Simultaneous Control and Recognition of Demonstrated Behavior

Erik Billing*, Thomas Hellström† and Lars-Erik Janlert‡
Department of Computing Science
Umeå University, Sweden

Abstract

A method for *Learning from Demonstration (LFD)* is presented and evaluated on a simulated Robosoft Kompai robot. The presented algorithm, called *Predictive Sequence Learning (PSL),* builds fuzzy rules describing temporal relations between sensory-motor events recorded while a human operator is tele-operating the robot. The generated rule base can be used to control the robot and to predict expected sensor events in response to executed actions. The rule base can be trained under different *contexts,* represented as fuzzy sets. In the present work, contexts are used to represent different behaviors. Several behaviors can in this way be stored in the same rule base and partly share information. The context that best matches present circumstances can be identified using the predictive model and the robot can in this way automatically identify the most suitable behavior for precent circumstances. The performance of PSL as a method for LFD is evaluated with, and without, contextual information. The results indicate that PSL without contexts can learn and reproduce simple behaviors. The system also successfully identifies the most suitable context in almost all test cases. The robot's ability to reproduce more complex behaviors, with partly overlapping and conflicting information, significantly increases with the use of contexts. The results support a further development of PSL as a component of a dynamic hierarchical system performing control and predictions on several levels of abstraction.

Index terms: Behavior Recognition, Context Dependent, Fuzzy Logic, Learning and Adaptive Systems, Learning from Demonstration

*Erik Billing (billing@cs.umu.se)
†Thomas Hellström (thomash@cs.umu.se)
‡Lars-Erik Janlert (lej@cs.umu.se)

1 Introduction

Learning from Demonstration (LFD) is a well-established technique for teaching robots new behaviors. One of the greatest challenges in LFD is to implement a learning algorithm that allows the robot pupil to generalize a sequence of actions demonstrated by the teacher such that the robot is able to perform the desired behavior under varying conditions. In earlier work (Billing et al., 2010b, 2011), we have developed and evaluated the algorithm *Predictive Sequence Learning (PSL)* as a method for LFD. PSL can be trained from demonstrations performed via tele-operation and used as a controller for robots. The algorithm treats control as a prediction problem, such that the next action is selected based on the sequence of recent sensory-motor events. In addition, PSL also produces predictions of expected sensor states. While these are not directly useful for control, predictions of sensor states appear to serve well as a method for behavior recognition (Billing et al., 2010a).

Here, we evaluate the possibility to use a context layer that interacts with the PSL algorithm, both during learning and reproduction of behaviors. The context layer activates relevant parts of the PSL knowledge base while inhibiting knowledge that could interfere with the current behavior, potentially allowing PSL to learn and reproduce more complex behaviors. The work can be seen as one step towards integrating PSL in a dynamic hierarchical learning system.

An introduction to LFD and hierarchical learning systems is presented in Section 2, followed by a more precise problem statement in Section 3. The PSL algorithm is presented in Section 4 and the problem of knowledge interference is described in Section 5. The experimental setup used for the present work is described in Section 6 and a formulation of expected results is given in Section 7. Finally, results are presented in Section 8 followed by a discussion in Section 9.

2 Background

One common approach to LFD is to map the demonstration onto a set of pre-programmed or previously learned primitives controllers (Billing & Hellström, 2010). The approach has strong connections to *behavior-based architectures* (Matarić & Marjanovic, 1993; Matarić, 1997; Arkin, 1998) and earlier reactive approaches (e.g. Brooks, 1986, 1991). When introducing behavior primitives, the LFD process can be divided into three tasks (Billing & Hellström, 2010):

1. *Behavior segmentation* where a demonstration is divided into smaller segments.

2. *Behavior recognition* where each segment is associated with a primitive controller.

3. *Behavior coordination*, referring to identification of rules or switching conditions for how the primitives are to be combined.

The approach represents one way of introducing good bias in learning and solve the generalization problem by relying on previously acquired behavioral knowledge. While there are many domain specific solutions to each one of these three subproblems, they appear very difficult to solve in the general case.

One argument for the use of behavior primitives in LFD is that known behaviors can constitute parts (i.e., primitives) of other, more complex, behaviors. If this process can be implemented in a general way, it would allow learnt behaviors to act as primitives in future learning sessions. The approach would potentially allow the robot to learn increasingly complex behaviors as its knowledge base grows, producing an hierarchical architecture of controllers (Byrne & Russon, 1998). While this approach appears to have great potential, it requires that not only pre-programmed primitives, but also controllers generated through learning, can be recognized as parts of a demonstrated behavior.

