Advanced Information and Knowledge Processing

For other titles published in this series, go to
www.springer.com/series/4738

Honghai Liu • Dongbing Gu • Robert J. Howlett • Yonghuai Liu
Editors

Robot Intelligence

An Advanced Knowledge Processing Approach

Springer

Editors
Dr. Honghai Liu
University of Portsmouth
Institute of Industrial Research
PO1 3QL Portsmouth
UK
honghai.liu@port.ac.uk

Prof. Dr. Dongbing Gu
University of Essex
Department of Computer Sc
Wivenhoe Park
CO4 3SQ Colchester
UK
dgu@essex.ac.uk

Dr. Robert J. Howlett
University Brighton
School of Engineering
Intelligent Signal Processing
Laboratories (ISP)
Moulsecoomb
BN2 4GJ Brighton
UK
rjhowlett@kesinternational.org

Prof. Dr. Yonghuai Liu
Aberystwyth University
Department of Computer Science
Ceredigion
SY23 3DB Aberystwyth
UK

AI&KP ISSN 1610-3947
ISBN 978-1-84996-328-2 e-ISBN 978-1-84996-329-9
DOI 10.1007/978-1-84996-329-9
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010931521

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

With the growing integration of machine learning techniques into robotics research, there is a need to address this trend in the context of robot intelligence. The multidisciplinary nature of robot intelligence provides a realistic platform for robotics researchers to apply machine learning techniques. One of the principal purposes of this book is to promote idea exchanges and interactions between different communities, which are beneficial and bringing fruitful solutions. Especially when the tasks robots are programmed to achieve become more and more complex, imprecise perception of the environments renders a difficult deliberative control strategy applied for robots for so many years. Understanding the environment where robots operate and then controlling robots gradually rely on machine learning techniques. It is more likely to better off with embedding control problems into the environment perception.

The major challenges for programming autonomous robots stem mainly from firstly the dynamic environment in which it is unable to predict when events will occur and the robots have to perceive their environment repeatedly, secondly uncertain sensory information that is inaccurate, noisy, or faulty, thirdly imperfect actuators that cannot guarantee perfect execution of actions due to mechanical, electrical, and servo problems, and finally limited time that constrains time intervals needed for sensor information processing, actuator control, and goal-oriented planning. As such, the robots cannot rely on their actions to predict motion results. Heavy computation would make the robots move and respond slowly to changes in the environment.

For autonomous mobile robots, early programming approaches followed a sequence: sensing the environment, planning trajectories, and controlling motors to move. With this kind of control strategies, the robot needs to "think" hard, consuming large amounts of time to model the environment and reason about what to do. In addition, modelling and reasoning methods vary with robot tasks and have not reached a widely accepted level of development. Furthermore, this type of control strategies is very fragile, as it can fail to deal with unpredictable events in dynamic environments even if the robot can model and reason precisely. Meanwhile, it is impossible to predict all the potential situations robots may encounter and to specify all the robot behaviors optimally in advance when programming them to achieve

complicated tasks in complex environments. Thus, robots have to learn from and adapt to their operating environments.

This volume aims to reflect the latest progresses made on central robotics issues, including robot navigation, human security and surveillance, human-robot interaction, flocking robots, multiple robot cooperation and coordination. The collected chapters not only represent the state-of-the-art research in robot development and investigation, but also demonstrate the application of a wide range of machine learning techniques that vary from artificial neural networks, evolutionary algorithms, fuzzy logic, reinforcement learning, k-means clustering, to multi-agent reinforcing learning. The book can be used as a valuable reference for robotics researchers, engineers, and practitioners for advanced knowledge, and university undergraduates and postgraduates who would like to specialize in robotics research and development.

Thirteen chapters are carefully selected from the extensive body of recent research work, which tackles the challenging issues of robotics development and applications with machine learning techniques. The selection is featured with the breadth of machine learning tools and emphasizes practical robot applications.

Skoglund et al. present a novel approach to robot skill acquisition from human demonstration. Usually the morphology of a robot manipulator is very different from that of the human arm. In this case, a human motion cannot be simply copied. The proposed approach uses a motion planner that operates in an object-related world-frame called hand-state to simplify a skill reconstruction and preserve the essential parts of the skill. In this way, the robot is able to generalize the learned skills to other similar skills without triggering a new learning process.

Palm et al. focus on the robot grasp recognition, which is a major part of the approach for Programming-by-Demonstration. Their work describes three different methods for grasp recognition for a human hand. The finger joint angle trajectories of human grasps are modeled by fuzzy modeling. Three methods for grasp recognition are compared with each other.

Cheng et al. investigate the multiple manipulators which need to achieve the same joint configuration to fulfill certain coordination tasks. Under the multi-agent framework, a robust adaptive control approach is proposed to deal with this consensus problem. Uncertainties and external disturbances in the robot's dynamics are considered, which is more practical in real-world applications. Due to the approximation ability of neural networks, the uncertain dynamics are compensated by the adaptive neural network scheme.

Ji et al. propose an exemplar-based view-invariant human action recognition framework to recognize the human actions from any arbitrary viewpoint image sequence. The proposed framework is evaluated in a public dataset and the results show that it not only reduces computational complexity, but it is also able to accurately recognize human actions using single cameras.

Khoury and Liu introduce the concept of fuzzy Gaussian inference as a novel way to build fuzzy membership functions that map underlying human motions to hidden probability distributions. This method is now combined with a genetic programming fuzzy rule based system in order to classify boxing moves from natural human motion capture data.

Zhou et al. consider the detection of hazards within the ground plane immediately in front of a moving pedestrian. Using epipolar constraints between two views, detected features are matched to compute the camera motion and reconstruct the 3-D geometry. For a less feature based scene a new disparity velocity based obstacle detection scheme is presented.

Tian and Tang explore the feasibility of using monocular vision for robot navigation. The path depth is learned by using the images captured in a single camera. Their work concentrates on finding passable regions from a single still color image and making the robot vision less sensitive to illumination changes.

Liu et al. propose a new model to characterize camera distortion in the process of the camera calibration. This model attempts to blindly characterize the overall camera distortion without taking the specific radial, decentering, or thin prism distortion into account. To estimate the parameters of interest, the well-known Levernburg-Marquardt algorithm is applied. To initialize the Levernburg-Marquardt algorithm, the results from the classical Tsai algorithm are estimated. After both the camera intrinsic and distortion parameters have been estimated, the distorted image points are corrected using again the Levernburg-Marquardt algorithm.

Wang and Gu present an approach to design a flocking algorithm by using fuzzy logic. The design of three basic behaviors in a flocking algorithm is discussed. They are alignment behavior, separation behavior, and cohesion behavior. Navigation control component is used in the design of cohesion behavior. To avoid becoming crowding or collision, an adaptive navigation gain is used. This gain changes with the number of neighbors. The flocking stability is analyzed and stability conditions are acquired from the stability analysis.

Oyekan et al. develop a behavior based control architecture for UAV surveillance mission. This architecture contains two layers: atomic action layer and behavior layer. They have also developed six atomic actions and ten behaviors for these layers. Various techniques have been used in the development, including adaptive PID controller, fuzzy logic controller, SURF algorithm, and Kalman filter.

Guo et al. present a novel anti-disturbance control strategy named hierarchical composite anti-disturbance control for a class of non-linear robotic systems with multiple disturbances. The strategy is established which includes a disturbance observer based controller and an H_∞ controller, stability analysis for two case studies are provided.

Ballantyne et al. present some of the key considerations for human guided navigation in the context of dynamic and complex indoor environments. Solutions and issues related to gesture recognition, multi-cue integration, tracking, target pursuing, scene association and navigation planning are discussed.

