Learning and Adaptation of Robot Skills using Fuzzy Models

Rainer Palm, Senior Member IEEE, and Boyko Iliev, Member IEEE

Abstract—Robot skills can be taught and recognized by a Programming-by-Demonstration technique where first a human operator demonstrates a set of reference skills. The operator’s motions are then recorded by a data-capturing system and modeled via fuzzy clustering and a Takagi-Sugeno modeling technique. The resulting skill models use the time as input and the operator’s actions as outputs. During the recognition phase, the robot recognizes which skill has been used by the operator in a novel demonstration. This is done by comparison between the time clusters of the test skill and those of the reference skills. Finally, the robot executes the recognized skill by using the corresponding reference skill model. Drastic differences between learned and real world conditions which occur during the execution of skills by the robot are eliminated by using the Broyden update formula for Jacobians. This method was extended for fuzzy models especially for time cluster models. After the online training of a skill model the updated model is used for further executions of the same skill by the robot.

I. INTRODUCTION

Programming of robots can be made easier by decomposing a robot task into robot skills which are low-level program units performing simpler robot tasks or subtasks. On the other hand, skill programming can be a time-consuming task because of which skills should be easy to program within a short period of time and with an appropriate accuracy. In this paper we deal with this problem using the so-called Programming by Demonstration (PbD) - approach. In the PbD framework a human operator shows or performs a task while the robot captures the data needed. Operator and experimental platform are equipped with data-capturing devices (e.g. data glove, cameras, haptic devices etc.) that deliver the necessary data to the robot. The robot analyzes the demonstrated actions and reactions and generates a corresponding robot skill. After the robot has generated (learned) a number of skills it is able to recognize a particular human skill in new demonstrations. Finally, motion trajectories and action/reaction patterns of the demonstrated task are automatically created by the robot using the skill models learned before. This approach can be used not only for industrial robots but also for prosthetics, humanoid service robots, remote control and teleoperation. Selected skills to be discussed are contour following, assembly (peg-in-hole insertion), handling of objects, or grasping of objects. Several problems may arise both with the recognition and the execution of skills learned by the robot.

Different techniques for modeling of skills have been applied for PbD.

Rainer Palm is adjunct professor at the AASS, Dept. of Technology Orebro University SE-70182 Orebro, Sweden, Email: rub.palm@t-online.de.
Boyko Iliev is with the AASS, Dept. of Technology Orebro University SE-70182 Orebro, Sweden, Emails: rub.palm@t-online.de, boyko.iliev@oru.se

Morrow and Khosla describe the generation of a library of robot capabilities by analysis and identification of tasks [1]. A sensorimotor layer is developed which integrates sensing (camera and force-torque sensor) into the robot programming primitives. Kaiser and Dillmann describe a neural net approach for the initial skill learning and reinforcement learning skill refinement and adaptation [2]. In the context of task learning Geib et. al. proposed an approach to integrating high-level artificial intelligence planning technology with low-level robotic control [3]. Chen [4] proposes the use of hybrid dynamic systems for construction of task-level assembly skills from human demonstrations. A general framework for robot tasks and robot skills has been presented by H. Liu [18]. In this paper fuzzy qualitative kinematic parameters are modeled by Gaussian Mixture Models.

In the field of recognition of robot behaviors the following publications are important: Kwun Han and M. Veloso describe an automated recognition of the behavior of robots using HMMs to represent and recognize strategic behaviors of robotic agents [5]. Zoellner et. al. [6] use a data glove with integrated tactile sensors for behavior recognition which is based on support vector machines (SVM). Ekvall and Kragic [7] apply Hidden Markov Models (HMM) and address the PbD-problem using the arm trajectory as an additional feature for grasp classification. Li et. al. [8] use the singular value decomposition (SVD) for the generation of feature vectors of human grasps and support vector machines (SVM) which are applied to the classification problem. Palm and Iliev presented two methods based on fuzzy models. In a direct comparison between the fuzzy approach and HMM method it could be shown that the fuzzy approach gives better results with respect to the recognition rates [9]. It turned out that a common fuzzy approach for modeling and recognition gives both better results and advantages in implementation than other methods such as HMM and SVM. In [10] the recognition and teaching of robot skills by fuzzy time-modeling is described. A human operator demonstrates a skill while his motions are recorded by a data-capturing device and subsequently modeled. The resulting skill models use the time as input and the operator’s actions and reactions as outputs. Given a test skill by the human operator the robot recognizes the individual phases of the skill and generates the type of skill shown by the operator.

