2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Behavior Recognition for Learning from Demonstration

Erik A. Billing
Department of Computing Science
Umeå University
Umeå, Sweden
Email: billing@cs.umu.se

Thomas Hellström
Department of Computing Science
Umeå University
Umeå, Sweden
Email: thomash@cs.umu.se

Lars-Erik Janlert
Department of Computing Science
Umeå University
Umeå, Sweden
Email: lej@cs.umu.se

***Abstract*—Two methods for behavior recognition are presented and evaluated. Both methods are based on the dynamic temporal difference algorithm *Predictive Sequence Learning (PSL)* which has previously been proposed as a learning algorithm for robot control. One strength of the proposed recognition methods is that the model PSL builds to recognize behaviors is identical to that used for control, implying that the controller (inverse model) and the recognition algorithm (forward model) can be implemented as two aspects of the same model. The two proposed methods, *PSLE-Comparison* and *PSLH-Comparison*, are evaluated in a Learning from Demonstration setting, where each algorithm should recognize a known skill in a demonstration performed via teleoperation. PSLH-Comparison produced the smallest recognition error. The results indicate that PSLH-Comparison could be a suitable algorithm for integration in a hierarchical control system consistent with recent models of human perception and motor control.**

***Index Terms*—Learning and Adaptive Systems, Neurorobotics, Autonomous Agents**

I. Introduction

In previous work [1], we present the dynamic temporal difference algorithm *Predictive Sequence Learning (PSL)* and apply it to a *Learning from Demonstration (LFD)* problem. In this application, PSL builds a model from a set of demonstrations, i.e., sequences of sensor and motor events recorded while a human teacher performs the desired task by teleoperating the robot. After training, PSL can be used to control the robot by continually predicting the next action based on the sequence of passed sensor and motor events.

PSL has many interesting properties seen as a learning algorithm for robots. It is model and parameter free, meaning that it introduces very few assumptions into learning and does not need any task specific configuration. Knowledge is stored in a *hypothesis library* H, where each *hypothesis* $h \in H$ describes a relation between a sequence of events $X = \left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)$ and a target event $Y = e_{t+1}$. $|h|$ denotes the length of h as the number of elements in X. PSL treats control as a prediction problem, and creates longer h when it fails to predict the next event. This corresponds to a dynamically growing state space similar to a *Variable order Markov Model (VMM)*. Hypotheses are only created when predictions fail, meaning that the learning rate is proportional to the prediction error and PSL will stop to learn in domains where it can predict future events perfectly.

Our evaluation of PSL indicates that the algorithm is suitable for learning problems up to a certain complexity. However, PSL is subject to combinatorial explosion and fails to reproduce more complex behavior properly. Specifically, PSL has problems capturing long-term variations within a behavior and some kind of higher level coordination is clearly necessary in these situations.

The design of PSL is inspired by several computational models of the human brain, MOSAIC [2], [3], [4], Predictive Coding [5], [6], [7], and Hierarchical Temporal Memory [8], [9]. These models propose a hierarchical organization of perception and control. Specifically, MOSAIC presents a modular view of central nervous system, where each module implements one *forward model* and one *inverse model*. The forward model predicts the sensory consequences as a result of a motor command, while the inverse model calculates the motor command that, in the current state, leads to the goal [4]. Each module works under a certain *context,* or bias, provided by higher ordinate modules in the hierarchy. One purpose of the forward model is to create a responsibility signal λ_β representing a measure of how well the present activity corresponds to the module's context. If the prediction error is small, the activity is familiar and λ_β is high. However, if the prediction error is large, the activity does not correspond to the module's context, and actions produced by the inverse model may be inappropriate. An overview of these approaches to intertwined control and perception for LFD is found in our previous work [10].

Placing PSL as a module in this kind of hierarchical structure could constitute one way to solve the problems with combinatorial explosion. In such an architecture, each PSL module would work under a certain context and only model the system variables that change quickly, while slower temporal dynamics are handled higher up in the hierarchy. However, this requires not only that PSL is useful as an inverse model, but also that it can constitute a forward model, able to compute λ_β. On the way to propose a fully developed hierarchical system based on the PSL algorithm, the present work proposes two ways of applying PSL as a forward model

and evaluates how well each approach is able to recognize a certain activity, a problem that within LFD is known as *behavior recognition*.

One of the few robot control architectures that employs this kind of organization is HAMMER [11], [12], [13], which focuses on direction of attention during action recognition. While HAMMER is in many respects further developed that the work presented here, it implements hard-coded forward models paired with inverse models. In the present work, both forward and inverse models are generated from demonstrated data in a model-free way.

The rest of this paper is organized as follows. Section II presents a short background to behavior recognition and introduces some of our earlier research relevant for the present work. Section III gives a detailed description of PSL, which is developed into the proposed methods for behavior recognition, presented in Section IV. Experimental setup and results from the conducted evaluation is presented in Section V. Finally, conclusions, limitations and future work are discussed in Section VI.

