Programming-by-Demonstration and Adaptation of Robot Skills by Fuzzy-Time-Modeling

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

***Abstract*—Complex robot tasks can be partitioned into motion primitives or robot skills that can directly be learned and recognized through Programming-by-Demonstration (PbD) by a human operator who demonstrates a set of reference skills. Robot motions are recorded by a data-capturing system and modeled by a specific fuzzy clustering and modeling technique where skill models use time instants as inputs and operator actions as outputs. In the recognition phase the robot identifies the skill shown by the operator in a novel test demonstration.**

Skill models are updated online during the execution of skills using the *Broyden update formula*. This method is extended for fuzzy models especially for time cluster models. The updated model is used for further executions of the same skill.

***Index Terms*—Programming-by-Demonstrations, Robot skills, fuzzy modeling, Broyden update**

I. Introduction

Robot skills are low-level motion and/or grasping capabilities that constitute the basic building blocks of robot tasks. One major challenge in robot programming is to be able to program robot tasks in an easy and fast way with high accuracy. *Programming-by-Demonstration (PbD)* is one such approach that has the afore mentioned properties. In PbD tasks are demonstrated as a sequence of skills by a human operator who is equipped with data-capturing devices (e.g., data glove, cameras, haptic devices etc.). While the demonstrator performs a skill, the robot captures the motion data, analyzes it and generates a robot-centered model of the demonstrated robot skill. After having learned a number of skills the robot is able to recognize skills in new test demonstrations. Finally, motion trajectories and action/reaction patterns of the demonstrated task are automatically generated by the robot using the skill models learned before. This approach can be used for industrial robots, prosthetics, humanoid service robots, remote control and teleoperation. Selected skills are e.g. *contour following, peg-in-hole insertion, handling* or *grasping of objects*.

see [1]

Morrow and Khosla describe the generation of a library of robot capabilities by analysis and identification of tasks [2]. Kaiser and Dillmann describe a neural net approach for the initial skill learning and adaptation and reinforcement learning for skill refinement [3]. Geib et. al. propose an approach to integrate high-level artificial intelligence planning technology with low-level robotic control [4]. Chen proposes the use of hybrid dynamic systems for construction of task-level assembly skills from human demonstrations [5] . A general framework for robot tasks and robot skills has been presented by H. Liu where fuzzy qualitative kinematic parameters are modeled by Gaussian Mixture Models [6].

Kwun Han and M. Veloso describe the recognition of robot behaviors using Hidden Markov Models (HMM) to represent and recognize strategic behaviors of robotic agents [7].

In [8] the recognition and teaching of robot skills by fuzzy time-modeling is described. In the case of new data the skill model will be changed *offline* so that the robot is enabled to recognize the skill under the new conditions.

An important aspect is the change of environment conditions while performing a skill. Here we consider two cases: The first case concerns disturbances or small deviations between learned and real world conditions. Small deviations can be eliminated by conventional feedback control methods using corresponding sensor information. The second case concerns larger discrepancies between learned and real world conditions caused by systematic changes of the robot environment. An example is the contour following problem for welding or gluing tasks. In this particular example the workpiece and the contour may have changed its position or even its shape so that a simple control strategy alone is not able to solve the problem. In this case *online model learning* or *model adaptation* should be introduced taking account the new situation.

The method described here is based on an iterative learning of system models [9] especially on learning of Jacobians in differential models which ends up with the so-called *Broyden update formula* usually applied [10]. Learning of Jacobians has been reported in [11] 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 [12]. Both publications deal with 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 [13]. Our work extends these ideas to general fuzzy models in particular to fuzzy time cluster models.

The paper is organized as follows: In Section II the general

Rainer Palm is adjunct professor at the AASS, Dept. of Technology Örebro University SE-70182 Örebro, 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

approach to skill learning is outlined. A brief introduction to fuzzy time-modeling of skills is discussed in Section III. Section IV describes the segmentation of skills into phases, the recognition of phases, and the decision process for the classification of skills. Section V describes the development of a Broyden update for fuzzy systems and online training of time cluster models. Section VI presents simulations and experimental results. The final Section VII draws some conclusions and directions for future work.

II. Programming-by-Demonstration of robot skills

Programming-by-Demonstration of robot skills requires the following steps:
1. A library of skill models is generated by human demonstration of the individual skills.
2. A user demonstrates a task to the robot consisting of a sequence of specific test skills.
3. Demonstrated test skills are compared with the trained skill models in the library.
4. The robot recognizes the full task consisting of the newly demonstrated skills.
5. The program of robot task including the recognized skills is automatically generated.
For the training of skills two main tasks are needed to perform: *segmentation* of human demonstrations into skill phases and *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 of skills 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.

