Behavior Recognition for Segmentation of Demonstrated Tasks

Erik A. Billing
Department of
Computing Science
Umeå University
Umeå, Sweden
phone: +4690-7869915
e-mail: billing@cs.umu.se

Thomas Hellström
Department of
Computing Science
Umeå University
Umeå, Sweden
phone: +4690-7867759
e-mail: thomash@cs.umu.se

***Abstract*—One common approach to the robot learning technique Learning From Demonstration, is to use a set of pre-programmed skills as building blocks for more complex tasks. One important part of this approach is recognition of these skills in a demonstration comprising a stream of sensor and actuator data. In this paper, three novel techniques for behavior recognition are presented and compared. The first technique is function-oriented and compares actions for similar inputs. The second technique is based on auto-associative neural networks and compares reconstruction errors in sensory-motor space. The third technique is based on S-Learning and compares sequences of patterns in sensory-motor space. All three techniques compute an activity level which can be seen as an alternative to a pure classification approach. Performed tests show how the former approach allows a more informative interpretation of a demonstration, by not determining "correct" behaviors but rather a number of alternative interpretations.**

***Index Terms*—Learning from demonstration, Segmentation, Generalization, Sequence Learning, Auto-associative neural networks, S-Learning.**

I. Introduction

A lot of research in Learning from Demonstration (LFD) deals with the problem of generating behaviors from data recorded during manual demonstrations. Behaviors are in this case direct mappings from sensor states to action states, where actions typically are low-level motor commands [12], [8]. This both challenging and interesting research has natural limitations in many real-world applications. A key problem is generalization, i.e. the robot´s ability to repeat a demonstrated behavior under conditions not identical to those present during the demonstration. One common way to overcome this is to transform the demonstration into a set of pre-programmed higher-level actions, called sub-tasks [20], motion primitives [23], motor primitives [1] or motor skills [18].

Most works in LFD deal with tasks such as robot arm motion, pole balancing, and robot gait, e.g., [2], [13], [4], and the higher-level actions are often short sequences of low-level motor commands. One of the few examples of LFD with complex high-level behaviors is the work presented by Nicolescu and Mataric´ [16], [15], [10]. The work presented in this paper uses similar types of higher-level skills. For this reason, we will adopt the terminology developed by Nicolescu [16], where *skills* can be understood as a relatively simple mapping from sensors to actuators, with an aptitude or ability to achieve or maintain a goal. Several skills can be combined to perform a task. A *task, which* has higher complexity than skills, may involve serveral goals and is represented as a sequence of skills.

The ability to represent a demonstration as a sequence of skills does not only serve as support for generalization, but is also a powerfull way to make the demonstrated data understandable to a human user. A labeled sequence of skills, for example, *following the wall to my right, passing through a door, going straight over the floor avoiding any obstacle*, is significantly easier to interprete than the raw sensor and motor data. A demonstration represented in this way should give the user a better understanding of what the robot did observe, and the opportunity to evaluate the robot's interpretation of the demonstrated behavior.

As such, the generalization part of LFD is in this context understood as describing a task-level demonstration in terms of the previously learned, or pre-programmed skills. This involves determining positions and characteristics of *segmentation points* where the skills change, and identifying the skills themselves. Identifying suitable parameters for parametrized skills may also be part of this process, e.g., [15]. Consequently, the result of performing LFD will be a task representation, i.e., a sequence of identified skills, together with a characterization of the segmentation points. This paper addresses the specific problem of determining positions for segmentation points and the identification of previously learned skills for each segment. However, the problem of determining the characteristics of each segmentation point is not addressed, and will be subject to future work.

Section 2 of our paper describes LFD and introduces a basic notation. Section 3 presents three novel techniques for behavior recognition. In Section 4, the test cases are described and the test results are reported in Section 5. Related work is reviewed in Section 6, and finally conclusions and ideas for future research are given.

II. Learning From Demonstration

The basic principle of the learning technique Learning From Demonstration (LFD) is that a robot should learn to repeat

a behavior after being teleoperated through one or several demonstrations performed by a human. Some key concepts will be introduced in this section.