Once a particular skill model has been generated it might be necessary to take new data into account which originate from different human operators to cover several ways of performing the same type of skill. According to these new data the skill model has to be changed offline so that the robot is enabled to recognize the skill under the new conditions (see [11]).
Another aspect is the change of environment conditions according to which the robot performs a task or a skill. Here we consider two cases: The first case is the presence of disturbances or small deviations between learned and real world conditions. These small deviations can be eliminated by appropriate control methods using corresponding sensor information. The second case concerns larger discrepancies between learned and real world conditions mostly having their origin in systematical changes of the robot environment. A good example is the contour following problem for robot applications like welding or gluing tasks. In this particular example the workpiece and the contour to be followed may have changed its position or even its shape so that a control strategy alone does not help to solve the problem. In this case online model learning or model adaptation regarding the new situation should be done. The method described here is based on an iterative learning of system models (see also [12]) especially on learning of Jacobians in differential models which ends up with the so-called Broyden update formula usually applied [13]. Learning of Jacobians has been reported in [14] where an algorithm for adaptive control of nonlinear multiple-input, multiple-output (MIMO) static systems is proposed. A nonlinear system is represented by a piecewise linear system model with a variable Jacobian matrix which is updated by the Broyden method using a fuzzy rule base. Jacobian learning for visual servoing can be found in [15] and [16]. Both publications deal with the problem of vision-guided robotic tracking of a moving target using a moving camera. For fuzzy system models presented in this paper it is essential that they consist of a set of fuzzily blended affine local models. Iterative learning of a fuzzy model requires the equal-ranking adaptation of each local model. Furthermore, the affine system model has to be transformed into a representation which makes a use of the broyden update formula applicable. Some of these problems have already been solved by Gorinevsky [17]. One contribution of our paper is the extension of this method to general fuzzy models in particular to fuzzy time cluster models. This aims at new skill models that are subsequently corrected by feedback control during execution. It should be mentioned that there is obviously a matching problem to be solved between human demonstration and robot execution of skills due to the degree of similarity between kinematic structures/constraints of human operator and robot concerning a specific task. An example for high/low similarity is the human grasp recorded from a data glove and executed by a human-like robotic hand/parallel gripper. For handling and contour following skills it is relatively easy to match between demonstration and execution. In this paper only skills with high similarity degrees between human demonstration and robot execution are discussed.

The paper is organized as follows: In Section II the general approach to skill learning is shortly outlined and a brief introduction to fuzzy time-modeling of skills is given. Section III describes the recognition of phases and the decision process for the classification of skills. Section IV describes the development of a Broyden update for fuzzy systems and the corresponding online training of time cluster models. Section V presents simulations and experimental results. The final Section VI draws some conclusions and directions for future work.

II. PROGRAMMING OF ROBOT SKILLS BY HUMAN DEMONSTRATION

Programming of robot skills requires two steps. First, we build a library of models of skills taught by human demonstrator(s). Next, the user can program a new task by demonstrating it to the robot. The newly demonstrated task is assumed to consist of skills. This leads to test models which are then compared with the trained models (skills) in the library. By such a comparison the robot is able to recognize these newly demonstrated skills. Finally a robot task including the recognized skills can automatically be generated. For the training phase two main tasks are needed to perform: segmentation of human demonstrations into skill phases and phase modeling of the segmented skill phases. Segmentation means a partition of the data record into a sequence of episodes, where each one contains a single skill phase. For the test phase three main tasks need to be performed: segmentation of the human test demonstrations, phase recognition, and skill classification. Phase recognition means to recognize the phases performed in each episode. The third task is to connect the recognized skill phases in such a way that a full skill can be identified.