Each phase starts and ends with a discrete event initiated either from a discrete sensor or from some preprocessing unit of continuous sensor signals. The structure of a skill can be described most appropriately by a hybrid automaton in which the nodes represent the continuous phases and the arcs the discrete transitions (switches) between them. The hybrid process is event-controlled [14] and is assumed to be stable both within the individual phases and with respect to the switching behavior between them. On the other hand the purpose of segmentation is to identify the discrete instants of time for the switches to occur in order to cut the whole skill into phases during demonstration.

The following section deals with fuzzy time-modeling in general that is used both for phase modeling and segmentation.

III. Fuzzy time-modeling

Recognition of a phase of a skill is achieved by a model that reflects the *behavior of the operator's end effector in time* during the episode (phase) considered. To collect enough samples of every skill phase each demonstration is repeated several times. From those data, models for each particular phase are developed by fuzzy time clustering and Takagi-Sugeno fuzzy modeling ([15], [16]). Fuzzy time clustering means that time instants are considered as model inputs and end effector coordinates as model outputs. Let the evolution of the end effector coordinate be defined by

$$\mathbf{x}(t) = \mathbf{f}(t) \tag{1}$$

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

$$\mathbf{x}(t) = \mathbf{x}(t_i) + \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \cdot (t - t_i) \tag{2}$$

which is a linear equation in t,

$$\mathbf{x}(t) = \mathbf{A}_i \cdot t + \mathbf{d}_i \tag{3}$$

where $\mathbf{A}_i = \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \in R^3$ and $\mathbf{d}_i = \mathbf{x}(t_i) - \frac{\Delta \mathbf{f}(t)}{\Delta t}|_{t_i} \cdot t_i \in R^3$. Using (3) as a local linear model one can express (1) in terms of a Takagi-Sugeno fuzzy model [17]

$$\mathbf{x}(t) = \sum_{i=1}^{c} w_i(t) \cdot (\mathbf{A}_i \cdot t + \mathbf{d}_i) \tag{4}$$

$w_i(t) \in [0, 1]$ 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 $\mathbf{x} = [x_1, x_2, x_3]^T$ be the 3 end effector position coordinates and t the time. The general clustering and modeling steps are

- Select an appropriate number of local linear models (data clusters) c
- Find c cluster centers $(t_i, x_{1i}, x_{2i}, x_{3i})$, $i = 1...c$, 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 [18]
- 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

$$w_i(t) = \frac{1}{\sum_{j=1}^{c} \left(\frac{(t-t_i)^T M_{ipro}(t-t_i)}{(t-t_j)^T M_{j\,pro}(t-t_j)} \right)^{\frac{1}{\tilde{m}_{proj}-1}}} \tag{5}$$

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

IV. Recognition of robot skills

In a first step the segmentation of a skill into phases is described. The second step addresses the recognition of phases (or sub-skills). The third step deals with the recognition of skills using an apriori known number of phases. Finally, the recognition of skills with an unknown number of phases is discussed.

A. Segmentation principle

Let for simplicity the signals of a skill be the end effector position coordinates $\mathbf{x}(t) \in R^3$ and the forces $\mathbf{f}(t) \in R^3$ in the end effector. In order to generate the 'events' determining the time bounds for the phases, $\mathbf{x}(t)$ is differentiated twice and $\mathbf{f}(t)$ only ones by time. The absolute values of the resulting vectors are collected in

$$\mathbf{X}(t) = [|\ddot{\mathbf{x}}(t)|^T, |\dot{\mathbf{f}}(t)|^T]^T. \tag{6}$$

For a segmentation we need the time-discrete case

$$\tilde{\mathbf{X}} = [\mathbf{X}(t_1)...\mathbf{X}(t_n)] \in R^{6\times n}. \tag{7}$$

Further define a vector of bounds $\mathbf{B} > 0 \in R^6$ above which $\mathbf{X}(t_i)$ are counted as 'events'. Then a vector $\mathbf{I} = [I_1...I_k...I_m]^T$ is generated where I_k are discrete time stamps t_i for which at least one component of $\mathbf{X}(t_i)$ lies above the corresponding component of the vector of bounds $\mathbf{B}$.

$$I_k = t_i \quad if \quad \mathbf{X}(t_i) > \mathbf{B}. \tag{8}$$

The next step is to select the number of time clusters c and find the clusters by time clustering for the data $\mathbf{Y} = ([\mathbf{X}(I_1); I_1]...[\mathbf{X}(I_k); I_k]...[\mathbf{X}(I_m); I_m])$. $\mathbf{Y}$ is a combination of 'events' $\mathbf{X}(t_i)$ and their corresponding time instants I_k. Once the time clusters are found the skill can be cut at these time instants into phases. However, for complicated skills the number of phases c might not be known in advance. Therefore we choose a higher number c_{ph} and merge close cluster centers into one.