II. Behavior recognition for LFD

Recent work in LFD is often concerned with identification and selection of *behavior primitives,* or *skills,* which can be seen as simple controllers and correspond to larger parts of the demonstration [14]. Behavior primitives implement hard-coded or previously learned behaviors that the robot can execute. By matching these primitives with a demonstration, selected primitives can be compiled into a new, more complex, controller that will be able to repeat the demonstrated behavior under varying environmental conditions [15], [14].

This approach transforms the general LFD process into the three activities of *behavior segmentation*, *behavior recognition* and *behavior coordination* [16]. Behavior segmentation refers to the process of dividing the observed event sequence into segments which can be explained by a single primitive. Behavior recognition is the process of matching each segment with a primitive. Finally, behavior coordination involves identifying switching criteria that control when the robot should switch between different primitives. Identification of switching criteria corresponds to finding sub-goals in the demonstrated behavior.

The behavior recognition problem is closely related to the creation of a *metric of imitation performance* which is a common concept in the literature on LFD and imitation learning, e.g. [17], [18], [15]. The metric acts as a cost function for imitation of a skill and is in this sense very similar to the computation of a responsibility signal. Identification of a metric of imitation performance is often focused on finding the critical components of a skill by identifying invariants within a set of demonstrations. One promising approach to construct such a metric is to use the demonstrations to impose constraints in a dynamical system [19], [20]. However, we take an alternative approach: Using forward models to compute a measure of how well observed events correspond to respective controller. Both behavior recognition algorithms presented here compute λ_β as a direct or indirect function of prediction error. We believe that this approach has larger potential to provide a generalizable solution to the behavior recognition problem and it also has many interesting connections to neurological models. A longer discussion of these issues can be found in [21].

Another important aspect of the metric of imitation performance is to solve the correspondence problem, i.e., comparing actions when the body of the teacher is different from that of the student. This problem does not exist in LFD using teleoperation and is not considered in the present work.

There are many possible ways to demonstrate new behavior to a robot. Good overviews can be found in [15], [17]. In the present work, we focus on demonstrations performed by controlling the robot via teleoperation such that it performs the desired behavior. In this case, a demonstration is a sequence of sensor and motor events $\eta = (u_1, y_1, u_2, y_2 \ldots, u_\kappa, y_\kappa)$, where κ denotes the most recent stage. An *observation* $y_k \in Y$ is defined as the combination of all sensors readings and an *action* $u_k \in U$ is defined as the combination of the motor commands sent to the robot. The *observation space* Y and *action space* U constitute the complete set of possible events, known as an *event alphabet* $\sum = Y \cup U$. A *stage* (u_k, y_k) comprises one action and the directly following observation. In some situations we do not distinguish between observations and actions and define an event sequence $\eta = (e_1, e_2, \ldots, e_t)$ as a sequence of discrete events $e \in \sum$ up to the current time t.

In earlier work, we have developed and evaluated three methods for behavior recognition [1]. The focus of the work was to propose methods for constructing several interpretations from a single sequence of events, using the set of known skills. Seen as a pure classification problem, one skill would have to be selected as the one best representing that segment in η. However, in our methods, all recognition algorithms produces *activity level* λ_β for each skill β. The activity level is in this context identical to the notion of a *responsibility signal* λ_β, as discussed in the introduction.

While one of the evaluated recognition techniques showed clear limitations, the other two, known as *AANN-Comparison* and *S-Comparison*, showed promising results. AANN-Comparison is based on a set of Auto-Associative Neural Networks, one for each skill β. The network's reconstruction error for each stage (u_k, y_k) from the demonstration η is used as a measure of $\lambda_\beta(k)$.

S-Comparison is based on S-Learning, a prediction-based control algorithm inspired by the human neuro-motor system [22], [23]. S-Learning is a dynamic *temporal difference* (TD) algorithm able to extract temporal patterns ρ in presented data. S-Comparison computes a similarity measure $\delta_\rho(k)$ for each pattern ρ, and uses the highest similarity value $\delta_{max}(k)$ to compute $\lambda_\beta(k)$. Details of both S-Comparison and AANN-Comparison are found in [1].

III. Predictive Sequence Learning

PSL is trained on an *event sequence* $\eta = (e_1, e_2, \ldots, e_t)$, where each *event* e is a member of an alphabet $\sum$. η is

defined up to the current time t from where the next event e_{t+1} is to be predicted.