Kubota and Nishida discuss the adaptation of perceptual modules of a partner robot based on classification and prediction through actual interactions with a human. They proposed a prediction-based perceptual system consisting of the input layer, clustering layer, prediction layer, and perceptual module selection layer. They apply the proposed method to the actual interaction between a human and a human-like partner robot.

We would like to express our sincere thanks to all the authors who have contributed to the book and support during the book preparation. Without their support,

it is impossible to see the advent of this book. Thanks also go to Natasha Harding from Springer UK who kindly and effectively communicated between the publisher and our editors of this book. We feel especially grateful to our publisher, Springer, who kindly supports the research direction of robot intelligence and the publication of the book. Finally, it would be our pleasure that this book would be valuable, for in-depth understanding of robot intelligence from the advanced knowledge processing point of view, to a wide range of audience from multi-disciplinary research communities and industrial practitioners.

Portsmouth, UK — Honghai Liu
Colchester, UK — Dongbing Gu
Brighton, UK — Robert J. Howlett
Aberystwyth, UK — Yonghuai Liu

Contents

Contributors

Ala Al-Obaidi Smart Light Devices, Ltd., Aberdeen AB24 2YN, UK, ala@sldltd.com

James Ballantyne Institute of Biomedical Engineering, Imperial College of London, London, UK, james.ballantyne@imperial.ac.uk

Long Cheng Key Laboratory of Complex Sciences, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, chenglong@compsys.ia.ac.cn

Patrick R. Green Heriot-Watt University, Edinburgh, UK, P.R.Green@hw.ac.uk

Dongbing Gu School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO3 4SQ, UK, dgu@essex.ac.uk

Lei Guo School of Instrument Science and Opto-Electronics Engineering, Beihang University, Beijing 100191, China; Research Institute of Automation, Southeast University, Nanjing 210096, China, lguo@buaa.edu.cn

Zeng-Guang Hou Key Laboratory of Complex Sciences, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, hou@compsys.ia.ac.cn

Huosheng Hu School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO3 4SQ, UK, hhu@essex.ac.uk

Boyko Iliev Department of Technology, Orebro University, 70182 Orebro, Sweden, boyko.iliev@aass.oru.se

Anthony Jakas Department of Computer Science, Aberystwyth University, Ceredigion SY23 3DB, UK, ajj08@aber.ac.uk

Xiaofei Ji Intelligent Systems and Biomedical Robotics Group, School of Creative Technologies, The University of Portsmouth, Eldon Building, Portsmouth PO1 2DJ, UK; School of Automation, Shenyang Institute of Aeronautical Engineering, No. 37 Daoyi South Avenue, Shenyang 110136, China, xiaofei.ji@port.ac.uk

Edward Johns Institute of Biomedical Engineering, Imperial College of London, London, UK, edward.johns09@imperial.ac.uk

Bourhane Kadmiry Department of Technology, Orebro University, 70182 Orebro, Sweden, bourhane.kadmiry@tech.oru.se

Mehdi Khoury Intelligent Systems and Biomedical Robotics Group, School of Creative Technologies, The University of Portsmouth, Eldon Building, Portsmouth PO1 2DJ, UK, mehdi.khoury@port.ac.uk

Naoyuki Kubota Department of System Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan, kubota@tmu.ac.jp

Bo Li School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO3 4SQ, UK

Yibo Li School of Automation, Shenyang Institute of Aeronautical Engineering, No. 37 Daoyi South Avenue, Shenyang 110136, China, lyb20040612@yahoo.com.cn

Honghai Liu Intelligent Systems and Biomedical Robotics Group, School of Creative Technologies, The University of Portsmouth, Eldon Building, Portsmouth PO1 2DJ, UK, honghai.liu@port.ac.uk

Junjie Liu Department of Computer Science, Aberystwyth University, Ceredigion SY23 3DB, UK, jul06@aber.ac.uk

Yonghuai Liu Department of Computer Science, Aberystwyth University, Ceredigion SY23 3DB, UK, yyl@aber.ac.uk

Bowen Lu School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO3 4SQ, UK, blv@essex.ac.uk

Kenichiro Nishida Multimedia Soc Development Dept., Toshiba Corporation Semiconductor Company, 580-1, Horikawa-Cho, Saiwai-Ku, Kawasaki 212-8520, Japan, Kenichiro.nishida@toshiba.co.jp

John Oyekan School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO3 4SQ, UK, jooyek@essex.ac.uk

Rainer Palm Department of Technology, Orebro University, 70182 Orebro, Sweden, rub.palm@t-online.de

Alexander Skoglund Department of Technology, Orebro University, 70182 Orebro, Sweden, alexander.skoglund@aass.oru.se

Min Tan Key Laboratory of Complex Sciences, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, tan@compsys.ia.ac.cn

Yandong Tang State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, ytang@sia.cn

Jiandong Tian State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China; Graduate School of the Chinese Academy of Sciences, Beijing, China, tianjd@sia.cn

Salman Valibeik Institute of Biomedical Engineering, Imperial College of London, London, UK, salman.valibeik05@imperial.ac.uk

Andrew M. Wallace Heriot-Watt University, Edinburgh, UK, A.M.Wallace@hw.ac.uk

Xu Wang Key Laboratory of Complex Sciences, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, wangxu.zju@163.com

Zongyao Wang School of Computer Science and Electronic Engineering, University of Essex, Essex, UK, zwangf@essex.ac.uk

Xin-Yu Wen Research Institute of Automation, Southeast University, Nanjing 210096, China,

Charence Wong Institute of Biomedical Engineering, Imperial College of London, London, UK, charence.wong05@imperial.ac.uk

Xin Xin Faculty of Computer Science and Systems, Okayama Prefectural University, 111 Kuboki, Japan,

Guang-Zhong Yang Institute of Biomedical Engineering, Imperial College of London, London, UK, gzy@doc.ic.ac.uk

Huiyu Zhou Queen's University Belfast, Belfast, UK, H.Zhou@ecit.qub.ac.uk

Chapter 1
Programming-by-Demonstration of Robot Motions

Alexander Skoglund, Boyko Iliev, and Rainer Palm

Abstract In this chapter a novel approach to skill acquisition from human demonstration is presented. Usually the morphology of a robot manipulator is very different from the human arm and cannot simply copy a human motion. Instead the robot has to execute its own version of the skill demonstrated by the operator. Once a skill has been acquired by the robot it must also be able to generalize to other similar skills without starting a new learning process. By using a motion planner that operates in an object-related world-frame called hand-state, we show that this representation simplifies a skill reconstruction and preserves the essential parts of the skill.

1.1 Introduction

This article presents a method for imitation learning based on fuzzy modeling and a next-state-planner in a Programming-by-Demonstration (PbD) framework. For a recent comprehensive overview of PbD, (also called Learning from Demonstration) see [1]. PbD refers to a variety of methods where the robot learns how to perform a task by observing a human teacher, which greatly simplifies the programming process [2–5]. One major scientific challenge in PbD is how to make the robot *capable* of imitating a human demonstration. Although the idea of copying human motion trajectories using a simple teaching-playback method seems straightforward, it is not realistic for several reasons. Firstly, there is a significant difference in morphology between the human and the robot, known as the correspondence problem in imitation [6]. The difference in the location of the human demonstrator and the robot might force the robot into unreachable parts of the workspace or singular arm configurations even if the demonstration is perfectly feasible from human viewpoint. Secondly, in grasping tasks the reproduction of human hand motions is not possible

A. Skoglund (✉), B. Iliev, and R. Palm
Department of Technology, Orebro University, 70182 Orebro, Sweden
e-mail: alexander.skoglund@aass.oru.se; boyko.iliev@aass.oru.se; rub.palm@t-online.de

H. Liu et al. (eds.), *Robot Intelligence,*
Advanced Information and Knowledge Processing,
DOI 10.1007/978-1-84996-329-9_1,

since even the most advanced robot hands cannot match neither the functionality of the human hand nor its sensing capabilities. However, robot hands capable of autonomous grasping can be used in PbD provided that the robot can generate an appropriate reaching motion towards the target object, as we will demonstrate in this article.