B. Recognition of phases using the distance between fuzzy clusters

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

$$V_{ref,ph_l} = [\mathbf{X}_1, \ldots, \mathbf{X}_i, \ldots, \mathbf{X}_{c_{ph}}]_{ref,ph_l} \tag{9}$$
$$\mathbf{X}_i = [x, y, z, f_x, f_y, f_z]_i^T$$

where matrix V_{ref,ph_l} includes the output cluster centers $\mathbf{X}_i$ for the l-th phase reference model.

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

$$V_{test,ph} = [\mathbf{X}_1, \ldots, \mathbf{X}_i, \ldots, \mathbf{X}_{c_{ph}}]_{test,ph} \tag{10}$$

A decision about the type of phase is made by the Euclidean matrix norm

$$N_l = ||V_{ref,ph_l} - V_{test,ph}|| \tag{11}$$

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

C. 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 sequence of phases (hybrid automaton) that represents the skill. For this purpose a list of possible skills and their phases should be produced. Table I and Figs. 1,2, and 3 show 3 robot skills and 7 skill phases and the correspondence among them.

TABLE I
SKILLS AND PHASES

Phases Skills:	handling	contour following	assembly
1. Grasp object	x		x
2. Free motion	x		
3. Search contact	x	x	x
4. Keep contact	x		
5. Follow with contact	x	x	x
6. Peg-in hole			x
7. release object/contact	x	x	x

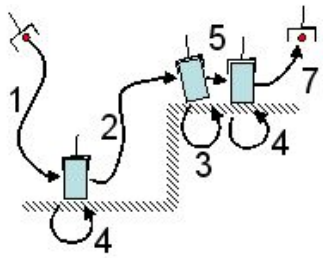

Fig. 1. Handling skill

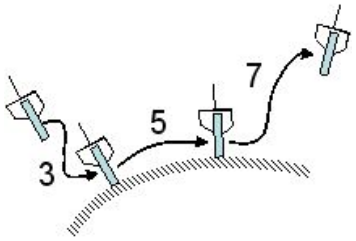

Fig. 2. Contour following skill

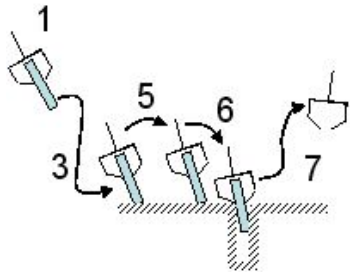

Fig. 3. Assembly skill

D. 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 among phases is difficult to obtain. One of these 'difficult' classes is *assembly* where transitions among phases like 'follow with contact' and 'peg-in-hole insertion' are uncertain to detect by the sensors available and the segmentation software. This can lead to a mismatch among the number of phases for the reference skill and the test skill. This in turn makes a proper comparison and recognition of phases impossible. A solution to this problem is to define a constant number of clusters for the whole skill to be the same for each skill. Then, instead of comparing the cluster centers of phases we can perform a comparison of the cluster centers of the total skills. This can be done by the following assumptions:

- The number of phases is restricted by a predefined upper bound
- The total 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 $e_{skill} = \int_t |[\mathbf{x}, \mathbf{f}] - [\mathbf{x}, \mathbf{f}]_{model}| dt \leq e_{max}$

The total number of clusters c_{skill} for a skill can be determined as follow: Let $T_{skill\,min}$ be the minimum number of data points among all skills to be considered. Let, furthermore, $c_{sg,max}-1$ be the maximum 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 $T_{skill\,min}$ then we obtain the following conservative bounds for c_{skill}

$$2 \cdot (c_{sg,max} - 1) \leq c_{skill} \leq T_{skill\,min}/2 \tag{12}$$

from which c_{skill} can be determined taking into account 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

$$c_{sg_{i_j}} = \left[\frac{T_{phase,i_j}}{T_{skill,j}} \cdot c_{skill} \right] \tag{13}$$

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 $\sum_i c_{sg_{i_j}}$ 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 $c_{sg_{i_j}}$.