PSL stores its knowledge as a set of hypotheses, known as a *hypothesis library* H. A *hypothesis* $h \in H$ expresses a dependence between an event sequence $X = (e_{t-n}, e_{t-n+1}, \ldots, e_t)$ and a target event $I = e_{t+1}$:

$$h : X \Rightarrow I \tag{1}$$

X_h is referred to as the *body* of h and I_h denotes the *head*. Each h is associated with a *confidence* c reflecting the conditional probability $P(I|X)$. For a given η, c is defined as $c(X \Rightarrow I) = s(X, I)/s(X)$, where the *support* $s(X)$ is the proportion of transactions in η that contains X. (X, I) denotes the concatenation of X and I. A transaction is defined as a sub-sequence of the same size as X, occurring after the creation of h. The length of h, denoted $|h|$, is defined as the number of elements in X_h. Hypotheses are also referred to as *states*, since a hypothesis of length $|h|$ corresponds to VMM state of order $|h|$.

A. Detailed description of PSL

Let the library H be the empty set of hypotheses. During learning, described in Algorithm 1, PSL tries to predict the future event e_{t+1}, based on the observed event sequence η. If the prediction is wrong, a new hypothesis h_{new} is created and added to H. h_{new} is one element longer than the longest hypothesis $h_c \in H$ that would have produced a correct prediction (see Algorithm 1 for an exact description of how h_c is selected). In this way, PSL grows the library only when it produces incorrect predictions.

For example, consider the event sequence $\eta = ABCCABCC$. Let $t = 1$. PSL will search for a hypothesis with a body matching A. H is initially empty and consequently PSL will not be able to perform a prediction. Instead, PSL creates a new hypothesis $(A) \Rightarrow B$ which is added to H. The same procedure will be executed at $t = 2$ and $t = 3$ so that $H = \{(A) \Rightarrow B; (B) \Rightarrow C; (C) \Rightarrow C\}$. At $t = 4$, PSL will find a matching hypothesis $h_{max} : (C) \Rightarrow C$ producing the wrong prediction C. Consequently, a new hypothesis $(C) \Rightarrow A$ is added to H. The predictions at $t = 5$ and $t = 6$ will be successful while $h : (C) \Rightarrow A$ will be selected at $t = 7$ and produce the wrong prediction. As a consequence, PSL will create a new hypothesis $h_{new} : (B, C) \Rightarrow C$. PSL has now learned the pattern and will not add any more hypotheses to H until it observes another η containing elements that do not follow this pattern.

Algorithm 1 Predictive Sequence Learning (PSL)

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_n)$

1: $t \leftarrow 1$
2: $H \leftarrow \emptyset$
3: $M \leftarrow \left\{h \in H \mid X_h = \left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)\right\}$
4: **if** $M = \emptyset$ **then**
5: **let** $h_{new} : (e_t) \Rightarrow e_{t+1}$
6: add h_{new} to H
7: **goto** 20
8: **end if**
9: $\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ for\, all\, h' \in M\}$
10: **let** $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ for\, all\, h' \in \hat{M}\right\}$
11: **if** $e_{t+1} \neq I_{h_{max}}$ **then**
12: **let** h_c be the longest hypothesis $\{h \in M \mid I_h = e_{t+1}\}$
13: **if** $h_c = null$ **then**
14: **let** $h_{new} : (e_t) \Rightarrow e_{t+1}$
15: **else**
16: **let** $h_{new} : \left(e_{t-|h_c|}, e_{t-|h_c|+1}, \ldots, e_t\right) \Rightarrow e_{t+1}$
17: **end if**
18: add h_{new} to H
19: **end if**
20: update the confidence for h_{max} and $h_{correct}$ as described in Section III
21: $t \leftarrow t + 1$
22: **if** $t < n$ **then**
23: **goto** 2
24: **end if**

IV. Methods for behavior recognition

Two methods based on the PSL algorithm are here presented. The first method, *PSLE-Comparison*, is inspired by the HMOSAIC architecture [2] and computes the responsibility signal λ_β as an inverse function of the normalized prediction error, produced by PSL. The second method, *PSLH-Comparison*, is more closely built on the PSL algorithm. λ is in this method a function of hypothesis activation match.

A. PSLE-Comparison

The responsibility signal $\lambda_\beta(t)$ of skill β at time t is given by:

$$\lambda_\beta(t) = \sum_{i=t-\nu}^{t} \frac{1 - \Delta_i^\beta}{\nu} \tag{2}$$

where ν is a constant describing the temporal extension of the behavior, i.e., λ_β is defined as an average of prediction performance over ν time steps. The *prediction error* Δ_i^β is given by:

$$\Delta_i^\beta = \begin{cases} 0 & if\, e_i = \hat{e}_i^\beta \\ 1 & otherwise \end{cases} \tag{3}$$

where $\hat{e}_i^\beta$ is the output of the forward model at position i in η_β.

By training a PSL library H_β on each event sequence η_β, one forward model for each skill β is created. The precise training procedure is described in Algorithm 1. The event sequence η_β used for training is a demonstration of skill β. In practice, several demonstrations of each skill may of course be used, but for simplicity we here consider them to be a single sequence of events.

