Thus the training data are weighted based on the distance between the training query points and the current query point. This means that nearby training points influence the result more. In this sense, GPR can be viewed as a local regression method.

To generate a new movement, the robot is given a desired query point $\mathbf{q}^*$, e. g. the desired reaching location. For each of the parameters defining a generalized DMP (τ, g_l, and w_{kl}), which encodes a suitable motion trajectory for this task situation, we need to calculate (16) on-line, whereas (15) can be stored in memory. Note also that matrix $\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}$ depends only on query points and not on the data points. Hence this matrix is the same for all parameters defining a DMP and thus needs to be inverted only once.

2.3. Comparison with previous local approaches

In our next set of experiments we compare the performance of the proposed approach and the performance of a method that uses complete trajectories without an intermediate trajectory conversion step (as proposed in [22]) for the purpose of task-specific generalization of dynamic movement primitives. In this section we examine the advantages and disadvantages of both approaches and their suitability for on-line generalization.

In contrast to our new approach, which converts the training data into DMPs, the approach proposed in [22] keeps complete trajectories in memory

and generalizes to new DMPs without the intermediate trajectory conversion step. In this case locally weighted regression (LWR) instead of Gaussian processes regression has to be used in some calculations that are needed for trajectory generalization. LWR is a memory-oriented, non-parametric method for statistical approximation. The basic idea is to compute local models using data from a small neighborhood of the desired query point. Since raw trajectories are used for estimation, the resulting systems of linear equations are much too large to be resolved on-line. Unlike this approach, which defers most of the calculations to the moment when a query needs to be answered, our new, GPR-based method performs most of the calculations off-line once all of the training data have been acquired. The most expensive off-line calculations are needed for the calculation of (15) and for the estimation of hyper-parameters l_i, σ_f and σ_n as defined in (13) and (14). After the end of learning the training data can be discarded and only simple calculations shown in (16) are needed to answer a new query or in other words generalize to new situations.

Here we note that in case of LWR we need to specify one additional parameter, i. e. influence radius, which determines how many nearby trajectories will be taken into account for generalization by LWR. Some approaches for the selection of the optimal radius can be found in [22]. Our new, GPR-based approach does not require such a parameter. In the following we call the approach proposed in this paper **MPG** (Movement Primitives Generalization) and the approach from [22] **RTG** (Raw Trajectories Generalization).

3. Experimental results

3.1. Simulation Results

We first show how good the proposed method is at generalizing motor knowledge, which was created in simulation. For this test we generated 75 minimum jerk trajectories in Cartesian space with zero velocity and acceleration at the beginning and at the end of each movement. It has been shown that minimum jerk trajectories are similar to human arm point-to-point reaching movements [6]. The simulated Cartesian space trajectories were converted into joint space trajectories of the right arm of humanoid robot HOAP-3 using inverse kinematics. This resulted in 75 four-dimensional joint space trajectories, which were used as training data. The Cartesian end-positions of trajectories were employed as query points. The query points were uniformly distributed 3 cm apart within a cuboid with dimensions 6 x 12 x 12 cm (see Figure 1). In our tests we compared how close the generalized trajectories are to the ideal minimum jerk trajectories and how the proposed method compares to the method developed in [22], which uses complete trajectories as input data without the intermediate trajectory conversion step.

We tested the performance of both approaches by calculating the generalized trajectories at query points different from the training queries. We compared the generalized joint space trajectories with the ideal minimum jerk trajectories in the Cartesian space. In summary, our experiments show that MPG calculates

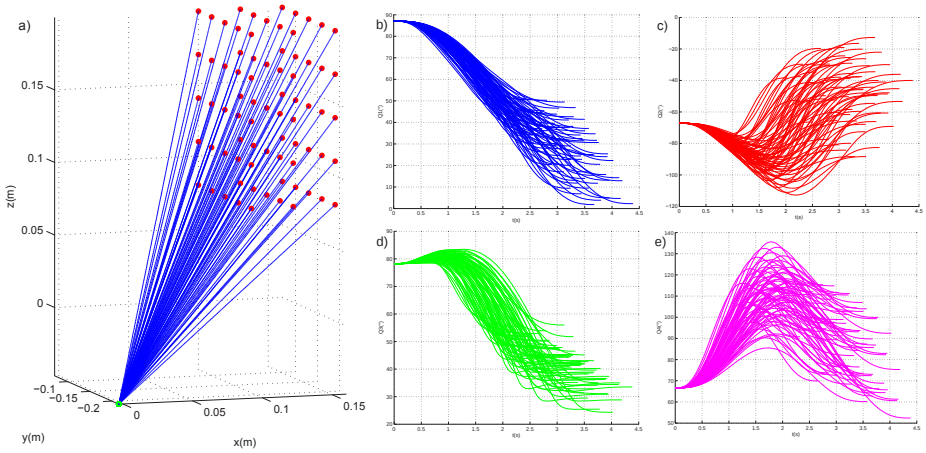

Figure 1: 75 minimum jerk trajectories were used as training data to test the generalization abilities of our approach. The 3-D graph a) shows the generated Cartesian space minimum jerk trajectories and 2-D graphs show the associated joint trajectories. b) shows shoulder flexion-extension, c) shoulder abducion-adduction, d) upper arm rotation and e) elbow flexion-extension of the right arm of the HOAP-3 robot.

the generalized trajectory much faster but with a slightly larger deviation than RTG. In the following, we analyze the error in the calculation of DMP goals $\mathbf{g}$ and the errors over the complete course of the trajectories estimated by the two different approaches.

As shown in (10), at each query point $\mathbf{q}$ we need to calculate the DMP parameters $\mathbf{w}$, $\mathbf{g}$, and τ. In our simulation example, query points are the end-points on the trajectories given in the Cartesian space and $\mathbf{g}$ are the associated joint angles at the end of the corresponding joint space trajectories. Thus in this case GPR attempts to estimate one particular solution of the inverse kinematics of the arm of HOAP-3. For testing, query points in Cartesian space given to GPR were uniformly distributed within the training cuboid with distance of 1 cm between them. The error was measured by calculating the generalized goal $\mathbf{g}$ ($\mathbf{g}_{joint,gen} = \mathbf{G}\left(\{\mathbf{w}_i, \mathbf{g}_i, \tau_i; \mathbf{q}_i\}_{i=1}^{NumEx}; \mathbf{q}_{xyz}\right)$), transforming the generalized goal from the joint space back to the Cartesian space using known forward kinematics of the arm ($\mathbf{q}_{xyz,gen} = FK(\mathbf{g}_{joint,gen})$), and calculating the distance between this transformed position and the original query point $e = \|\mathbf{q}_{xyz,gen} - \mathbf{q}_{xyz}\|$. The average difference between these points was 0.8 mm (see also Figure 2), which is significantly smaller than the distance between the training data (3 cm).

In simulation we also tested the accuracy of the complete generalized trajectories as estimated MPG and RTG. We compared the results of both approaches with the ideal minimum jerk trajectories in joint space over the entire trajectories, which end in query points situated between the training points (see Figure 3.a). We also tested how the approximation by different number of weighted radial basis functions effect the average error in joint space and in Cartesian

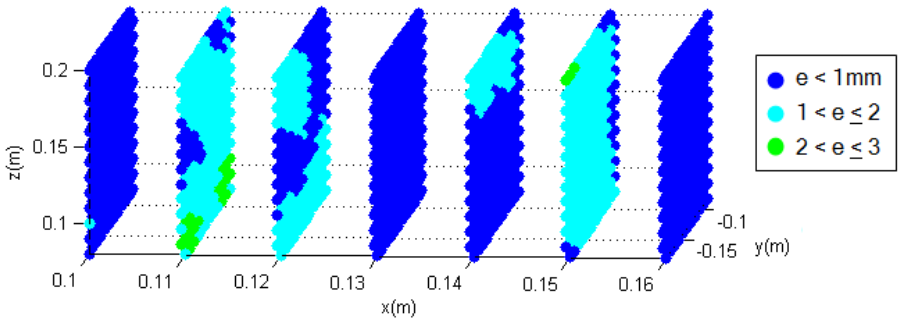

Figure 2: Error in the estimation of the DMP goal parameter **g** (see also the text). At dark blue stars the estimation error is smaller than 1 mm, at light blue start the error is between 1 and 2 mm, while at green stars the error is above 2 mm. The average error is 0.8 mm.