E. Recognition of skills using the distance between fuzzy clusters

Recognition of a skill is performed in a similar way like with models (9) and (10). Let the model of the k-th skill be composed by the sequence of the corresponding phase models $k = 1...K$

$$\begin{aligned} V_{ref,sk_k} &= [V_{ref,ph_1} ... V_{ref,ph_m}]; \quad m = c_{ph,i_j\,ref} \\ V_{ref,ph_l} &= [\mathbf{X}_1, \ldots, \mathbf{X}_i, \ldots, \mathbf{X}_{c_{ph}}]_{ref,ph_l}; l = 1...m \\ \mathbf{X}_i &= [x, y, z, f_x, f_y, f_z]_i^T \end{aligned} \tag{14}$$

Furthermore, let the test skill be composed by a sequence of phase models being different from (14)

$$\begin{aligned} V_{test,sk_k} &= [V_{test,ph_1} ... V_{test,ph_n}]; \quad n = c_{ph,i_j\,test} \\ V_{test,ph_l} &= [\mathbf{X}_1, \ldots, \mathbf{X}_i, \ldots, \mathbf{X}_{c_{ph}}]_{test,ph_l}; l = 1...n \\ \mathbf{X}_i &= [x, y, z, f_x, f_y, f_z]_i^T \end{aligned} \tag{15}$$

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

$$N_k = ||V_{ref,sk_k} - V_{test,sk}|| \tag{16}$$

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.

V. On-line training of time cluster models using the Broyden update

Online training of models addresses the correction of drastic differences between learned and real world conditions which occur during the execution of skills by the robot. 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 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 is responsible to cope with remaining uncertainties and disturbances.

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 updating principle

The output $Y \in \Re^{n,1}$ of a real system let be described by

$$Y = f(t) \tag{17}$$

Furthermore, let

$$\begin{aligned} \hat{Y} &= G(\tau, t) \cdot U + Z(\tau, t) \\ \hat{Y}, Z \in \Re^{n,1}; \quad U \in \Re^{m,1}; \quad G \in \Re^{n,m} \end{aligned} \tag{18}$$

be the corresponding model with
$\tau \in \Re^1$ - optimization time
$t \in \Re^1$ - real time
To apply the Broyden update for Jacobians we rewrite (18) by the substitution

$$\Theta = [Z/c \quad G] \in \Re^{n,m+1}, \quad W = [c \quad U^T]^T \in \Re^{m+1,1} \tag{19}$$

leading to

$$\hat{Y} = \Theta \cdot W \tag{20}$$

with $c > 0$ as a constant parameter to be defined. To minimize the error between the outputs of the real system and the model a *quadratic Lyapunov function* is formulated

$$V(\tau) = \frac{1}{2}(Y - \hat{Y})^T(Y - \hat{Y}) \rightarrow min \tag{21}$$

Derivation of (21) by τ yields

$$V' = -(\Theta' W)^T(Y - \hat{Y}) \leq 0 \tag{22}$$

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

In order to meet (22) we choose

$$\Theta' W = \tilde{\lambda}(Y - \hat{Y}) \tag{23}$$

from which we obtain

$$\Theta' = \tilde{\lambda}(Y - \hat{Y})\frac{W^T}{W^T W} \tag{24}$$

where $\tilde{\lambda} > 0$ is the learning rate. Substituting (19) into (24) leads to

$$[Z'/c \quad G'] = \tilde{\lambda}(Y - \hat{Y})\frac{[c \quad U^T]}{c^2 + U^T U} \tag{25}$$

Finally we obtain the updating law for Z and G

$$Z' = \tilde{\lambda}(Y - \hat{Y})\frac{c^2}{c^2 + U^T U}; \tag{26}$$
$$G' = \tilde{\lambda}(Y - \hat{Y})\frac{U^T}{c^2 + U^T U}$$

B. The discrete case

The discrete case follows directly from (26)

$$Z^{(k+1)} = Z^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)})\frac{c^2}{c^2 + {U^{(k)}}^T U^{(k)}} \tag{27}$$
$$G^{(k+1)} = G^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)})\frac{{U^{(k)}}^T}{c^2 + {U^{(k)}}^T U^{(k)}}$$

where
$Z' \approx \Delta Z/\Delta\tau$, $G' \approx \Delta G/\Delta\tau$, $\Delta Z = Z^{(k+1)} - Z^{(k)}$, $\Delta G = G^{(k+1)} - G^{(k)}$, $\Delta\tau = \tau^{(k+1)} - \tau^{(k)}$, $\lambda = \tilde{\lambda}\Delta\tau$. Superscript k denotes the kth optimization step.