After training, H_β is used for prediction as described in Algorithm 2. In principle, any discrete prediction algorithm

Algorithm 2 Making predictions using PSL

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_{t-1})$
Require: the trained library $H = \left(h_1, h_2, \ldots, h_{|H|}\right)$

1: $M \leftarrow \left\{h \in H \mid X_h = \left(e_{t-|h|}, e_{t-|h|+1}, \ldots, e_{t-1}\right)\right\}$
2: $\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ for\, all\, h' \in M\}$
3: **let** $h_{max} \in \left\{h \in \hat{M} \mid c\,(h) \geq c\,(h') \ for\, all\, h' \in \hat{M}\right\}$
4: **return** the prediction $\hat{e}_t = I_{h_{max}}$

could be used as forward model, but an advantage of PSL is that the same algorithm constitutes both forward and inverse model. As enforced by the MOSAIC framework [24], [3], the forward and inverse models should be paired, meaning that the forward model should be able to predict the consequences of actions produced by the inverse model. This pairing is built into PSL, since the prediction and control is actually performed by the same model.

B. PSLH-Comparison

One problem with comparison methods based directly on prediction error was observed during our previous investigation of methods for behavior recognition [1]. Prediction error can be seen as a measure of how consistent an event sequence η is with some skill β. However, the prediction error does not tell whether η demonstrates all aspects of β, or only a fraction of these aspects.

In an attempt to approach this problem, *PSLH-Comparison* was designed. PSL is here used to create a single library H from skill demonstrations η_β of all β. $\lambda_\beta\,(t)$ is defined as the intersection between the hypotheses activated during the demonstrations of β, and the hypotheses activated by η within the time span $t - \nu$ to t:

$$\lambda_\beta\,(t) = \frac{\sum\limits_{h \in H} \min\left(a_h^{\eta_\beta}, a_h^{\eta_t}\right)}{\sum\limits_{h \in H} a_h^t} \tag{4}$$

where $a_h^{\eta_\beta}$ = *hAct*(η_β, H, h_η, 1, $|\eta_\beta|$) and $a_h^{\eta_t}$ = *hAct*(η, H, h_η, $t-\nu$, t). Equation 4 is the Bayes P_e, the minimum error probability between the two hypothesis activation distributions [25]. The hypothesis activation function *hAct* is defined in Algorithm 3, calculating the prediction contribution for a specific hypothesis h_η, given a certain time interval in η. Similarly to PSLE-Comparison, ν is a constant describing the temporal extension of the skill. The minimum error probability gives reward for hypotheses that are activated in both behaviors (similar to inverted prediction error), but also gives penalty for hypotheses that are only activated by one of the event sequences η or η_β.

Algorithm 3 Hypothesis activation

function *hAct*(η, H, h_η, t_{start}, t_{stop})

1: $t \leftarrow t_{start}$
2: $a_h \leftarrow 0$
3: $M \leftarrow \left\{h \in H \mid X_h = \left(e_{t-|h_\eta|}, e_{t-|h_\eta|+1}, \ldots, e_{t-1}\right)\right\}$
4: $M' \leftarrow \{h \in M \mid I_h = e_t\}$
5: $\hat{M} \leftarrow \{h \in M' \mid |h| \geq |h'| \ for\, all\, h' \in M'\}$
6: **let** $h_{max} \in \left\{h \in \hat{M} \mid c\,(h) \geq c\,(h') \ for\, all\, h' \in \hat{M}\right\}$
7: **if** $h_{max} = h$ **then**
8: $\quad a_h \leftarrow a_h + \frac{1}{t_{stop} - t_{start}}$
9: **end if**
10: $t \leftarrow t + 1$
11: **if** $t \leq t_{stop}$ **then**
12: $\quad$ **goto** 3
13: **end if**

V. Evaluation

The two proposed methods for behavior recognition, PSLE-Comparison and PSLH-Comparison, are here tested in an LFD setting using a Khepera II miniature robot [26]. A *load-transport-unload* task is defined, consisting of three sub-behaviors or skills. *Skill 1* involves the robot moving forward in a corridor approaching an object (cylindrical wood block). When the robot gets close to the object, it should stop and wait for the human teacher to "load" the object, i.e., place it upon the robot. After loading, the robot turns around and goes back along the corridor. *Skill 2* involves general corridor driving, taking turns in the right way without hitting the walls and so on. *Skill 3* constitutes the "unloading" procedure, where the robot stops in a corner and waits for the teacher to remove the object and place it to the right of the robot. Then the robot turns and pushes the cylinder straight forward for about 10 centimeters, backs away and turns to go for another object. The sequence of actions expected by the robot is illustrated in Figure 1 and the experimental setup can be seen in Figure 2. Even though the setup was roughly the same in all experiments, the starting positions and exact placement of the walls varied between demonstration and repetition.