The recognition of a skill phase is achieved by a model that reflects the behavior of the operator’s end-effector in time during the episode considered. Each demonstration is repeated several times to collect enough samples of every particular skill phase. From those data, models for each individual phase are developed using fuzzy clustering and Takagi–Sugeno fuzzy modeling ([19], [10]). Time instants are considered as model inputs and end-effector coordinates as model outputs. Let the end-effector coordinate be defined by

\[ x(t) = f(t) \]  

where \( x(t) \in R^3, f \in R^3, \) and \( t \in R^+ \). Furthermore, linearize (1) at selected time points \( t_i \)

\[ x(t) = x(t_i) + \frac{\Delta f(t)}{\Delta t} |_{t_i} \cdot (t - t_i) \]  

which is a linear equation in \( t \),

\[ x(t) = A_{i} \cdot t + d_{i} \]  

where \( A_{i} = \frac{\Delta f(t)}{\Delta t} |_{t_i} \in R^3 \) and \( d_{i} = x(t_i) - \frac{\Delta f(t)}{\Delta t} |_{t_i} \cdot t_i \in R^3 \). Using (3) as a local linear model one can express (1) in terms of a Takagi–Sugeno fuzzy model [20]

\[ x(t) = \sum_{i=1}^{c} w_i(t) \cdot (A_{i} \cdot t + d_{i}) \]  

\( w_i(t) \in [0, 1] \) is the degree of membership of a time point \( t \) to a cluster with the cluster center \( t_i \), \( c \) is the number of clusters, and \( \sum_{i=1}^{c} w_i(t) = 1 \).
Let \( x = [x_1, x_2, x_3]^T \) be the 3 end-effector coordinates and \( t \) the time. The general clustering and modeling steps are described as follows:

- Select an appropriate number of local linear models (data clusters) \( c \).
- Find \( c \) cluster centers \( (t_i, x_1, x_2, x_3), i = 1 \ldots c \), in the product space of the data quadruples \((t, x_1, x_2, x_3)\) by Fuzzy-c-elliptotype clustering.
- Find the corresponding fuzzy regions in the space of input data \((t)\) by projection of the clusters of the product space into Gustafson-Kessel clusters (GK) within the input space [21].
- Calculate \( c \) local linear (affine) models (4) using the GK clusters from step 2.

The membership degree \( w_i(t) \) of an input data point \( t \) in an input cluster \( C_i \) is calculated by

\[
w_i(t) = \frac{1}{\sum_{j=1}^{c} \frac{(t-t_j)^T M_{ipro j} (t-t_j)}{\| M_{ipro j} \|^2}}
\]

The projected cluster centers \( t_j \) and the induced matrices \( M_{ipro j} \) define the input clusters \( C_i \) \((i = 1 \ldots c)\). The parameter \( \tilde{\mu}_{ipro} > 1 \) determines the fuzziness of an individual cluster.

### III. RECOGNITION OF ROBOT SKILLS

In this section we assume that the segmentation of skills into phases has already been done and will therefore not be addressed in the following. In a first step the recognition of phases (i.e. sub-skills) is discussed. The second part deals with the recognition of skills using an a priori known number of phases. Finally, the recognition of skills with an unknown number of phases is discussed.

#### A. Recognition of phases using the distance between fuzzy clusters

Let the model of each phase have the same number of clusters \( c_{ph} \) so that each duration \( T_l \) \((l = 1 \ldots L)\) of the \( l \)-th phase is divided into \( c_{ph} \) - 1 time intervals \( \Delta t_i \), \( i = 2 \ldots c_{ph} \) of the same length. Let the phases be executed in an environment comparable with the modeled phase in order to avoid calibration and re-scaling procedures. Furthermore let

\[
V_{ref,ph1} = [X_1, \ldots, X_i, \ldots, X_{c_{ph}}]_{ref,ph1}
\]

\[
X_i = [x, y, z, f_x, f_y, f_z]^T
\]

where matrix \( V_{ref,ph1} \) includes the output cluster centers \( X_i \) for the \( l \)-th phase reference model.