This approach has many connections to biology and specifically the mirror system (e.g. Rizzolatti et al., 1988; Gallese et al., 1996; Brass et al., 2000; Rizzolatti & Craighero, 2004). While the role of the mirror system is still highly debated, several groups of researchers propose computational models where perception and action are tightly interweaved. Among the most prominent examples are the work by Demiris & Khadhouri (2006) proposing an architecture called *Hierarchical Attentive Multiple Models for Execution and Recognition (HAMMER)*. A similar theoretical framework is presented by Haruno et al. (2003) under the name *Hierarchical Modular Selection and Identification for Control* (*HMOSAIC)*. Both these architectures implement a set of modules, where each module is an inverse model (controller) paired with a forward model (predictor). The inverse and forward models are trained together such that the forward model can predict sensor data in response to the actions produced by the inverse model. The inverse model is tuned to execute a certain behavior when the forward model produces good predictions. The prediction error is used to compute a bottom-up signal for each module. Based on the bottom-up signal, a top-down responsibility signal or confidence value is computed and propagated to each module. The output of the system is a combination of the actions produced by each inverse model, proportional to their current responsibility. The responsibility signal also controls the learning rate of each module, such that modules are only updated when their responsibility is high. In this way, modules are tuned to a specific behavior or parts of a behavior. Since the prediction error of the forward model is used as a measure of how well the specific module fits present circumstances, it can be seen as a metric of imitation (Billard et al., 2003) that is learnt together with the controller. The architecture can be composed into a hierarchical system where modules are organized in layers, with the lowest layer interacting with sensors and actuators. The bottom-up signal constitutes sensor input for the layer above and actions produced by higher levels constitute the top-down responsibility signal.

One motivation for this architecture lies in an efficient division of labor between different parts of the system. Each module can be said to operate at a specific temporal resolution. Modules at the bottom layer are given the highest temporal resolution while modules higher up in the hierarchy have decreasing

resolution, allowing these modules to express dependencies over longer periods of time. State variables that change slowly compared to a specific module's resolution are ignored by that module and are instead handled by modules higher up in the hierarchy. Slowly changing states that lead to high responsibility for the module is referred to as the module's *context*. In a similar fashion, variables that change fast in relation to the temporal resolution are handled lower in the hierarchy. This allows each module to implement a controller where the behavior depends on relatively recent states, at its level of temporal resolution. Long temporal dependencies are modeled by switching between modules, which removes the requirement for each model to capture these dependencies. Furthermore, updates of a single behavior or parts of a behavior will only require updates of a few modules and will not propagate changes to other modules. See Billing (2009) for a longer discussion on these aspects of hierarchical architectures.

The HAMMER and MOSAIC architectures make few restrictions on what kind of controllers each module should implement. We argue however, that modules should be *semi-reactive*, meaning that action selection and predictions of sensor events should be based on recent sensor and motor events. Strictly reactive modules are not desirable since each module must be able to model any dependency with a temporal resolution too high for modules at the layer above.

However, the division of behavior into modules also has a number of drawbacks. The possibility for the system to share knowledge between behaviors is limited. Moreover, the system has to combine actions produced by different modules, which may be difficult in cases when more than one module receives high responsibility.

One architecture with similarities to HAMMER and MOSAIC able to share knowledge between different behaviors is *Recurrent Neural Network with Parametric Bias (RNNPB)* (Tani et al., 2004). Both input and output layer of the network contain sensor and motor nodes as well as nodes with recurrent connections. In addition, the input layer is given a set of extra nodes, representing the PB vector. The network is trained to minimize prediction error, both by training the network using back-propagation and by changing the PB input vector. The PB vector is however updated slowly, such that it organizes into what could be seen as a context layer for the rest of the network. In addition to giving the network the ability to represent different behaviors that share knowledge, the PB vector can be used for behavior recognition.

All these architectures can be seen as examples of a larger body of work employing the motor system for perception and imitation (e.g. Atkeson & Schaal, 1997; Billard, 2001; Demiris & Hayes, 2002; Demiris & Johnson, 2003; Alissandrakis et al., 2002; George, 2008), with an emphases on being biologically plausible. While there are many important differences between these works, both in proposed architectures and the claims that they make, there is also a significant common ground. One attempt to describe this common ground was made by Billing (2009), proposing four criteria for general learning ability:

1. **Hierarchical structures**

   Knowledge gained from learning should be represented in hierarchies.