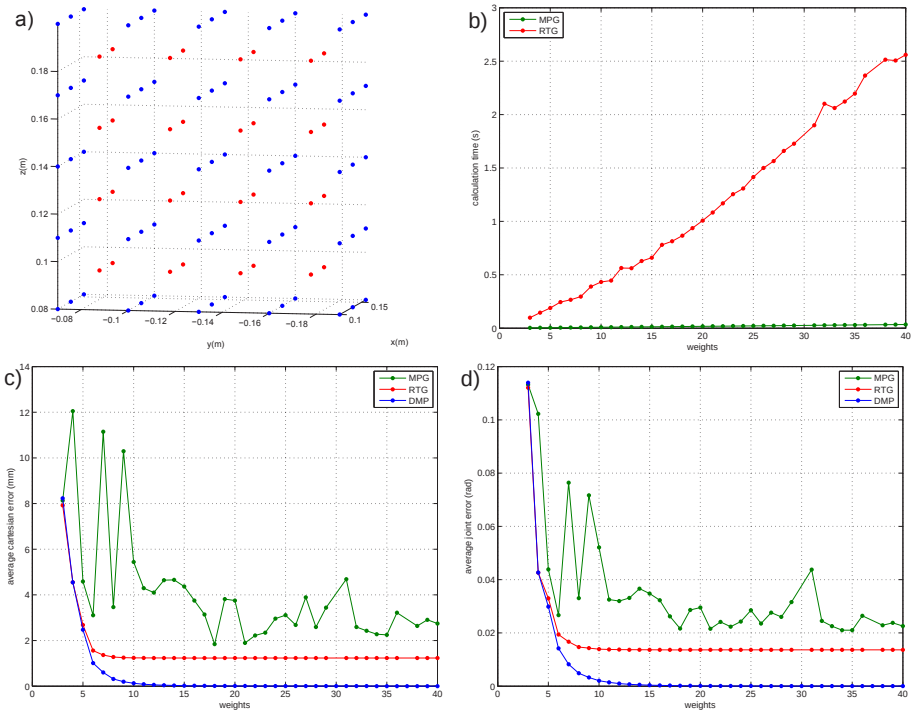

Figure 3: Results of MPG and RTG were compared with the ideal minimum jerk trajectories (in joint space) ending in query points situated between the training points as shown in graph a). The blue line in graph c) shows an error in Cartesian space of the reconstructed minimum jerk trajectories trajectories in dependence on the number of weights (here the error is due inaccurate approximation by DMPs), the red line shows an error of RTG, and the green line shows an error of MPG. Graph d) depicts an error in joint space and graph b) shows calculation times that are needed to perform RTG and MPG.

space:

$$error_k = \frac{1}{T_{end,k}} \sum_{i=1}^{T_{end,k}} \|\mathbf{y}_{i,k,gen} - \mathbf{y}_{i,k,ideal}\| \quad , \quad k = 1, ..., NumEx, \tag{17}$$

$$error_{W_n} = \frac{1}{NumEx} \sum_{k=1}^{NumEx} error_{k,n} \quad , \quad n = 3, ..., MaxN, \tag{18}$$

where $\mathbf{y}_{i,k}$ are the points on the trajectory k, given either in Cartesian or in joint space.

Results shown in Figure 3.c and d demonstrate that RTG reaches the minimum error sooner and is more stable then MPG, but MPG also reaches comparable error minimum when a few more radial basis functions are used to encode the DMPs. Figure 3.b shows a big difference in computation times for MPG and RTG as the number of DMP basis functions increases. MPG generalizes much faster and is therefore more suitable for on-line calculation. These results also show that 18 radial basis functions are enough to approximate the simulated reaching trajectories. With more than 18 weights the average error does not change significantly regardless of the selected method. Figure 4 shows how well some of the generalized trajectories (encoded with 18 radial basis functions) fit the ideal minimum jerk trajectories.

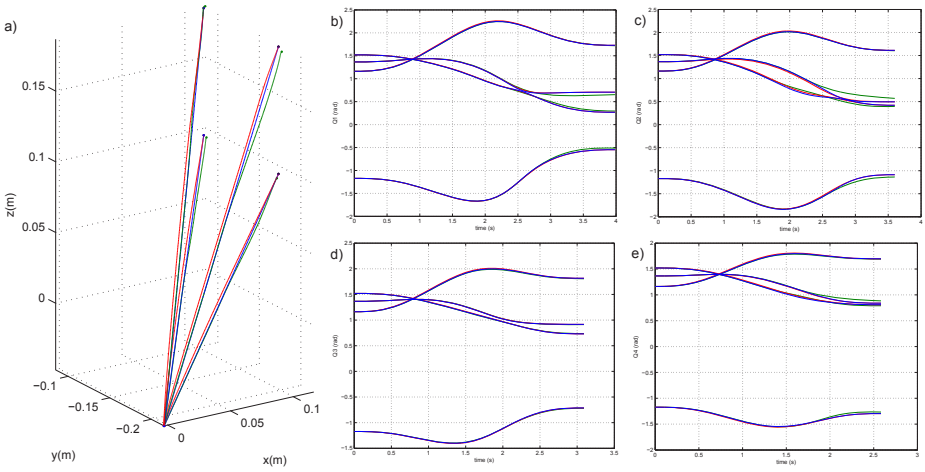

Figure 4: The 3-D graph a) shows a few calculated minimum jerk trajectories compared with generalized trajectories using MPG and RTG. Blue curves are the real minimum jerk trajectories, green curves are the generalized trajectories estimated by MPG and red curves are the generalized trajectories generated by RTG. 2-D graphs b), c), d) and e) show these trajectories in joint space (as a function of time).

The number of radial basis functions needed to approximate the trajectories depends on the type of movement. With MPG we need to use the same number of basis function to estimate all training trajectories, otherwise it is not

possible to apply Gaussian process regression. The longer and more complex the trajectories are, the more radial basis functions are needed to approximate the movements. The automatic selection of the number of basis functions is discussed in [17].

To test the MPG and RTG in real experiments, where the correct trajectories are not known, we applied the leave-one-out cross validation (L1OCV) to determine the number of necessary basis functions. In L1OCV, each of the demonstrated trajectories is taken out from the training data and re-estimated by generalization from remaining trajectories. The generalized and the skipped trajectory are then compared to determine an average error over the entire trajectory. The L1OCV score is given by an average error over all trajectories in the training data. To make comparison with real experiments easier, we tested the leave-one-out cross validation also with the simulated trajectories.

Results in Figure 5 show that the average Cartesian and joint space L1OCV scores follow a similar pattern as in Figure 4. The differences are due to the different distribution of training and test query points (here the test query point is always as far away as possible from the training points, whereas in the previous simulation experiment we tested the full distribution of query points).

3.2. Reaching and grasping with a humanoid robot HOAP-3

The aim of our next experiment was to show that our method can be used to teach the humanoid robot HOAP-3 how to reach for an object and grasp it using data coming from its own visual system. The training was done by collecting a number of example reaching movements using kinesthetic guiding along the desired reaching trajectories. We held the robot's arm near the right elbow joint and manually moved it from the initial position to the desired end-points in front of the robot (Figure 6). All four joints of the right arm were collected along the demonstrated trajectories. Altogether we collected 140 training movements to points the robot can see with its cameras. Endpoints were about 5 cm apart from each other distributed in the workspace of the robot's right arm (Figure 7), which is approximately 30x30x10 cm.

Besides the trajectories, we also measured the final reaching positions as estimated by the active stereo vision of the robot. The positions estimated by vision were used as query points. Despite accurate calibration of the active camera system of a humanoid robot, some noise and errors are to be expected [23]. By comparing the results of vision-based 3-D position estimation with the results of the robot's forward kinematics, we estimated the systematic vision error to be 1.8 cm on the average. However, the systematic vision error can be in part learned by Gaussian process regression. To collect the vision data, we put a small, colored spherical object into the robot hand and estimated its position at the final configuration on the trajectory (see right graph in Figure 7). Since stereo vision is also used to estimate the object position when generating a new movement, the vision errors that arise in training and the errors in query points used for generalization cancel each other out.

By changing the signs of all four joints of the right arm, we were able to transfer the training movements from the right to the left robot arm (Figure

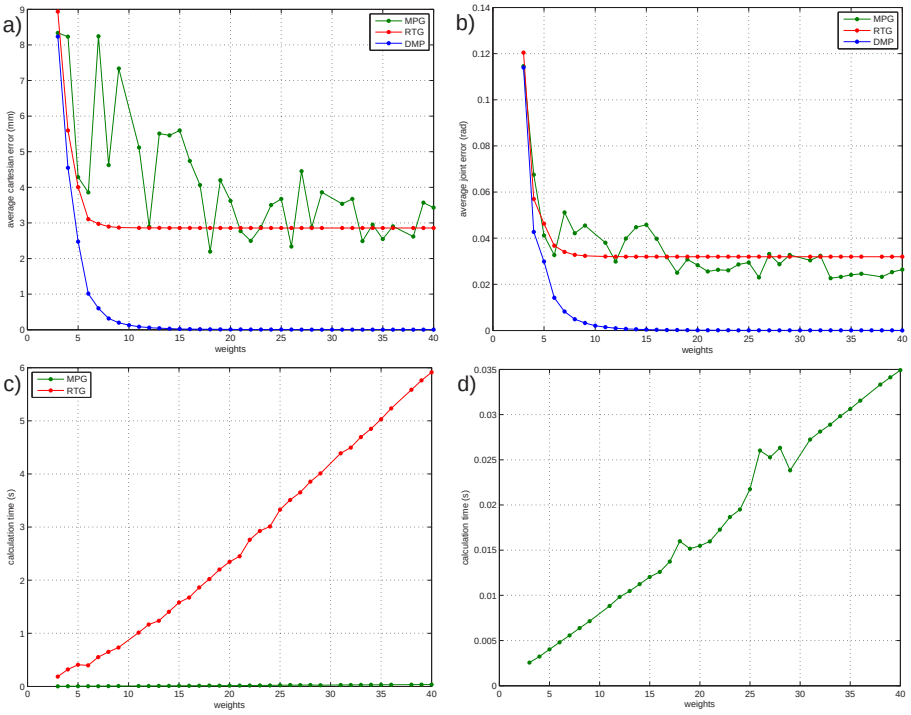

Figure 5: The evaluation of MPG and RTG using leave-one-out cross validation method. The blue line in graph a) shows the Cartesian space error of DMPs encoding the simulated minimum jerk trajectories in dependence on the number of weights, the green line shows the error of generalization by MPG, while the red line depicts the error of generalization by RTG. Graph b) shows the same errors in joint space, graph c) shows the computational times needed for generalization by MPG and RTG in dependence on the number of weights and graph d) takes a closer look on calculation times of MPG.