A test model of the phase to be classified is built by the matrix

\[
V_{test,ph} = [X_1, \ldots, X_i, \ldots, X_{c_{ph}}]_{test,ph}
\]

A decision about which phase is present is made by applying the Euclidean matrix norm

\[
N_l = ||V_{ref,ph1} - V_{test,ph}||
\]

Once the unknown phase is classified to the phase model with the smallest norm \( \min(N_l) \), \( l = 1 \ldots L \) then the recognition of the phase is finished.

#### B. Recognition of skills using an a priori known number of phases

Once the phases of a test skill are recognized (identified) one should be able to recognize the skill as a whole and finally to reconstruct a hybrid automaton that represents the skill. For this purpose a list of possible skills and their phases should be produced. In the following we will discuss the three robot skills
- handling
- contour following
- assembly

The corresponding phases can be found in Table I. Switching between phases leads to a hybrid automaton already described in [11]. As an example Fig. 1 shows the correspondence between the contour following skill and its individual phases.

#### C. Recognition of skills with an a priori unknown number of phases

In the last subsections the recognition of skills with a priori known phases per skill has been discussed. However, experiments have shown that for a certain class of skills a clear distinction between phases is difficult to obtain. One of these 'difficult' classes is assembly where transitions between phases like 'follow with contact' and 'peg-in-hole insertion' are uncertain to detect by the sensors available and the segmentation software. This in turn can lead to a mismatch between the number of phases for the reference and the test skill, which makes it impossible to compare and recognize.

A solution to this problem is to refrain from comparing reference and test phases but rather comparing reference and test skills instead. In doing this, the number of clusters for each phase is left open for the time being, whereas a constant number of clusters is chosen for each skill. So, instead of comparing the cluster centers of phases, the cluster centers of complete skills are compared with each other.
This can be done by the following assumptions:
- The number of phases is restricted by a predefined upper bound.
- The number of clusters for a skill is defined in advance and is identical for each skill.
- The modeling error for a skill has an upper bound $e_{skill} = \int [x, f] - [x, f]_{model} dt \leq e_{max}$

The total number of clusters $c_{skill}$ for a skill can be determined as follows: Let $T_{skill_{\text{min}}}$ be the minimum number of data points among all skills to be considered. Let, furthermore, $c_{sg,\text{max}} - 1$ be the maximum possible number of phases. If we require at least two time clusters per phase and allow the maximum total number of clusters to be half of $T_{skill_{\text{min}}}$ then we obtain the following conservative bounds for $c_{skill}$

$$2 \cdot (c_{sg,\text{max}} - 1) \leq c_{skill} \leq T_{skill_{\text{min}}}/2 \quad (9)$$

from which $c_{skill}$ can be determined also taking account into the upper bound $e_{skill} \leq e_{max}$.

The next step is to compute the number $c_{ph,i,j}$ for the $i_j$ phases of a skill where $i_j$ denotes the $i$-th phase of the $j$-th skill. This number is obtained by the relation between the time length $T_{phase,i,j}$ of the $i_j$-th phase and the total time length $T_{skill,j}$ of the $j$-th skill. A simple calculation between the time lengths and the cluster numbers yields

$$c_{sg,i,j} = \left[ \frac{T_{phase,i,j}}{T_{skill,j}} \cdot c_{skill} \right] \quad (10)$$

where the brackets [...] mean a round-operation to the nearest integer. Such a round-operation can lead to a difference between the sum over all clusters $\sum c_{sg,i,j}$ and the total number of clusters $c_{skill}$. In order to avoid any mismatch a possible difference is added/subtracted to/from that phase with the maximum number of clusters $c_{sg,i,j}$.