The robot was given no previous knowledge about itself or its surroundings. The only obvious design bias is the thresholding of proximity sensors into three levels, *far*, *medium* and *close,* corresponding to distances of a few centimeters. This thresholding was introduced to decrease the size of the observation space Y, limiting the amount of training required. An *observation* $y \in Y$ is defined as the combination of the eight proximity sensors, producing a total of 3^8 possible observations. An *action* $u \in U$ is defined as the combination of the speed commands sent to the two motors.

To put the performance of PSLE-Comparison and PSLH-Comparison in a larger context, two of our previously proposed methods for behavior recognition, AANN-Comparison and S-Comparison [1], are included in the evaluation.

All four comparison methods are trained on the same set of demonstrations. Skill 1 is demonstrated seven times for a total of about 2.6 minutes. Skill 2 is demonstrated for about 8.7 minutes and Skill 3 is demonstrated nine times, in total 4.6 minutes. For Task 4, the demonstrations from all three partial tasks were used, plus a single 2 min demonstration of the entire task. Exactly the same set of demonstrations have previously been used for evaluating the PSL algorithm

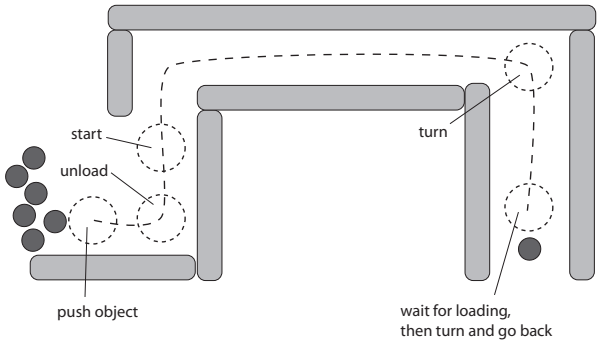

Figure 1. Schematic overview of the *load-transport-unload* task. Light gray rectangles mark walls, dark gray circles mark the objects and dashed circles mark a number of key positions for the robot. The robot starts by driving upwards in the figure, following the dashed line. until it reaches the object at the loading position. After loading, the robot turns around and follows the dashed line back until it reaches the unload position. When the cylinder has been unloaded (placed to the left of the robot), the robot turns and pushes the object. Finally, it backs away from the pile and awaits more instructions.

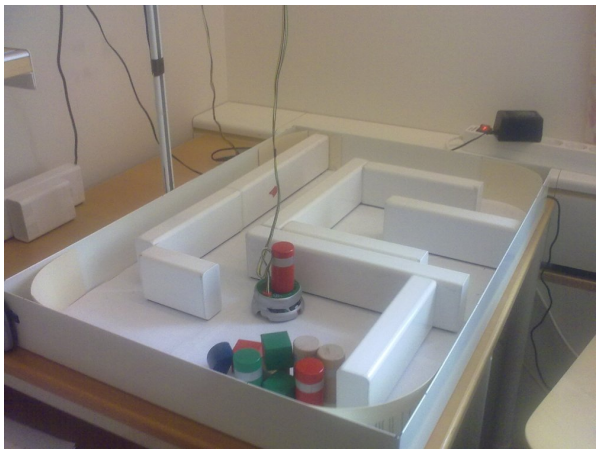

Figure 2. Experimental setup.

as a controller [21]. PSL is then able to repeat each of the three skills successfully, but unable to reproduce the complete *load-transport-unload* task. The reason PSL was unable to repeat the complete behavior is that knowledge from the three skills interfered. The algorithm is unable to separate the unloading activity from the turning, loading from pushing and so on. In these situations, some kind of higher level coordination is needed to prevent knowledge about the wrong activity from interfering. If PSL, when used as an algorithm for behavior recognition, is able to identify the present activity, it should constitute a good basis for building a coordination system separating the different activities into skills, preventing knowledge interference.

Ten demonstrations of the full *load-transport-unload* task are used for testing. A responsibility signal template is defined for each of the demonstrations, specifying which parts of the demonstration that corresponds to respective skill. See Figure 3 for an example template. The templates are manually constructed, based on time-synced video recordings of each demonstration. Parts of the templates contained over-

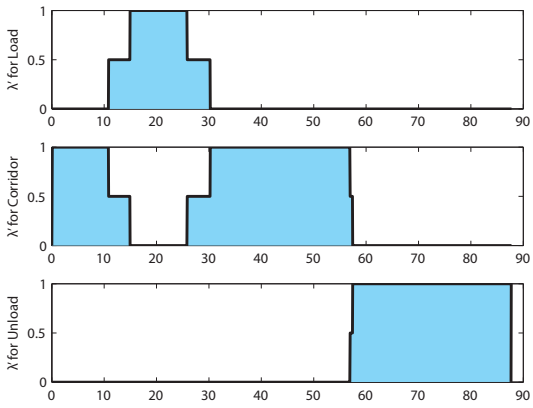

Figure 3. Example template for a single demonstration of the *load-transport-unload* task. The tick black line in top, middle and bottom plots indicates λ' for the Skill 1, 2 and 3, respectively. The green area below the line indicates the parts of the demonstration where respective skill should gain high responsibility. Overlapping periods are normalized such that the sum of activity levels for all skills equals 1.