2. **Functional specificity**

   Knowledge gained from learning should be organized in functionally specialized modules.

3. **Forward and inverse**

   Prediction error reflects how well the state definition satisfies the Markov assumption, and by consequence a forward model can be used to improve knowledge representation when paired with an inverse model.

4. **Bottom-up and top-down**

   Both bottom-up and top-down signals must be propagated through the hierarchical structure. Bottom-up signals represent the state of modules, and the top-down signals specify context.

These criteria can be seen as typical properties of a system able to internalize a simulation of percepts, in response to actions. That should be understood as one way to give the robot an inner world, a simulation of the physical world that does not rely on a pre-defined physics simulator but is generated from interactions with the world. Such a simulation is inherently grounded in the robot's sensors and actuators and is therefore not subject to the symbol grounding problem (Harnad, 1990). A minimalistic implementation of this approach can be found in the work by Ziemke et al. (2005). This approach also has tight connections with the work by Barsalou and colleagues (e.g. Barsalou et al., 2003; Barsalou, 2009), describing the brain as a system simulating sensor percepts in relation to motor activity.

Rohrer & Hulet (2006) proposed an architecture called *Brain Emulating Cognition and Control Architecture* (BECCA). The focus of BECCA was to capture the discrete episodic nature of many types of human motor behavior, while limiting the use of task-specific prior knowledge. BECCA was presented as a very general reinforcement learning system, applicable to many types of learning and control problems. One of the core elements of BECCA is the temporal difference (TD) algorithm *Sequence Learning* (SL) (Rohrer, 2007; Rohrer et al., 2009). SL builds sequences of passed events which is used to predict future events, and can in contrast to other TD algorithms base its predictions on a sequence of previous states.

3 Problem statement

Inspired by BECCA (Rohrer & Hulet, 2006) and specifically SL (Rohrer, 2007; Rohrer et al., 2009), we developed the PSL algorithm as a method for LFD (Billing et al., 2010a,b). PSL has many interesting properties seen as a learning algorithm for robots. It is model free, meaning that it introduces very few assumptions into learning and does not need any task specific configuration. PSL can be seen as a variable-order Markov model. Starting out from a reactive (first-order) model, PSL estimates transition probabilities between discrete

sensory-motor states. For states that do not show Markov property, the order is increased and PSL models the transition probability based on several passed events. In this way, PSL will progressively gain memory for parts of the behavior that cannot be modeled in a reactive way.

While previous evaluations of PSL (Billing et al., 2010a,b, 2011) show that the algorithm can be used both for control and recognition of several different behaviors, PSL is subject to combinatorial explosion when the demonstrated behavior requires modeling of long temporal dependencies. PSL can however efficiently model short temporal dependencies in a semi-reactive way and is a good candidate for implementation of forward and inverse models in an architecture similar to those described above. The fact that PSL is not able to implement an arbitrary controller is here seen as an important bias and serves as a way to get around the "no free lunch" theorems (Wolpert & Macready, 1997; Ho & Pepyne, 2002). In the present work we combine PSL control with behavior recognition in order to reduce these limitations and take one step closer to a hierarchical learning systems satisfying all criteria for general learning ability (Section 2).

4 Predictive Sequence Learning

PSL builds fuzzy rules, referred to as *hypotheses* h, describing temporal dependencies between a sensory-motor event e_{t+1} and a sequence of passed events $\left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)$, defined up until current time t.

$$h : \left(\Upsilon_{t-|h|+1} \; is \; E^h_{|h|} \wedge \Upsilon_{t-|h|+2} \; is \; E^h_{|h|-1} \wedge \ldots \wedge \Upsilon_t \; is \; E^h_1\right) \Rightarrow \Upsilon_{t+1} \; is \; \bar{E}^h. \tag{1}$$

Υ_i is the event variable and $E^h\left(e\right)$ is a fuzzy membership function returning a membership value for a specific e. The right hand side $\bar{E}^h$ is a membership function comprising expected events at time $t+1$. $|h|$ denotes the length of h, i.e., the number of left-hand-side conditions of the rule. Both E and $\bar{E}$ are implemented as standard cone membership functions with base width ε (e.g. Klir & Yuan, 1995).

