Programming-by-Demonstration of Reaching Motions using a Next-State-Planner

Alexander Skoglund, Boyko Iliev
and Rainer Palm
Örebro University
Sweden

1. Introduction

Programming-by-Demonstration (PbD) is a central research topic in robotics since it is an important part of human-robot interaction. A key scientific challenge in PbD is to make robots capable of imitating a human. PbD means to instruct a robot how to perform a novel task by observing a human demonstrator performing it. Current research has demonstrated that PbD is a promising approach for effective task learning which greatly simplifies the programming process (Calinon et al., 2007), (Pardowitz et al., 2007), (Skoglund et al., 2007) and (Takamatsu et al., 2007). In this chapter a method for imitation learning is presented, based on fuzzy modeling and a next-state-planner in a PbD framework. For recent and comprehensive overviews of PbD, (also called "Imitation Learning" or "Learning from Demostration") see (Argall et al., 2009), (Billard et al., 2008) or (Bandera et al., 2007).

What might occur as a straightforward idea to copy human motion trajectories using a simple teaching-playback method, it turns out to be unrealistic for several reasons. As pointed out by Nehaniv & Dautenhahn (2002), there is significant difference in morphology between body of the robot and the robot, in imitation learning known as the correspondence problem. Further complicating the picture, the initial location of the human demonstrator and the robot in relation to task (i.e., object) might force the robot, into unreachable sections of the workspace or singular arm configurations. Moreover, in a grasping scenario it will not be possible to reproduce the motions of the human hand since there so far do not exist any robotic hand that can match the human hand in terms of functionality and sensing. In this chapter we will demonstrate that the robot can generate an appropriate reaching motion towards the target object, provided that a robotic hand with autonomous grasping capabilities is used to execute the grasp.

In the approach we present here the robot observes a human first demonstrating the environment of the task (i.e., objects of interest) and the the actual 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 invariant with respect to perception, and includes the mapping between the human and robot hand, i.e., the correspondence. To

enable a *next-state-planner* (NSP) to perform reaching motions from an arbitrary robot configuration to the target object, the hand-state representation of the task is then embedded into the planner.
An NSP plans only one step ahead from its current state, which contrasts to traditional robotic approaches where the entire trajectory is planned in advance. In this chapter we use the term "next-state-planner", as defined by Shadmehr & Wise (2005), for two reasons. Firstly, because it emphasizes on the next–state planning ability, the alternative term being "dynamic system" which is very general. Secondly, the NSP also act as a *controller* which is an appropriate name, but "next-state-planner" is chosen because the controller in the context of an industrial manipulator refers to the low level control system. In this chapter the term planner is more appropriate. Ijspeert et al. (2002) were one of the first researchers to use an NSP approach in imitation learning. A humanoid robot learned to imitate a demonstrated motion pattern by encoding the trajectory in an autonomous dynamical system with internal dynamic variables that shaped a "landscape" used for both point attractors and limit cycle attractors. To address the above mention problem of singular arm configurations Hersch & Billard (2008) considered a combined controller with two controllers running in parallel; one controller acts in joint space, while the other one acts in Cartesian space. This was applied in a reaching task for controlling a humanoid's reaching motion, where the robot performed smooth motion while avoiding configurations near the joint limits. In a reaching while avoiding an obstacle task, Iossifidis & Schöner (2006) used attractor dynamics, where the target object acts as a point attractor on the end effctor. Both the end-effector and the redundant elbow joint avoided the obstacle as the arm reaches for an object.
The way to combine the demonstrated path with the robots own plan distinguishes our use of the NSP from from previous work (Hersch & Billard, 2008), (Ijspeert et al., 2002) and (Iossifidis & Schöner, 2006). Another difference is the use of the hand state space for PbD; most approaches for motion planning in the literature uses joint space (Ijspeert et al., 2002) while some other approaches use the Cartesian space.
To illustrate the approach we describe two 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 as well as the ability to learn from self observation. In the experiments we test how well the skills perform the demonstrated task, how well they generalize over the workspace and how skills can be adapted from self execution. 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* (Oztop & Arbib, 2002) 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 shows how the robot can generalize and use the demonstration despite the fundamentally difference in morphology.
4. We present a performance metric for the NSP, which enables the robot to evaluate its performance and to adapt its actions to fit its own morphology instead of following the demonstration.

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. We assume the demonstrator to be aware of the particular restrictions of the robot. Given this assumption, the demonstrator shows the task by performing it in a way that seems to be feasible for the robot. In this work the approach is entirely based on proprioceptive information, i.e., we consider only the body language of the demonstrator. Furthermore, interpretation of human demonstrations is done under two assumptions: firstly, the type of tasks and grasps that can be demonstrated are *a priori* known by the robot; secondly, we consider only demonstrations of power grasps (e.g., cylindrical and spherical grasps) which can be mapped to–and executed by–the robotic hand.

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 (Oztop & Arbib, 2002). The aim is to resemble the functionality of the MNS to enable a robot to interpret human goal-directed reaching 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 the corresponding robot trajectory and experience from multiple demonstrations is used to control/improve the execution of new skills. Thus, when the robot execute the encoded hand-state trajectory of a reach and grasp motion, it has to move its own end-effector 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 denoted G_i, it is assumed that there must be a target object that matches this observed grasp type. In other words, the object has certain grasp-related features which *affords* this particular type of grasp (Oztop & Arbib, 2002). The position of the object can either 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 (e.g., magnetic trackers). A subset of suitable object affordances is identified a priori and learned from a set of training data for each grasp type G_i. In this way, the robot can associate observed grasp types G_i with their respective affordances A_i.

According to Oztop & Arbib (2002), 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 general definition of the hand-state is in the form:

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

where $h_1 \dots h_{k-1}$ are *hand-specific components* which describe the motion of the fingers during a reach-to-grasp motion. The remaining components $h_k \dots h_p$ describe the motion of the hand in relation to the target 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 in Eqn. 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 origin of the hand-state space is in the target object, a displacement of the object will not affect the hand-state trajectory. However, when an object is firmly grasped then, the hand-state become fixated and will not capture a motion relative to the base coordinate system. This implies that for object handling and manipulation the use of a single hand-state trajectory is insufficient.

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. Five important and desirable properties for encoding movements have been identified, and Ijspeert et al. (2001) enumerates them as follows:

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.

A number of methods for encoding human motions have previously been proposed including splines (Ude, 1993); Hidden Markov Models (HMM) (Billard et al., 2004); HMM combined with Non-Uniform Rational B-Splines (Aleotti & Caselli, 2006); Gaussian Mixture Models (Calinon et al., 2007); dynamical systems with a set of Gaussian kernel functions (Ijspeert et al., 2001). The method we propose is based on fuzzy logic which deals with the above properties in a sufficient manner (Palm et al., 2009).
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) (Palm & Iliev, 2006). To change a learned trajectory into a new one for a similar task with preserved characteristics of a motion, we proposed a modification of the fuzzy time modeling algorithm (Iliev et al., 2007), thus addressing property 3. The method also satisfies property 4, as it was successfully used for grasp recognition by (Palm et al., 2009). In (Skoglund, Iliev & Palm, 2009) we show that our fuzzy modeling based NSP is robust to short perturbations, like NSPs in general are known to be robust to perturbations (Ijspeert et al., 2002) and (Hersch & Billard, 2008).
Here it follows a description of the fuzzy time modeling algorithm for motion trajectories. Takagi and Sugeno proposed a structure for fuzzy modeling of input-output data of dynamical systems (Takagi & Sugeno, 1985). 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:

$$\text{Rule } i: \text{ IF } x \text{ IS } \mathbf{X}_i \text{ THEN } y = A_i x + B_i \tag{2}$$

$\mathbf{X}_i$ denotes the i:th 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 i:th fuzzy region $\mathbf{X}_i$. A_i and B_i are fixed parameters of the local linear equation on the right hand side of Eqn. 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:

$$y = w_i(x)(A_i x + B_i) \tag{3}$$

A composition of all rules $R_1 \ldots R_c$ results in a summation over all outputs from Eqn. 3:

$$y = \sum_{i=1}^{c} w_i(x)(A_i x + B_i) \tag{4}$$

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 nonlinear characteristics of input-output data-sets (see (Gustafson & Kessel, 1979) and (Palm & Stutz, 2003)). For more details about fuzzy systems see (Palm et al., 1997).
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 become the outputs of the model.

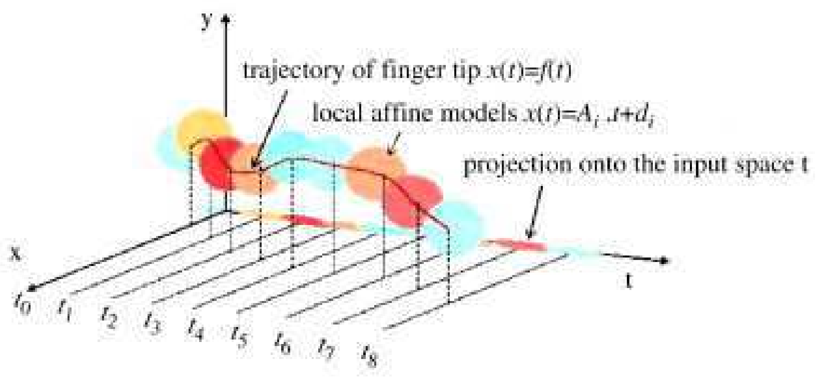

y
trajectory of finger tip $x(t)=f(t)$
local affine models $x(t)=A_i \cdot t+d_i$
projection onto the input space t
x
t
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

Fig. 1. Time-clustering principle.