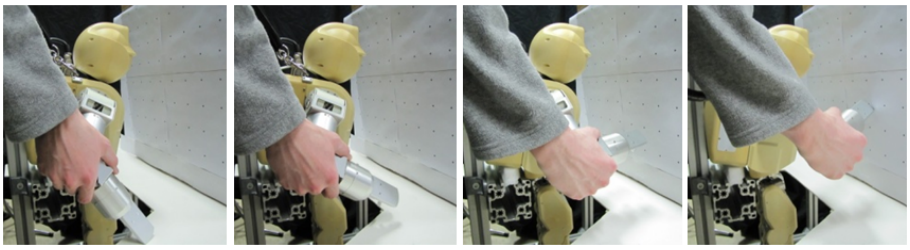

Figure 6: Image sequence showing the teaching of a reaching movement to humanoid robot HOAP-3 with kinesthetic guiding. We moved the arm towards the plate in front of the robot with points plotted at a distance of five centimeters between them. We gradually moved the plate away from the robot and demonstrated a series of movements at four different distances from the robot.

7 left graph). Such simplification was possible due to the design of the arm of the humanoid robot HOAP-3. Note that it is still necessary to estimate the end

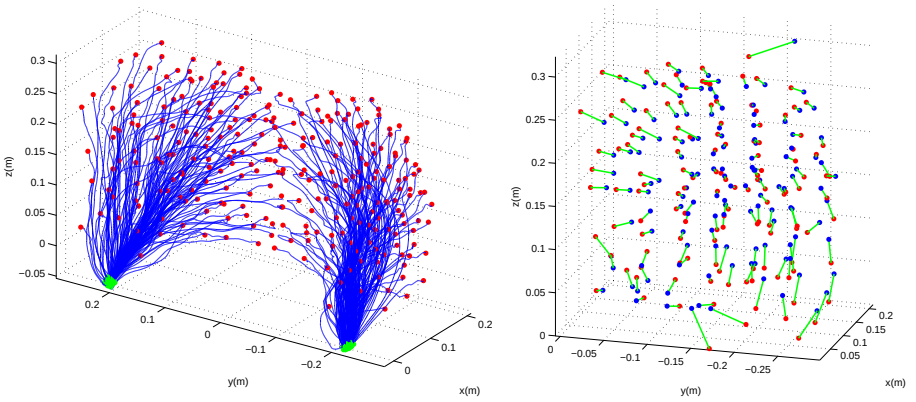

Figure 7: The left graph shows example reaching trajectories. The final positions on the trajectories are shown as red dots and the starting positions as green dots. The trajectories for the left arm of the HOAP-3 robot were created by changing the signs of all four joints of the right arm. On the right graph red dots depict the original end-points of the right arm of all demonstrated trajectories as calculated by the robot's forward kinematics, while the blue dots are the same end-points as estimated by stereo vision. The green lines illustrate the shift of end-points, which is 1.8 cm on the average and represents a systematic error of the stereo vision.

positions on the trajectories by vision because the error in the estimation of the object position depends on the configuration of the robot.

After collecting the training trajectories, we calculated the associated dynamic movement primitives (DMPs). The estimated DMP parameters (weights associated with radial basis functions, time durations, and end-points of all reaching trajectories in joint coordinates) were used as input for learning by Gaussian process regression. To define how many radial basis functions were needed to properly approximate the acquired reaching movements, we used leave-one-out cross validation as explained above. Results can be seen in Figure 8. They are similar to our simulation results. Again, the calculation time increases with more radial basis functions defining a DMP. Based on these results we applied 27 radial basis functions to define DMPs because with a larger number of weights the L1OCW score does not change significantly. A comparison between the demonstrated trajectories, which were omitted from the training data, and trajectories generalized by MPG and RTG can be seen in Figure 9.

Given a new desired reaching position in Cartesian coordinates (as estimated by the cameras), i. e. a new query point, the robot calculates the DMP parameters using Gaussian process regression as described in Section 2. The generated DMP is then integrated with Euler's method and the integration results are used to control the robot arm. If the object appears anywhere in the robot's workspace, HOAP-3 reaches towards the object with a movement similar to the demonstrated movements (Figure 10).

When the robot hand reaches the final position, it grasps the object and moves back to the initial position. If the object position changes while the

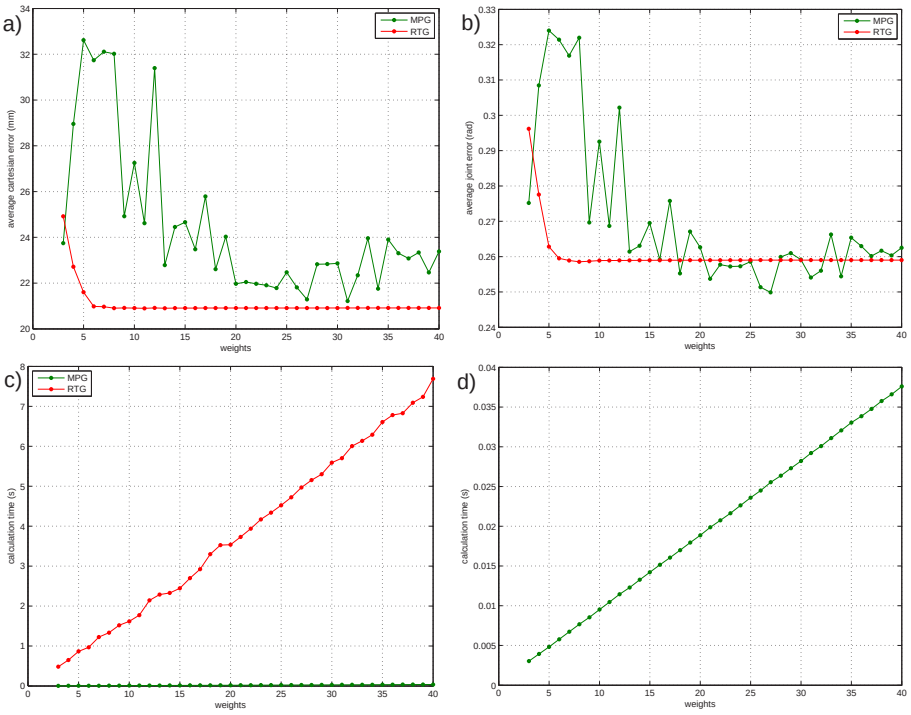

Figure 8: The results of MPG and RTG were compared with the example trajectories by leave-one-out cross validation method. In graph a), red line shows L1OCW score obtained by RTG and green line represents an L1OCW score obtained by MPG, all in Cartesian space. Graph b) represents the L1OCW score in joint space and graph c) shows calculation times that are needed for MPG and RTG, all in dependence on the number of weights. Graph d) takes a closer look at calculation times of MPG.

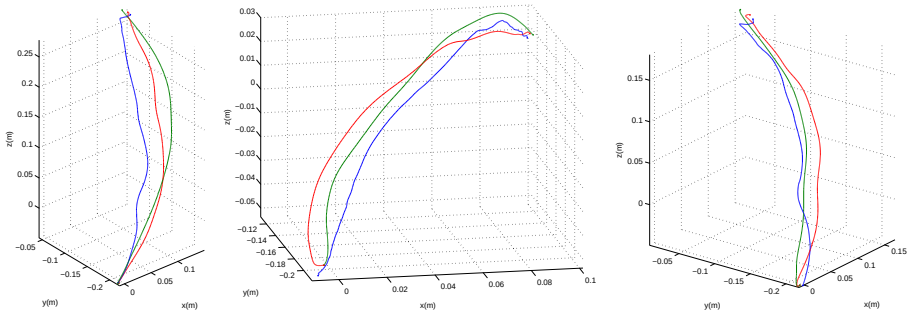

Figure 9: A few examples comparing the demonstrated trajectories (blue curves), which were omitted from the training data, and trajectories calculated by MPG (green curves) and by RTG (red curves).

arm is moving, the robot uses its vision to estimate the new object position and calculates the new goal parameters using Gaussian process regression. The form

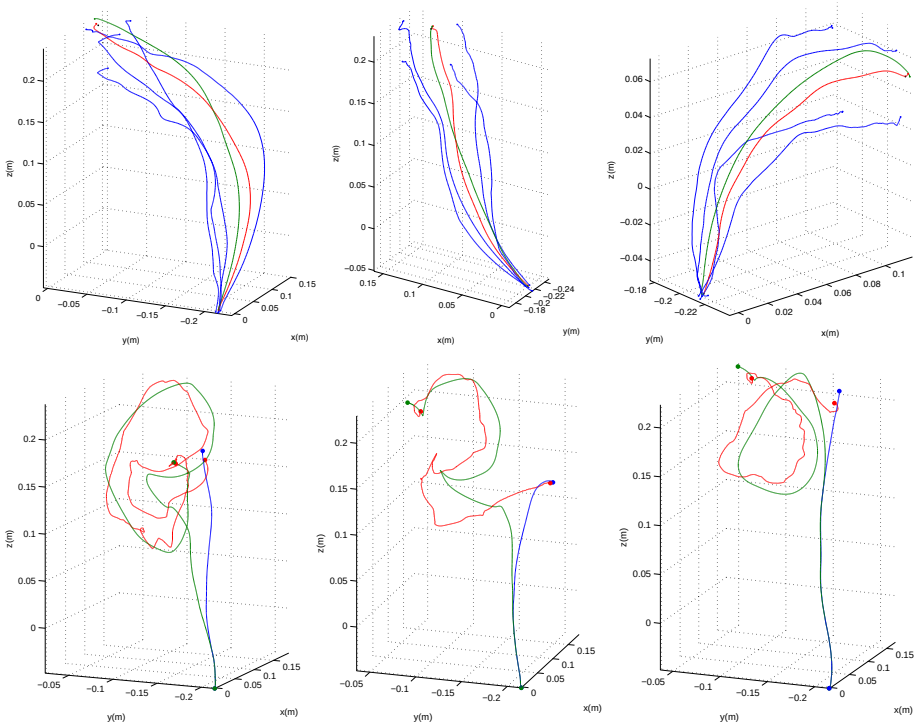

Figure 10: **Top row**: the blue trajectories are nearby demonstrated trajectories, the green curves are the generalized trajectories estimated by MPG, and the red curves are generated by RTG. Notice the shape similarity between these trajectories. **Bottom row**: the blue trajectories show the generalized right hand movement if the object keeps still. The green trajectories depict the actual right hand movement, which changed because the attractor point **g** was continuously adapted using the results of vision and Gaussian process regression. The red trajectories show the movement of the object estimated by vision.

of the movement trajectory remains similar to the training trajectories until the end of the estimated duration time, afterwards it takes the form dictated by the critically damped system (1) (the nonlinear part is close to zero once the time exceeds the estimated duration). If the robot has not reached the object within the expected time, we continue to estimate the goal parameters **g** by vision and GPR, which ensures that the robot eventually reaches the object.