The robot represents its view of itself and of the environment at time t with a sensor vector x_t comprising observed sensor variables. Some variables represent exteroceptive physical sensors, such as infra-red distance sensors, ambient light sensors, cameras, bump sensors, accelerometers, gyros, and GPS. Other variables represent interoceptive sensors, such as joint angles, wheel encoders, actual motor speed, and battery state. Dynamics and time dependencies can be handled by adding lagged variables, derivatives, or sub sequences to the sensor vector. The robot affects the world and itself through its effectors represented by the action vector y_t. Typical actions are motor speed and steering angle controlling propulsion, or joint angles controlling the motion of a robot arm.

A behavior β can be represented by a function from a subset of sensor vector space to a subset of action vector space. Thus, the expression $y_t = \beta(x_t)$ means that β suggests action y_t for a sensor vector x_t. β is sometimes called "policy" or, in the control systems community, "control law". By adding lagged state variables to the sensor vector, the behaviors can be viewed as purely reactive while still being able to handle dynamical models.

Often, no distinction between sensor data and actions is made, and an event e_t is defined as

$$e_t = (x_t, y_t)\,. \tag{1}$$

A demonstration θ is represented by a time series comprising N such events:

$$\theta = (e_1, e_2, ..., e_N)\,. \tag{2}$$

For each t, x_t is the observed sensor vector and y_t is the action vector issued by the demonstrator. Hereby, β most often denotes a relatively simple behavior with a single goal. In these cases, β represents a skill, as defined in the introduction. Common skills are *avoid obstacles, follow wall* or *drive towards goal*.

III. Methods for behavior recognition

In the presented work, three methods for behavior recognition are suggested and evaluated. Each method defines a function f_β for each skill β, mapping a sequence of events $\theta = \{e_1, e_2, ..., e_N\}$ and a time index t in $[1, \ldots, N]$ to a a real number representing the *activity level* $\alpha_t \in [0, 1]$:

$$\alpha_t = f_\beta\,(\theta, t)\,. \tag{3}$$

α_t could, informally, be interpreted as the probability of β controlling the robot at time t given the observations θ, i.e.:

$$\alpha_t \sim P\,(\beta|\theta)\,. \tag{4}$$

If f_β for all β are computed for each time t, the activation levels can be used both for positioning the segmentation points and determining the most suitable skills for each segment. The three suggested methods for behavior recognition are described below, followed by two examples where the methods are applied and evaluated.

A. β-Comparison

This method is based on the notion that two skills are equal if they produce similar actions given the same sensor input. f_β is defined as the distance between the action y_t observed in θ and the action produced by the specific β :

$$f_\beta(\theta, t) = 1 - d\,(\beta\,(x_t)\,, y_t) \tag{5}$$

where d is a function computing a relevant distance measure for action vectors in the specific application. d should reflect the *relevant* difference between the two actions, and the implementation of d is as such dependent on both the robot and the behavior which is being learned. This is a limitation compared to the other methods described in Section III-B and III-C, which do not require any application-specific functions. The precise implementation of d used in the present work is described in Section IV.

B. AANN-Comparison

Autoassociative Neural Networks AANNs are regular feed-forward neural networks with the same size of input and output layers. The input and output parts of the training data are identical, such that the net learns to map input values onto the same values in the output layer. With a small hidden layer, the network performs data compression with a least-squares criterion [6]. When exposed to a new data vector, the difference between input and output (reconstruction error) is a measure of how similar the new data vector is to the training data. In this particular case, the network input at time t consists of the vector e_t comprising both sensor data and action data (Equation 1). The network output is denoted τ_t. One network for each skill is created and trained with an event sequence θ_β observed while performing behavior β. In this way, the characteristics of the sensory-motor patterns from β will be modeled by the network. When exposed to a new input vector, the reconstruction error can be used to define the f function and hence the activity level scaled to $[0, 1]$:

$$f_\beta(\theta, t) = 1/\,(1 - \|\tau_t - e_t\|)\,. \tag{6}$$

C. S-Comparison

This algorithm is based on S-Learning, a prediction-based control algorithm inspired by the human neuromotor system,[21], [22]. S-Learning is able to extract temporal patterns in presented data, a very attractive property when comparing sequential data, such as sensor readings and motor commands. The temporal dimension allows S-Comparison to make decisions based on several recent samples, in contrast to β-Comparison (III-A) and AANN-Comparison (III-B) which both treat each sample separately.

S-Comparison differs in many respects from the S-Learning algorithm. Some changes are a direct consequence of the algorithm being used to compute a similarity measure rather than future actions. Other modifications improve the handling

of continuous data, since S-Learning was originally designed for discrete data.