C. The fuzzy case

Theorem:
For the fuzzy system

$$\hat{Y}^{(k)} = \sum_{i=1}^{c_l} w_i(t^{(k)})(A_i^{(k)} t^{(k)} + B_i^{(k)}) \tag{28}$$

with
c_l - number of fuzzy clusters
k - adaptation step

we obtain

$$B_i^{(k+1)} = B_i^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)})\frac{w_i c^2}{\sum_{i=1}^{c_l} w_i^2(c^2 + {t^{(k)}}^2)} \tag{29}$$
$$A_i^{(k+1)} = A_i^{(k)} + \lambda(Y^{(k)} - \hat{Y}^{(k)})\frac{w_i t^{(k)}}{\sum_{i=1}^{c_l} w_i^2(c^2 + {t^{(k)}}^2)}$$

Proof: A general fuzzy system

$$\hat{Y} = \sum_{i=1}^{c_l} w_i(U)(G_i U + Z_i) \tag{30}$$

where $w_i(U) \in [0, 1]$ is the degree of membership of U to a cluster with the cluster center $U = U_i$. c_l is the number of clusters, and $\sum_{i=1}^{c_l} w_i(U) = 1$ is rewritten into

$$\hat{Y} = \Theta \cdot \Phi(p, U) \tag{31}$$

where

$$\Theta = [[Z_1/c \quad G_1], ..., [Z_{c_l}/c \quad G_{c_l}]] \tag{32}$$
$$\Phi(p) = [w_1, ..., w_{c_l}]^T \in \Re^{c_l,1}$$

and

$$\Phi(p, U) = \Phi(p) \otimes W = \tag{33}$$
$$[[w_1 c \quad w_1 U^T], ..., [w_{c_l} c \quad w_{c_l} U^T]]^T$$

with $\otimes$ as Kronecker (direct matrix) product.
In (24) we replace W by $\Phi(p, U)$

$$\Theta' = \tilde{\lambda}(Y - \hat{Y})\frac{\Phi(p, U)^T}{\Phi(p, U)^T \Phi(p, U)} \tag{34}$$

which can be rewritten into

$$\Theta' = [[Z_1'/c \quad G_1'], ..., [Z_{c_l}'/c \quad G_{c_l}']] = \tag{35}$$
$$\frac{\lambda}{||\Phi(p, U)||^2}(Y - \hat{Y})[[w_1 c \quad w_1 U^T], ..., [w_{c_l} c \quad w_{c_l} U^T]]$$

from which we get

$$Z_i' = \lambda(Y - \hat{Y})\frac{w_i c^2}{||\Phi(p, U)||^2}; \tag{36}$$
$$G_i' = \lambda(Y - \hat{Y})\frac{w_i U^T}{||\Phi(p, U)||^2}$$

where $||\Phi(p, U)||^2 = \sum_{i=1}^{c_l} {w_i}^2(c^2 + U^T U)$. By using the substitutions $B_i = Z_i$, $A_i = G_i$, $t = U$ and discretizing (36) we finally obtain (29). ■

VI. 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 hand and fingers can be tracked. In addition, tactile sensors are mounted at each fingertip to detect the contact between fingertips and objects. 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 starting position at a distance from the contour. In the approach phase 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. We recorded 10 samples for each skill to perform a reliable modeling.

TABLE II
Experiments

Straight lines	**Meanders**
1. slow speed (see Fig. 5)	7. meander slow (see Fig. 6)
2. fast speed	8. meander fast
3. ramp downhill slow	**Loops**
4. ramp downhill fast	9. loop slow 1 (see Fig. 7)
5. ramp uphill slow	10. loop fast 1
6. ramp uphill fast	