In this article, we present an approach to learning of reaching motions where the robot uses human demonstrations in order to collect essential knowledge about the task. This knowledge, i.e., grasp-related object properties, hand-object relational trajectories, and coordination of reach and grasp motions is encoded and generalized in terms of *hand-state space* trajectories. The hand-state components are defined such that they are perception-invariant and define the correspondence between the human and robot hand. The hand-state representation of the task is then embedded into a next-state-planner (NSP) which enables the robot to perform reaching motions from an arbitrary robot configuration to the target object. The resulting reaching motion ensures that the robot hand will approach the object in such way that the probability for a successful grasp is maximized.

An NSP plans one step ahead from its current state. This contrasts to traditional robotic approaches which plan the entire trajectory in advance. One of the first researchers to use a NSP approach in imitation learning were Ijspeert et al. [7], where they encode the trajectory in an autonomous dynamical system with internal dynamic variables that shapes a "landscape" used for both point attractors and limit cycle attractors. For controlling a humanoid's reaching motion, Hersch and Billad [8] considered a combined controller with two controllers running in parallel; one controller acts in joint space, while the other one acts in Cartesian space. To generate reaching motions and avoiding obstacles simultaneously Iossifidis and Schöner [9] used attractor dynamics, where the target object acts as a point attractor on the end effector. The end-effector as well as a redundant elbow joint avoids an obstacle as the arm reaches for an object.

In our approach, a human demonstration guides the robot to grasp an object. Our use of an NSP differs from previous work [7–9] in the way it combines the demonstrated path with the robot's own plan. The use of hand-state trajectories distinguishes our work from most previous work on imitation. According to [7], most approaches in the literature use the joint space for motion planning while some other approaches use the Cartesian space.

To illustrate the approach we describe three scenarios where human demonstrations of goal-directed reach-to-grasp motions are reproduced by a robot. Specifically, the generation of reaching and grasping motions in pick-and-place tasks is addressed. In the experiments we test how well the skills perform the demonstrated task and how well they generalize over the workspace. The contributions of the work are as follows:

1. We introduce a novel next-state-planner based on a *fuzzy modeling* approach to encode human and robot trajectories.
2. We apply the *hand-state concept* [10] to encode motions in hand-state trajectories and apply this in PbD.

3. The combination of the NSP and the hand-state approach provides a tool to address the *correspondence problem* resulting from the different morphology of the human and the robot. The experiments show how the robot can generalize and use the demonstration despite the fundamental difference in morphology.

1.2 Learning from Human Demonstration

In PbD the idea is that the robot programmer (here called demonstrator) shows the robot what to do and from this demonstration an executable robot program is created. In our case, the demonstrator shows the task by performing it in a way that seems to be feasible for the robot. This means that we assume the demonstrator to be aware of the particular restrictions of the robot. In this work we consider only the body language of the demonstrator, i.e., the approach is entirely based on proprioceptive information. Interpretation of human demonstrations is done under two assumptions: the type of tasks and grasps that can be demonstrated are *a priori* known by the robot; we consider only demonstrations of power grasps (e.g., cylindrical and spherical grasps) which can be mapped to–and executed by–the robotic hand.

1.2.1 Interpretation of Demonstrations in Hand-State Space

To create the associations between human and robot reaching/grasping we employ the hand-state hypothesis from the Mirror Neuron System (MNS) model of [10]. The aim is to mimic the functionality of the MNS to enable a robot to interpret human goal-directed motions in the same way as its own motions. Following the ideas behind the MNS-model, both human and robot motions are represented in hand-state space. A hand-state trajectory encodes a goal-directed motion of the hand during reaching and grasping. Thus, the hand-state space is common for the demonstrator and the robot and preserves the necessary execution information. Hence, a particular demonstration can be converted into executable robot code and experience from multiple demonstrations is used to control/improve the execution of new skills. Thus, when the robot tries to imitate an observed reach and grasp motion, it has to move its own hand so that it follows a hand-state trajectory similar to the demonstrated one. If such a motion is successfully executed by the robot, a new robot skill is acquired. Seen from a robot perspective, human demonstrations are interpreted as follows.

If hand motions with respect to a potential target object are associated with a particular grasp type G_i, it is assumed that there must be a target object that matches the observed grasp type. In other words, the object has certain grasp-related features, also called *affordances* [10], which makes this particular grasp type appropriate. The position of the object can be retrieved by a vision system, or it can be estimated

from the grasp type and the hand pose, given some other motion capturing device. For each grasp type G_i, a subset of suitable object affordances is identified a priori and learned from a set of training data. In this way, the robot is able to associate observed grasp types G_i with their respective affordances A_i.

According to [10], the hand-state must contain components describing both the hand configuration and its spatial relation with respect to the affordances of the target object. Thus, the hand-state is defined in the form:

$$H = \{h_1, h_2, \ldots, h_{k-1}, h_k, \ldots, h_p\} \tag{1.1}$$

where $h_1 \ldots h_{k-1}$ are *hand-specific components* which describe the motion of the fingers during grasping. The remaining components $h_k \ldots h_p$ describe the motion of the hand in relation to the object. Thus, a hand-state trajectory contains a record of both the reaching and the grasping motions as well as their synchronization in time and space.

The hand-state representation equation (1.1) is invariant with respect to the actual location and orientation of the target object. Thus, demonstrations of object-reaching motions at different locations and initial conditions can be represented in a common domain. This is both the strength and weakness of the hand-state approach. Since the hand-state space has its origin in the goal object, a displacement of the object will not affect the hand-state trajectory. However, when an object is firmly grasped then the hand-state is fixed and will not capture a change in the object position relative to the base coordinate system. This implies that for object handling and manipulation the use of hand-state trajectories is limited.

1.2.2 Skill Encoding Using Fuzzy Modeling

Once the hand-state trajectory of the demonstrator is determined, it has to be modeled for several reasons. In [11] Ijspeert enumerates five important desirable properties for encoding movements have been identified. These are:

1. The representation and learning of a goal trajectory should be simple.
2. The representation should be compact (preferably parameterized).
3. The representation should be reusable for similar settings without a new time consuming learning process.
4. For recognition purpose, it should be easy to categorize the movement.
5. The representation should be able to act in a dynamic environment and be robust to perturbations.

Several methods for encoding human motions include Splines [12]; Hidden Markov Models (HMM) [13]; HMM combined with Non-Uniform Rational B-Splines [14]; Gaussian Mixture Models [2]; dynamical systems with a set of Gaussian kernel functions [11]. We developed a method based on fuzzy logic which deals with the above properties in a sufficient manner [15].

Let us examine the properties of fuzzy modeling with respect to the above enumerated desired properties. Fuzzy modeling is simple to use for trajectory learning and is a compact representation in form of a set of weights, gains and offsets (i.e., they fulfill property 1 and 2) [16]. To change a learned trajectory into a new one for a similar task with preserved characteristics of a motion, we proposed modification to the fuzzy time modeling algorithm [17], thus addressing property 3. Furthermore, the method satisfies property 4, as it was successfully used for grasp recognition by [15].