Similarly to AANN-Comparison, one model of each skill is first created from a separate demonstration θ_β. Each model, or *pattern library* $\lambda = \{\rho_1, \rho_2, \ldots\}$, is a set of *patterns* $\rho = (e_1, e_2, ..., e_m)$, where each ρ is a sub sequence of θ_β. $\rho(k)$ denotes the k-th element e_k in ρ. The concatenation operator $\rho||e$ combines ρ and e into a new pattern including all elements in ρ and with e as last element.

Algorithm 1 Training of S-Comparison

1) $\lambda = \{\varnothing\}$
2) $\rho_{max} \leftarrow null$; $\delta_{max} \leftarrow H$
3) For each $\rho \in \lambda$ then
   $\delta \leftarrow \sum_{k=1}^{|p|} 1 - \frac{|\rho(k) - \theta(k)|}{\sigma}$
   If $\delta > \delta_{max}$ then $\delta_{max} \leftarrow \delta$; $\rho_{max} \leftarrow \rho$
4) If $\rho_{max} \neq null$ then
   $e_{new} \leftarrow \theta(|\rho_{max}| + 1)$
   $\rho_{new} \leftarrow \rho_{max}||e_{new}$
   Else $\rho_{new} \leftarrow \theta(1)$
5) Add ρ_{new} to λ then
6) Remove the first $|\rho_{new}|$ elements from θ
7) If $\theta \neq \{\varnothing\}$ then go to 2
   Else: *Training finished*

The initially empty pattern library is populated by traversing θ_β, a detailed description of this training procedure is found in Algorithm 1. Two constants control the result: The *creation threshold* H controls how frequently the algorithm creates completely new patterns, and the *error tolerance* σ is a real number between 0 and 1, balancing pattern length against correctness. A small σ produces many short patterns, resulting in an algorithm less prone to fall into false interpretations, but also with limited ability to recognize temporal patterns.

After training, S-Comparison can be used to define the f_β function, and hence the activity level for each position in θ. In the same way as during the training phase, a similarity measure δ is computed for each $\rho \in \lambda$ given a set of past events, i.e., all elements in θ up to time t. f_β are defined as

$$f_\beta(\theta, t) = \begin{cases} \frac{2}{\pi} \arctan(\delta_{max}/d) : & if\ \delta_{max} > 0 \\ 0 : & otherwise \end{cases} \quad (7)$$

where d denotes the dimensionality of θ. Since the similarity measure δ is arbitrary and depends on the amount of training data, the creation level, and the error tolerance, it has no obvious maximum value. For this reason, the $\arctan$ function is used as a squashing function to keep the activity levels between 0 and 1.

IV. Experimental setup

The three methods for behavior recognition presented in Section III were evaluated using a Khepera robot from K-Team [9]. As discussed earlier, the *correct* way to generalize any demonstration depends on, among other things, which skills are available. In the present work, five skills were used, which all produce common movement behaviors: FLW - *Drive along a wall on left side*, FRW - *Drive along a wall on right side*, AVOID - *Go straight ahead but avoid obstacles*, CORRIDOR - *Drive in a narrow corridor without hitting the walls*, and SLALOM - *A slalom drive around circular cones*. Each skill was first demonstrated manually using a standard keyboard interface as remote control. The physical test environment can be seen in Figure 1. Values from the eight infrared proximity sensors constituted the sensor vector x_t, while the speed of the two wheels constituted the action vector y_t. None of the presented recognition methods assumes that the skills are created from a single demonstration, and consequently θ_β should be understood as a general notation for all demonstrations of a specific β. However, in the present work, each skill is created from a single demonstration with a length of about 4000 samples. All values were rescaled to a number between 0 and 1 and logged at about 10 Hz.

Figure 1. Experimental setup. During both training and test sessions, the Khepera robot was placed in a large rectangular box with movable walls and cones, creating a steady and well controlled environment. Sensor readings and motor commands were recorded while the Khepera was remotely controlled, demonstrating one or several behaviors. The present image was taken during demonstration of the Slalom test case, cf. Figure 3.

To ensure that each log file contained all information necessary to achieve the specific goal, one neural network for each skill was created. Each network was trained on its corresponding log file, using the proximity values and wheel speeds as input and output data, respectively. The networks had one hidden layer with five nodes. After training, each network was used as controller β (See Section III) of the robot, which was then able to repeat the corresponding demonstrated behavior.