Figures 11 and 12 show two reaching examples where the robot motion is continuously adapted to the current situation. In this experiment we show the integration between the proposed approach and walking o grasp an object. A similar experiment on a humanoid robot but based on more classic robotics approaches have been reported in [20]. If the object falls outside of the robot's workspace, HOAP-3 estimates the distance between the object and the robot's base coordinate system with its cameras and starts walking to come closer to the object. A predefined walking pattern was used for this task. While walking, the robot tracks the object and keeps estimating the distance to the object.

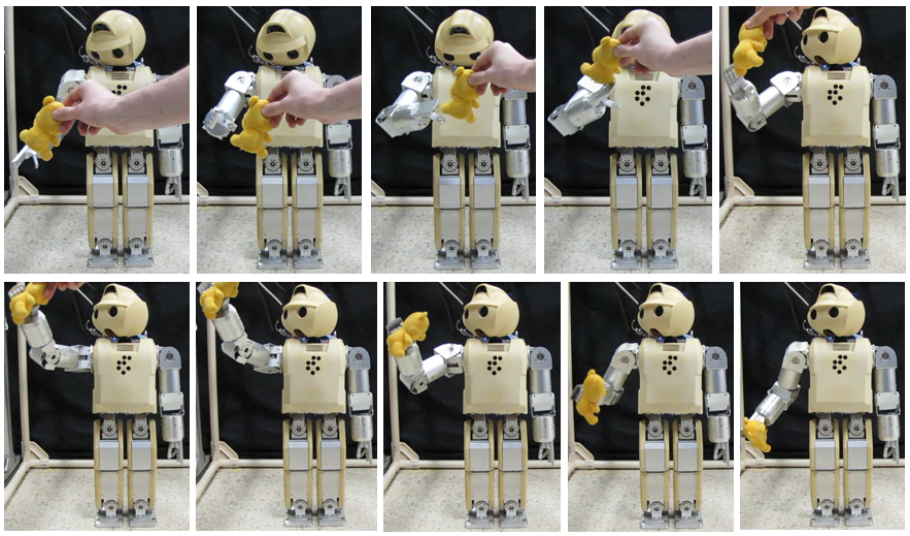

Figure 11: HOAP-3 detects the object and keeps tracking it. Once the object stops moving, the robot can grasp it and move the arm back to its starting position.

This information is used to estimate how many intermediate steps are needed to reach the goal. The robot also adapts its orientation to ensure that it walks towards the object and finally grasp it.

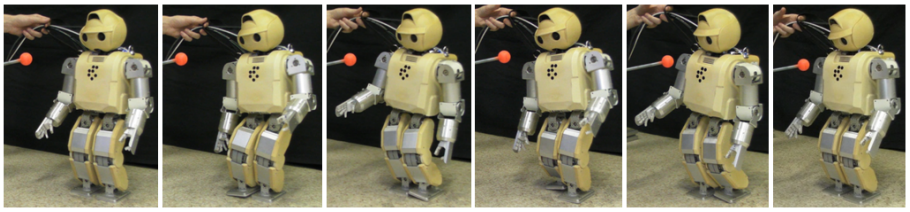

Figure 12: If the object is outside of the robot's workspace, HOAP-3 uses vision to estimate the distance between the object and its base coordinate system and starts walking to come closer to the object. A predefined walking pattern was used for this task.

As mentioned previously, the systematic error of stereo vision was partly learned by GPR and was consecutively reduced, but the error due to imperfect generalization by GPR remains. To test the generalization accuracy of GPR in a real experiment, the error of the method was estimated in the same way as in the simulation example. The test points, where the regression error was measured, were distributed on a regular grid with a distance of 1 cm between the end points inside the right arm workspace. Figure 13 shows the difference between test points and 3-D position of that points as estimated by Gaussian process regression and robot forward kinematics. Here the error of GPR is 7.3 mm on average, which is low enough for the given task of grasping. The error

in the case of real data is significantly larger than in simulation, but this can be expected because the distribution of the simulated data was more compact, more regular and there was no noise.

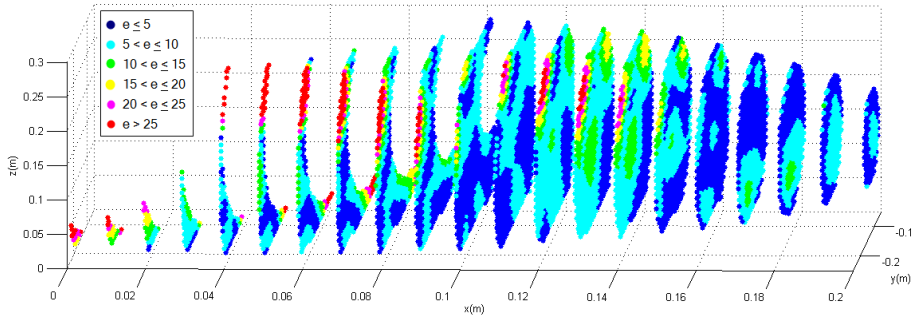

Figure 13: Dark blue fields in the graph represent GPR error of less than 5 mm, light blue between 5 and 10 mm, green between 10 and 15 mm, yellow between 15 and 20 mm, rose between 20 and 25 mm and above 25 mm error are the red fields. The average error of this real-task endpoints is 7.3 mm.

3.3. Switching between two different movement primitives

Our final experiment was performed with the 7 DOF KUKA Light-Weight Robot arm. The main goal of this experiment was to demonstrate on-line generalization of trajectories, which can be accomplished only with our new approach (MPG) because RTG is too slow. The task was to reach towards an object and grasp it. The object can be grasped either from its right or from its left side. Thus the robot needs to learn how to grasp the object from both sides. We demonstrated 144 reaching movements (72 from the right side and 72 from the left side of the object), which were all acquired by kinesthetic guiding of the arm (see Figure 14). We show the performance of real-time generalization in a task in which the robot switches between two different types of reaching movements (for left- and right-side grasps) in case of perturbations.

The KUKA arm was controlled in stiffness mode. While reaching the stiffness is high enough to properly perform the generalized reaching move. The external joint torques are monitored during execution. If joint torques exceed a threshold (determined empirically), the algorithm switches to a lower stiffness mode, knowing that a physical disturbance occurred. During low stiffness mode the robot is compliant enough to move in the direction of push. Meanwhile, new generalized reaching movements are constantly calculated (every 0.03 seconds) based on the current position of the robot's end-effector. When the perturbation stops, the newest generalized reaching movement starts being executed. If for example the robot starts reaching from the right side of the object and the perturbation causes it to move to the left side, the algorithm switches from right- to left-side reaching movement. The object is grasped once the robot reaches the end-position on the reaching trajectory. We used a BarrettHand BH-8 Series

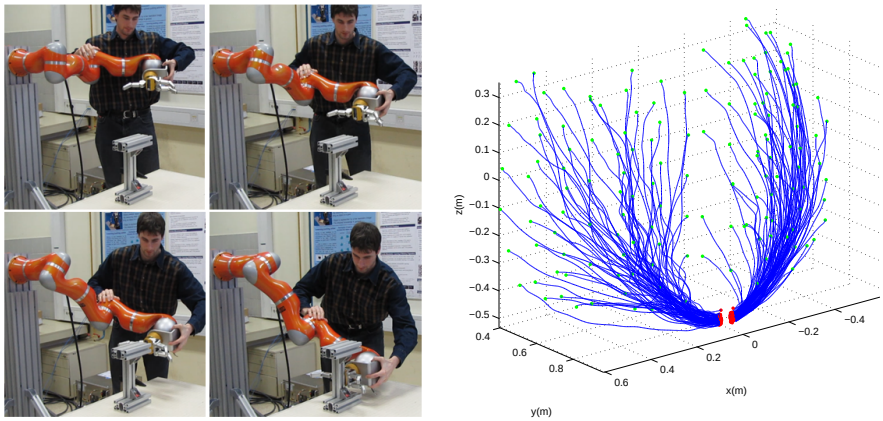

Figure 14: Pictures show the acquisition of reaching movements by kinesthetic guiding of KUKA Light-Weight Robot arm. The graph shows 144 reaching moves that were performed, 72 from the right side and 72 from the left side of the object.

attached at the top of the arm for grasping. The described reaching process is shown in Figures 15, 16 and 17.