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

$$\mathbf{x}(t) = \mathbf{f}(t) \tag{5}$$

where $\mathbf{x}(t) \in \mathbb{R}^3$, $\mathbf{f} \in \mathbb{R}^3$, and $t \in \mathbb{R}^+$.
Equation (5) is linearized at selected time points t_i with

$$\mathbf{x}(t) = \mathbf{x}(t_i) + \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \cdot (t - t_i) \tag{6}$$

resulting in a locally linear equation in t.

$$\mathbf{x}(t) = \mathbf{A}_i \cdot t + \mathbf{d}_i \tag{7}$$

where $\mathbf{A}_i = \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \in \mathbb{R}^3$ and $\mathbf{d}_i = \mathbf{x}(t_i) - \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \cdot t_i \in \mathbb{R}^3$. Using Eqn. 7 as a local linear model one can express Eqn. 5 in terms of an interpolation between several local linear models by applying Takagi-Sugeno fuzzy modeling (Takagi & Sugeno, 1985) (see Fig. 1)

$$\mathbf{x}(t) = \sum_{i=1}^{c} w_i(t) \cdot (\mathbf{A}_i \cdot t + \mathbf{d}_i) \tag{8}$$

$w_i(t) \in [0,1]$ 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

$$w_i(t) = \frac{1}{\sum_{j=1}^{c} \left(\frac{(t-t_i)^T M_{i\,pro}(t-t_i)}{(t-t_j)^T M_{j\,pro}(t-t_j)} \right)^{\frac{1}{\tilde{m}_{proj}-1}}} \tag{9}$$

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 (Gustafson & Kessel, 1979).

2.3 Model Selection using Q-learning

The actions of the robot have to be evaluated to enable the robot to improve its performance. Therefore, we use the state-action value concept from reinforcement learning to evaluate each action (skill) in a point of the state space (joint configuration). The objective is to assign a metric to each skill to determine its performance, and include this is a reinforcement learning framework.
In contrast to most other reinforcement learning applications, we only deal with one state-action transition, meaning that from a given position only one action is performed and then judged upon. A further distinction to classical reinforcement learning is to ensure that all actions initially are taken so that all existing state-action transitions are tested. Further improvement after the initial learning and adaption is possible by implementing a continuous learning scheme. Then, the robot will receive the reward after each action and continues to improve the performance over a longer time. However, this is beyond the scope of this chapter, and is a topic of future work.
The trajectory executed by the robot is evaluated based on three criteria:

- Deviation between the demonstrated and executed trajectories.
- Smoothness of the motion, less jerk is preferred.
- Successful or unsuccessful grasp.

In a Q-learning framework, the reward function can be formulated as:

$$r = -\frac{1}{t_f} \sum_{t=0}^{t=t_f} |H^r(t) - H(t)| - \frac{1}{t_f} \sum_{t=0}^{t=t_f} \dddot{x}^2(t) + r_g \tag{10}$$

where

$$r_g = \begin{cases} -100 & \text{if} \quad \text{Failure} \\ 0 & \text{if} \quad \text{Success, but overshoot} \\ +100 & \text{if} \quad \text{Success} \end{cases} \tag{11}$$

where $H^r(t)$ is the hand-state trajectory of the robot, $H(t)$ is the hand-state of the demonstration. The second term of the Eqn. 10 is proportional to the jerk of the motion, where t_f

is the duration of the motion, t_0 is the starting time and $\dddot{x}$ is the third derivative of the motion, i.e., jerk. The third term in Eqn. 10 r_g, is the reward from grasping as defined in Eqn. 11 where "failure" meaning a failed grasp and "success" means that the object was successfully grasped. In some cases the end-effector performs a successful grasp but with a slight overshoot, which is an undesired property since it might displace the target. An overshoot means that the target is sightly missed, but the end-effector then returns to the target.
When the robot has executed the trajectories and received the subsequent rewards and the accumulated rewards, it determines what models to employ, i.e., action selection. The actions that received the highest positive reward are re-modeled, but this time as robot trajectories using the hand-state trajectory of the robot as input to the learning algorithm. This will give less discrepancies between the modeled and the executed trajectory, thus resulting in a higher reward. In Q-learning a value is assigned for each state-action pair by the rule:

$$Q(s,a) = Q(s,a) + \alpha * r \tag{12}$$

where s is the state, in our case the joint angles of the manipulator, a is the action, i.e., each model from the demonstration, and α is a step size parameter. The reason for using the joint space as state space is the highly non-linear relationship between joint space and hand-state space (i.e., a Cartesian space): two neighboring points in joint space are neighboring in Cartesian space but not the other way around. This means that action selection is better done in joint space since the same action is more likely to be suitable for two neighboring points than in hand-state space.
To approximate the Q-function we used Locally Weighted Projection Regression (LWRP)[1] as suggested in (Vijayakumar et al., 2005), see their paper for details.

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 we achieve the mapping from human to robot hand and how to define the hand-state components. Then, section 3.3 describes the next-state-planner, which produces the actual robot trajectories.

3.1 Mapping between Human and Robot Hand States

The definition of H is *perception invariant* and must be able to update from any type of sensory information. The hand-sate components $h_1, \ldots h_p$ are such that they can be recovered both from human demonstration and from robot perception. Fig. 2 shows the definition of the hand-state in this article.
Let the human hand be at some initial state H_1. The hand then moves along the demonstrated path until the final state H_f is reached where the target object is grasped by the hand (Iliev et al., 2007). That is, the recorded motion trajectory can be seen as a sequence of states, i.e.,

$$H(t) : H_1(t_1) \rightarrow H_2(t_2) \rightarrow \ldots \rightarrow H_f(t_f) \tag{13}$$

Since the hand state representation is defined in relation to the target object, the robot must have access to the complete trajectory of hand of the demonstrator. Therefore the hand-state trajectory can only be computed *during* a motion if the target object is known in advance.

[1] Available at: http://www-clmc.usc.edu/software/lwpr

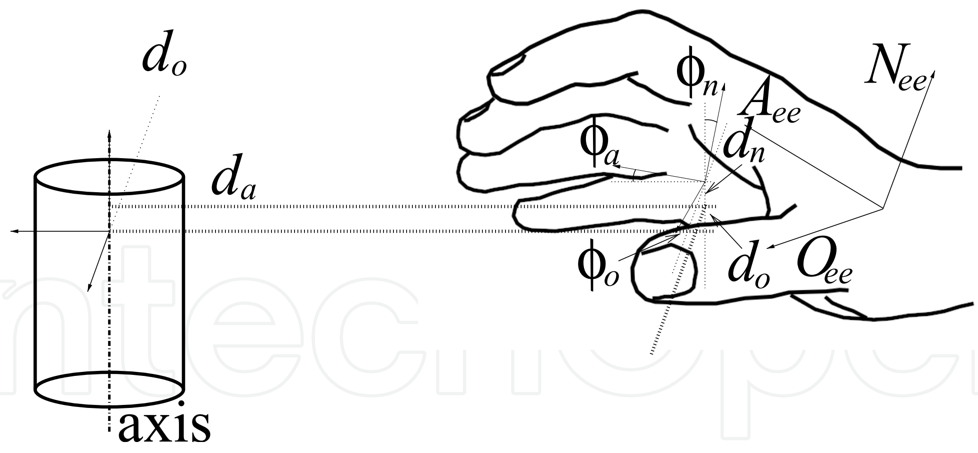


Fig. 2. The hand-state describes the relation between the hand pose and the object affordances. N_{ee} is the the normal vector, O_{ee} the side (orthogonal) vector and A_{ee} is the approach vector. The vector Q_{ee} is the position of the point. The same definition is also valid for boxes, but with the restriction that the hand-state frame is completely fixed, it cannot be rotated around the symmetry axis.

Let $H_{des}(t)$ be the desired hand-state trajectory recorded from a demonstration. In the general case the desired hand-state $H_{des}(t)$ cannot be executed by the robot without modification. Hence, a *robotic* version of $H_{des}(t)$ have to be constructed, denoted by $H^r(t)$, see Fig. 3 for an illustration.

One advantage of using only one demonstrated trajectory as the desired trajectory over trajectory averaging (e.g., (Calinon et al., 2007) or (Delson & West, 1996)) is that the average might contain two essentially different trajectories (Aleotti & Caselli, 2006). By capturing a human demonstration of the task, the synchronization between reach and grasp is also captured, demonstrated in (Skoglund et al., 2008). Other ways of capturing the human demonstration, such as kinesthetics (Calinon et al., 2007) or by a teach pendant (a joystick), cannot capture this synchronization easily.

To find $H^r(t)$ a mapping from the human grasp to the robot grasp a transformation is needed, denoted T^r_h. This transformation can be obtained as a static mapping between the pose of the demonstrator hand and the robot hand while they are holding the same object at a fixed position. Thus, the target state H^r_f 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^r_f, using the transformation T^r_h:

$$H^r_f = T^r_h H_f \tag{14}$$

The pose of the robot hand at the start of a motion defines the initial state H^r_1. Since H^r_f represents the robot hand holding the object , it has to correspond to a stable grasp. For a known object, suitable H^r_f can either be obtained by simulation (Tegin et al., 2009), grasp planning or by learning from experimental data. Thus, having a human hand state H_f and their corresponding robot hand state H^r_f, T^r_h is obtained as:

$$T^r_h = H^r_f H_f^{-1} \tag{15}$$

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.