The three behavior recognition methods were evaluated using two test cases. The first test case, referred to as the *L-demonstration*, involved controlling the robot from start to goal, along the dashed line in Figure 2. Given the skills listed above, it should be generalized into FLW t=0 to 140, FRW t=140 to 215, CORRIDOR t=215 to 320 and finally FLW t=320 to 390.

The second test case, named *Slalom-demonstration*, involves

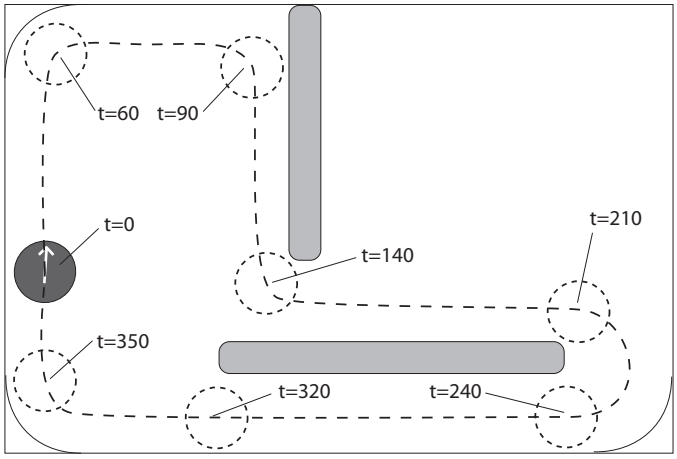

Figure 2. L-demonstration

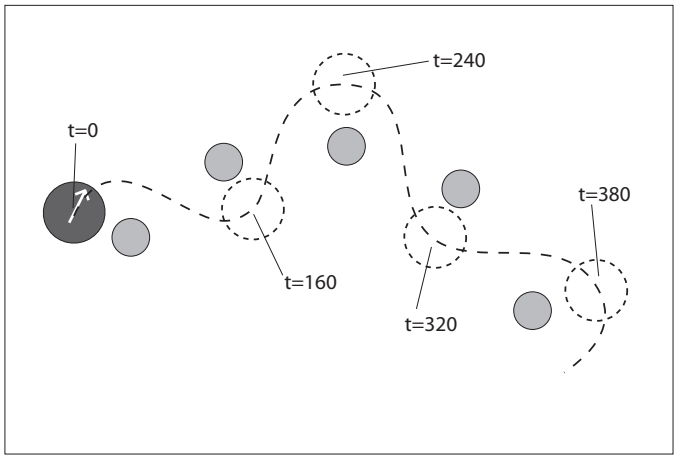

Figure 3. Slalom-demonstration

driving zigzag through a five-cone track, as illustrated in Figure 3. This demonstration differs in several ways from the demonstration used to create the SLALOM skill, both in number of cones and their relative positioning. The demonstration could be understood as an instance of the SLALOM skill, or as a sequence of FLW, FRW, FLW, and so on.

The dark gray circles in Figures 2 and 3 mark the robot's initial position, and the dashed circles mark key positions with the corresponding time stamps. Walls and obstacles in the environment are illustrated as light gray areas. Note that the dashed trajectory simply is a coarse illustration of the robots motion, the real trajectory is significantly more jagged due to the binary behavior of the keyboard control.

For the $\beta - Comparison$, the neural network controllers created from respective θ_β, as described above, were used. All five skills used in the present work are speed invariant in the sense that the goal does not depend on the speed of the robot. Inspired by the work of Olenderski and co-workers [17], the difference function d (Equation 5) is defined as the absolute difference in turning rates:

$$d\left(\beta\left(x_t\right), y_t\right) = |w_t - v_t| \tag{8}$$

where w_t represents the turning rate produced by $\beta(x_t)$ and v_t represents the turning rate observed in θ at time t. The turning rate is given by the difference in speed between the left and right wheels of the Khepera robot.

The AANN-Comparison and S-Comparison are performed by first training the algorithms with each θ_β. Each AANN has one hidden layer with five nodes, and is trained for 100 epochs, (very similar results are achieved for 50, 150 and 200 epochs). Neither AANN nor S-Comparison distinguishes between sensors and actions, and the action vector is not converted to a turning rate as in β-Comparison.

V. Results

The estimated activity levels produced for the two test cases are visible in Figure 4 to 9.