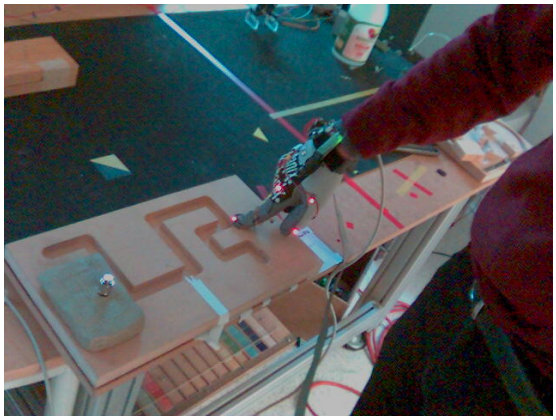

Fig. 4. Experiment with a meander-like contour

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

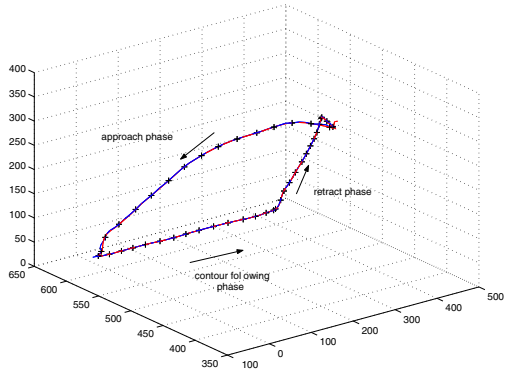

Fig. 5. Contour following: line

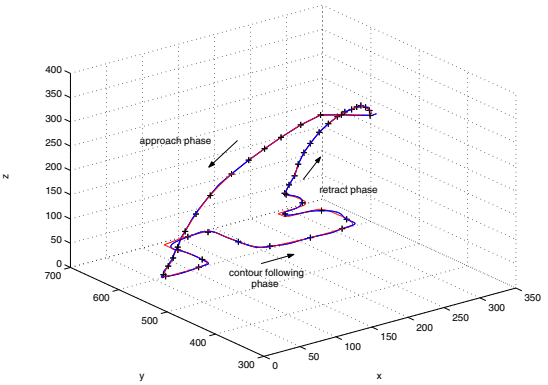

Fig. 6. Contour following: meander

phases has been done using the forces applied to the tip of the index finger. Figure 8 shows the time plots for the meander experiment 7. Using force f applied to the fingertip and its derivative df the segmentation can be obtained very easily because of the distinct derivatives of the force signals df. The recognition of skills have been achieved by comparing each test skill with all model skills. Since we only discuss contour following experiments, the norms over the total skill have been computed instead of considering the phases separately. The results are shown in Table III. To read this table, let us consider experiment 3, *ramp downhill slow*, for example (3rd row). Compared to a model skill of the same type the norm of the differences between model and test skill is near zero. The

TABLE III
Identification results, − means < 0.1

skill	1	2	3	4	5	6	7	8	9	10
1	-	0.2	0.6	0.5	0.6	0.4	0.7	0.9	1.8	1.8
2	0.2	-	0.7	0.6	0.6	0.5	0.7	0.9	1.8	1.8
3	0.5	0.7	-	0.3	0.8	0.6	0.9	1.0	1.9	1.8
4	0.5	0.6	0.3	-	0.8	0.6	0.8	0.9	1.8	1.8
5	0.6	0.6	0.8	0.8	-	0.5	0.9	1.0	1.9	1.9
6	0.4	0.5	0.6	0.6	0.5	-	0.8	1.0	1.8	1.9
7	0.7	0.7	0.9	0.8	0.9	0.8	-	0.4	1.4	1.4
8	0.9	0.9	1.0	0.9	1.0	1.0	0.4	-	1.5	1.3
9	1.8	1.8	1.9	1.8	1.9	1.8	1.4	1.5	-	0.4
10	1.8	1.8	1.8	1.8	1.9	1.9	1.4	1.4	0.4	-