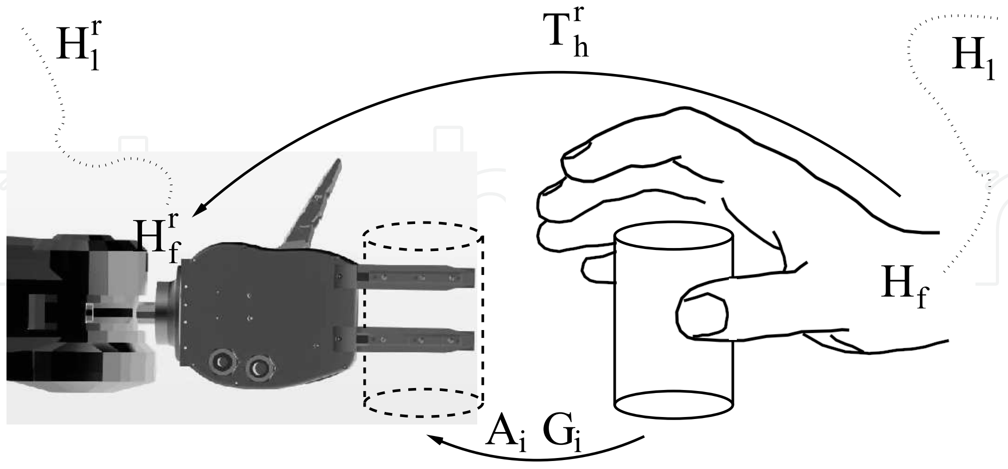


Fig. 3. Mapping from human hand to robotic gripper.

3.2 Definition of Hand-States for Specific Robot Hands

hen the initial state H_1^r and the target state H_f^r are defined, we have to generate a trajectory between the two states. In principle, it is possible to use Eqn. 14 to $H_{des}(t)$ such that it has its final state in H_f^r. The robot then starts at H_1^r and approaches the displaced demonstrated trajectory and then track this desired trajectory 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.
The end-effector trajectory is reconstructed from the recorded demonstration, and is represented by a time dependent homogeneous matrix $T_{ee}(t)$. Each element is represented by the matrix

$$T_{ee} = \begin{pmatrix} N_{ee} & O_{ee} & A_{ee} & Q_{ee} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

where N_{ee}, O_{ee} and A_{ee} are the normal vector, the side vector, and the approach vector, respectively of the end effector. The position is represented by the vector Q_{ee}. It is important to note that the matrix T_{ee} is defined differently for different end-effectors, for example, the human hand is defined as in Fig. 2.
From the demonstrated trajectory, a handstate trajectory can be obtained as afunction of time. We formulate the hand-state as:

$$H(t) = [d_n(t)\ d_o(t)\ d_a(t)\ \phi_n(t)\ \phi_o(t)\ \phi_a(t)] \tag{17}$$

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. 2.

Note that by omitting the finger specific components of the hand-state we get a simplified definition of H, but cannot determine the type of human grasp. In (Skoglund et al., 2008) we give an account of how the finger specific components and object relation components can be used to synchronize reaching with grasping. Another reason for omitting finger specific components is that grasp classification is out of scope of this chapter; only power grasps are used the subsequent experiments. Thus, the grasp type is assumed to be known $G = \{\textit{cylindrical, spherical, plane}\}$; the affordances are: position, size, and cylinder axis $A = \{\textit{width, axis}\}$ or box $A = \{\textit{width, length, N-axis, O-axis, A-axis}\}$. See (Palm & Iliev, 2007) for grasp taxonomy.

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 (Skoglund et al., 2008). The NSP we use is inspired by the Vector Integration To Endpoint (VITE) planner propsed by Bullock & Grossberg (1989) as a model 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 NSP generates a hand-state trajectory using the TS fuzzy-model of a demonstration. Since the resulting hand-state trajectory $H^r(t)$ can easily be converted into Cartesian space, the inverse kinematics provided by the arm controller can be used directly.
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^r_{des} is computed from the TS fuzzy time-model (Eqn. 8). The desired hand-state H_{des} is fed to the NSP. Instead of using only one goal attractor as in VITE (Bullock & Grossberg, 1989), and additional attractor–the desired hand-state trajectory–is used at each state. The system has the following dynamics:

$$\ddot{H} = \alpha(-\dot{H} + \beta(H_g - H) + \gamma(H_{des} - H)) \tag{18}$$

where H_g is the hand-state goal, H_{des} 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 (not to be confused with the step size parameter α in Eqn. 12) and β, γ are positive weights for the goal and tracking point, respectively.
If the last term $\gamma(H_{des} - H)$ in Eqn. 18 is omitted, i.e., $\gamma = 0$, then the dynamics is *exactly* as the VITE planner Bullock & Grossberg (1989). 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. To determine β and γ, which controls the behavior of the NSP, we use a time dependent weighting mechanism. The weighting is a function of time left t_l to reach the goal at t_f; $\gamma = K(t_l/t_f)^2$, where K is 2; $\beta = 1 - \gamma$. Since the environment demonstration provides better accuracy than the task demonstration, it is reasonable to give the target a hight weight at the end of the trajectory (i.e., $\beta = 1$), Spherical linear interpolation is used. To interpolate between initial and final orientation along the trajectory spherical linear interpolation is used (Shoemake, 1985).
It is also possible to determine β and γ by the variance across multiple demonstrations (Skoglund, Tegin, Iliev & Palm, 2009).
Analytically, the poles in Eqn. 18 are:

$$p_1, p_2 = -\frac{\alpha}{2} \pm \sqrt{\frac{\alpha^2}{4} - \alpha\gamma}. \tag{19}$$

The real part of p_1 and p_2 will be ≤ 0, which will result in a stable system (Levine, 1996). Moreover, $\alpha \not\lesssim 4\gamma$ and $\alpha \geq 0$, $\gamma \geq 0$ will contribute to a critically damped system, which is fast and has small overshoot. Fig. 4 shows how different values γ affect the dynamics of the planner.

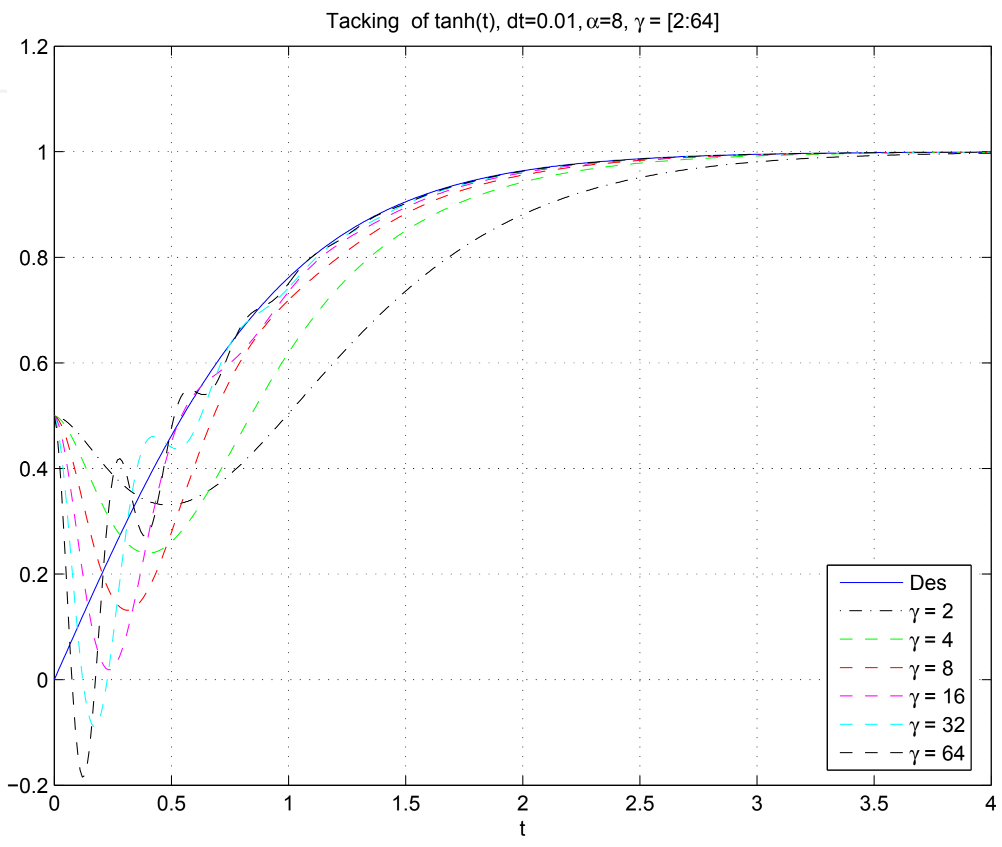


Fig. 4. The dynamics of the planner for six different values of γ. The tracking point is $tanh(t)$, with $dt = 0.01$ and α is fixed at 8. A low value on $\gamma = 2$ produces slow dynamics (black dot-dashed line), while a high value $\gamma = 64$ is fast but overshoots the tracking point (black dashed line).

The controller has a feedforward structure as in Fig. 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.