A. Results from β-Comparison

When looking at the results for the two test cases it is obvious that the β-Comparison approach has problems. The results from the L-demonstration, plotted in Figure4, show roughly correct estimations of FLW for t=20 to 90, FRW for t=130 to 210 and CORRIDOR for 150 to 320. However, during parts of the demonstration, β-Comparison gives the AVOID and SLALOM skills higher activity levels than any of the other, although none of these behaviors is part of the demonstration.

The reason for this poor performance can primarily be derived from the fact that β-Comparison only compares action vectors. For example, when driving along a wall on the left side, β-Comparison has no information about the high values on the leftmost proximity sensor. The algorithm only measures the difference in turning rate between the observed and produced data. When the robot is closer to the wall than the FLW controller is configured for, the controller returns a right turn to increase the distance to the wall. However, the human demonstrator might be more tolerant, accept the current distance to the wall, and consequently not find it necessary to turn. As a result, β-Comparison recognizes a relatively large difference in turning rate, and a low activity level is returned, even though the episodes from the demonstrator's point of view is very similar. This is the reason for the low FLW ratings for t=90 to 140.

B. Results for AANN-Comparison

The activity level computed by AANN-Comparison varies a lot between the different skills. In the L-demonstration (Figure 5), FLW and FRW receive activity levels around 0.8 during their respective parts of the demonstration, wile the CORRIDOR skill is only given an activity level of about 0.1 during the period where it should receive the highest levels. However, when looking only at the maximum levels at each time, the correct skill is identified during almost the entire demonstration. In the Slalom-demonstration (Figure 8), a sequence of FRW, FLW, FRW, FLW, and finally FRW is identified. This is, if not the best, at least a reasonable interpretation of the demonstration.