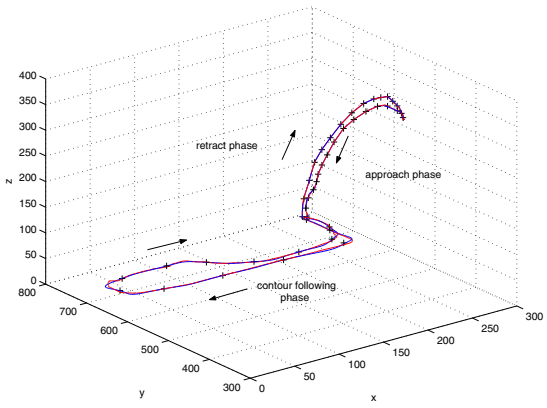

Fig. 7. Contour following: loop

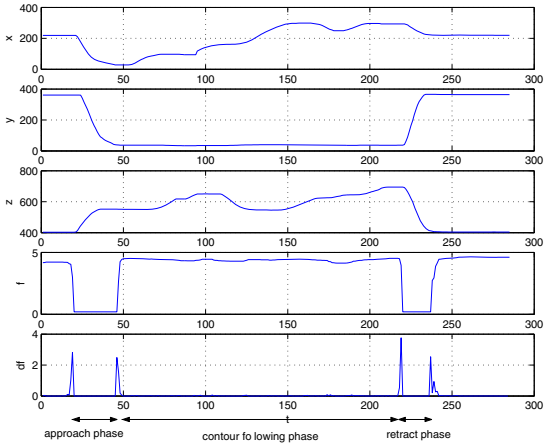

Fig. 8. Contour following: loop, time plots

next higher norm difference can be observed for experiment 4, *ramp downhill fast*. This corresponds completely with the idea that 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 identified. 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 one another. Furthermore, one can see that, from the recognition point of view, groups A and B are more related than B and C or A and C.

B. Updating of models

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 flat surface
2. Modeling of the three phases: *approach, follow contour, retract*
3. Following of the flat 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 flat 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

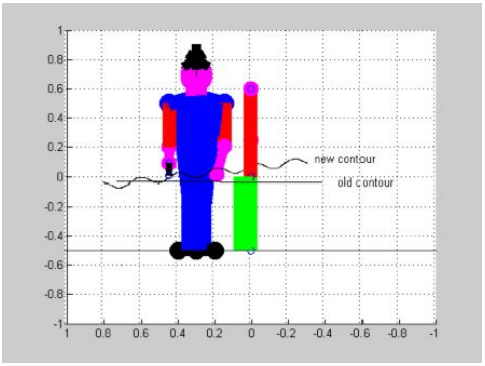

Fig. 9. Change of contour from plane to ripple

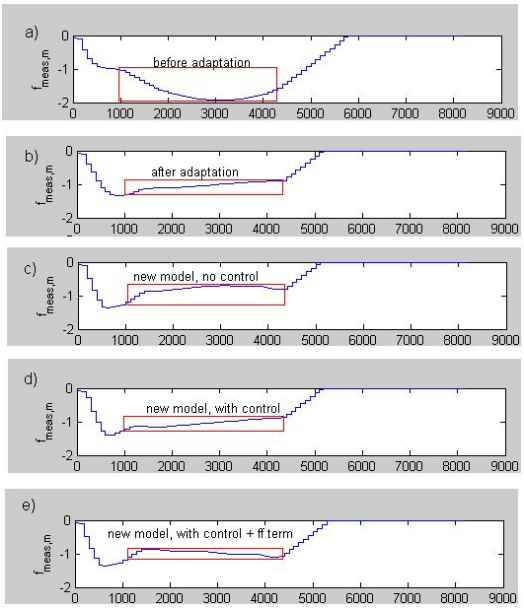

Fig. 10. Change of contour: results

7. Following the new surface using the new skill model and an additional control loop

Notice that in this simulation only phase 2 is updated. The approach phase 1 persists until the contact force $|f_{meas_m}| < |f_0|$ 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_{meas_m} is simulated as a spring force $f_{meas_m} = K\Delta x$, where
K - stiffness parameter
Δx - 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 flat surface, attended with a loss of contact between tool and surface. However because of the new condition of a rising slope, tool and surface still keep in 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.

To further improve the contour following results the updating procedure was performed with a lower speed. Here the robot has more time to adapt to the new contour. Once the contour model has been updated the real contour following process can run with the original higher speed. Figure 11 shows the results where the modeling speed was half of the real speed of the contouring operation. Figure 11 a shows the result after adaptation with slow speed but with no control included. Figures 11 b and 11 c show the results with a PD-force controller and with an additional feedforward term, respectively. Comparing the results from Figures 10 e and 11 c we can notice a further improvement of the contour following quality for a slow speed of adaptation.

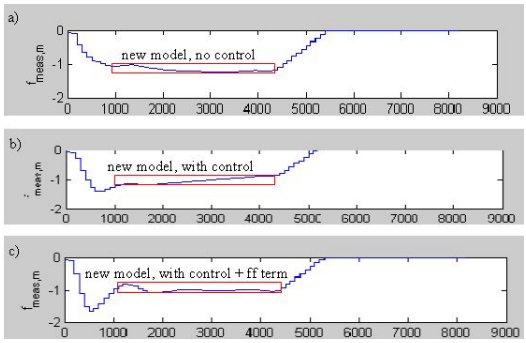

Fig. 11. Change of contour: results1

VII. Conclusions

A new approach to Programming-by-Demonstration of manipulation and handling tasks using skills based on fuzzy time clustering has been presented. The advantages of the method are: modeling and recognition of robot skills with both continuous-time and discrete-time characteristics and excellent modeling accuracy. In this context the focus is directed to the skills "handling", "contour following" and "assembly". Skills are partitioned into phases and their modeling by fuzzy time clustering is discussed. Phases and skills are recognized by comparing fuzzy time clusters of model skills and test skills. Experiments with human demonstrations of different contouring tasks ("straight line", "meander", or "loop") show very good up to excellent modeling and recognition results.