D. Recognition of skills using the distance between fuzzy clusters

The recognition of a skill is done in a similar way than by using the models (6) and (7). Let the model of the $k$-th skill be composed by the sequence of the corresponding phase models $k = 1..K$

$$V_{ref,sk} = [V_{ref,ph_1},...,V_{ref,ph_m}]; \quad m = c_{ph,i,j_{ref}}$$

$$V_{ref,ph_i} = [X_1,\ldots,X_i,\ldots,X_{c_{ph,ph_i}}]_{ref,ph_i}; \quad l = 1..m$$

$$X_i = [x, y, z, f_x, f_y, f_z]_i^T \quad (11)$$

Let, furthermore, the test skill to be classified be composed by another sequence of phase models

$$V_{test,sk} = [V_{test,ph_1},...,V_{test,ph_l}]; \quad l = c_{ph,i,j_{test}}$$

$$V_{test,ph_i} = [X_1,\ldots,X_i,\ldots,X_{c_{ph,ph_i}}]_{test,ph_i}; \quad l = 1..n$$

$$X_i = [x, y, z, f_x, f_y, f_z]_i^T \quad (12)$$

A decision about the type of test skill is made by applying the Euclidean matrix norm

$$N_k = ||V_{ref,sk} - V_{test,sk}|| \quad (13)$$

Once the unknown skill is classified to the skill reference model with the smallest norm $\min(N_k), \ k = 1..K$, then the recognition of the skill is finished.

IV. ONLINE TRAINING OF TIME CLUSTER MODELS USING THE BROYDEN UPDATE

Online training of models deals with the correction of drastic differences between learned and real world conditions which occur during the execution of skills by the robot (see Fig. 2). These differences are hard to be eliminated just by a simple control strategy. Instead, based on sensor information the models are changed online by some optimization procedure. As already mentioned in Sect. I this is done by using the Broyden update formula for Jacobians. After the online training of a skill model the updated model is used for further executions of the same skill by the robot. Furthermore, an additional control loop deals with the remaining uncertainties and disturbances. Fig. 3 shows the corresponding learning and control scheme.

In what follows, first the general principle of the update process will be outlined. In a next step the learning principle is extended to fuzzy system models especially for fuzzy time cluster models.

A. The general principle

Let the output $Y \in \mathbb{R}^{m,1}$ of a real system be described by

$$Y = f(t) \quad (14)$$

Let furthermore

$$\dot{Y} = G(\tau, t) \cdot U + Z(\tau, t) \quad (15)$$

be the corresponding model with $\tau \in \mathbb{R}^1$ - optimization time.
Let us reformulate (15) by the substitution

\[ \Theta = [Z/c \ G] \in \mathbb{R}^{n,m+1}, \quad W = [c \ U^T]^T \in \mathbb{R}^{m+1,1} \]  

leading to

\[ \hat{Y} = \Theta \cdot W \]  

(17)

with \( c > 0 \) as a constant parameter to be chosen. In order to minimize the error between the outputs of the real system and the model a quadratic Lyapunov function is formulated

\[ V(\tau) = \frac{1}{2} (Y - \hat{Y})^T (Y - \hat{Y}) \rightarrow \min \]  

(18)

Derivation of (18) by \( \tau \) results in

\[ V' = -(\Theta'W)^T (Y - \hat{Y}) \leq 0 \]  

(19)

where \( V' = \frac{\partial V}{\partial \tau} \) and \( \Theta'W \in \mathbb{R}^{n,1} \) and \( V' \leq 0 \) is required for convergence of the optimization.

In order to meet (19) we set

\[ \Theta'W = \hat{\lambda}(Y - \hat{Y}) \]  

(20)

from which we obtain

\[ \Theta' = \frac{\hat{\lambda}(Y - \hat{Y})}{c^2 + U^TU} \]  

(21)

where \( \hat{\lambda} > 0 \) is the learning rate. Substituting (16) into (21) we get

\[ [Z'/c \ G'] = \hat{\lambda}(Y - \hat{Y}) \frac{[c \ U^T]}{c^2 + U^TU} \]  

(22)

and finally

\[ Z' = \frac{\hat{\lambda}(Y - \hat{Y})}{c^2 + U^TU} c^2 \]

\[ G' = \frac{\hat{\lambda}(Y - \hat{Y})}{c^2 + U^TU} U^T \]  

(23)