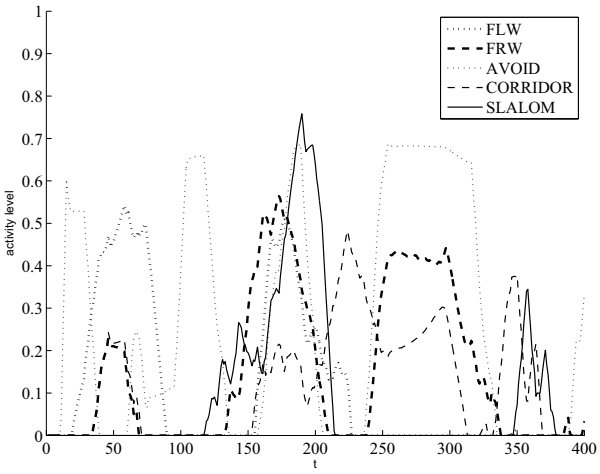

Figure 4. Recognition of the L-demonstration using β-Comparison

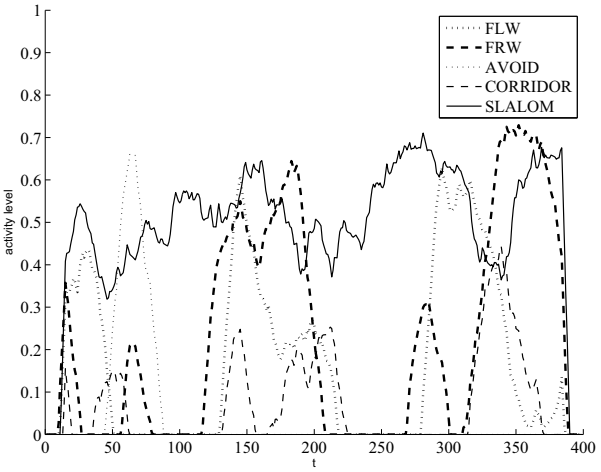

Figure 7. Recognition of the Slalom-demonstration using β-Comparison

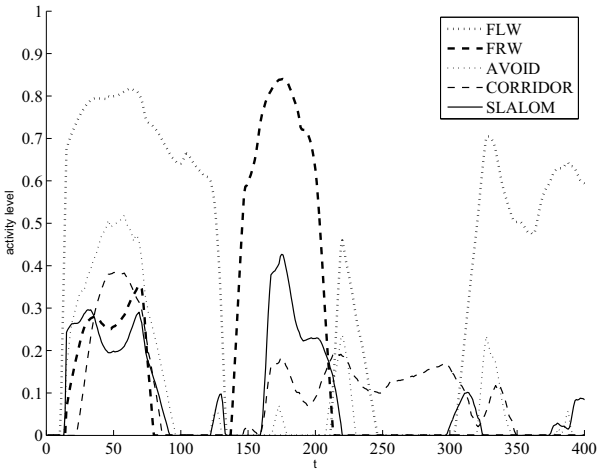

Figure 5. Recognition of the L-demonstration using AANN-Comparison

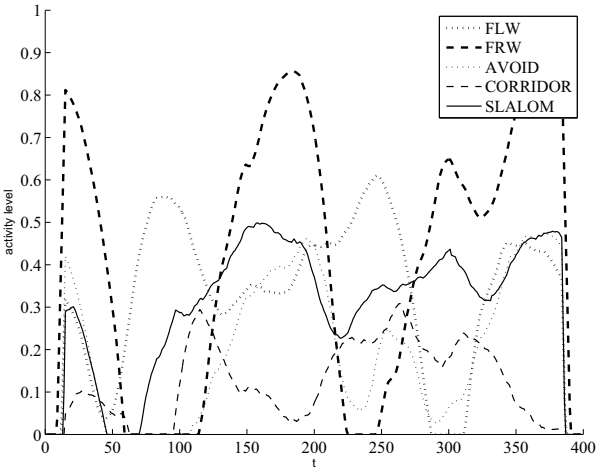

Figure 8. Recognition of the Slalom-demonstration using AANN-Comparison

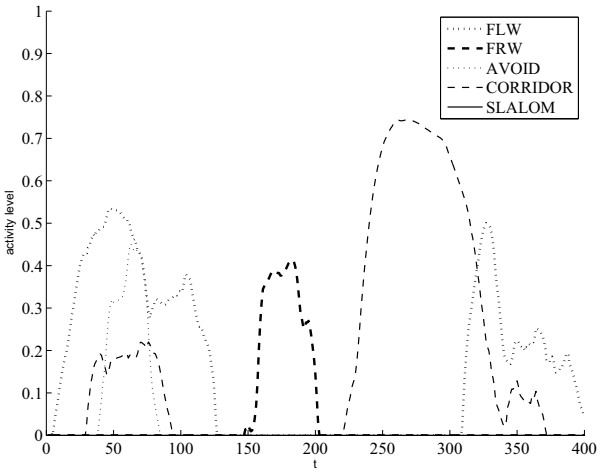

Figure 6. Recognition of the L-demonstration using S-Comparison

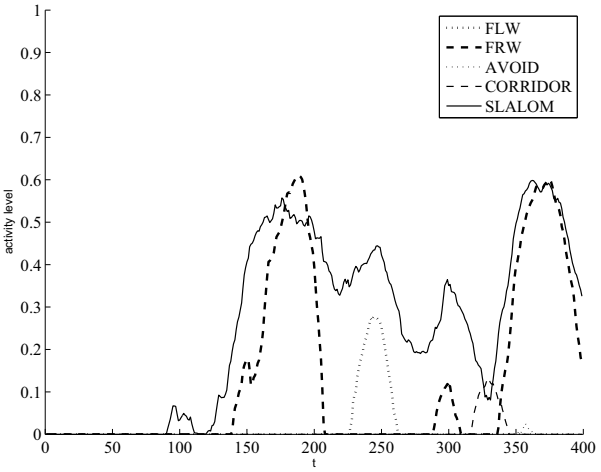

Figure 9. Recognition of the Slalom-demonstration using S-Comparison

C. Results for S-Comparison

S-Comparison is the most restrained comparison method, and causes significantly fewer false alarms than the other methods, in the sense that it does not give activity levels over 0 to behaviors that clearly do not belong to the demonstrated data. With the exception of relatively high ratings for the AVOID and CORRIDOR skills early in the L-demonstration (Figure 6), inappropriate skills are never given an activity level over 0.1 in either of the two test cases.

However, even appropriate skills receive relatively low activity values compared to the other recognition methods. This can be, at least to some extent, explained by the fact that the activity levels computed by S-Comparison are arbitrary, see Section III-C for details.