3.4 Demonstrations of Pick-and-Place Tasks

In our setup a demonstration is done in two stages: environment- and task demonstration. During the first stage, the *environment demonstration*, the target objects in the workspace are

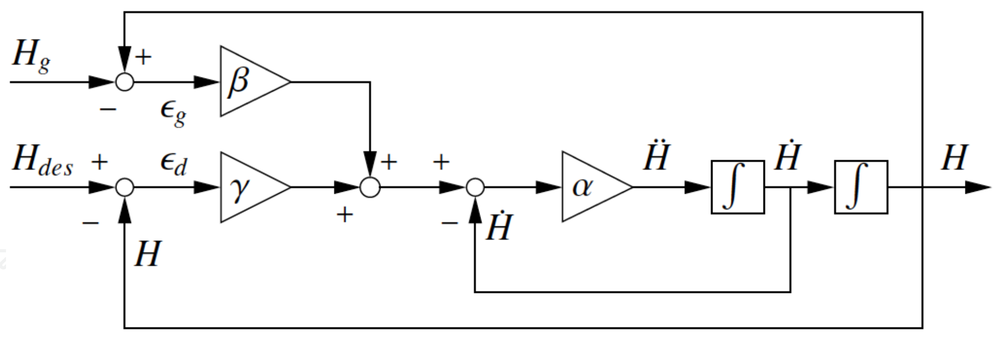


Fig. 5. Hand-state planner architecture. H_g is the desired hand-state goal, H_{des} is the desired hand-state at the current distance to target.

shown to the robot. The environment demonstration can provide a more accurate workspace model than the task demonstration, since this is a separated stage and the demonstrator can focus on the objects and don't consider the dynamics of the task. In the experiment in section 5.2, we use a simplistic modeling of the environment where all objects are modeled as box shaped objects. The result from an environment demonstration is shown in Fig. 6, where the bounding boxes were created from information on hand/fingerpose and tactile data. A bounding box represents each object with position of the center and length, width and height, which are used to compute the orientation of the object. A more sophisticated modeling–based on point clouds–could be used if a better granularity of the workspace is needed (Charusta et al., 2009).
In the *task demonstration*, i.e., a pick-and-place of an object, only the task is shown. Once the workspace model is available, only the demonstrated trajectory is used. If an environment demonstration is unavailable the target object can be determined from task demonstration, where the center point of the grasp can be estimated from the grasp type. However, this not as accurate as using data form an environment demonstration. The task demonstration contains the trajectories which the robot should execute to perform the task. Trajectories recorded from a task demonstration are shown in Fig. 7.

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. The Impulse system consists of four cameras (in experiment 1), and five (in Experiment 2) 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 have 12 Mega pixel resolution resulting in sub-millimeter precision. One of the cameras can be seen in the right picture of Fig. 8. The operator wears a glove with LEDs attached to it, left picture see Fig. 8. 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 of the time visible to 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. Moreover, we have

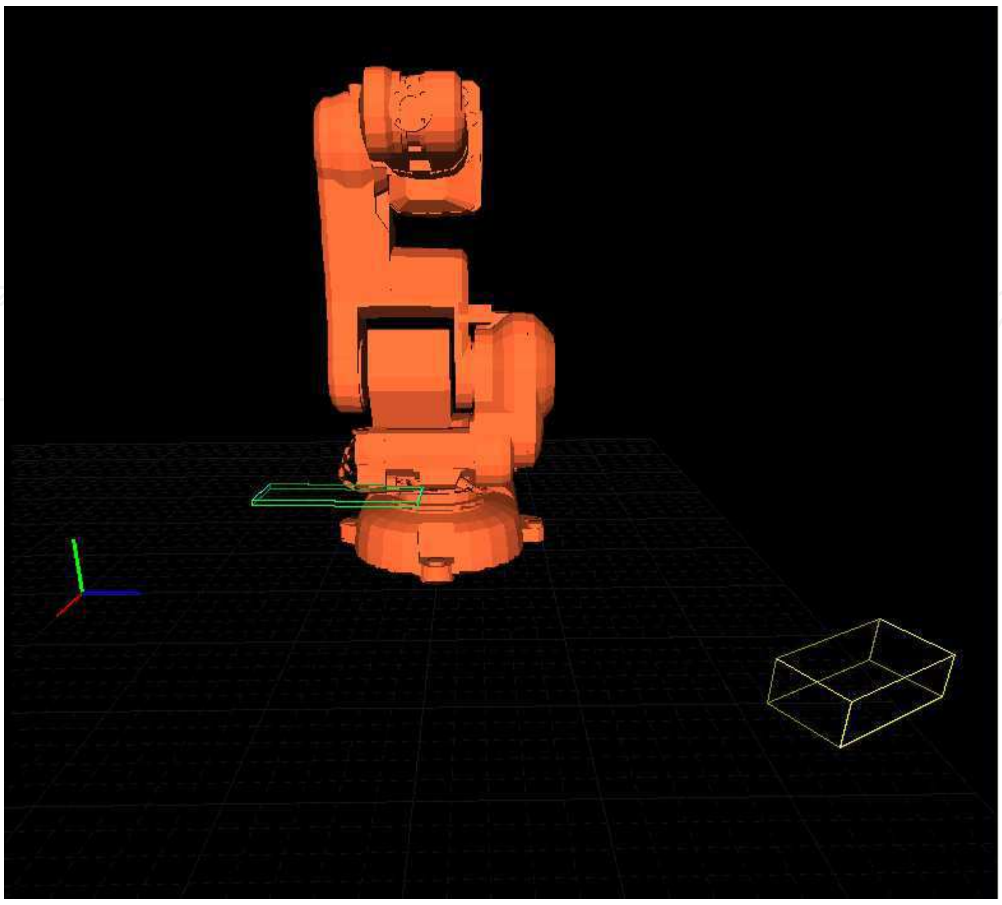


Fig. 6. The result of an environment demonstration.

mounted tactile sensors (force sensing resistors) to detect contact with objects. The sensors are mounted on the fingertips of the glove, shown in the middle of Fig. 8. We define a grasp as when contact is detected at the thumb sensor and one additional finger. This means that only grasps which include the thumb and one other finger can be detected. No grasp recognition is necessary since the gripper only allows one grasp type. When a grasp is detected the distance to each object in the workspace is measured and the nearest object, if below some distance threshold, is identified as 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. 9), which can perform power grasps (i.e., cylindrical and spherical grasps) using a hybrid position/force controller. For details on the KTHand, see (Tegin et al., 2008).
The demonstrations were performed with the teacher standing in front of the robot. First, the environment is demonstrated by tactile exploration of the workspace. The demonstrator touches the objects of interest; with special care so the boundaries of each object are correctly captured. Second, the task is demonstrated where the teacher starts with the hand in a position similar the robot's home position, i.e., $\Theta = [0\ 0\ 0\ 0\ 0\ 0]^T$. The results from the environment and task demonstrations are shown in Fig. 6 and 7 respectively, with the base frame of the

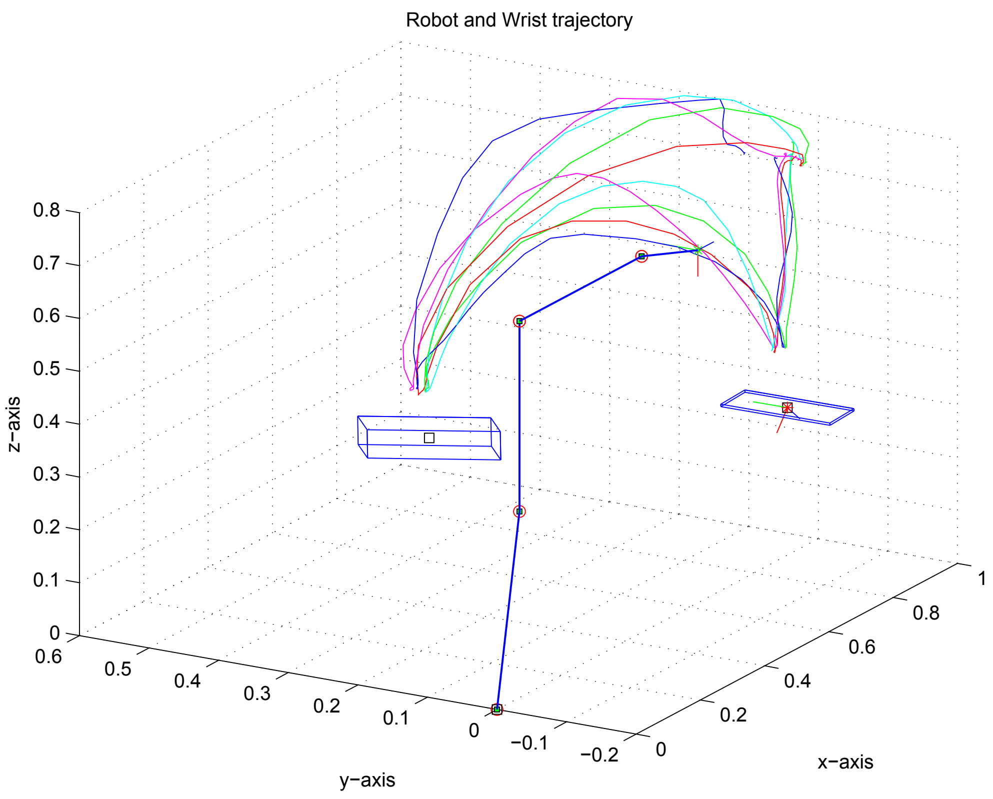


Fig. 7. Five sample demonstrated trajectories (the wrist is plotted), the robot and the two objets of interest. The object to the right is marked with a red star to indicate that this object was the nearest when a grasp was detected in the task demonstration.