A set of hypotheses can be used to compute a prediction $\hat{e}_{t+1}$ given a sequence of passed sensory-motor events η, defined up to the current time t:

$$\eta = \left(e_1, e_2, \ldots, e_t\right). \tag{2}$$

The process of matching hypotheses to data is described in Section 4.1. The PSL learning process, where hypotheses are generated from a sequence of data, is described in Section 4.2 and interaction with the context layer is described in Section 4.3. The description of PSL given here is similar, but not identical, to Fuzzy PSL as described in our previous evaluation of this algorithm (Billing et al., 2011). A few changes to the algorithm was introduced as a result of optimizations made in order to allow on-line predictions with multiple contexts. These changes are further discussed in Section 4.4.

4.1 Matching hypotheses

Given a sequence of sensory-motor events η (Equation 2), a match $\alpha_t(h)$ of the rule is given by:

$$\alpha_t(h) = \bigwedge_{i=1}^{|h|-1} E_i^h(e_{t-i+1}) \tag{3}$$

where $\wedge$ is implemented as a *min*-function.

Hypotheses are grouped in fuzzy sets C whose membership value $C(h)$ describes the *confidence* of h at time t:

$$C(h) = \frac{\sum_{k=t^h}^{t} \alpha_k(h)\, \bar{E}^h(e_{k+1})}{\sum_{k=t^h}^{t} \alpha_k(h)} \tag{4}$$

where t^h is the creation time of h or 1 if h existed prior to training. I.e., $C(h)$ is a weighted average of how well the h predicts the event e_{k+1}, over all observation up to time t. Each set C represents a *context* and can be used to implement a specific behavior or part of a behavior. The *responsibility signal* $\lambda_t(C)$ is used to control which behavior is active at a specific time. The combined confidence value $\tilde{C}_t(h)$, for hypothesis h, is a weighted average over all C:

$$\tilde{C}_t(h) = \frac{\sum_C C(h)\, \lambda_t(C)}{\sum_C \lambda_t(C)} \tag{5}$$

$\tilde{C}_t$ can be seen as a fuzzy set representing the active context at time t. Hypotheses contribute to a prediction in proportion to their membership in $\tilde{C}$ and the *match set* $\hat{M}$. $\hat{M}$ is defined in three steps. First, longest matching hypotheses are selected:

$$M = \{h \mid |h| \geq |h'| \ for\, all\ \{h' \mid \alpha(h') > 0\}\} \tag{6}$$

The best matching $h \in M$ is selected:

$$\tilde{M} = \{h \mid \alpha(h) \geq \alpha(h') \ for\, all\ \{h' \in M\}\} \tag{7}$$

Finally, the match set $\hat{M}$ is defined as:

$$\hat{M}(h) = \begin{cases} \tilde{C}(h) & h \in \tilde{M} \\ 0 & otherwise \end{cases} \tag{8}$$

The aggregated prediction $\hat{E}(e_{t+1})$ is computed using the Larsen method (e.g. Fullér, 1999):

$$\hat{E}(e_{t+1}) = \bigvee_h \bar{E}_h(e_{t+1})\, \hat{M}(h) \tag{9}$$

$\hat{E}$ is converted to crisp values using a *center of max* defuzzification (e.g. Klir & Yuan, 1995, p. 337):

$$\hat{e} = \frac{\min\left\{e \mid \hat{E}(e) = \max\left(\hat{E}\right)\right\} + \max\left\{e \mid \hat{E}(e) = \max\left(\hat{E}\right)\right\}}{2} \tag{10}$$

4.2 Generating hypotheses

Hypotheses can be generated from a sequence of sensory-motor events η. During training, PSL continuously makes predictions and creates new hypotheses when no matching hypothesis produces the correct prediction $\bar{E}$. The exact training procedure is described in Algorithm 1.

For example, consider the event sequence $\eta = abccabccabcc$. Let $t = 1$. PSL will search for a hypothesis with a left hand side matching a. Initially, the context set C is empty and PSL will create a new hypothesis $(a) \Rightarrow b$ which is added to C with confidence 1, denoted $C(a \Rightarrow b) = 1$. The same procedure will be executed at $t = 2$ and $t = 3$ such that $C((b) \Rightarrow c) = 1$ and $C((c) \Rightarrow c) = 1$. At $t = 4$, PSL will find a matching hypothesis $(c) \Rightarrow c$ producing the wrong prediction c. Consequently, a new hypothesis $(c) \Rightarrow a$ is created and confidences are updated such that $C((c) \Rightarrow c) = 0.5$ and $C((c) \Rightarrow a) = 1$. The new hypothesis receives a higher confidence since confidence values are calculated from the creation time of the hypothesis (Equation 4). The predictions at $t = 5$ and $t = 6$ will be correct and no new hypotheses are created. At $t = 7$, both $(c) \Rightarrow a$ and $(c) \Rightarrow c$ will contribute to the prediction $\hat{E}$. Since the confidence of $(c) \Rightarrow a$ is higher than that of $(c) \Rightarrow c$, $\hat{E}$ will defuzzify towards a, producing the wrong prediction (Equation 10). As a result, PSL creates a new hypothesis $(b, c) \Rightarrow c$. Similarly, $(c, c) \Rightarrow a$ will be created at $t = 8$. PSL is now able to predict all elements in the sequence perfectly and no new hypotheses are created.