VI. Related Work

Skills and segmentation points in demonstrated sequences can be identified in several ways. Segmentation points may be identified by statistical features in data, for example thresholding the variance for certain sensor modalities [11], [19], thresholding the mean velocity of joints [7], [14], or entropy measures [5]. Another way is to observe the outcome of the robot actions. In [16], segmentation points are identified by constantly matching current sensory states with post-conditions for all pre-programmed skills. Once a post-condition is matched, both segmentation point and skills are identified. Other techniques try to directly identify the skills in the demonstrated data. Bentivegna [3] uses a nearest-neighbor classifier on state data to identify skills in a marble maze task. Pook and Ballard [20] present an approach where sliding windows of data are classified using Learning Vector Quantization in combination with a k-nn classifier.

To compare the approaches [11], [19], [7], [14], [5], [16], [3], [20] mentioned above with the work presented in this paper, it is important to first observe that the complexity of the skills is a crucial factor when choosing techniques for segmentation and skill identification. Approaches that look for general statistical features in data to detect segmentation points are not sufficient for the high-level behaviors that we are using. Our second test case, presented in Section IV, shows how both the location of segmentation points, and the identity of the actual skills depend on much more than the fluctuations in data. The Slalom demonstration can be understood as both an instance of the SLALOM primitive, and a sequence of FLW and FRW primitive, and the segmentation points have to be placed differently for these two cases. The activation levels suggested in this paper serve as a tool to deal with this ambiguity, e.g. by behavior arbitration or parallel behaviors, based on fuzzy logic, for example.

Nicolescu's approach using post-conditions [16] focuses on the segmentation points and ignores the actual behavior. This works well for behaviors where the goal determines the wanted behavior completely, but would fail with other types of behaviors.

In contrast, the sliding window k-nn classifier presented by Pook and Ballard [20] focuses directly on the identification of skills, and is in this sense similar to our approach. In fact, S-Comparison can be understood as a 1-nn classifier with a dynamic sliding window, even though it has not been used as a classifier in this case .

VII. Conclusions and Future Work

We have developed and evaluated three different techniques for behavior recognition in an LFD setting. All techniques compute an activity level which can be seen as an alternative to a pure classification approach. The examples show how the former approach allows a more informative interpretation of a demonstration, by not determining "correct" behaviors, but rather a number of alternative interpretations. As shown in Figure 8 for example , there is no reason to claim that a sequence of FLW, FRW, FLW, ... is more, or less "correct" than the SLALOM skill. The final decision of how to interpret the observed event sequence should be left to higher cognitive levels in a final LFD system. This decision depends, among other things, on the meaning of "generalization", which we intend to deal with in our future research.

The three presented techniques differ in the way they utilize the demonstrated data for behavior recognition. β-Comparison focuses on actions and ignores the input part entirely. As discussed in Section V, this leads to unavoidable problems. The technique is not suitable for segmentation purposes, but the average activity levels for an entire demonstration may be useful for work with behavior fusion, such as described in [17].

AANN-Comparison models the sensory-motor space for each skill and performs recognition based on how well the demonstrated data fits into these models. The results are clearly better than for β-Comparison, both in terms of skill identification and localization of segmentation points. This can be explained by AANN-Comparison using the sensor vectors directly in the comparison process, and consequently it has much more information available than does β-Comparison.

S-Comparison uses most information of the three evaluated algorithms. Similarly to AANN-Comparison, it treats sensors and actuators as a single event vector. In addition, it models temporal patterns in the event stream. However, the modeling power of S-Comparison does not show up as increased performance in the results, compared to AANN-Comparison. S-Comparison is less noisy than the other two methods and has both fewer false positives (high activity level where it should be low) but also more false negatives (low activity level where it should be high) compared with AANN-Comparison. This is clearly visible in the first part of the Slalom test case, Figure 9.

Furthermore, S-Comparison fails to identify exact positions of the segmentation points. This can be seen as a direct consequence of the temporal dimension of S-Comparison. Since S-Comparison has not been trained on data describing the transitions between different skills, such periods yield a low similarity measure. As seen in Figures 6 and 9, this problem shows up as gaps in the activity level curves. Similar problems have been reported by Pook and Ballard [20], evaluating their window-based approach.

Even though some effort has been put on comparing these techniques, this work should be seen as an attempt to eval-

uate a concept of behavior recognition, rather than test the exact performance of presented algorithms. Furthermore, the problem of identifying characteristics of segmentation points required to autonomously repeat a demonstrated behavior is not addressed here. However, by identifying the location of the segmentation points, we have drastically narrowed down the problem.