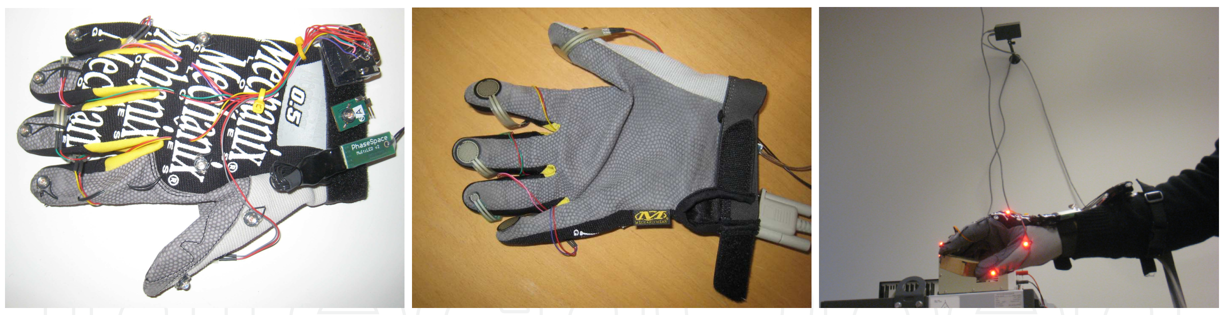


Fig. 8. **Left:** The glove used in the Impulse motion capturing system from PhaseSpace. The glove from the top showing the LEDs. **Middle:** The glove with the tactile sensors mounted on each finger tip. **Right:** The system in use showing one of the cameras and the LED on the glove.

robot as the reference frame. The object is determined from the environment demonstration, since it is more exact compared to the task demonstration.

5. Experimental Evaluation

In this section we provide an experimental evaluation of the presented method.

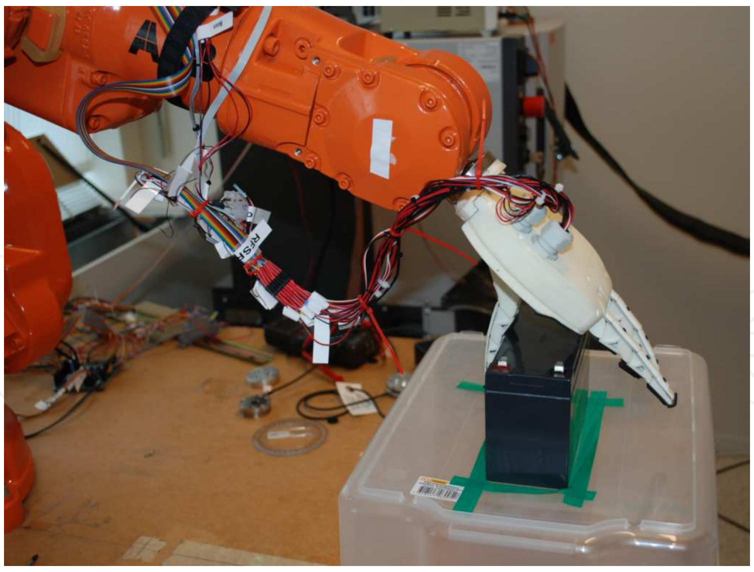


Fig. 9. The anthropomorphic gripper KTHand used in the second experiment.

5.1 Experiment 1 – A Complete Pick-and-Place Task

To test the approach on an integrated system the KTHand is mounted on the ABB manipulator and a pick-and-place task is executed, guided by a demonstration showing pick-and-place task of a box (110 $\times$ 56 $\times$ 72 mm). The synchronization between reach and grasp is performed by a simple finite state machine. After the grasp is executed, the motion to the placing point is performed by following the demonstrated trajectory (see section 2.2). Since the robot grasp pose corresponds approximately to the human grasp pose it is possible for the planner to reproduce the human trajectory almost exactly. This does not mean that the robot actually can execute the trajectory, due to workspace constraints. The retraction phase follows the same strategy as the reaching motion, but in reverse. Fig. 10 shows the complete task learned from demonstration.

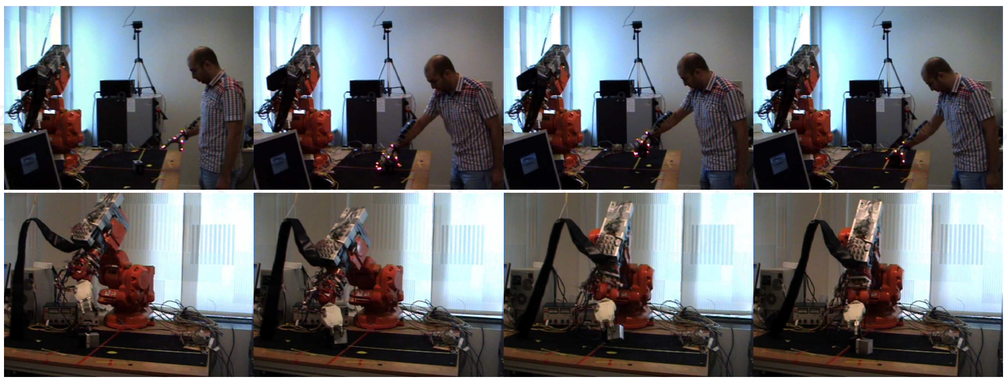


Fig. 10. Industrial manipulator programmed using a demonstration. A movie of the sequence is available at: http://www.aass.oru.se/Research/Learning/arsd.html.

5.2 Experiment 2 – Skill Improvement

In this experiment the robot uses the self executed trajectories for skill modeling and compare the performance to the skills models from human demonstration. The aim is to enable to robot to improve its performance. In this experiment 20 task demonstrations of the same pick-and-place task were performed, where only the reaching part of the task was used to train a reaching skill. Five of these demonstrations are shown in Fig. 7. In addition, one environment demonstration was also recorded, shown in Fig. 6. All demonstrations were modeled using fuzzy time-modeling, where the position of the index finger and orientation of the wrist were used to record the trajectory. Three of the modeled trajectories are shown in figure 11 in the top graphs as dashed lines. The reason for using the index finger instead of–what would be more appropriate–the center point between the index finger and the tip of the thumb is that the LED on the thumb was often hidden resulting from occlusions during the motion. Thus, the number of occlusions is minimized.

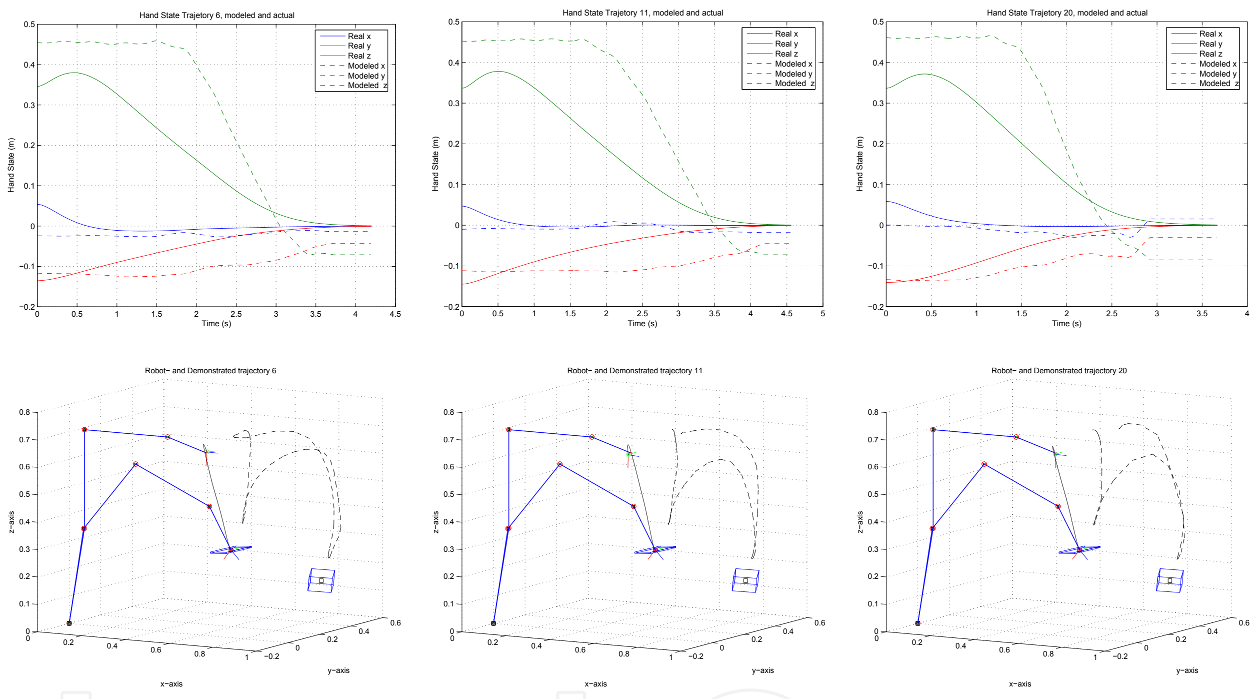


Fig. 11. Demonstrations number H_6, H_{11} and H_{20}, of which the two former were the two best performing models, and the latter was the worst performing model in generating the reaching trajectory for the robot to execute. **Top:** The dashed lines in the top graphs are the modeled trajectories from the demonstration H_{des}. The solid lines are the trajectory executed by the robot, H^r. **Bottom:** Both as H_{des} (dashed) as the human demonstrated in, and H^r (solid) as the robot executed it, Cartesian space.