B. The discrete case

For the discrete case one obtains directly from (23)

\[ Z^{(k+1)} = Z^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)}) \frac{c^2}{c^2 + U^{(k)}T U^{(k)}} \]  

(24)

\[ G^{(k+1)} = G^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)}) \frac{U^{(k)}T}{c^2 + U^{(k)}T U^{(k)}} \]  

(25)

where

\[ Z^{(k)} \approx Z/\Delta \tau, \quad G^{(k)} \approx \Delta G/\Delta \tau, \quad \Delta Z = Z^{(k+1)} - Z^{(k)}, \quad \Delta G = G^{(k+1)} - G^{(k)}, \quad \Delta \tau = \tau^{(k+1)} - \tau^{(k)}, \quad \lambda = \hat{\lambda} \Delta \tau. \]

Superscript \( k \) denotes the \( k \)th optimization step.

C. The fuzzy case

For the fuzzy system

\[ \hat{Y}^{(k)} = \sum_{i=1}^{c_1} w_i(t^{(k)}) (A_i^{(k)} t^{(k)} + B_i^{(k)}) \]  

(25)

with

\[ c_i - \text{number of fuzzy clusters} \]

\[ k - \text{adaptation step} \]

we obtain

\[ B_1^{(k+1)} = B_1^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)}) \frac{w_i c^2}{\sum_{i=1}^{c_1} w_i^2 (c^2 + t^{(k)}/2) \} \]  

(26)

\[ A_1^{(k+1)} = A_1^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)}) \frac{w_i c^2}{\sum_{i=1}^{c_1} w_i^2 (c^2 + t^{(k)}/2) \} \]  

(27)

which will be explained in the following:

A general fuzzy system

\[ \hat{Y} = \sum_{i=1}^{c_i} w_i(p) (G_i U + Z_i) \]  

(28)

where \( p \) is an exogenous parameter vector, can be rewritten into

\[ \hat{Y} = \Theta \cdot \Phi(p, U) \]  

(29)

where

\[ \Theta = [[Z_1/c \ G_1], ..., [Z_{c_1}/c \ G_{c_1}]] \]  

\[ \Phi(p) = [w_1, ..., w_{c_1}]^T \in \mathbb{R}^{c_1,1} \]  

(30)

and

\[ \Phi(p, U) = \Phi(p) \otimes W = [[w_1 c \ w_1 U], ..., [w_{c_1} c \ w_{c_1} U]]^T \]  

(31)

where \( \otimes \) is the Kronecker (direct matrix) product.

Now we replace \( W \) by \( \Phi(p, U) \) in (21)

\[ \Theta' = \lambda(Y - \hat{Y}) \frac{\Phi(p, U)^T}{\Phi(p, U)^2 \Phi(p, U)} \]  

(32)

Using (29) and (30), (31) can be rewritten as

\[ \Theta' = \lambda(Y - \hat{Y}) \frac{\Phi(p, U)^T}{\Phi(p, U)^2 \Phi(p, U)} \]  

from which we get

\[ Z_i' = \lambda(Y - \hat{Y}) \frac{w_i c^2}{\Phi(p, U)^2 \Phi(p, U)^2} \]  

(33)

\[ G_i' = \lambda(Y - \hat{Y}) \frac{w_i U^T}{\Phi(p, U)^2} \]  

(34)

where \( ||\Phi(p, U)||^2 = \sum_{i=1}^{c_1} w_i^2 (c^2 + U^TU) \). By using the substitutions \( B_i = Z_i, \ A_i = G_i, \ t = U \) and discretizing (33) we finally obtain (26).
V. EXPERIMENTS AND SIMULATIONS

A. Recognition of skills

The experimental platform comprises a data glove with diodes mounted at the fingertips and links, see Fig. 4. A system of 5 stereo cameras takes records of the positions of the diodes so that the position of the hand and its fingers can be tracked. In addition, tactile sensors are mounted at each finger tip in order to detect the contact between the fingertips and an object or a surface, respectively. In the experiment only the tip of the index finger is tracked in order to identify the contour followed by this finger. The experiments include contour following examples using different contours running at different speeds. We recorded 10 samples for each skill to perform a reliable modeling. Each example consists of three phases: the approach phase, the contour following phase, and the retract phase. The experiment starts with the index fingertip being in contact with a defined start location at a distance from the contour. Next comes the approach phase where the finger moves towards the starting point of the contour. In the contour following phase the index finger moves along the path while the contact is preserved until the end of the contour is reached. During the retract phase there is no contact until the index finger reaches again the start location. The experiments can be divided into 3 groups, see Table II.