Both skills and task level representations [16] are created in this approach from manual demonstrations. A thrilling possibility is to use the task level representations, i.e., sequences of skills, as primitives in even more complex tasks. This would allow the robot to reuse its experience, both from manual demonstrations and successful automatic drives. These issues will be subject to future work.

The present work should also be seen as a step towards a developed interaction between the robot and its user. The presented methods transform a complex, multi-dimensional stream of data into a relatively simple sequence of named skills, easily read and understood by a human user. From an interaction perspective, this feature appears as one of the strongest motivations behind the use of previously learned skills in LFD, and possibly also in other areas of robotics.

REFERENCES

[1] R. Amit and M. Mataric. Parametric primitives for motor representation and control, 2002.

[2] C. G. Atkeson and S. Schaal. Learning tasks from a single demonstration. In *In Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997.

[3] Darrin C. Bentivegna. *Learning from observation using primitives*. PhD thesis, 2004. Director-Christopher G. Atkeson.

[4] E.U. Braun, H. Mayer, I. Nagy, A. Knoll, S.M. Wildhirt, R. Lange, and R. Bauernschmitt. An instrumentation system with force feedback, automatic recognition and skills for cardiac telemanipulation. In *Proceedings 33rd Annual International Conference of Computers in IEEE Comp Cardiol*, volume 33, pages 553–556, 2006.

[5] Paul Cohen, Niall Adams, and Heeringa Brent. Voting experts: An unsupervised algorithm for segmenting. *To appear in Journal of Intelligent Data Analysis*.

[6] K. I. Diamantaras and S. Y. Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[7] A. Fod, M. Mataric, and O. Jenkins. Automated derivation of primitives for movement classification, 2000.

[8] Thomas Hellström. Teaching a robot to behave like a cockroach. In *Proceedings of the Third International Symposium on Imitation in Animals and Artifacts in Hatfield UK*, pages 54–61, 2005.

[9] K-Team. Khepera ii mobile robot. www.k-team.com, 2007.

[10] N. Koenig and M. J. Matarić. Demonstration-based behavior and task learning. Working Notes, AAAI Spring Symposium, 2006.

[11] Nathan Koenig and Maja J Matarić. Behavior-based segmentation of demonstrated tasks. In *In International Conference on Development and Learning (ICDL),*, Bloomington, IN., May 2006.

[12] Paul Martin and Ulrich Nehmzow. Programming by teaching: Neural network control in the manchester mobile robot. In *Proceedings Intelligent Autonomous Vehicles*, 1995.

[13] J. Nakanishi, J. Morimoto, G. Endo, S. Cheng, G. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:2–3:79Ű81, 2004.

[14] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi. Recognition and generation of leg primitive motions for dance imitation by a humanoid robot, 2003.

[15] Monica Nicolescu and Maja Matarić. Linking perception and action in a control architecture for human-robot domains. In *Thirty-Sixth Hawaii International Conference on System Sciences, HICSS-36*, Hawaii, USA, January 2003.

[16] Monica Nocolescu. *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*. PhD thesis, University of Southern California, 2003.

[17] Adam Olenderski, Monica Nicolescu, and Sushil Louis. Robot learning by demonstration using forward models of schema-based behaviors. In *Proceedings of the Second International Conference on Informatics in Control, Automation, and Robotics.*, volume 3, pages 263–26, 2005.

[18] J. Peters and S. Schaal. Policy learning for motor skills. In *Proceedings of 14th international conference on neural information processing (iconip)*, 2007.

[19] Richard Alan Peters II and Christina L. Campbell. Robonaut task learning through teleoperation. In *Proceedings of the 2003 IEEE, International Conference on Robotics and Automation*, pages 23 — 27, Taipei, Taiwan, September 2003.

[20] Polly K. Pook and Dana H. Ballard. Recognizing teleoperated manipulations. In *ICRA (2)*, pages 578–585, 1993.

[21] B. Rohrer and S. Hulet. Becca – a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Sandria National Laboratories, Alberquerque, NM, USA, 2006.

[22] B. Rohrer and S. Hulet. A learning and control approach based on the human neuromotor system. In *Biomedical Robotics and Biomechatronics, 2006. BioRob.*, pages 57–61, February 2006.

[23] H. Urbanek, A. Albu-Schäffer, and P. Smagt van der. Learning from demonstration repetitive movements for autonomous service robotics. In *IROS 2004 IEEE RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, Sept. 28-Oct.2, 2004*, 2004. LIDO-Berichtsjahr=2004.