5.2.1 Evaluation of Hand-State Trajectories

All the recorded demonstrations were modeled and executed by the robot in the same manner as in Exp. 1. In addition the performance of each model was evaluated, the criteria in Eqn. 10. Three of the executed trajectories are shown in Fig. 11 in the top graph as solid lines and in the bottom graph as Cartesian trajectories. At the end of the demonstrations (around 3.5 to 4.0 sec) in Fig. 11 the difference between the model and the actual trajectory is up to 8 cm (in

y-direction). Since the model is created from the trajectory of the fingertip of the index finger, the end of the trajectory will be displaced from the center of the target object. This is due to the fact that the index finger will be at the side of the box shaped object during the grasp. Recall from the definition of the hand-state space (section 2.1) that the target object is the frame in which all motions are in relation to. This means that the origin of the hand-state frame is the top center of the target object. The point attractor of Eqn. 18 for the trajectory generation is switching from the task demonstration to the target object, as described below. Therefore, the point $[0, 0, 0]$ becomes the point attractor at the end of the motion.
The hand-state error and minimum jerk can be evaluated already in the generation state in simulation, i.e., before the actual execution on the real robot. The grasp success is evaluated from real execution. In the real execution the robot starts in the home position $\Theta = [0\ 0\ 0\ 0\ 0\ 0]^T$ deg, with a small perturbation added to the last three joints since the home position is a singular configuration. All models succeeded in generating a trajectory which positioned the end-effector in such a way that a successful grasp can be performed, resulting in a positive reward of $+100$. This reward is then decreased by adding a negative reward from hand-state error and jerk, as described by equations 10-11.

Human Actions	H_1	H_2	H_3	H_4	H_5	H_6	H_7
Q-values	0.7572	0.7854	0.8120	0.8596	0.6864	0.9521	0.7410
Human Actions	H_8	H_9	H_{10}	H_{11}	H_{12}	H_{13}	H_{14}
Q-values	0.8966	0.7738	0.8659	0.9964	0.8555	0.8349	0.8334
Human Actions	H_{15}	H_{16}	H_{17}	H_{18}	H_{19}	H_{20}	
Q-values	0.8319	0.8481	0.7656	0.7572	-1.9459	0.6745	
Robot Actions	R_1	R_2					
Q-values	1.0769	1.0437					

Table 1. The Q-values at the home configuration, $\Theta = [0\ 0\ 0\ 0\ 0\ 0]^T$ deg. When the robot imitated human actions H_6 and H_{11} (light gray) from its home configuration, it received the highest Q-value and used these two to create the two new action models R_1 and R_2. To compare the two original motions H_6 and H_{11} were selected since they performed best and H_{20} which had the lowest positive Q-value.

In Table 1 this is shown in the rows with joint configuration $\Theta = [0\,0\,0\,0\,0\,0]^T$ deg. The models from human demonstration H_6 and H_{11} performed best when the robot executed them. Hence, these values were selected to be remodeled into robot skills: R_1 and R_2, meaning that the robot trajectory is used to create the fuzzy models. Worst performance was obtained in H_{19} and H_{20}, where H_{19} received a negative Q-value. Finally, R_1 and R_2 were tested from the home configuration and evaluated using the same criteria, shown in Fig. 12. As expected, their performance was better than all others, since the hand-state error is reduced.

5.2.2 Generalization of Robot Skills

Generalization over the workspace is used as a test on how well the skills modeled from robot execution perform in comparison to the skills modeled from human demonstration. This is done by positioning the manipulator in twelve starting configurations, different from the home configuration $\Theta = [0\ 0\ 0\ 0\ 0\ 0]^T$, while keeping the target object at the same position as in the demonstration. The models H_6 and H_{11}, acquired from human demonstrations, received the highest Q-values (see Table 1) and are therefor selected for comparison. In addition we also selected one of the worst scoring models of the remaining models, namely H_{20}. A set of twelve joint configurations were chosen as the initial starting positions. The configurations were chosen to approximately correspond to 1/4 of the full joint range at each joint, except last joint where 1/8 of the range was used. Two exceptions from this were the configurations $\Theta = [0\ 20\ 40\ 1\ 1\ 1]^T$ and $\Theta = [0\ 0\ 25\ 1\ 1\ 1]^T$ due to workspace restrictions. The resulting Q-values after executing a reaching motion form these joint configurations can be seen in Table 2. In the table the highest Q-values are highlighted. From two of the starting configurations none of the models could execute a successful reaching motion, namely $\Theta = [0\ -45\ 0\ 1\ 1\ 1]^T$ deg and $\Theta = [0\ 0\ 0\ 1\ -55\ 1]^T$ deg (dark rows in Table 2). This means that new demonstrations should be done from these positions. From the other joint configurations neither H_6 nor H_{11} performed best. Surprisingly, the worst model H_{20} was the only one to perform a successful reaching motion from configuration $\Theta = [0\ 0\ -50\ 1\ 1\ 1]^T$ deg, resulting in the highest Q-value. From the rest of the configurations robot skill R_1 or R_2 performed best, thus confirming our hypothesis that skills modeled from own experience are better adapted to the robot than skills directly modeled from observation. Fig. 13 shows four sample configurations where the robot skill R_1 and R_2 are used to generate reaching actions from configurations different from the trained position. In Fig. 14 two sequences are shown where the real robot executes the reaching skills from the trained position, and from the tested position.

	Q-values				
	Human Actions			Robot Actions	
Joint configuration, **q**	H_6	H_{11}	H_{20}	R_1	R_2
$[0\ 0\ 0\ 0\ 0\ 0]^T$	0.9521	0.9964	0.6745	1.0769	1.0437
$[-45\ 0\ 0\ 1\ 1\ 1]^T$	1.6079	1.7029	1.1207	−0.0670	1.7847
$[0\ -45\ 0\ 1\ 1\ 1]^T$	−0.3890	−0.2960	−0.8784	−0.0670	−0.2143
$[0\ 0\ -50\ 1\ 1\ 1]^T$	−0.3895	−0.2968	1.1208	−0.0676	−0.2150
$[0\ 0\ 0\ -50\ 1\ 1]^T$	1.6104	1.7032	1.1208	−0.0675	1.7850
$[0\ 0\ 0\ 1\ -55\ 1]^T$	−0.3895	−0.2968	−0.8792	−0.0676	−0.2150
$[0\ 0\ 0\ 1\ 1\ -50]^T$	1.6102	1.7031	1.1208	−0.0675	1.7850
$[45\ 0\ 0\ 1\ 1\ 1]^T$	1.6102	1.7030	1.1206	1.9320	1.7847
$[0\ 20\ 40\ 1\ 1\ 1]^T$	1.6104	1.7032	1.1207	1.9324	1.7850
$[0\ 0\ 25\ 1\ 1\ 1]^T$	1.5848	1.6515	1.0994	1.9017	1.7275
$[0\ 0\ 0\ 50\ 1\ 1]^T$	1.6105	1.7031	1.1207	1.9324	1.7850
$[0\ 0\ 0\ 1\ 55\ 1]^T$	1.6105	1.7032	1.1208	1.9324	1.7850
$[0\ 0\ 0\ 1\ 1\ 50]^T$	1.6105	1.7032	1.1208	1.9324	1.7850

Table 2. The Q-values for selected joint configurations. Twelve other configurations than the home configuration were selected to test how well the robot skills R_1 and R_2 performed. As comparison, the two original motions H_6 and H_{11} were selected since they performed best. In addition H_{20} was selected, because it had the lowest positive Q-value.

6. Conclusions and Future Work

In this article, we present a method for programming-by-demonstration of reaching motions which enables robotic grasping. To allow the robot to interpret both the human motions and