<p>| TABLE II |
| EXPERIMENTS |</p>
<table>
<thead>
<tr>
<th>Straight lines</th>
<th>Meanders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. slow speed (see Fig. 5)</td>
<td>7. meander slow (see Fig. 6)</td>
</tr>
<tr>
<td>2. fast speed</td>
<td>8. meander fast</td>
</tr>
<tr>
<td>3. ramp downhill slow</td>
<td>Loops</td>
</tr>
<tr>
<td>4. ramp downhill fast</td>
<td>9. loop slow 1 (see Fig. 7)</td>
</tr>
<tr>
<td>5. ramp uphill slow</td>
<td>10. loop slow 2</td>
</tr>
<tr>
<td>6. ramp uphill fast</td>
<td>11. loop fast 1</td>
</tr>
</tbody>
</table>

Three modeling examples are shown in Figs.5 - 7. The blue curves represent the modeled phases whereas the red curves represent the original data. Each skill phase is modeled by 15 cluster centers. The crosses depict the cluster centers. It can be observed that the modeling/approximation quality
of the fuzzy time models is excellent. The partition of the skill into phases has been done by means of the forces applied to the tip of the index finger. Figure 8 shows the time plots for the meander experiment 7. By means of the force \( f \) applied to the fingertip and its derivative \( df \), the segmentation can be done very easily because of the distinct derivatives of the force signals \( df \). The recognition of skills have been done by comparing each skill with all other skills using the method described in the last section. Since we only deal with contour following experiments the norms over the whole skill have been taken into account instead of considering the phases separately. The recognition rates are shown in Table III. To explain this table, let us consider experiment 3, ramp downhill slow, as an example (3rd row). Compared to itself the norm of the differences between model and test skill is very small (< 0.1). The next higher norm difference can be observed for experiment 4, ramp downhill fast. This corresponds completely with the idea that a similarity of trajectories should lead to small norm differences. Going through all 11 experiments it turns out that almost all contour following skills can be indentified. One exception is experiment 6 where skill 1 or 2 are identified instead of skill 5 as expected. It can also be noticed that the groups ‘A: Straight lines’, ‘B: Meander’ and ‘C: Loops’ can be significantly distinguished from each other. Furthermore, one can see that, from the recognition point of view, the groups A and B are more related than B and C or A and C.

### Table III

<table>
<thead>
<tr>
<th>skill</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2</td>
<td>0.6</td>
<td>0.5</td>
<td>0.6</td>
<td>0.4</td>
<td>0.1</td>
<td>0.9</td>
<td>1.8</td>
<td>1.8</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.7</td>
<td>-</td>
<td>0.3</td>
<td>0.8</td>
<td>0.6</td>
<td>0.9</td>
<td>1.0</td>
<td>1.9</td>
<td>1.9</td>
<td>1.8</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.6</td>
<td>0.3</td>
<td>-</td>
<td>0.8</td>
<td>0.6</td>
<td>0.8</td>
<td>0.9</td>
<td>1.8</td>
<td>1.9</td>
<td>1.8</td>
</tr>
<tr>
<td>4</td>
<td>0.6</td>
<td>0.6</td>
<td>0.8</td>
<td>0.8</td>
<td>-</td>
<td>0.5</td>
<td>0.9</td>
<td>1.0</td>
<td>1.9</td>
<td>2.0</td>
<td>1.9</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.6</td>
<td>0.5</td>
<td>-</td>
<td>0.8</td>
<td>1.0</td>
<td>1.8</td>
<td>2.0</td>
<td>1.9</td>
</tr>
<tr>
<td>6</td>
<td>0.7</td>
<td>0.2</td>
<td>0.9</td>
<td>0.8</td>
<td>0.9</td>
<td>0.8</td>
<td>-</td>
<td>0.4</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>7</td>
<td>0.9</td>
<td>0.9</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.4</td>
<td>-</td>
<td>1.5</td>
<td>1.4</td>
<td>1.3</td>
</tr>
<tr>
<td>8</td>
<td>1.8</td>
<td>1.8</td>
<td>1.9</td>
<td>1.8</td>
<td>1.9</td>
<td>1.9</td>
<td>1.4</td>
<td>1.4</td>
<td>0.6</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>9</td>
<td>1.8</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>2.0</td>
<td>1.9</td>
<td>1.4</td>
<td>1.4</td>
<td>1.4</td>
<td>0.4</td>
<td>0.5</td>
</tr>
</tbody>
</table>