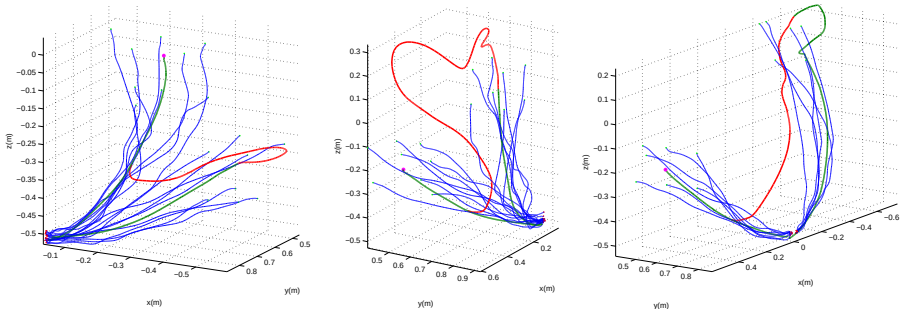

Figure 15: Graphs shows reaching examples of Light-Weight Robot arm. Green curves are generalized trajectories while blue curves are nearby demonstrated trajectories. Red curves represent robot movement under physical disturbance. In all graphs initial generalized reaching moves encounter physical disturbance and are pushed to different positions where new generalized moves are performed. Graphs also shows how initial reaching moves would look like, if there wouldn't be any physical disturbances. In the right graph reshape of the robot can be seen when passing to different type of reaching moves, where the beginning of the second green curve shows reshaping of the robot.

4. Conclusion

We developed a new approach for on-line generalization of discrete movements based on Gaussian process regression. The proposed methodology was inspired by motor tapes theories [1] and motor schemas [19], in which example movement trajectories are stored directly in memory [15]. Unlike previous

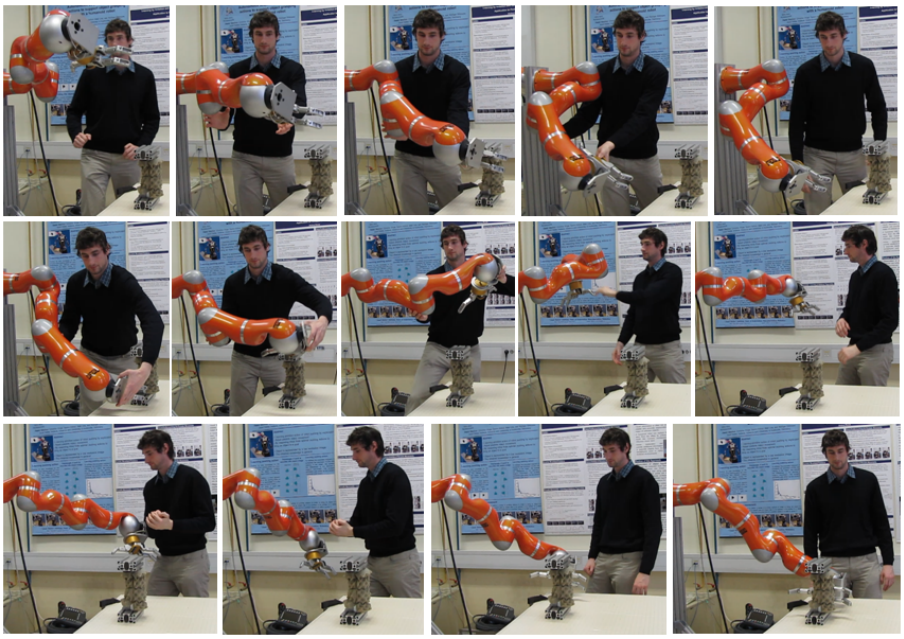

Figure 16: Reaching with Light-Weight Robot arm controlled in stiffness mode. During reaching the stiffness is high. The joint torques are monitored and if physical disturbance occurs the algorithm switches to low stiffness mode where the robot is compliant enough to be pushed to any position from where the newest generalized reaching movement is performed.

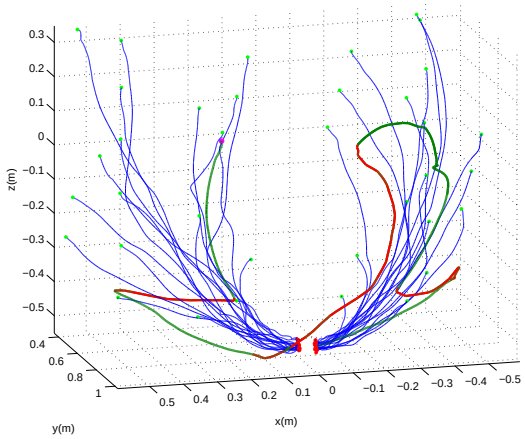

Figure 17: 3-D graph illustrates movement of the Light-Weight Robot arm represented in Figure 16. Green curves are generalized trajectories while blue curves are some of the demonstrated trajectories. Red curves represent robot movement under physical disturbance. Beginning of the third green curve shows reshaping of the robot when passing to different type of reaching moves.

generalization approaches, which either required significant on-line calculations [22] or global optimization [7] prone to local minima, the proposed approach can avoid both. Our experiments have shown that despite significant data reduction, which provides the basis for a real-time implementation of the proposed methodology, the generated movements remain close to the ideal movements. The real-time implementation enabled us to realize tasks such as on-line switching between movement primitives based on perceptual feedback, which would not be possible with previous memory-based approaches.

Acknowledgement: This work was supported in part by the European Union Cognitive Systems Project Xperience under Grant FP7-ICT-2009-6-270273 and by SRBPS, MEXT.

References

[1] C. G. Atkeson, J. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, and M. Kawato. Using humanoid robots to study human behavior. *IEEE Intelligent Systems*, 15(4):46–56, 2000.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *AI Review*, 11:11–73, 1997.

[3] D. Bentivegna, C. G. Atkeson, and G. Cheng. Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47(2-3):163–169, 2004.

[4] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. Billard. Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression. *IEEE Robot. Automat. Mag.*, 7(2):44–54, 2010.

[5] R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, 2004.

[6] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, 1985.

[7] E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard. Learning non-linear multivariate dynamics of motion in robotic manipulators. *The International Journal of Robotics Research*, 30(1):80–117, 2011.

[8] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Trans. Robotics*, 24(6):1463–1467, 2008.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 958–963, Lausanne, Switzerland, 2002.

[10] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398–1403, Washington, DC, 2002.

[11] D. Kulić, W. Takano, and Y. Nakamura. Online segmentation and clustering from continuous observation of whole body motions. *IEEE Trans. Robotics*, 25(5):1158 – 1166, 2009.

[12] C. Liu and C. G. Atkeson. Standing balance control using a trajectory library. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3031 – 3036, 2009.

[13] D. Nguyen-Tuong, M. Seeger, and J. Peters. Model learning with local Gaussian process regression. *Advanced Robotics*, 23:2015–2034, 2009.

[14] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 763–769, Kobe, Japan, 2009.

[15] T. Poggio and E. Bizzi. Generalization in vision and motor control. *Nature*, 431:768–774, 2004.

[16] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

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

[18] S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics systems vs. optimal control – a unifying view. *Progress in Brain Research*, 165(6):425–445, 2007.

[19] R. A. Schmidt. A schema theory of discrete motor skill learning. *Psychological Review*, 82:225–260, 1975.

[20] O. Stasse, B. Verrelst, A. Davison, N. Mansard, F. Said, B. Vanderborght, C. Esteves, and K. Yokoi. Integrating walking and vision to increase humanoid autonomy. *International Journal of Humanoid Robotics*, 5(2):287–310, 2008.

[21] A. Ude, C. G. Atkeson, and M. Riley. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems*, 47(2-3):93–108, 2004.

[22] A. Ude, A. Gams, T. Asfour, and J. Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Trans. Robot.*, 26(5):800–815, 2010.

[23] A. Ude and E. Oztop. Active 3-D vision on a humanoid head. In *Proc. 14th Int. Conf. Advanced Robotics*, Munich, Germany, 2009.

2011 11th IEEE-RAS International Conference on Humanoid Robots
Bled, Slovenia, October 26-28, 2011

Segmentation and learning of unknown objects through physical interaction

David Schiebener*, Aleš Ude*†, Jun Morimoto†, Tamim Asfour‡ and Rüdiger Dillmann‡

*Jožef Stefan Institute, Dept. of Automatics, Biocybernetics and Robotics, Ljubljana, Slovenia
†Department of Brain Robot Interface, ATR Computational Neuroscience Laboratories, Kyoto, Japan
‡Karlsruhe Institute of Technology, Humanoids and Intelligence Systems Lab, Karlsruhe, Germany

***Abstract*—This paper reports on a new approach for segmentation and learning of new, unknown objects with a humanoid robot. No prior knowledge about the objects or the environment is needed. The only necessary assumptions are firstly, that the object has a (partly) smooth surface that contains some distinctive visual features and secondly, that the object moves as a rigid body. The robot uses both its visual and manipulative capabilities to segment and learn unknown objects in unknown environments. The segmentation algorithm is based on pushing hypothetical objects by the robot, which provides a sufficient amount of information to distinguish the object from the background. In the case of a successful segmentation, additional features are associated with the object over several pushing-and-verification iterations. The accumulated features are used to learn the appearance of the object from multiple viewing directions. We show that the learned model, in combination with the proposed segmentation process, allows robust object recognition in cluttered scenes.**

I. Introduction

Autonomous learning of the visual appearance of unknown objects from camera images requires that the robot is able to detect and segment new objects in the acquired images. If no prior knowledge about the object and the environment is available, it is in general very difficult to segment it accurately and reliably based on visual information only. Although humans are usually very successful at this task, it is not easy to replicate the equivalent ability in artificial (passive) vision systems [1][2]. The main reason for this is that no clear and comprehensive definition for the concept "object" has been found so far. For each principle that could be used to define the concept of object, e. g. closure, connectedness, etc., counterexamples can be found. Thus in general a sufficient criterion to decide if some part of an observed scene constitutes a part of an object is not known.

Even though simple principles are not sufficient to define the concept of object, they can give hints to generate hypotheses about the existence of objects. The generated hypotheses must then be tested using stronger criteria. When a robot is not constrained to passively observing a scene, but can use its manipulation abilities to physically interact with the scene, it can observe the outcome of its own actions to provide an additional source of information. Like humans, the robot can use its (partial) control over the objects and the resulting visual input to observe - and learn about - the effects of its actions [3]. For example, moving an object can help to extract its boundaries [4]. In [5], the kinematic properties of an unknown articulated object are obtained by moving its parts.