The algorithm for fuzzy time modeling of motion trajectories is briefly described as follows. Takagi and Sugeno proposed a structure for fuzzy modeling of input-output data of dynamical systems [18]. Let $\mathbf{X}$ be the input data set and $\mathbf{Y}$ be the output data set of the system with their elements $x \in \mathbf{X}$ and $y \in \mathbf{Y}$. The fuzzy model is composed of a set of c rules R from which rule R_i reads:

$$\text{Rule } i\text{: IF } x \text{ IS } \mathbf{X}_i \text{ THEN } y = A_i x + B_i \tag{1.2}$$

$\mathbf{X}_i$ denotes the ith fuzzy region in the fuzzy state space. Each fuzzy region $\mathbf{X}_i$ is defuzzified by a fuzzy set $\int w_{x_i}(x)|x$ of a standard triangular, trapezoidal, or bell shaped type. $W_i \in \mathbf{X}_i$ denotes the fuzzy value that x takes in the ith fuzzy region $\mathbf{X}_i$. A_i and B_i are fixed parameters of the local linear equation on the right hand side of (1.2).

The variable $w_i(x)$ is also called degree of membership of x in $\mathbf{X}_i$. The output from rule i is then computed by:

$$y = w_i(x)(A_i x + B_i). \tag{1.3}$$

A composition of all rules $R_1 \ldots R_c$ results in a summation over all outputs from (1.3):

$$y = \sum_{i=1}^{c} w_i(x)(A_i x + B_i) \tag{1.4}$$

where $w_i(x) \in [0, 1]$ and $\sum_{i=1}^{c} w_i(x) = 1$.

The fuzzy region $\mathbf{X}_i$ and the membership function w_i can be determined in advance by design or by an appropriate clustering method for the input-output data. In our case we used a clustering method to cope with the different non linear characteristics of input-output data-sets (see [19] and [20]). For more details about fuzzy systems see [21].

In order to model time dependent trajectories $x(t)$ using fuzzy modeling, the time instants t take the place of the input variable and the corresponding points $\mathbf{x}(\mathbf{t})$ in the state space becomes the outputs of the model.

The Takagi-Sugeno fuzzy model is constructed from captured data from the end-effector trajectory described by the nonlinear function:

$$\mathbf{x}(t) = \mathbf{f}(t) \tag{1.5}$$

where $\mathbf{x}(t) \in R^3$, $\mathbf{f} \in R^3$, and $t \in R^+$.

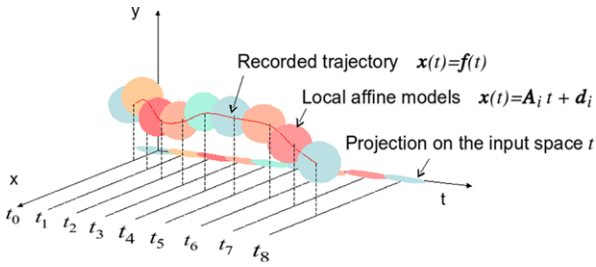

Fig. 1.1 Time-clustering principle

Equation (1.5) is linearized at selected time points t_i with

$$\mathbf{x}(t) = \mathbf{x}(t_i) + \left.\frac{\Delta \mathbf{f}(t)}{\Delta t}\right|_{t_i} \cdot (t - t_i) \tag{1.6}$$

resulting in a locally linear equation in t.

$$\mathbf{x}(t) = \mathbf{A}_i \cdot t + \mathbf{d}_i \tag{1.7}$$

where $\mathbf{A}_i = \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \in R^3$ and $\mathbf{d}_i = \mathbf{x}(t_i) - \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \cdot t_i \in R^3$. Using (1.7) as a local linear model one can express (1.5) in terms of an interpolation between several local linear models by applying Takagi-Sugeno fuzzy modeling [18] (see Fig. 1.1)

$$\mathbf{x}(t) = \sum_{i=1}^{c} w_i(t) \cdot (\mathbf{A}_i \cdot t + \mathbf{d}_i) \tag{1.8}$$

$w_i(t) \in [0, 1]$ is the degree of membership of the time point t to a cluster with the cluster center t_i, c is number of clusters, and $\sum_{i=1}^{c} w_i(t) = 1$.

The degree of membership $w_i(t)$ of an input data point t to an input cluster C_i is determined by

$$w_i(t) = \frac{1}{\sum_{j=1}^{c} \left(\frac{(t-t_i)^T M_{i\,pro}(t-t_i)}{(t-t_j)^T M_{j\,pro}(t-t_j)}\right)^{\frac{1}{\tilde{m}_{proj}-1}}}. \tag{1.9}$$

The projected cluster centers t_i and the induced matrices $M_{i\,pro}$ define the input clusters C_i $(i = 1 \ldots c)$. The parameter $\tilde{m}_{pro} > 1$ determines the fuzziness of an individual cluster [19].

1.3 Generation and Execution of Robotic Trajectories Based on Human Demonstration

This section covers generation and execution of trajectories on the actual robot manipulator. We start with a description of how the mapping from human to robot hand is achieved and how the hand-state components are defined. Then follows a description of the next-state-planner, which produces the actual robot trajectories.

1.3.1 Mapping Between Human and Robot Hand States

In the PbD framework, the hand-state components $h_1, \ldots h_p$ must be such that they can be recovered from both human demonstrations and the perception system of the robot. That is, the definition of H is *perception invariant* and can be updated from arbitrary types of sensory information. Figure 1.2 shows the definition of the hand-state in this article.

Let the human hand be at some initial state H_1. Then the hand moves along a certain path and reaches the final state H_f where the target object is held by the hand [17]. That is, the recorded motion trajectory can be seen as a sequence of states, i.e.,

$$H(t) : H_1(t_1) \rightarrow H_2(t_2) \rightarrow \cdots \rightarrow H_f(t_f). \tag{1.10}$$

To determine the hand-state representation of a demonstration the robot needs to have access to the complete motion trajectories of the teacher's hand since the motion must be in relation to the target object. This means that the hand-state trajectories can only be computed *during* a motion if the target object is known in advance.

Let $H_{des}(t)$ be the desired hand-state trajectory recorded from a demonstration. Since $H_{des}(t)$ cannot be executed by the robot without modification in the general case, we have to construct the *robotic* version of $H_{des}(t)$, denoted by $H^r(t)$, see Fig. 1.3 for an illustration.

To find $H^r(t)$ a mapping from the human grasp to the robot grasp a transformation is needed, denoted by T_h^r. This mapping is created as follows. We can measure the pose of the demonstrator hand and the robot hand holding the same object at fixed position and obtain T_h^r as a static mapping between the two poses. Thus, the target state H_f^r will be derived from the demonstration by mapping the goal configuration of the human hand H_f into a goal configuration for the robot hand H_f^r, using the transformation T_h^r:

$$H_f^r = T_h^r H_f. \tag{1.11}$$

The pose of the robot hand at the start of a motion defines the initial state H_1^r. Since H_f^r represents the robot hand holding the object, it has to correspond to a stable

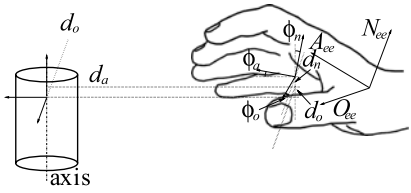

Fig. 1.2 The hand-state describes the relation between the hand pose and the object affordances. N_{ee} is the normal vector, O_{ee} the side (orthogonal) vector and A_{ee} is the approach vector. The vector Q_{ee} is the position of the point. The same definition is also valid for boxes, but with the restriction that the hand-state frame is completely fixed, it cannot be rotated around the symmetry axis