#### B. Updating of models

The updating of a skill model is tested by the simulation of a contour following process with the steps:

1. Demonstration of the contour following process along an even surface
2. Modeling of the three phases: approach, follow contour, retract
3. Following of the even surface by the robot using the skill model
4. Following of a wavy and rising slope by the robot using the old skill model for the even surface (see Fig. 9)
5. Following the new surface by updating the old skill model
6. Following the new surface using the new skill model
7. Following the new surface using the new skill model and an additional control loop

It has to be emphasized that in this simulation only phase 2 is updated. The approach phase 1 persists until the contact force \( |f_{\text{meas}}| < |f_0| \) where \( |f_0| \) is a minimal force above which the contact between robot tool and object (surface) is established and phase 2 begins. The contact force \( f_{\text{meas}} \) is simulated as a spring force \( f_{\text{meas}} = K \Delta x \), where \( K \) - stiffness parameter \( \Delta x \) - deviation of position

Phase 2 is the actual contour following part of the skill during which a desired contact force \( f_d = 1N \) is required. After phase 2 it follows the retract phase 3 which is, for an even surface, attended with a lost of contact between tool and surface. However because of the new condition of a rising slope, tool and surface keep still contact. Fig. 10 a) shows the result for phase 2 before the model update. Because of the rising slope of the surface and the use of the old model, high contact forces are the result. Fig. 10 b)
shows the contact forces during model adaptation. Fig. 10 c) shows the behavior after adaptation with no further control included. This plot shows that changing the model according to the new conditions leads to an acceptable result. This result is further improved by a PD - force controller (Fig. 10 d)) plus a feedforward control term ( Fig. 10 e)) consisting of an additional position term acting in the direction of the surface.

VI. CONCLUSIONS

We have presented a new approach to Programming-by-demonstration of manipulation and handling tasks using skills based on fuzzy time clustering. The advantages of the method are: modeling and recognition of robot skills with both continuous-time and discrete-time characteristics and excellent modeling accuracy. The focus is directed to the skills ‘handling’, ‘contour following’ and ‘assembly’. In this context the partitioning of skills into phases and their modeling using fuzzy time clustering is discussed. The recognition of phases and skills is done by comparing fuzzy time clusters of model skills and test skills. To illustrate the approach, experiments with human demonstrations of different contouring tasks (’straight line’, ’meander’, or ’loop’) were performed. Modeling and recognition results were very good to excellent.

The aim of an online training of models is a correction of differences between learned and real world conditions during the execution of skills by the robot. Based on sensor information the old models are changed online by using the Broyden update formula for Jacobians. This method was extended for fuzzy models especially for time cluster models. The updated models are used for further executions of the same skill by the robot. After that, an additional control loop deals with the remaining uncertainties and disturbances. Simulation results shows the practicability of the method presented. In the future, experiments will be done for handling and assembly tasks for industrial robots. The recognition of skills using fuzzy time modeling and hybrid automata will be further developed and extended to different test persons. Updating of fuzzy system models in robotics will be an important focus in the future. A further focus will be the integration of high-level AI planning techniques with low-level robotic skills in a single modeling framework.

REFERENCES