Table I
AVERAGE RECOGNITION ERRORS AND λ VARIANCE ON THE *load-transport-unload* TASK.

Algorithm	$\tilde{\lambda}$	σ^2_λ	$\tilde{\lambda}_{Load}$	$\tilde{\lambda}_{Corr}$	$\tilde{\lambda}_{Unload}$
AANN-Comparison	0.405	0.027	0.350	0.423	0.442
S-Comparison	0.228	0.030	0.231	0.298	0.155
PSLE-Comparison	0.217	0.050	0.234	0.310	0.107
PSLH-Comparison	0.147	0.036	0.198	0.212	0.032

lapping skills, implying that these segments could have been produced by more than one skill. While manually constructed interpretations of the demonstrations may not constitute the ideal environment for an absolute performance measure, they should still constitute a good frame for comparing the behavior recognition algorithms.

A. Results

Recognition errors for each of the four evaluated algorithms are presented in Table I. $\tilde{\lambda}_\beta(t) = \left|\lambda_\beta(t) - \lambda'_\beta(t)\right|$ is the recognition error at time t, where $\lambda_\beta(t)$ is the computed responsibility signal for skill β. $\lambda'_\beta(t)$ is the desired responsibility signal for β defined by the template. Both $\lambda_\beta(t)$ and $\lambda'_\beta(t)$ are normalized over all three skills.

In Table I, $\tilde{\lambda}$ is the total mean recognition error over all skills. σ^2_λ is the total variance over λ. $\tilde{\lambda}_{Load}$, $\tilde{\lambda}_{Corr}$ and $\tilde{\lambda}_{Unload}$ is the mean recognition error for each of the three skills. All values are normalized averages over 10 demonstrations. A standard t-test shows that both PSLE-Comparison and PSLH-Comparison have significantly smaller $\tilde{\lambda}$ than the other algorithms ($p < 0.005$) and that PSLH-Comparison is significantly better than PSLE-Comparison ($p < 0.005$).

Figure 4, 5 and 6 display the responsibility signals for skill 1, 2 and 3, respectively. Each figure shows both the desired responsibility signal λ', and the signals computed by each of the four recognition algorithms. Displayed values are from the same demonstration as the template signal in Figure 3 (and is consequently not an average over all ten

demonstrations, as opposed to values in Table I).

VI. Discussion

In the present work, two methods for behavior recognition are presented and evaluated. Both methods are based on the dynamic temporal difference algorithm Predictive Sequence Learning (PSL). PSL is both parameter-free and model-free in the sense that no ontological information about the robot or conditions in the world is pre-defined in the system. Instead, PSL creates a state space (hypothesis library) in order to predict the demonstrated data optimally.

The first method, *PSLE-Comparison*, takes inspiration from the MOSAIC architecture [2], [24] and computes the responsibility signal λ_β based on prediction error. The second method, *PSLH-Comparison*, is based on the minimum error probability between activation distributions over model H. PSLH-Comparison was designed to not only compute λ_β as a function of skill match (inverse prediction error) but also include a penalty for aspects of the skill not present in the demonstration.

The two algorithms are compared to two other methods for behavior recognition, *AANN-Comparison* and *S-Comparison* [1]. Performance is measured as the average recognition error. Both PSLE-Comparison and PSLH-Comparison shows significantly smaller recognition error than the other methods. However, the difference between PSLE-Comparison and S-Comparison is small. Overall, PSLH-Comparison is the winner with significantly smaller recognition errors than the other algorithms.

The present evaluation is based on ten demonstrations of a *load-transport-unload* task using a Khepera II robot [26]. This task is selected since the same data has previously been used to evaluate PSL as a controller [21]. In the previous evaluation, it was concluded that the PSL could learn each of the three skills (*load, corridor* and *unload*) but PSL was unable to repeat the overall task. The load-transport-unload task should consequently constitute a setting where some higher level coordination is necessary. Being able to identify each of the three skills in a demonstration of the whole behavior is one step towards creating such a coordination mechanism, allowing PSL to be placed within a hierarchical control architecture such as HMOSAIC [2].