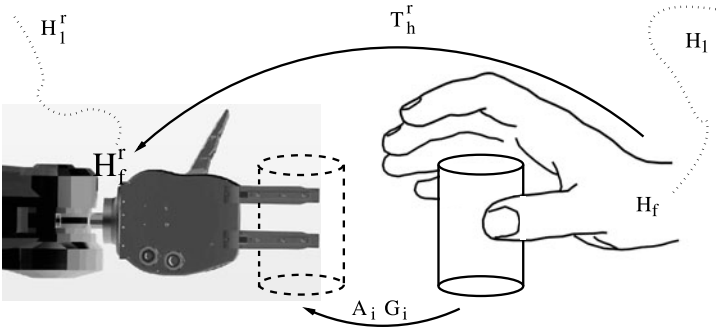

Fig. 1.3 Mapping from human hand to robotic gripper

grasp. For a known object, suitable H_f^r can either be obtained by simulation [22], grasp planning or by learning from experimental data. Thus, having a human hand-state H_f and their corresponding robot hand-state H_f^r, T_h^r is obtained as:

$$T_h^r = H_f^r H_f^{-1}. \tag{1.12}$$

It should be noted that this method is only suitable for power grasps. In the general case it might produce ambiguous results or rather inaccuarate mappings.

One advantage of using one demonstrated trajectory as the desired trajectory over trajectory averaging (e.g., [2] or [23]) is that the average might contain two essentially different trajectories [14]. By capturing a human demonstration of the task, the synchronization between reach and grasp is also captured, demonstrated in [24]. Other ways of capturing the human demonstrating, such as kinesthetics [2] or by a teach pendant (a joystick), cannot capture this synchronization easily.

1.3.2 Definition of Hand-States for Specific Robot Hands

Having the initial state H_1^r and the target state H_f^r defined, we have to generate the trajectory between the two states. In principle, we could transform $H_{des}(t)$ using (1.11) in such way that it has its final state in H_f^r. Then, the robot starts at H_1^r, approaches the displaced demonstrated trajectory and tracks it until the target state is reached. However, such an approach would not take trajectory constraints into account. Thus, it is also necessary to specify exactly how to approach $H_{des}(t)$ and what segments must be tracked accurately. driving the hand.

A hand-state trajectory must be constructed from the demonstrated trajectory. From the recorded demonstration we reconstruct the end-effector trajectory, represented by a time dependent homogeneous matrix $T_{ee}(t)$. Each element is represented by the matrix

$$T_{ee} = \begin{pmatrix} N_{ee} & O_{ee} & A_{ee} & Q_{ee} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.13}$$

where N_{ee}, O_{ee} and A_{ee} are the normal vector, the side vector, and the approach vector respectively. The last vector Q_{ee} is the position. The matrix T_{ee} is defined differently for different end-effectors, for example, the human hand is defined as in Fig. 1.2.

There is evidence that the internal models of arm dynamics found in biological systems are state-dependent rather than time-dependent [25]. Therefore, when we transform human demonstrations into robot motions we define *distance to object* d, as an additional scheduling variable for hand-state trajectories. To preserve the velocity profile from the human demonstration the distance to the target is modeled as a function of time using fuzzy time-modeling, see Sect. 1.2.2. The inputs to the fuzzy modeling is the Euclidean distance at each instance t of time:

$$d(t) = \sqrt{(Q_{ee}(t) - P)^2} \tag{1.14}$$

where Q_{ee} and P are the end-effector position and object position respectively.

The same procedure is applied to the hand-state trajectories. Two types of models are needed: one modeling of the hand-state as a function of time, and one as a function of distance. In this article a general formulation of the hand-state is adopted to fit the two states (open and close) for the anthropomorphic hand. We formulate the hand-state as:

$$H(t) = [d_n(t)\ d_o(t)\ d_a(t)\ \phi_n(t)\ \phi_o(t)\ \phi_a(t)]. \tag{1.15}$$

The individual components denote the position and orientation of the end-effector. The first three components, $d_n(t)$, $d_o(t)$ and $d_a(t)$, describe the distance from the object to the hand along the three axes n, o and a with the object as the base frame. The next three components, $\phi_n(t)$, $\phi_o(t)$ and $\phi_a(t)$, describe the rotation of the hand in relation to the object around the three axes n, o and a. The notion of the hand-state used in this section is illustrated in Fig. 1.2.

The components of the hand-state, as a function of distance, are given by:

$$H(d) = [d_n(d)\ d_o(d)\ d_a(d)\ \phi_n(d)\ \phi_o(d)\ \phi_a(d)] \tag{1.16}$$

where the hand-state components are the same as in (1.15), but with $d \in R^1$ instead of t. The role of the scheduling variable d is important since it expresses *when* the robot should move to the next state, while the hand-state variables reflect *where* the hand should move. Thus, d synchronizes the motions' *when* (dynamics and synchronization) and *where* (desired path) of the reach and grasp.

Note that with this simplified definition of H we cannot determine the human grasp type, since we have omitted the finger specific components of the hand-state. In [24] we give an account of how these components can be used to synchronize reaching with grasping. Grasp classification is out of scope of this article, because only power grasps are used in our experiments. Thus, the grasp type is assumed to be known $G = \{cylindrical, spherical, plane\}$; the affordances are: position, size, and

cylinder axis $A = \{width, axis\}$ or box $A = \{width, length, N\text{-}axis, O\text{-}axis, A\text{-}axis\}$. See [26] for grasp taxonomy.

1.3.3 Next-State-Planners for Trajectory Generation

In this section we present the next-state-planner (NSP) that balances its actions between *following a demonstrated trajectory* and *approaching the target*, first presented in [24]. The NSP is inspired by the Vector Integration To Endpoint (VITE) planner suggested by Bullock and Grossberg [27]. The VITE planner is a biologically inspired planner for human control of reaching motions. The NSP-approach requires a control policy, i.e., a set of equations describing the next action from the current state and some desired behavior.

The proposed NSP generates a hand-state trajectory for the robot using the TS fuzzy-model of a demonstration. As the resulting hand-state trajectory $H^r(t)$ can easily be converted to Cartesian space, we can use the inverse kinematics provided by the controller for the robot arm. The TS fuzzy-model serves as a motion primitive for the arm's reaching motion. The initial hand-state of the robot is determined from its current configuration and the position and orientation of the target object, since these are known at the end of the demonstration. Then, the desired hand-state H_d^r is computed from the TS fuzzy time-model (see (1.8)). The desired hand-state H_d is fed to the NSP. Instead of using only one goal attractor as in VITE [27], and additional attractor—the desired hand-state trajectory—is used at each state. The system has the following dynamics:

$$\ddot{H} = \alpha(-\dot{H} + \beta(H_g - H) + \gamma(H_d - H)) \tag{1.17}$$

where H_g is the hand-state goal, H_d the desired state, H is the current hand-state, $\dot{H}$ and $\ddot{H}$ are the velocity and acceleration respectively. α is a positive constant and β, γ are positive weights for the goal and tracking point, respectively.

If the last term $\gamma(H_d - H)$ in (1.17) is omitted, i.e., $\gamma = 0$, then the dynamics is *exactly* as the VITE planner [27]. Indeed, if no demonstration is available the planner can still produce a motion if the target is known. Similarly, if the term $\beta(H_g - H)$ is omitted, the planner becomes a trajectory following controller. If the final position from the demonstration can be used for gasping, as in [28], it is possible to set $\beta = 0$, which we do in our experiments in Sects. 1.5.1 and 1.5.2. The reason for setting β to zero is that the demonstration towards the goal will end at the goal, making the term $\beta(H_g - H)$ redundant if the goal cannot be estimated more accurately using some other sensor system as in [29]. Then the variance across multiple demonstrations is used to determine γ, which controls the behavior of the NSP.