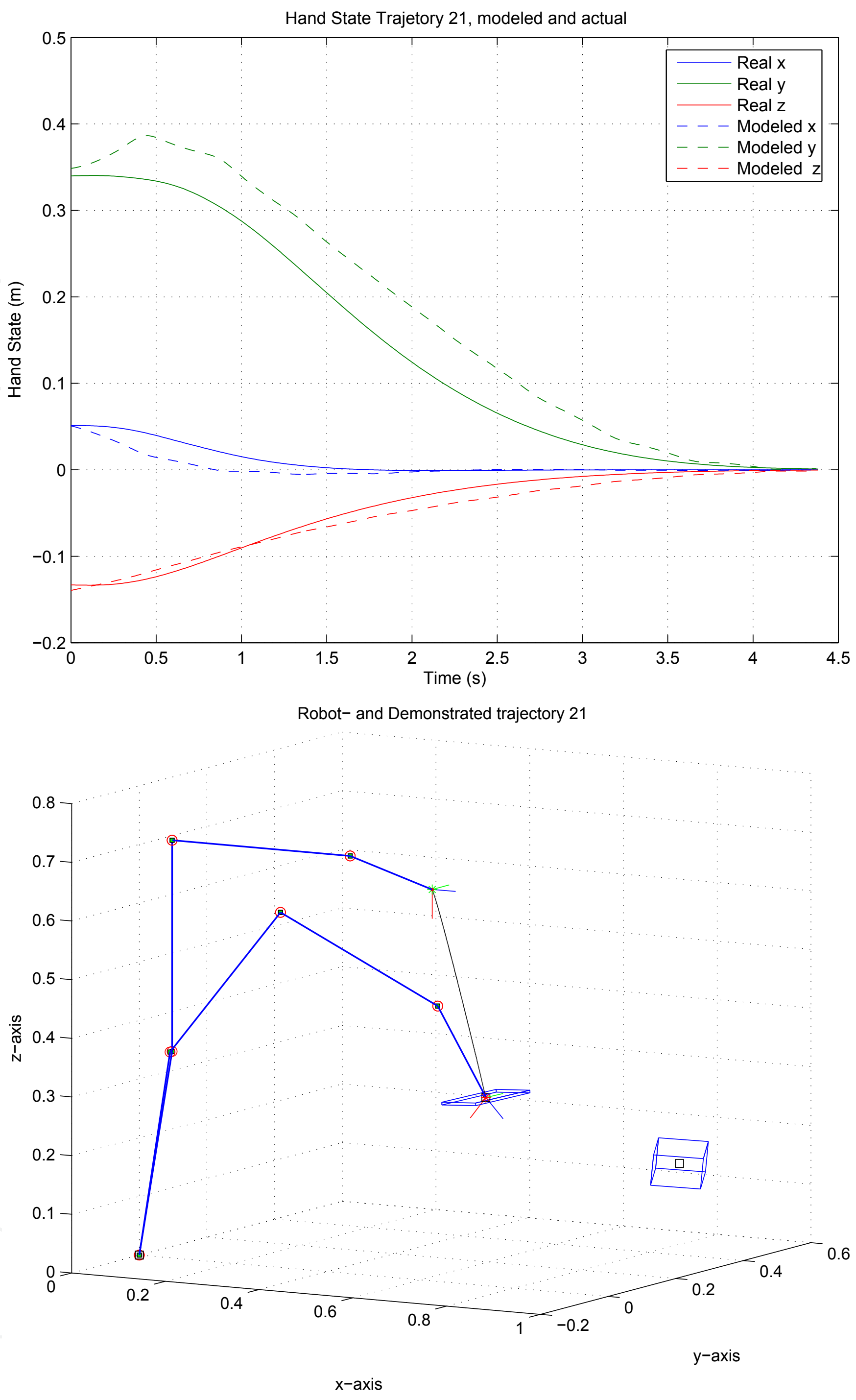


Fig. 12. Robot model R_1 and the generated reaching trajectory the robot executes.

its own in similar ways, we employ a hand-state space representation as a common basis between the human and the robot.
We presented the design of a NSP, which includes the advantages of fuzzy modeling and executes the motion in hand-state space.

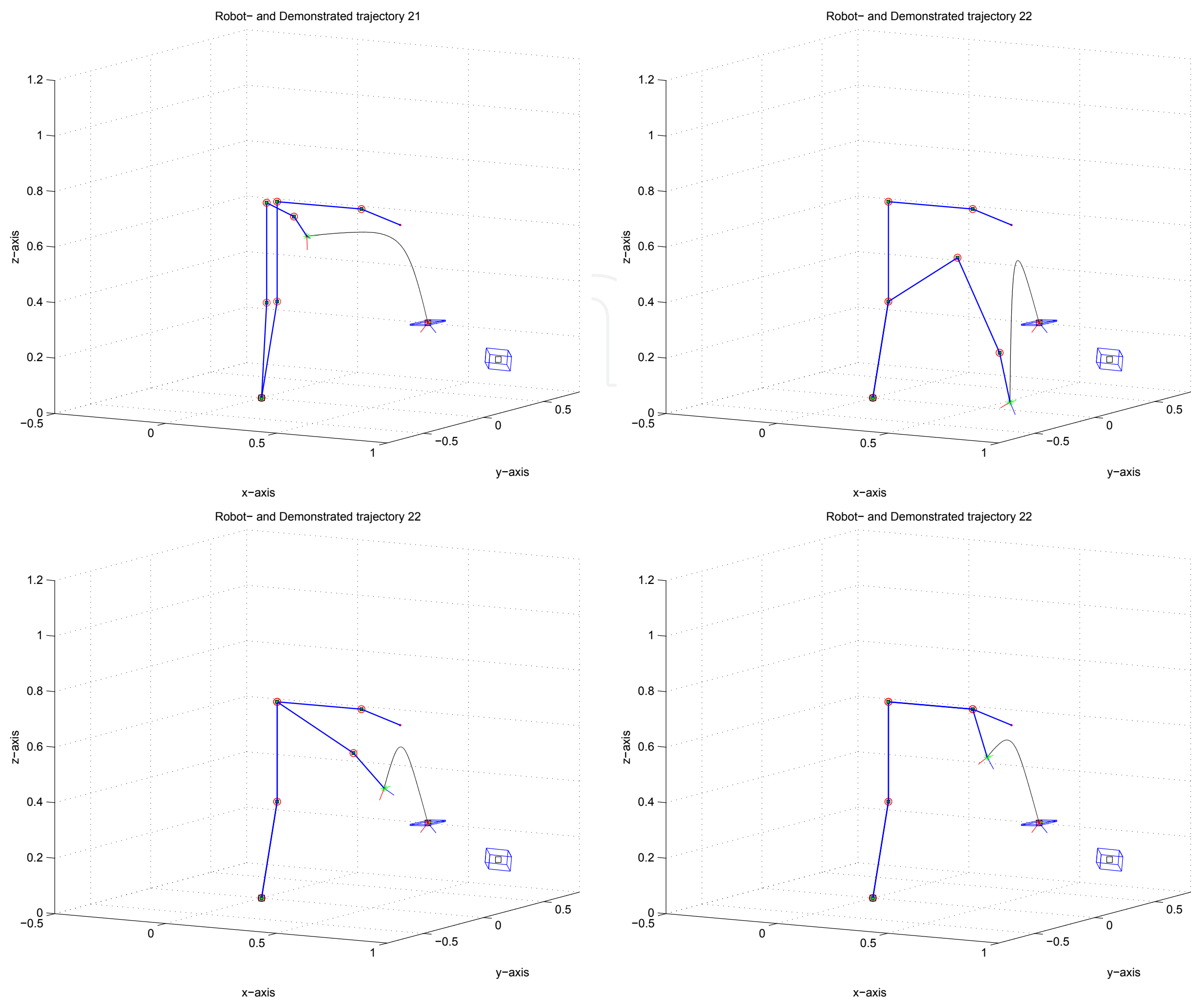
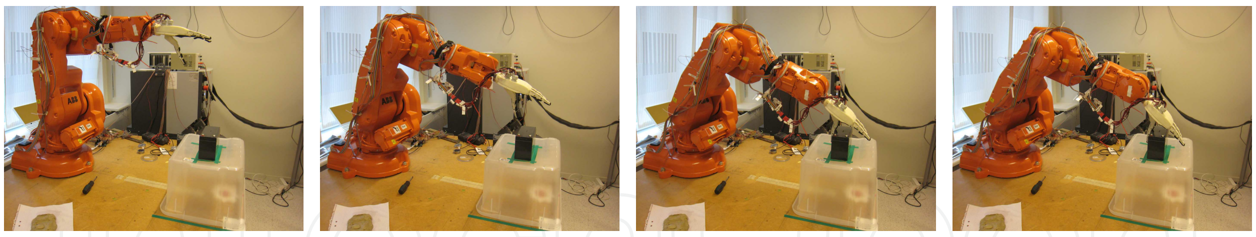


Fig. 13. Reaching from different initial positions. Four sample configurations, $\Theta = [-45\ 0\ 0\ 1\ 1\ 1]^T$, $\Theta = [0\ 20\ 40\ 1\ 1\ 1]^T$, $\Theta = [0\ 0\ 25\ 1\ 1\ 1]^T$ and $\Theta = [0\ 0\ 0\ 1\ 55\ 1]^T$.

Human demonstrations have been shown to provide sufficient knowledge to produce skill models good enough for the robot to use as its own skills.
It is shown that the suggested method can generate executable robot trajectories based on current and past human demonstrations despite morphological differences. The robot gains experience from human demonstration, and we have shown how the robot can improve when the selfexecuted motions are performed. The generalization abilities of the trajectory planner are illustrated by several experiments where an industrial robot arm executes various reaching motions and performs power grasping with a three-fingered hand.
To validate the performance of the skills learned from demonstration we included a metric which can be used in reinforcement learning. The performances of the initially learned skills were compared to the skills learned from self execution (robot skills). The result showed that robot skills indeed performed better than skills from human demonstration. In reinforcement learning the performance metric is used to formulate the reward function.
In our future work we plan to extend the theoretical and experimental work to include all feasible grasp types of the KTHand. To remedy the effect of the small workspace of the robot

R_2 from configuration $\Theta = [0\ 0\ 0\ 0\ 0\ 0]^T$



R_2 from configuration $\Theta = [0\ 20\ 40\ 1\ 1\ 1]^T$

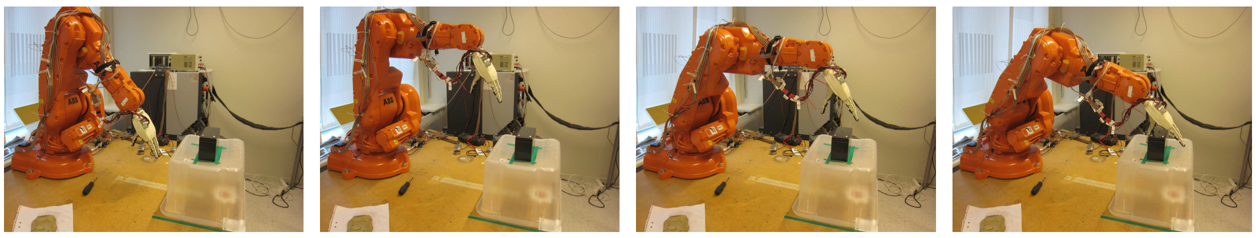


Fig. 14. The top series show the robot grasping the box shaped object from configuration $\Theta = [0\ 0\ 0\ 0\ 0\ 0]^T$ using robot skill R_2 . The bottom graph show the same skill R_2 but from a different initial position than the trained one: $\Theta = [0\ 20\ 40\ 1\ 1\ 1]^T$.

a different workspace configuration will be used. Furthermore, the robot's own perception will be incorporated into the loop to enable the robot to learn from its own experience.
Acknowledgment: Johan Tegin at Mechatronics Laboratory, at the Royal Institute of Technology, Stockholm, should be acknowledged for providing access to the KTHand. Krzysztof Charusta and Per Sporrong at the Center for Applied Autonomous Sensor Systems (AASS) at Örebro University, should be acknowledged for their help with the experiments.