The results show that the proposed recognition methods are significantly better than the benchmark methods used in the evaluation. An overall recognition error of less than 0.15 for PSLH-Comparison is in fact much better than expected. How well the algorithms would do in an on-line situation is however still an open question. The template signal $\lambda'_\beta(t)$ defined as the "correct" responsibility signal for skill β at time t merely reflects the teacher's high-level understanding of the demonstrated behavior, and is not necessarily the best way to separate the overall behavior into skills. It should also be mentioned that the algorithms are tested under conditions with relatively low noise levels. Even though S-Learning has been evaluated under noisy conditions with good results [23], it is expected that the PSL-based recognition methods is affected by noise in similar ways

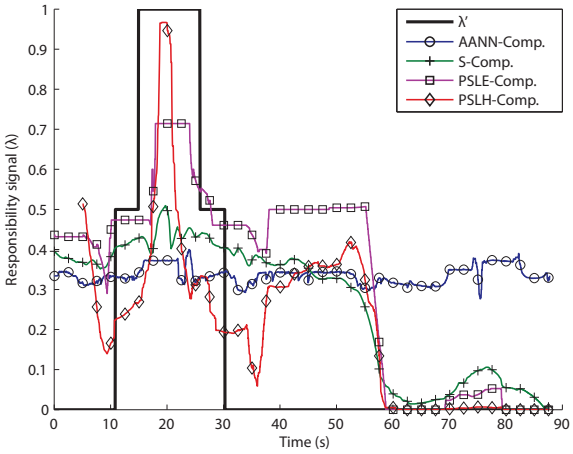

Figure 4. Responsibility signals for Skill 1 - Load. *AANN-Comp*, *S-Comp, PSLE-Comp* and *PSLH-Comp* indicates the responsibility signal computed with respective method.

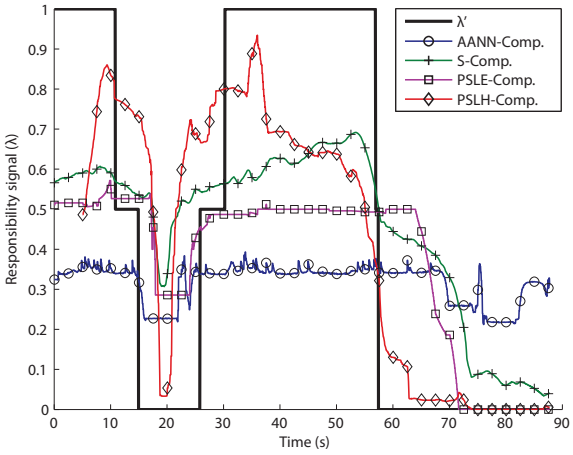

Figure 5. Responsibility signals for Skill 2 - Corridor. *AANN-Comp*, *S-Comp, PSLE-Comp* and *PSLH-Comp* indicates the responsibility signal computed with respective method.

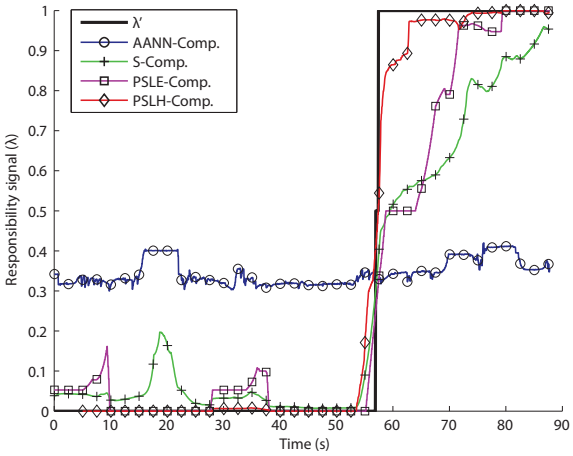

Figure 6. Responsibility signals for Skill 3 - Unload. *AANN-Comp*, *S-Comp, PSLE-Comp* and *PSLH-Comp* indicates the responsibility signal computed with respective method.

as PSL is subject to combinatorial explosion in large state spaces. Furthermore, both proposed algorithms compute the responsibility signal over a temporal extension ν, meaning that $\lambda_\beta(t)$ corresponds to how well β explains the events $(e_{t-\nu}, e_{t-\nu+1}, \ldots, e_t)$. I.e., we only know if the controller defined by β is the right choice for the present situation after these events have already occurred. Wolpert and coworkers [3] have also observed this problem, and introduce a *responsibility predictor* that estimates future responsibility signals. A corresponding mechanism is probably necessary when integrating the PSL based recognition methods with a controller.

A. Conclusions and future work

The results show that Bayes P_e (minimum error probability) over the activation pattern in the forward model (PSLH-Comparison) is a better method for behavior recognition than the prediction error (PSLE-Comparison), in the evaluated setting. While a more extensive study is necessary to draw any conclusions about the general performance of these algorithms, we find these results to be promising and intend to extend this evaluation to behavior recognition in other domains, and possibly to other types of data.

A big advantage of using PSL both for control (as described in [21]) and behavior recognition is that the forward and inverse computations are in fact based on the same model, i.e., the PSL library. This approach has several theoretical connections to the view of human perception and control as two heavily intertwined processes.