Analytically, the poles in (1.17) are:

$$p_1, p_2 = -\frac{\alpha}{2} \pm \sqrt{\frac{\alpha^2}{4} - \alpha\gamma}. \tag{1.18}$$

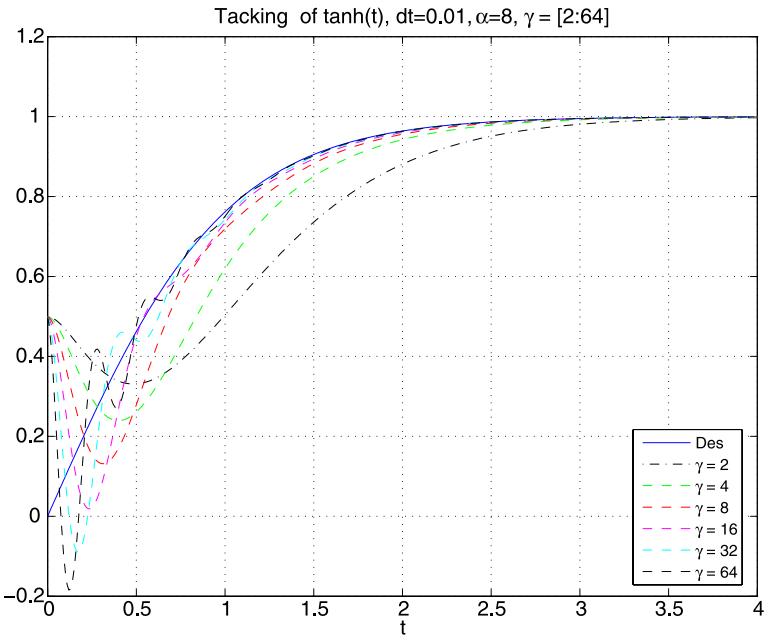

Fig. 1.4 The dynamics of the planner for six different values of γ. The tracking point is $\tanh(t)$, with $dt = 0.01$ and α is fixed at 8. A low value on $\gamma = 2$ produces slow dynamics (*black dot-dashed line*), while a high value $\gamma = 64$ is fast but overshoots the tracking point (*black dashed line*)

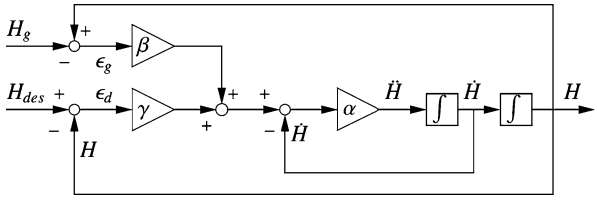

Fig. 1.5 Hand-state planner architecture. H_g is the desired hand-state goal, H_{des} is the desired hand-state at the current distance to target

The real part of p_1 and p_2 will be ≤ 0, which will result in a stable system [30]. Moreover, $\alpha \not\leq 4\gamma$ and $\alpha \geq 0$, $\gamma \geq 0$ will contribute to a critically damped system, which is fast and has small overshoot. Figure 1.4 shows how different values γ affect the dynamics of the planner.

The controller has a feedforward structure as in Fig. 1.5. The reason for this structure is that a commercial manipulator usually has a closed architecture, where the controller is embedded in the system. For this type of manipulators, a trajectory is usually pre-loaded and then executed. Therefore, we generate the trajectories in batch mode for the ABB140 manipulator. Since our approach is general, for a given different robot platform with hetroceptive sensors (e.g., vision) our method can be implemented in a feedback mode, but this requires that the hand-state $H(t)$ can be measured during execution.

1.3.4 Demonstrations of Pick-and-Place Tasks

The demonstration of the task is performed with the demonstrator (teacher) standing in front of the robot. Then the *task*, i.e., a pick-and-place of an object, is shown. The target object is determined from the task demonstration, where the center point of the grasp can be estimated from the grasp type. For example, grasp recognition (see [31]) can improve the estimate of the object position and orientation. In our experiments we assume a power grasp to determine the orientation of the object, while the motion capturing system is used to record the position of the object. The task demonstration contains the trajectories which the robot should execute to perform the task.

1.3.4.1 Variance from Multiple Demonstrations

When multiple demonstrations of a skill are available to the robot we can obtain a generalized version of that skill. We exploit the fact that when humans grasp the same object several times they seem to repeat the same grasp type which leads to similar approach motions. Based on that, multiple demonstrations of a skill become more and more similar to each other the closer one gets to the target state. This implies that successful grasping requires an accurate positioning of the hand in a region near the object while the path towards this area is subject to less restrictions. Therefore, by looking at the variance of several demonstrations the importance of each hand-state component can be determined. The variance of the hand-state as a function of the distance to target d is given by:

$$var({}^{k}h(d)) = \frac{1}{n-1}\sum_{i=1}^{n}({}^{k}h_i(d) - mean({}^{k}h(d)))^2 \tag{1.19}$$

where d is the Euclidean distance to the target, ${}^{k}h_i$ is the kth hand-state parameter of ith demonstration and n is the number of demonstrations. Figure 1.6 shows how the variance decreases as the distance to the object decreases. This means that the position and orientation of the hand are less relevant when the distance to the target increases.

1.4 Experimental Platform

For these experiments human demonstrations of a pick-and-place task are recorded with two different subjects, using the PhaseSpace Impulse motion capturing system described below. The Impulse motion capturing system consists of four cameras mounted around the operator to register the position of the LEDs. Each LED has a unique ID by which it is identified. Each camera can process data at 480 Hz and has 12 Mega pixel resolution resulting in sub-millimeter precision. The Impulse systems

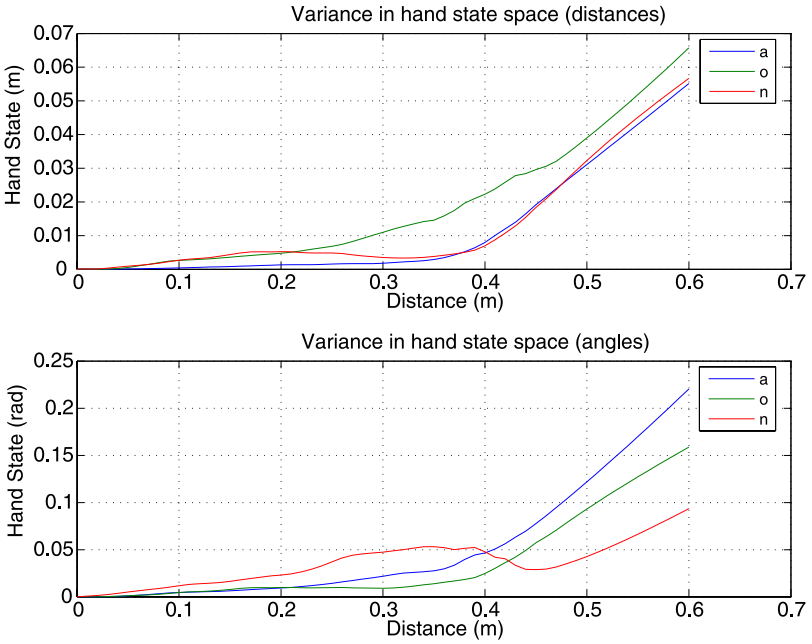

Fig. 1.6 Position- and orientation-variance of the hand-state trajectories as function of distance, across 22 demonstrations of a reaching to grasp motion. Note that distances over 0.47 are extrapolations made by the clustering method