7. References

Aleotti, J. & Caselli, S. (2006). Robust trajectory learning and approximation for robot programming, *Robotics and Autonomous Systems* **54**(5): 409–413.

Argall, B. D., Chernova, S., Veloso, M. & Browning, B. (2009). A survey of robot learning from demonstration, *Robotics and Autonomous Systems* **57**(5): 469–483.

Bandera, J. P., Marfil, R., Molina-Tanco, L., Rodríguez, J. A., Bandera, A. & Sandoval, F. (2007). *Humanoid Robots: Human-like Machines*, Itech, Vienna, Austria, chapter Robot Learning by Active Imitation, pp. 519–535.

Billard, A., Calinon, S., Dillmann, R. & Schaal, S. (2008). *Springer handbook of Robotics*, Springer, chapter Robot Programming by Demonstration, pp. 1371–1394.

Billard, A., Epars, Y., Calinon, S., Schaal, S. & Cheng, G. (2004). Discovering optimal imitation strategies, *Robotics and Autonomous Systems* **47**(2–3): 69–77.

Bullock, D. & Grossberg, S. (1989). VITE and FLETE: Neural modules for trajectory formation and postural control, *in* W. A. Hershberger (ed.), *Volitonal Action*, Elsevier Science Publishing B.V. (North-Holland), pp. 253–297.

Calinon, S., Guenter, F. & Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot, *IEEE Transactions on Systems, Man and Cybernetics, Part B* **37**(2): 286–298.

Charusta, K., Dimitrov, D., Lilienthal, A. J. & Iliev, B. (2009). Grasp-related features extraction by human dual-hand object exploration, *Proceedings of the 14:th International Conference on Advanced Robotics*.

Delson, N. & West, H. (1996). Robot programming by human demonstration: Adaptation and inconsistency in constrained motion, *IEEE International Conference on Robotics and Automation*, pp. 30–36.

Gustafson, D. & Kessel, W. (1979). Fuzzy clustering with a fuzzy covariance matrix., *Proceedings of the 1979 IEEE CDC* pp. 761–766.

Hersch, M. & Billard, A. G. (2008). Reaching with multi-referential dynamical systems, *Autonomous Robots* **25**(1–2): 71–83.

Ijspeert, A. J., Nakanishi, J. & Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots, *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 1398–1403.

Ijspeert, A., Nakanishi, J. & Schaal, S. (2001). Trajectory formation for imitation with nonlinear dynamical systems, *IEEE International Conference on Intelligent Robots and Systems (IROS 2001)*, Vol. 2, pp. 752–757.

Iliev, B., Kadmiry, B. & Palm, R. (2007). Interpretation of human demonstrations using mirror neuron system principles, *Proceedings of the 6th IEEE International Conference on Development and Learning*, Imperial College London, pp. 128–133.

Iossifidis, I. & Schöner, G. (2006). Dynamical systems approach for the autonomous avoidance of obstacles and joint-limits for an redundant robot arm, *Proceedings of 2006 IEEE International Conference on Robotics and Automation*, pp. 580–585.

Levine, W. S. (1996). The root locus plot, *in* W. S. Levine (ed.), *The Control Handbook*, CRC Press, chapter 10.4, pp. 192–198.

Nehaniv, C. L. & Dautenhahn, K. (2002). The correspondence problem, *in* K. Dautenhahn & C. Nehaniv (eds), *Imitation in Animals and Artifacts*, The MIT Press, Cambridge, MA, pp. 41–61.

Oztop, E. & Arbib, M. A. (2002). Schema design and implementation of the grasp-related mirror neurons, *Biological Cybernetics* **87**(2): 116–140.

Palm, R., Driankov, D. & Hellendoorn, H. (1997). *Model Based Fuzzy Control*, Springer.

Palm, R. & Iliev, B. (2006). Learning of grasp behaviors for an artificial hand by time clustering and Takagi-Sugeno modeling, *Proceedings of the IEEE International Conference on Fuzzy Systems*, Vancouver, BC, Canada, pp. 291–298.

Palm, R. & Iliev, B. (2007). Segmentation and recognition of human grasps for programming-by-demonstration using time-clustering and fuzzy modeling, *Proceedings of the IEEE International Conference on Fuzzy Systems*, London, UK.

Palm, R., Iliev, B. & Kadmiry, B. (2009). Recognition of human grasps by time-clustering and fuzzy modeling, *Robotics and Autonomous Systems* **57**(5): 484–495.

Palm, R. & Stutz, C. (2003). Generation of control sequences for a fuzzy gain scheduler, *International Journal of Fuzzy Systems* **5**(1): 1–10.

Pardowitz, M., Knoop, S., Dillmann, R. & Zöllner, R. D. (2007). Incremental learning of tasks from user demonstrations, past experiences, and vocal comments, *IEEE Transactions on Systems, Man and Cybernetics, Part B* **37**(2): 322–332.

Shadmehr, R. & Wise, S. P. (2005). *Computational Neurobiology of Reaching and Pointing - A Foundation for Motor Learning*, Computational Neuroscience, The MIT Press.

Shoemake, K. (1985). Animating rotation with quaternion curves, *ACM SIGGRAPH Computer Graphics* **19**(3): 245–254.

Skoglund, A., Iliev, B., Kadmiry, B. & Palm, R. (2007). Programming by demonstration of pick-and-place tasks for industrial manipulators using task primitives, *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Jacksonville, Florida, pp. 368–373.

Skoglund, A., Iliev, B. & Palm, R. (2008). A hand state approach to imitation with a next-state-planner for industrial manipulators, *Proceedings of the 2008 International Conference on Cognitive Systems*, University of Karlsruhe, Karlsruhe, Germany, pp. 130–137.

Skoglund, A., Iliev, B. & Palm, R. (2009). Programming-by-demonstration of reaching motions – a next-state-planner approach, *Robotics and Autonomous Systems* **In press.**

Skoglund, A., Tegin, J., Iliev, B. & Palm, R. (2009). Programming-by-demonstration of reach to grasp tasks in hand-state space, *Proceedings of the 14:th International Conference on Advanced Robotics*, Munich, Germany.

Takagi, T. & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control., *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-15**(1): 116–132.

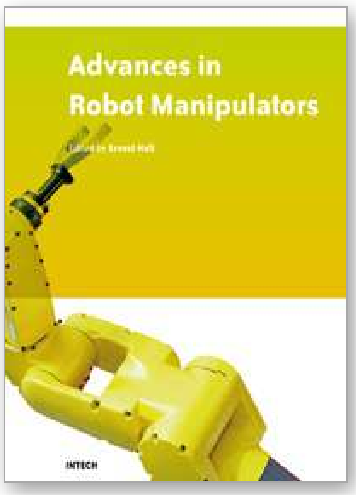
Takamatsu, J., Ogawara, K., Kimura, H. & Ikeuchi, K. (2007). Recognizing assembly tasks through human demonstration, *International Journal of Robotics Research* **26**(7): 641–659.

Tegin, J., Ekvall, S., Kragic, D., Wikander, J. & Iliev, B. (2009). Demonstration based learning and control for automatic grasping, *Journal of Intelligent Service Robotics* **2**(1): 23–30.

Tegin, J., Wikander, J. & Iliev, B. (2008). A sub €1000 robot hand for grasping – design, simulation and evaluation, *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, Coimbra, Portugal.

Ude, A. (1993). Trajectory generation from noisy positions of object features for teaching robot paths, *Robotics and Autonomous Systems* **11**(2): 113–127.

Vijayakumar, S., D'Souza, A. & Schaal, S. (2005). Incremental online learning in high dimensions, *Neural Computation* **17**(12): 2602–2634.



Advances in Robot Manipulators
Edited by Ernest Hall

ISBN 978-953-307-070-4
Hard cover, 678 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

The purpose of this volume is to encourage and inspire the continual invention of robot manipulators for science and the good of humanity. The concepts of artificial intelligence combined with the engineering and technology of feedback control, have great potential for new, useful and exciting machines. The concept of eclecticism for the design, development, simulation and implementation of a real time controller for an intelligent, vision guided robots is now being explored. The dream of an eclectic perceptual, creative controller that can select its own tasks and perform autonomous operations with reliability and dependability is starting to evolve. We have not yet reached this stage but a careful study of the contents will start one on the exciting journey that could lead to many inventions and successful solutions.