The goal of 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 copes with the remaining uncertainties and disturbances. Simulation results show the practicability of the method presented. In the future, the recognition of skills using fuzzy time modeling and hybrid automata will be developed and extended to different test persons. Updating of fuzzy system models will keep being 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 common modeling framework.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems.*, Vol 57:469–483, 2009.

[2] J. D. Morrow and P. K. Khosla. Manipulation task primitives for composing robot skills. *Proceedings of the 1997 IEEE Intemational Conference on Robotics and Automation, Albuquerque, New Mexico*, pages 3354–3359, 1997.

[3] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. *Proceedings of the 1996 IEEE Intemational Conference on Robotics and Automation, Minneapolis, Minnesota*, 1996.

[4] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. Woergoetter. Object action complexes as an interface for planning and robot control. In *Proc. IEEE RAS Int Conf. Humanoid Robots(Genova)*, Genova, Dec. 4-6 2006. IEEE, IEEE.

[5] J. Chen. Constructing task-level assembly strategies in robot programming by demonstration. *Int. Journal of Robotic Research*, 24:1073–1085, December 2005.

[6] H. Liu. A fuzzy qualitative framework for connecting robot qualitative and quantitative representations. *IEEE Trans. on Fuzzy Systems*, 16(6):1522–1530, 2008.

[7] Kwun Han and M. Veloso. Automated robot behavior recognition. *Proceedings of the 1999 ijcai Conference*, 1999.

[8] R. Palm, B. Iliev, B. Kadmiry, and D. Driankov. Recognition and teaching of robot skills by fuzzy time-modeling. In *Proceedings IFSA/EUSFLAT 2009*, Lisbon, July 20-24 2009. IFSA/EUSFLAT.

[9] Hyo-Sung Ahn, Yang Quan Chen, and K. L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Trans. on Syst., Man, and Cybern. Part C: Applications and Reviews.*, VOL. 37, NO. 6:1099–1121, NOVEMBER 2007.

[10] C.G. Broyden. A class of methods for solving nonlinear simultanuous equations. *Mathematics of Computation*, Vol. 19, No. 92:577–593, Oct. 1965.

[11] D. P. Filev, S. Bharitkar, and Meng-Fu Tsai. Nonlinear control of static systems with unsupervised learning of the initial conditions. In *18th Intern. Conf. of the North American Fuzzy Informantion Processing Society - NAFIPS*, New York, NY,USA, June 10-12 1999. NAFIPS.

[12] N. Mansard, M. Lopes, J. Santos-Victor, and F. Chaumette. Jacobian learning methods for tasks sequencing in visual servoing. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 9 - 15 2006. ASME.

[13] D. Gorinevsky. An approach to parametric nonlinear least square optimization and application to task-level learning control. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, VOL. 42, NO. 7:912–927, JULY 1997.

[14] M.S.Branicky, V.S.Borkar, and S.K.Mitter. A unified framework for hybrid control: Background, model, and theory. *Proceedings of the 11th Intern. Conference on Analysis ond Optimization of Systems. INRIA. France*, June 15-17, 1994.

[15] R. Palm and Ch. Stutz. Open loop dynamic trajectory generator for a fuzzy gain scheduler. *Engineering Applications of Artificial Intelligence*, Vol. 16:213–225, 2003.

[16] R. Palm, B. Iliev, and B. Kadmiry. Recognition of human grasps by time-clustering and fuzzy modeling. *Robotics and Autonomous Systems*, Vol. 57, No. 5.:484–495, 2009.

[17] T. Takagi and M. Sugeno. Identification of systems and its applications to modeling and control. *IEEE Trans. on Syst., Man, and Cyb.*, Vol. SMC-15. No.1:116–132, January/February 1985.

[18] D.E. Gustafson and W.C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *Proceedings of the 1979 IEEE CDC*, pages 761–766, 1979.