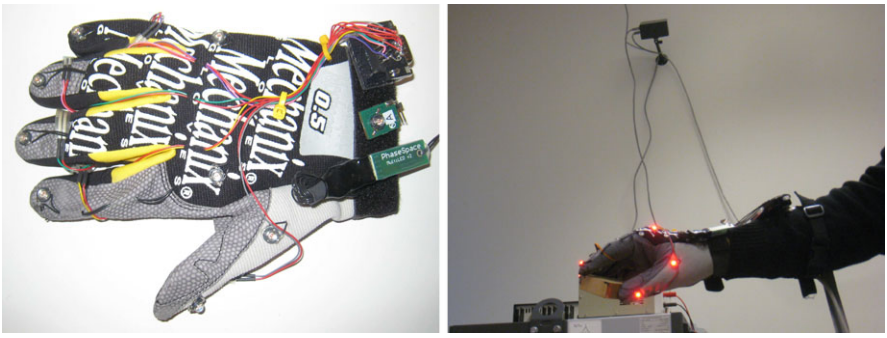

Fig. 1.7 *Left*: The glove used in the Impulse motion capturing system from PhaseSpace. The glove from the top showing the LEDs. *Right*: The system in use showing one of the cameras and the LED on the glove

can be seen in the upper right picture in Fig. 1.7. The operator wears a glove with LEDs attached to it, see upper left picture in Fig. 1.7. Thus, each point on the glove can be associated with a finger, the back of the hand or the wrist. To compute the orientation of the wrist, three LEDs must be visible during the motion. The back of the hand is the best choice since three LEDs are mounted there and they are most

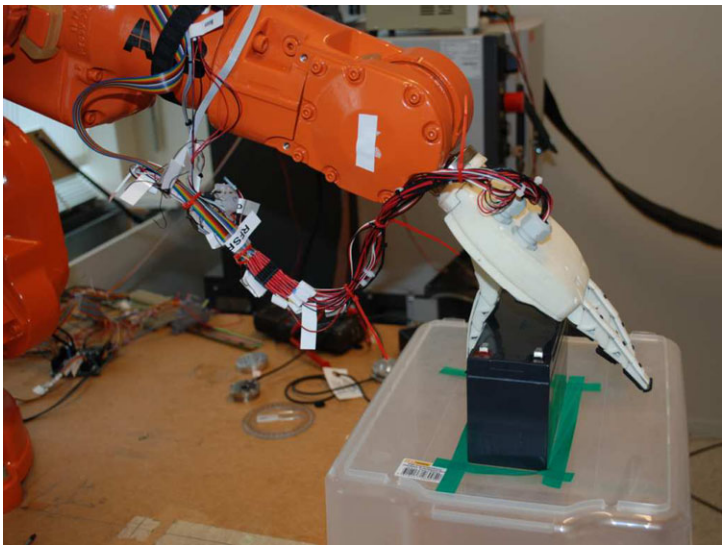

Fig. 1.8 The anthropomorphic gripper KTHand used in the experiments

of the time visible for at least three cameras. One LED is mounted on each finger tip, and the thumb has one additional LED in the proximal joint. One LED is also mounted on the target object.

The motions are automatically segmented into reach and retract motions using the velocity profile and distance to the object. The robot used in the experiments is the industrial manipulator ABB IRB140. In this experiment we use the anthropomorphic gripper KTHand (Fig. 1.8), which can perform power grasps (i.e., cylindrical and spherical grasps) using a hybrid position/force controller. For details on the KTHand, see [32].

1.5 Experimental Evaluation

In this section we provide an experimental evaluation of the methods presented. The 1st experiment deals with the task to learn a robot skill from human demonstration. The 2nd experiment shows how well the learned trajectories can be generalized w.r.t. different workspaces especially the workspace of the human operator and the robot's workspace. In the 3rd experiment a complete pick-and-place task is executed.

1.5.1 Experiment 1: Learning from Demonstration

For this experiment 26 task demonstrations of a pick-and-place task were performed using a soda can for performing a spherical grasp. To make the scenario more real-

istic the object is placed with respect to what is convenient for the human operator and what *seems* to be feasible for the robot.

Five of the 26 demonstrations were discarded in the segmentation (see [4]) and modeling process for reasons such as failure to segment the demonstrations into three distinct motions (approach, transport and retract) or the amount of data were not enough for modeling because of occlusions. Only the reach-to-grasp phase is considered in this experiment. All 21 demonstrations were used for trajectory generation and to compute the variance, shown in Fig. 1.6. Moreover, the variance is used to compute the γ-gain, which determines how much the robot can deviate from the followed trajectory. The trajectory generator produced 21 reaching motions, one from each demonstration, which are loaded to the robot controller and executed. By using each demonstrated trajectory as the desired trajectory H_d instead of building an average of them we avoid fusing of essentially different trajectories into a possibly incoherent trajectory. Large differences will instead affect the variance, resulting in a small γ-gain. In eight attempts, the execution succeeded while 13 attempts failed because of unreachable configurations in joint space. This could be prevented by placing the robot at a different location with better reachability. Moreover, providing the robot with more demonstrations, with higher variations in the path, will lead to fewer constraints. Two sample hand-state trajectories of the successfully generated ones are shown in Fig. 1.9. In the top graphs it is shown how for different initial locations the generated trajectory converges towards the desired trajectory. The bottom graphs shows how γ varies over time, to make the generated trajectory H^r follow the desired H_d.

In the eight successfully executed reaching motions we measured the variation in position of the gripper, shown in Fig. 1.10, which is within the millimeter range. This means that the positioning is accurate enough to enable successful grasping using an autonomous gripper, such as the Barrett hand [33] or the KTHand.

1.5.1.1 Importance of the Demonstration

The weight γ reflects the importance of the path, acquired from variance, see Sect. 1.3.4.1. For experiment 1 and 2, we have empirically found γ to produce satisfying results at:

$$\gamma_{pos} = 0.3 \frac{1}{\sqrt{Var(H_{xyz}(d))}},$$

$$\gamma_{ori} = 5 \frac{1}{\sqrt{Var(H_{rpy}(d))}}$$

where γ_{pos} and γ_{ori} are the weights for position and orientation, respectively. $Var(H_{xyz}(d))$ and $Var(H_{rpy}(d))$ are the variance for the position and orientation respectively, from (1.19), of the respective hand state component. α_{pos} and α_{ori} are fixed during our experiments at 8 and 10, respectively, with a time step $dt = 0.01$.