Source code from the implementation used in the present work is available online (Billing, 2011).

4.3 Computing context responsibility

PSL is not only used as a controller but also as a method for behavior recognition. By letting PSL compute one prediction for each context, the responsibility of each context can be changed based on the size of respective prediction error. The method used here has strong similarities with the responsibility update mechanism used in the MOSAIC architecture (Haruno et al., 2001). Similar mechanisms can also be found in other learning and control frameworks with hierarchical structure (e.g. Demiris & Khadhouri, 2006).

One important difference between PSL and most other approaches is however that the context layer of PSL allows partial knowledge overlap between contexts. Furthermore, this overlap may be fuzzy in the sense that each hypothesis is a member of the context to a certain degree. This allows a much more flexible organization of knowledge compared to an architecture that requires each module

Algorithm 1 Predictive Sequence Learning (PSL)

Require: $\psi = (e_1, e_2, \ldots, e_T)$ where T denotes the length of the training set
Require: $\hat{\alpha}$ as the precision constant, see text

1: **let** $t \leftarrow 1$
2: **let** $\eta = (e_1, e_2, \ldots, e_t)$
3: **let** $C \leftarrow \varnothing$
4: **let** $\hat{E}$ as Equation 9
5: **if** $\hat{E}(e_{t+1}) < \hat{\alpha}$ **then**
6: **let** $h_{new} = CreateHypothesis(\eta, C)$ as defined by Algorithm 2
7: $C(h_{new}) \leftarrow 1$
8: **end if**
9: Update confidences $C(h)$ as defined by Equation 4
10: **set** $t = t + 1$
11: **if** t<T **then**
12: **goto** 2
13: **end if**

Algorithm 2 CreateHypothesis

Require: $\eta = (e_1, e_2, \ldots, e_t)$
Require: $C : h \rightarrow [0, 1]$
Require: α as defined by Equation 3

1: **let** $\hat{M}(h)$ as Equation 8
2: **let** $\bar{M} = \left\{ h \mid \bar{E}^h(e_{t+1}) \geq \hat{\alpha} \wedge \hat{M}(h) > 0 \right\}$ where $\hat{\alpha}$ is the precision constant, see Section 4.4
3: **if** $\bar{M} = \emptyset$ **then**
4: **let** E^* be a new membership function with center e_t and base ε
5: **return** $h_{new} : (\Upsilon_t \; is \; E^*) \Rightarrow \Upsilon_{t+1} \; is \; \bar{E}$
6: **else**
7: **let** $\bar{h} \in \bar{M}$
8: **if** $C(\bar{h}) = 1$ **then**
9: **return** *null*
10: **else**
11: **let** E^* be a new membership function with center $e_{t-|\bar{h}|}$ and base ε
12: **return** $h_{new} : \left(\Upsilon_{t-|\bar{h}|} \; is \; E^*, \Upsilon_{t-|\bar{h}|+1} \; is \; E^{\bar{h}}_{|\bar{h}|-1}, \ldots, \Upsilon_t \; is \; E^{\bar{h}}_0 \right) \Rightarrow \Upsilon_{t+1} \; is \; \bar{E}$
13: **end if**
14: **end if**

to be strictly separated from other modules. Several contexts may also be active simultaneously, without the need for a separate action coordination mechanism.

Let the $\hat{E}_t^C$ be the prediction for context C at time t, as given by Equation 9. The prediction for each context is calculated with the responsibility signal $\lambda_t(C) = 1$ and the responsibility of all other contexts equal to zero (see Equation 5). Based on these predictions, the responsibility signal for each context is updated using Bayes' rule:

$$\lambda_t(C) = \frac{\lambda_{t-1}(C) \exp\left(\frac{\left(E_t^C(e_t)-1\right)^2}{2\sigma^2}\right)}{\sum_{i=1}^{N}\left[\lambda_{t-1}(C_i) \exp\left(\frac{\left(E_t^{C_i}(e_t)-1\right)^2}{2\sigma^2}\right)\right]} \tag{11}$$

where e_t represents the observed sensory-motor event at time t. N is the number of contexts and σ^2, corresponding to the variance, is used as a scaling constant controlling the size of confidence changes in relation to prediction error size.