The present work should be seen as one step towards a hierarchical control architecture that can learn and coordinate itself, based on the PSL algorithm. The model-free design of PSL introduces very few assumptions into learning, and should constitute a good basis for many types of learning and control problems. Integrating PSLE-Comparison with a PSL-based control algorithm, to achieve a two-layer modular control system, is the next step in this process and will be part of our future work.

REFERENCES

[1] E. A. Billing and T. Hellström, "Behavior recognition for segmentation of demonstrated tasks," in *IEEE SMC International Conference on Distributed Human-Machine Systems*, Athens, Greece, March 2008, pp. 228 – 234.

[2] M. Haruno, D. M. Wolpert, and M. Kawato, "Hierarchical MOSAIC for movement generation," in *International Congress Series 1250*. Elsevier Science B.V., 2003, pp. 575– 590.

[3] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7–8, pp. 1317—1329, 1998.

[4] D. M. Wolpert, "A unifying computational framework for motor control and social interaction," *Phil. Trans. R. Soc. Lond.*, vol. B, no. 358, pp. 593–602, Mar. 2003.

[5] K. J. Friston, "Functional integration and inference in the brain," *Progress in Neurobiology*, vol. 68, no. 2, pp. 113–143, Oct. 2002.

[6] ——, "Learning and inference in the brain," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 16, no. 9, pp. 1325–52, 2003, PMID: 14622888.

[7] J. M. Kilner, K. J. Friston, and C. D. Frith, "Predictive coding: an account of the mirror neuron system," *Cogn Process*, vol. 8, pp. 159–166, 2007.

[8] D. George, "How the brain might work: A hierarchical and temporal model for learning and recognition," Ph.D. dissertation, Stanford University, Department of Electrical Engineering, 2008.

[9] D. George and J. Hawkins, "A hierarchical bayesian model of invariant pattern recognition in the visual cortex," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN'05)*, vol. 3, 2005, pp. 1812–1817 vol. 3.

[10] E. Billing, "Cognition reversed - robot learning from demonstration," Ph.D. dissertation, Umeå University, Department of Computing Science, Umeå, Sweden, December 2009.

[11] Y. Demiris and M. Johnson, "Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning," *Connection Science*, vol. 15, no. 4, pp. 231–243, 2003.

[12] Y. Demiris and A. Dearden, "From motor babbling to hierarchical learning by imitation: a robot developmental pathway," in *Proceedings of the 5th International Workshop on Epigenetic Robotics*, 2005, pp. 31—37.

[13] Y. Demiris and B. Khadhouri, "Hierarchical attentive multiple models for execution and recognition of actions," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 361–369, May 2006.

[14] M. Nicolescu, "A framework for learning from demonstration, generalization and practice in Human-Robot domains," Ph.D. dissertation, University of Southern California, 2003.

[15] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008.

[16] E. A. Billing and T. Hellström, "A formalism for learning from demonstration," in *Cognition Reversed - Robot Learning from Demonstration*. Umeå, Sweden: Print & Media, Umeå University, 2009, pp. 73–102.

[17] C. L. Nehaniv and K. Dautenhahn, "Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications," in *Learning Robots: An Interdisciplinary Approach*, J. Demiris and A. Birk, Eds. World Scientific Press, 2000, vol. 24, pp. 136–161.

[18] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, "Action, state and effect metrics for robot imitation," in *15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN 2006)*, Hatfield, Sep. 2006, pp. 232–237.

[19] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *RSJ Advanced Robotics, Special Issue on Imitative Robots*, vol. 21, no. 13, pp. 1521–1544, 2007.

[20] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, vol. 37, no. 2, pp. 286–298, 2007.

[21] E. A. Billing, T. Hellström, and L. E. Janlert, "Model free learning from demonstration," in *Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART)*, J. Filipe, A. Fred, and B. Sharp, Eds. Valencia, Spain: INSTICC, January 2010, pp. 62–71.

[22] B. Rohrer and S. Hulet, "BECCA - a brain emulating cognition and control architecture," Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA, Tech. Rep., 2006.

[23] ——, "A learning and control approach based on the human neuromotor system," in *Proceedings of Biomedical Robotics and Biomechatronics, BioRob*, 2006.

[24] M. Haruno, D. M. Wolpert, and M. M. Kawato, "MOSAIC model for sensorimotor learning and control," *Neural Comput.*, vol. 13, no. 10, pp. 2201–2220, 2001.

[25] S. Cha and S. N. Srihari, "On measuring the distance between histograms," *Pattern Recognition*, vol. 35, no. 6, pp. 1355–1370, Jun. 2002.

[26] K-Team, "Khepera robot," http://www.k-team.com, 2007. [Online]. Available: http://www.k-team.com