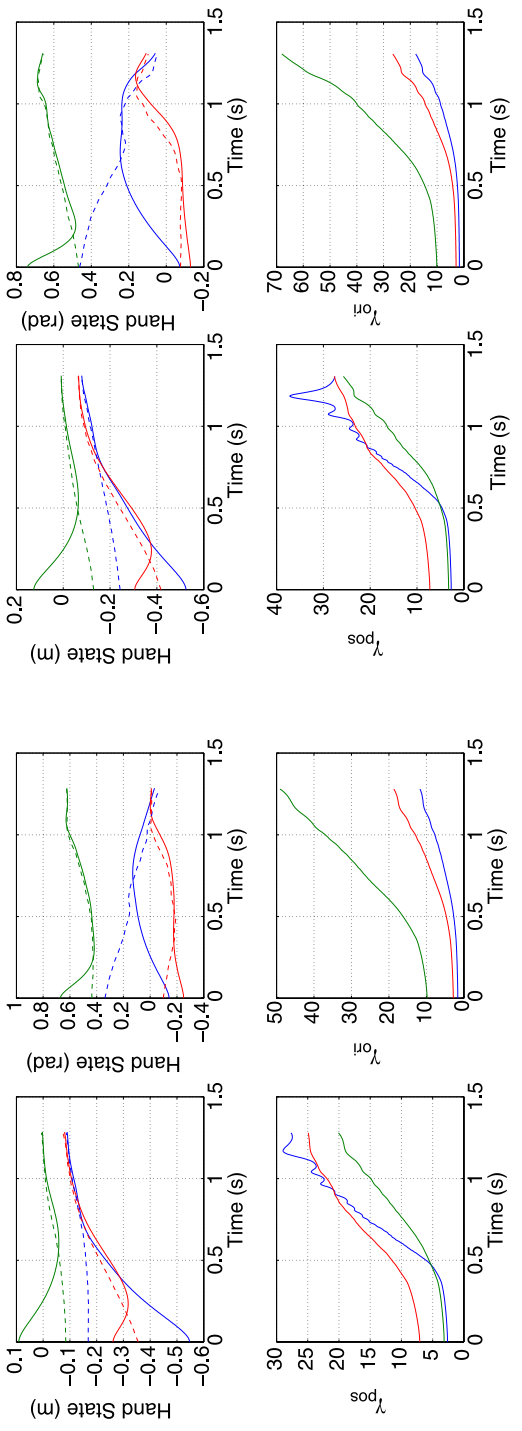

Fig. 1.9 Two samples of demonstrations, and the corresponding imitations. The *top graphs* shows the hand-state trajectories, position and orientation for each of the demonstrations/imitation. The *solid lines* are the trajectories produced by the planner, *dashed lines* are the model constructed from the recorded demonstration

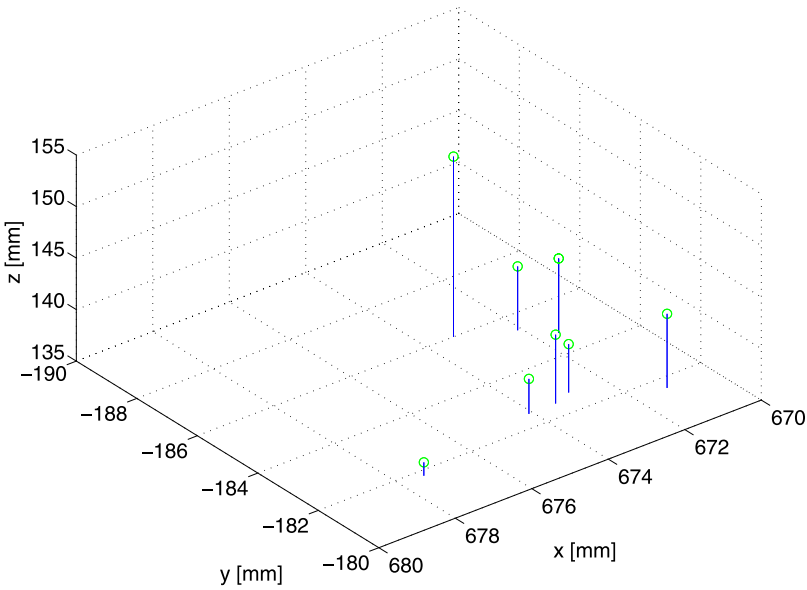

Fig. 1.10 The end effector position at the end of the motion for the 8 successfully executed trajectories. The positioning accuracy is within the millimeter range; 6 mm along x, 4 mm along y and 12 mm along z

These gains were chosen to provide dynamic behavior similar to the demonstrated motions, but other criteria can also be used.

The next-state planner uses the demonstration to generate a similar hand-state trajectory, using the distance as a scheduling variable. Hence, the closer to the object the robot is the more important it becomes to follow the demonstrated trajectory. This property is reflected by adding a higher weight to the trajectory-following dynamics when we get closer to the target; in reverse a long distance to the target leads to a lower weight to the trajectory following dynamics.

1.5.2 Experiment 2: Generalization in Workspace

In this experiment, the generalization of the method is tested. This is done by examining whether feasible trajectories are generated when the object is placed at arbitrary locations and when the initial configuration of the manipulator is very different from the demonstration. This determines how the trajectory planner handles the correspondence problem in terms of morphological differences. In experiment 2 the same data set was used as in experiment 1. Three tests were performed to evaluate the trajectory generator in different parts of the workspace.

1. Trajectories are generated when the manipulator's end-effector starts directly above the object at the desired final position with the desired orientation, that

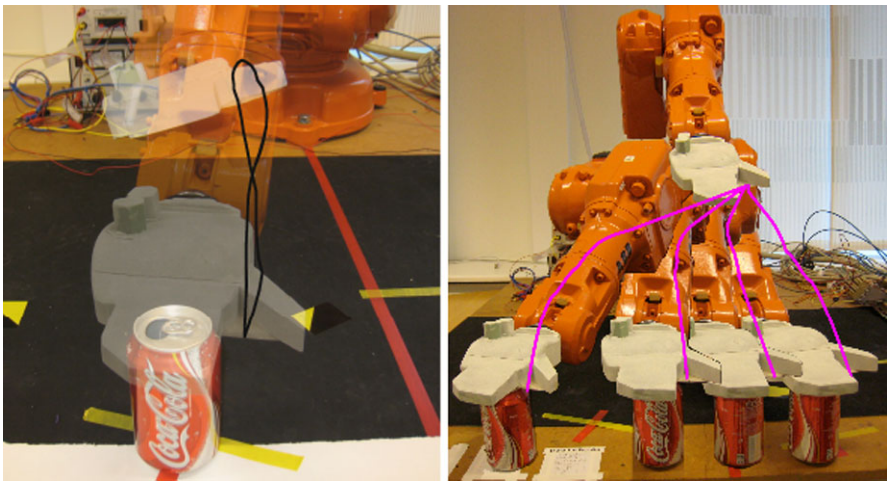

Fig. 1.11 *Left*: A trajectory generated when the initial position is the same as the desired final position, showing that the method generate trajectories as similar to the demonstration as possible based on the distance. *Right*: The object is placed at four new locations within the workspace

is $H_1^r = H_f^r$. The resulting trajectory is shown to the left in Fig. 1.11. Four additional cases are also tested displacing the end-effector by 50 mm in $+x$, $-y$, $+y$, and $+z$ direction from H_f, all with very similar results (from the robot's view: x is forward, y left and z up).

2. The object is placed at four different locations within the robot's workspace; displaced 100 mm along the x-axis, and -100 mm, $+100$ mm, $+200$ mm, and $+300$ mm along the y-axis, seen to the right in Fig. 1.11. The initial pose of the manipulator is the same in all reaching tasks. The planner successfully produces four executable trajectories to the respective object position.
3. We tested the reaching of the object at a fixed position from a random initial configuration. Figure 1.12 shows the result from two random initial positions where one trajectory is successfully tracked but the other one fails. The failure is a result of operation in hand-state space instead of joint space, and it might therefore have a tendency to go onto unreachable joint space configurations, as seen in the right column of Fig. 1.12. To prevent this it is possible to combine two controllers: one operating in joint space and the other in hand-state space, similar to the approach suggested by [34], but at the price of violating the demonstration constraints.

The conclusion from this experiment is that the method generalizes well in the tested scenarios, thus adequately addressing the correspondence problem. However, the unreachability problem has to be addressed in future research to investigate how the robot should balance the two contradiction goals: reaching an object in its own way, with the risk of collision, or reaching an object as the demonstrator showed. Indeed, if the robot has more freedom to choose the path it is more likely to avoid unreachable configurations. However, such freedom increases the risk for collision.