If the robot can grasp an object it is interested in, it can move it in a controlled way. In this case, the object can be segmented reliably and its visual appearance from multiple viewing directions can be learned [6][7]. But grasping of a completely unknown, unsegmented object is in general very difficult, and in some cases it may be impossible anyway because of the size or shape of the object. A simpler alternative is to just push the object. This will result in rather uncontrolled object movements, but has been shown to be sufficient to acquire affordances of unknown objects [3].

In our previous work [8] we showed that pushing can be useful for object segmentation. Here we extend this initial work by providing a methodology to discover more candidate surfaces that give hints about the existence of the object. More importantly, we developed a new approach that allows for reliable feature accumulation across a number of different snapshots. Based on these results we developed an object recognition system, which supports both autonomous object learning and object recognition. The developed system has been tested in a number of experiments that involved both object learning and recognition.

II. Overview

Our method for learning new objects consists of the following four procedures:

- **Generation of object hypotheses:** Visual features that seem to lie on a smooth surface patch are detected and grouped together.
- **Verification by pushing:** The hypothetical object is pushed. The resulting feature motion allows to verify which features belong to the object. Additional features are added if they move concurrently.
- **Feature accumulation:** The above step can be repeated arbitrarily many times to accumulate object features from multiple viewpoints.
- **Learning of a classifier:** Since it is often difficult to reliably extract and track the same feature point across multiple views, we based our recognition system on a bag-of-features approach, which does not require that all features are tracked and matched across different views.

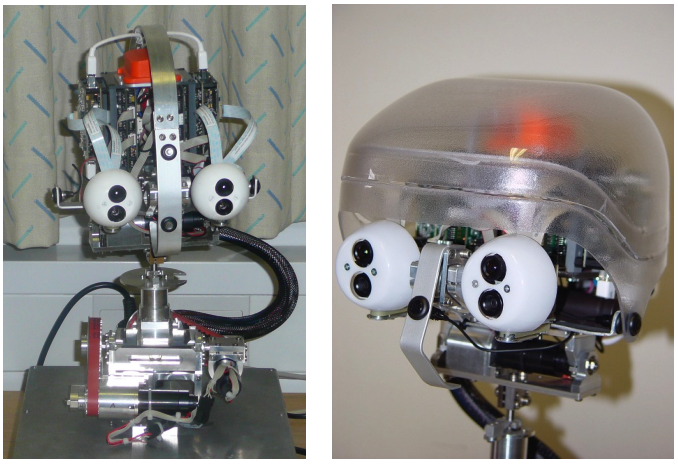

Fig. 1. The Karlsruhe Humanoid Head [9], which is equipped with two pairs of stereo cameras.

III. Hypothesis generation

The first step of our approach for segmenting and learning unknown objects is to form hypotheses about possible objects. They are generated using only the visual information that the robot perceives from its cameras (see Fig. 1). As pointed out in the introduction, the visual information may be misleading, and therefore these hypotheses can only be a starting point and must later be examined further by pushing the hypothetical object and observing the induced feature motion.

The intended scenario for our system is a household environment. Most objects in such environments consist of planar or curved surfaces. Hence it is reasonable to look for planar or cylindrical surface patches, which are mathematically simple to describe, to generate hypothesis about the existence of the objects.

We apply the Harris corner detector [10] to choose interest points that can be used both for hypothesis generation and object learning and recognition. The points determined by this detector are usually distinctive enough to allow for reliable matching in the two images from the stereo cameras. We can calculate the position of the corresponding 3-D point using the calibration of the camera pair [11]. The calibration also allows to use epipolar geometry which reduces the matching problem to a search along the epipolar line. There may still be some incorrect points due to mismatches, but they are too few to affect the hypothesis generation.

Given a set of 3-D points, our goal is to find planes and cylinders that contain as many of these points as possible. For each surface patch, we have to expect that only a rather small part of all features belongs to it. To enable the detection of surface patches among many outliers, we apply the RANSAC algorithm [12], which enables us to find the parameters defining the surface patch that contains maximal subsets of feature points belonging to the parametric surfaces. RANSAC achieves this by randomly selecting a minimal number of points, which is sufficient to calculate the parameters of the sought for surface, and then counting how many points of the whole set lie within a tolerance of the defined surface.

The plane or cylinder containing the largest number of points is added to the list of hypotheses and its points are removed from the set. RANSAC can then be run again on the remaining points. This is repeated until no surface with more than a minimal number of points can be found. The specific approaches to finding planes and cylinders using RANSAC are described in more detail in the following two subsections.

A. Plane detection

A 3-D plane is defined by the equation $ax+by+cz+d=0$ and contains all points (x, y, z) that fulfill this equation. The vector (a, b, c) is the surface normal. If it has unit length, then the above equation gives the distance of the point (x, y, z) to the plane (a, b, c, d). A plane is uniquely defined by three points that are not collinear. With this in mind, the implementation of RANSAC for planes is straightforward:

- repeat N_p times:
  - select 3 different points at random
  - calculate the plane parameters
  - check for each point if it lies within tolerance t_p of the plane, count the inliers
- return the parameters of the plane with maximal number of inliers

It can occur that a hypothesis extends to two or more objects which by chance contain points lying in the same plane. To avoid misled attempts of pushing in this case, we group the features of each plane using X-means clustering [13], which is a k-means based algorithm that also estimates the number of clusters. Single points that are far away from the cluster centers are discarded, because they are with high probability outliers. Sometimes a hypothesis containing a large object is accidentally divided by the above clustering process. However, this is not a serious problem for our system because the initial hypothesis will be expanded after the push (as other feature points on the object will move in unison with the initial hypothesis).

B. Cylinder detection

Finding cylinders in a point cloud is more complicated because the parameters of a cylinder can not be determined so easily from a few points on its surface. We applied the algorithm proposed in [14], which uses a 2-stage RANSAC approach, first estimating the cylinder axis and then the appropriate radius and offset from the origin for that axis.

In the first stage, the algorithm uses local surface normals to find promising candidates for possible cylinder axes. To this end, for each 3-D point a local surface normal is estimated using the point and its nearest neighbours. The set of normalized surface normals lies on the unit sphere and is called the *Gaussian image* of the points, as it is the result of applying the *Gaussian map* operation to the set of points. Points belonging to an arbitrary cylinder are mapped to a great circle on the Gaussian sphere. A great circle on the sphere is equivalent to the intersection of this sphere with a plane which passes

Fig. 2. Hypotheses generation: The left image shows all detected Harris interest points, the other images display the generated hypotheses for each scene. Usually, the hypotheses correspond to a textured region on an object's surface. When objects are close to each other and points on their surfaces lie on a common plane or cylinder, it may happen that these points are subsumed in one hypothesis.

Fig. 3. Hypotheses generation for cylindrical surfaces. The left image shows all Harris interest points, the central and right images show the generated cylindrical hypotheses. Although the two objects in the central image do not have an exactly cylindrical shape, a large part of their surfaces can be captured by the cylinder hypotheses.

through its origin. Therefore, we only need to find the plane passing through the origin that contains the maximal number of points on the Gaussian sphere. This problem is identical to that of finding a plane, where one of the three sample points is always the origin. The normal of the resulting plane is the sought cylinder axis.

Once the cylinder axis has been detected, we still need to find the radius of the cylinder and its offset from the origin. This problem can be reduced to finding a 2-dimensional circle: all points are projected onto the plane orthogonal to the cylinder axis and we need to find a circle with the maximal number of points lying on it. Three non-collinear 2-D points (x_i, y_i) define a circle, its center coordinates (x_c, y_c) are given by

$$x_c = \frac{(y_3 - y_2)(x_1^2 + y_1^2) + (y_1 - y_3)(x_2^2 + y_2^2) + (y_2 - y_1)(x_3^2 + y_3^2)}{2\delta}$$

$$y_c = \frac{(x_3 - x_2)(x_1^2 + y_1^2) + (x_1 - x_3)(x_2^2 + y_2^2) + (x_2 - x_1)(x_3^2 + y_3^2)}{2\delta}$$

where

$$\delta = x_1(y_3 - y_2) + x_2(y_1 - y_3) + x_3(y_2 - y_1)$$

and the radius is simply the distance of one of these points to the center. Finding an optimal circle can therefore easily be done by another application of RANSAC. Here we need to consider only the points that contributed to the great circle on the Gaussian sphere that defines the examined cylinder axis. The radius of the resulting circle is the radius of the cylinder, and the cylinder axis passes through the center of the circle.

When the number of points lying on a cylinder candidate is being determined, only those points are accepted which would lie on the side of the cylinder that is turned towards the camera. To test if a point fulfills this criterion, we check if it lies on the correct side of the plane spanned by the cylinder axis and the vector that is orthogonal both to the cylinder axis and the viewing direction of the camera. This turned out to be very helpful for reducing the number of incorrect hypotheses because sometimes objects are arranged in a way that their sides form a half cylinder opened towards the camera. To further reduce the number of false hypotheses, only cylinders with a rather small radius are accepted, which again avoids the "fusion" of several objects into one big cylindrical surface.

In every iteration of the outer RANSAC loop, a new possible cylinder axis is determined. After a fixed number of iterations, or when no new axis with more than a minimal support in the Gaussian sphere can be found anymore, the parameters of the cylinder with the maximal number of inliers are returned. Just like in the case of planes, we next discard all points that lie far away from the others to reduce the probability that outliers are included. In our experiments, the clustering of points belonging to one of the detected cylinders was not necessary.

IV. Hypothesis validation by pushing

Additional information need to be provided to verify or discard the generated object hypotheses. By inducing the object to move, visual features can be analyzed for coherent motion, which is a very strong evidence for deciding if they belong to the same object or not. Such information could not be obtained by passive observation. The most common assumption, which we also make, is that the object moves as a rigid body. A more general model of motion would be, for example, an articulated motion [5] or a deformable motion.

Inducing motion on the object, even if it is rather uncontrolled, resolves most of the ambiguities about object segmentation. We use simple pushing movements to verify the initial object hypotheses and to extend them to features that move coherently with the initial features. The initial hypotheses serve as a cue for promising points and directions of pushing. An obvious choice for the hypothesis on which a push is attempted is the one that contains the largest number of features because a large number of features usually result in a more robust estimation of object motion.

A necessary prerequisite for the estimation of feature point motion is to be able to match the features before and after the push. For its descriptiveness and robustness to small rotations, we use SIFT descriptors [15] to find matches of the features in the images before the push and after it. For all initial features for which a corresponding feature is found, the new 3-D positions are calculated using stereo images.

Due to occlusions or too large rotations caused by the induced object motion, some features may not be found again after the push. There may also be mismatches, especially if the object contains non-unique features. Again, RANSAC is a good choice to get a robust estimation of the object motion. The parameters of a transformation associated with the rigid body motion can be obtained from three different pairs of corresponding points before and after the push [16]. If $\mathbf{x}_o$ is the initial position of a point, then its new position $\mathbf{x}_n$ is given by the transformation $\mathbf{x}_n = \mathbf{R}\mathbf{x}_o + \mathbf{t}$, where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ a translation vector.

After the object has been pushed, the initial hypothesis is evaluated to confirm whether the hypothetic feature points have moved as a rigid body or not. RANSAC is applied to estimate the transformation with which most of the points of the hypothesis concur. The norm of the translation vector $\mathbf{t}$ and the angle of rotation φ, which can be calculated from $\mathbf{R}$, give a measure for the amount of motion resulting from that transformation. The hypothesis is considered confirmed if the weighted sum of $\|\mathbf{t}\|$ and φ is above a threshold. In this case, the features that moved coherently are considered validated and those who did not are discarded. The hypothesis is ignored if the estimated parameters suggest that the hypothetical features did not move. If none of the generated hypotheses moved, another attempt to push one of them is made. If at least one of the hypotheses has moved, and it still contains a sufficient number of features, we assume to have found an object whose appearance needs to be learned.

V. Object learning and recognition

To learn the appearance of the segmented object from multiple viewpoints, the object must be moved, e. g. by pushing, several times. At every step, new points are added to the hypothesis if they seem to belong to the object, and can be verified after the next push. The accumulated set of all verified points, as well as the set of only those verified points that are visible at a given instant, are admissible candidates for representing the appearance of the object. As we use SIFT descriptors for feature matching between stereo image frames, it is an obvious choice to use these features for describing the object. However, it is possible to use any other desired local descriptor at the locations of the confirmed points. Object recognition based on SIFT descriptors, especially when their spatial relationships are incorporated, has been shown to be very successful and reliable [15][17]. Another possibility is the "bag-of-features" approach [18]. Here a so-called "visual vocabulary" is learned first by clustering a large number of training features. When working with descriptors later, each of them is assigned to the most similar "visual word", i. e. cluster center. A histogram of the occurrences of each visual word on the object is calculated and stored in a database of histograms. To recognize an object, its bag-of-features histogram is calculated for the current, segmented image and matched to the histograms in the database of known objects. We use the bag-of-features approach to memorize the object appearances from different viewpoints and, as we have several histograms for each object, we can apply a k-nearest-neighbours decision for recognition.

A. Object learning

The object needs to be pushed several times to acquire snapshots from different viewpoints. This data can be used to learn a multi-view representation of a successfully segmented object. In this process, the already verified features are tracked as long as they are visible, which enables the system to estimate the underlying object motion. At every step, new Harris interest points are detected in the image, and they are added to the object model if

- they moved in unison with the object during the last pushing action, which implies that they belong to the same rigid body, or
- they lie "inside" the object, i.e. their distance from the object center is small compared to the extent of the object.

In both cases, the new features have to be verified after the next push before they are confirmed and included in the learned object description. To estimate the object's motion caused by the push, we use only the confirmed features.

After every push, two bag-of-features histograms of the object are created and saved. One contains all confirmed descriptors that have been accumulated up to the last push. The other histogram contains only the confirmed features that are visible after the last push. While the intent of the first histogram is to have a more comprehensive description of the object, the second one has a snapshot-like character and is

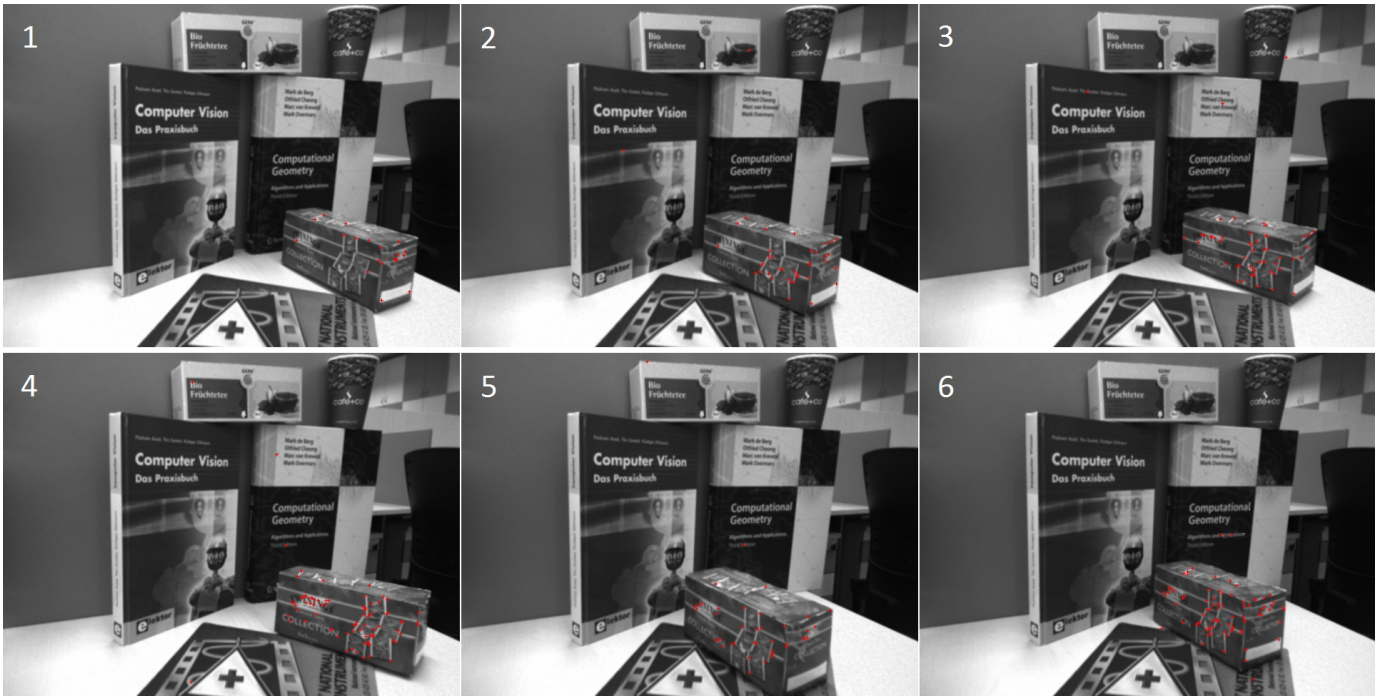

Fig. 4. An object is learned by accumulating verified feature points on its surface during repeated pushing. At each step, new candidate points are added to the object hypothesis and verified after the next push.

more specific to the appearance of the object from the current viewpoint. In our experiments, both types of histograms turned out to be helpful for recognition.

Although the SIFT descriptor is robust to minor viewpoint changes, feature matching fails once the rotation in depth becomes too large, which normally happens after a few pushes. Therefore after each push new descriptors are calculated from the current image for each of the visible, verified feature points. A new descriptor is added to the list of descriptors associated with the feature point if it is significantly different from the old descriptors.

When a confirmed feature becomes invisible, there is a possibility of a mismatch, resulting in an assignment to a point in the image that does not belong to the object. To avoid problems that may arise from such mismatches, confirmed points that do not follow the object's motion two times are not used for the motion estimation anymore. If they do not move in unison with the object four times in succession, they are discarded completely.

The learning process can be continued as long as required. Due to the uncontrolled character of the object motion, there is no guarantee that a complete description of the object will ever be obtained. Still, the chances are good that with a moderate number of pushes a large part of the possible view directions onto the object will be covered.

B. Object recognition

To recognize an object using the bag-of-features approach, its features have to be segmented in the image. Then each of them is assigned to the most similar word of the visual vocabulary and the histogram of word occurrences is calculated. Now the corresponding known object needs to be found, which can be done by comparing the current histogram with the histograms of all known objects using the χ^2 histogram distance. As several histograms of each object are available, conventional classification techniques can be applied for reliable recognition. We use a k-nearest-neighbours classifier to identify the object.

The main difficulty in recognizing objects based on the bags-of-features technique is to correctly segment the hypothetical object that needs to be recognized. If the segmentation contains only some of the object features or many features that do not belong to the object, the histogram is distorted, which makes a correct recognition improbable. Classical approaches to segmentation include feature clustering with k-means or regular and randomized windowing [19]. In our setting – since the object segmentation problem is equivalent to the one we face when learning object histograms – we can apply the same active segmentation algorithm as during the learning process. Moreover, to improve the recognition rate, we can push the object several times, which improves the quality of the segmentation by adding more features and by discarding the unstable ones.

VI. Experimental evaluation

We conducted experiments to evaluate the generation of object hypotheses in complex scenes, the segmentation and learning of unknown objects by pushing them repeatedly, and the recognition of objects using both our initial hypotheses and segmentation results that were improved by pushing the

TABLE I
QUALITY OF THE INITIAL OBJECT HYPOTHESES.

good	part of an object	wrong
50 %	39 %	11 %

object several times.

In our experiments, the number of initial hypotheses was not limited, but a hypothesis had to consist of at least 10 points. To find planes, 1000 iterations of RANSAC were performed, and the tolerance was 2 mm. With around 500 3-D points, this takes about 15 ms on a standard PC with a 2.67GHz Intel i-7 CPU. For cylinder detection, the local surface normals for the Gaussian sphere were computed by fitting a plane through each point and its 4 nearest neighbours. To find a cylinder axis in the Gaussian sphere, 500 iterations of RANSAC were executed. At most 30 different axes were evaluated, where for each axis at most 10000 RANSAC iterations were executed to find the optimal cylinder radius and offset from the origin (less iterations if there are only few candidate points). The tolerance for deciding if a point lies on a hypothetical cylinder surface was 4 mm. Finding a cylinder in a set of 500 3-D points takes about 150-200 ms. When the first (and largest) planes or cylinders are found and their points are removed, the computation time is reduced significantly. On the average it takes around 350 ms to find all hypotheses. As RANSAC can easily be parallelized, this time can be reduced considerably on a multicore CPU.

The generated hypotheses can fall into three categories of correctness: Firstly, the hypotheses can be approximately identical with an object or at least those parts of it that contain visual features. Secondly, it can contain a part of the object, which frequently happens in the case of large objects. This is acceptable because such a hypothesis still allows a successful manipulation of the underlying object. Thirdly, the hypothesis may span over more than one object. This can lead to failed manipulation attempts unless the majority of the points lie on the pushed object. We carried out a number of experiments in different complex scenes, each containing 5-8 objects that stand close together and partly occlude each other. Table I shows the quality of the hypotheses in such scenes. "Good" means that the hypotheses approximately coincided with an object, "part of object" indicates that they contained a part of an object, and the "wrong" hypotheses contained parts of two or more objects. In simple scenes the hypotheses are usually correct or contain a part of a large object.

We applied our system to the learning of 15 different objects. The number of features contained in each initial object hypothesis varied strongly between the different objects. For the initial hypotheses, the numbers of features ranged from 21 to 153, the average was 53. During the learning process, after each pushing movement 20 - 150 new candidate points were added to the hypothesis, where the actual number strongly depended on the object (54 on average). The percentage of candidate points that were confirmed with the next push

TABLE II
OBJECT RECOGNITION SUCCESS RATE OF THREE EXAMPLE OBJECTS, AND THE AVERAGE OF ALL 15 OBJECTS THAT WERE LEARNED.

	init. hyp.	1 push	2 pushes	3 pushes	5 pushes
Book	57 %	54 %	77 %	85 %	90 %
Tea	65 %	77 %	91 %	93 %	97 %
Bottle	69 %	68 %	73 %	78 %	81 %
Average	68 %	65 %	79 %	86 %	92 %

appeared to be approximately the same for all objects, on the average 32%. The percentage of feature points of the initial hypothesis which were validated after the first push was approximately the same.

For the evaluation of the object recognition system, in addition to the 15 test objects mentioned above, another 25 objects were learned from presegmented images. Thus the complete database contained 40 objects. We tried to recognize the learned objects in complex scenes containing 5-8 objects. For the bag-of-features, a visual vocabulary of 1000 words was learned from 50000 features that were extracted from 25 images, each containing several objects. For each object, 15-20 histograms were learned, and we used 3-nearest-neighbours classification with χ^2 distance for recognition.

For three exemplary objects and the average of all 15 tested objects, table II shows the recognition results for the initial hypotheses and after n iterations of pushing and validation. On the average, the initial hypotheses lead to a recognition rate of 68%, which also gives an idea about their usefulness for segmentation. While hypotheses that approximately contain an object (compare table I) are usually classified correctly, those hypotheses which contain only a part of an object are frequently rejected or misclassified. Hypotheses that contain two or more objects are usually rejected.

After the first push and the subsequent verification of the hypothetical feature points, the average recognition rate is 65%, which is – somewhat surprisingly – slightly lower than for the initial hypothesis. As now only the confirmed points are used for recognition, the effect of this first push was mainly to remove the features from the object hypothesis that did not move in unison with the majority of feature points, or were not found in the next image. By that, the number of features is reduced to around 32% of the size of the initial hypothesis (see above). Apparently, this affects the recognition so strongly that the positive effect of eliminating the false features is voided. But after the second push, new confirmed features are added at each iteration, and now the positive effect is significant. The recognition rate immediately rises to 79% after the second push, 86% after the third and 92% after the fifth. It finally converges to a value between 92% and 95%.

This general tendency is also visible when looking at the particular objects. As the book is frequently divided into two or three initial hypotheses, it profits significantly from the accumulation of more features from the first to the second push. The tea can be recognized very reliably, while the bottle

has only very few features and is therefore more difficult to identify even with a good segmentation.

VII. Conclusion and future work

In this paper, we presented a method for the segmentation and learning of unknown objects in unstructured environments. We generate initial object hypotheses from 3-D points, which were obtained through stereo vision, by detecting planar and cylindrical surfaces amongst them. The hypotheses are then verified, corrected and extended by pushing them repeatedly. Objects are learned using bag-of-feature histograms based on the SIFT descriptors of the points belonging to the object. We have shown experimentally that the objects learned this way can later be recognized, and that the segmentation by pushing can serve as a powerful methodology for recognition in complex scenes.

One possibility to extend our method would be to allow other and more complex geometrical shapes for the initial hypothesis, like spheres, ellipsoids, superquadrics, geons etc. But since many common household objects can roughly be modeled by planes and cylinders and since the accumulation of features after the pushing movements is independent from the shape of the initial hypothesis, the benefit would probably be very limited. A more promising enhancement would be to additionally use different local descriptors. Especially the use of color information could prove to be helpful in complementing the greyscale-based SIFT descriptors. It is also an interesting question if our approach can be adapted to deal with more uniformly colored objects, e. g. by using maximally stable extremal regions (MSER) [20].

Acknowledgment

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/20072013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience. It was also supported by SRBPS, MEXT. A. Ude would like to thank NICT for its support within the JAPAN TRUST International Research Cooperation Program.

References

[1] G. Kootstra, J. Ypma, and B. de Boer, Active exploration and keypoint clustering for object recognition, in: *Proc. IEEE Int. Conf. Robotics and Automation*, Pasadena, CA, 2008.

[2] G. Metta and P. Fitzpatrick, Early integration of vision and manipulation, *Adaptive Behavior*, vol. 11, no. 2., 2003.

[3] G. Metta and P. Fitzpatrick, Grounding vision through experimental manipulation, *Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences*, vol. 361, no. 1811, 2003.

[4] P. Fitzpatrick, First contact: An active vision approach to segmentation, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, Nevada, 2003.

[5] D. Katz and O. Brock, Manipulating Articulated Objects With Interactive Perception, *IEEE Int. Conf. Robotics and Automation*, Pasadena, CA, 2008.

[6] A. Ude, D. Omrčen, and G. Cheng, Making object learning and recognition an active process, *Int. Journal of Humanoid Robotics*, vol. 5, no. 2, 2008.

[7] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann, Autonomous Acquisition of Visual Multi-View Object Representations for Object Recognition on a Humanoid Robot, *IEEE Int. Conf. Robotics and Automation*, Anchorage, Alaska, 2010.

[8] E. Stergaršek Kuzmič and A. Ude, Object segmentation and learning through feature grouping and manipulation, *10th IEEE-RAS Int. Conf. Humanoid Robots*, 2010.

[9] T. Asfour, K. Welke, P. Azad, A. Ude, and R. Dillmann, The Karlsruhe Humanoid Head, in: *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Daejeon, Korea, 2008.

[10] C. Harris and M. Stephens, A combined corner and edge detector, in: *Alvey Vision Conference*, pp. 147151, 1988.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004.

[12] M. A. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, vol. 24, no. 6, 1981.

[13] D. Pelleg and A. Moore, X-means: Extending K-means with Efficient Estimation of the Number of Clusters, in: *Proc. 17th Int. Conf. Machine Learning*, San Francisco, CA, 2000.

[14] T. Chaperon and F. Goulette, Extracting Cylinders in Full 3D Data Using a Random Sampling Method and the Gaussian Image, in: *Proc. Vision Modeling and Visualization Conference*, 2001.

[15] D. G. Lowe, Object recognition from local scale-invariant features, in: *Proc. Int. Conf. Computer Vision*, Corfu, Greece, 1999.

[16] B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, in: *Journal Optical Society America A*, vol. 4, 1987.

[17] P. Azad, T. Asfour, and R. Dillmann, Stereo-based 6D object localization for grasping with humanoid robot systems, *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2007.

[18] G. Csurka, C. Dance, L. X. Fan, J. Willamowski, and C. Bray, Visual categorization with bags of keypoints, in: *Proc. ECCV Int. Workshop on Statistical Learning in Computer Vision*, 2004.

[19] A. Ramisa, S. Vasudevan, D. Scaramuzza, R. L. de Mántaras, and R. Siegwart, A tale of two object recognition methods for mobile robots, in: *Proc. 6th Int. Conf. Computer Vision Systems*, 2008.

[20] J. Matas, O. Chum, M. Urba, and T. Pajdla, Robust wide baseline stereo from maximally stable extremal regions, in: *Proc. British Machine Vision Conference*, 2002.