4.4 Parameters and implementation

A clear description of parameters is important for any learning algorithm. Parameters always introduce the risk that the learning algorithm is tuned towards the evaluated task, producing better results than it would in the general case. We have therefore strived towards limiting the number of parameters of PSL. The original design of PSL was completely parameter free, with the exception that continuous data was discretized using some discretization method. The version of PSL proposed here can be seen as a generalization of the original algorithm (Billing et al., 2010b,a) where the width ε of the membership function E determines the discretization resolution. In addition, a second parameter is introduced, referred to as the *precision constant* $\hat{\alpha}$. $\hat{\alpha}$ is with fuzzy logic terminology an α-*cut*, i.e., a threshold over the fuzzy membership function in the interval $[0, 1]$ (e.g., Klir & Yuan, 1995).

ε controls how generously PSL matches hypotheses. A high ε makes the algorithm crisp but typically increases the precision of predictions when a match is found. Conversely, a low ε reduces the risk that PSL reaches unobserved states at the cost of a decreased prediction performance. The high value of ε can be compared to a fine resolution data discretization for the previous version of PSL.

$\hat{\alpha}$ is only used during learning, controlling how exact a specific $\bar{E}$ has to be before a new hypothesis with a different $\bar{E}$ is created. A large $\hat{\alpha}$ reduces prediction error but typically results in more hypotheses being created during learning.

Both ε and $\hat{\alpha}$ control the tolerance to random variations in data and can be set based on how precise we want that PSL to model the data. Small ε in combination with large $\hat{\alpha}$ will result in a model that closely fits training data, typically producing small prediction errors but also requires more training data in order to cover the state space.

Updates of the responsibility signal (Equation 11) introduces a third parameter (the variance σ^2) scaling the prediction error and consequently controlling how quickly the responsibility signal changes. While this parameter could be estimated form actual data, it was manually set to a fixed value in the present work.

The original implementation of Fuzzy PSL (Billing et al., 2011) requires that the prediction for each context is computed separately, significantly increasing the computational load for each new context. For the present work, the algorithm was therefore reimplemented such that a majority of computations to be shared for all contexts, resulting in a minor extra load when multiple contexts are used. Even though no deeper study comparing the two versions of the algorithm has been made, our initial tests did not indicate any significant difference other than reduced computational requirements. In earlier work (Billing et al., 2010b,a), we used a discrete version of PSL that however differs from the present algorithm in several ways. See Billing et al. (2011) for details.

5 Knowledge interference

In early evaluations of PSL (Billing et al., 2010a,b), we noticed that increased training could affect the performance in both a positive and a negative way. On the positive side, more demonstrations provide more sensory-motor patterns that PSL can reuse in many situations. As long as the local sensory-motor history provides enough information to reliably separate between two situations, more training is always positive. However, when the recent sensory-motor history does not provide reliable information to select the right action, PSL produces longer hypotheses in order to increase prediction performance. While this in itself is not a large problem, it increases the risk for inappropriate action selection when PSL is used as a controller. If the current sensory-motor history does not match any long hypothesis, PSL falls back on shorter, less reliable, hypotheses. As a result, PSL sometimes selects an inappropriate action. We call this problem *knowledge interference*, since knowledge of one behavior or part of a behavior is interfering with the behavior currently being executed.

One potential solution to this problem is to separate behavioral knowledge into several contexts, and let a behavior recognition mechanism select one or several context that should be responsible for the present situation. Hypotheses that are strongly associated with the active contexts are prioritized over other hypotheses, and hypotheses that would have interfered with the current behavior can in this way be ignored. This can be seen as one way to achieve the criterion of Functional specificity presented in Section 2.

We have previously evaluated several techniques for behavior recognition (Billing & Hellström, 2008) and also shown that PSL can be used for behavior recognition (Billing et al., 2010a), based on the same model as used for control. We are however not aware of any previous work that connects the behavior recognition capabilities of PSL with a PSL based controller, such that the robot can continuously evaluate the responsibility of each context and in this way

reduce the problem of knowledge interference.

6 Experimental setup

In order to evaluate PSL, a simulated Robosoft Kompai robot (Robosoft, 2011) was used in the Microsoft RDS simulation environment (Microsoft, 2011). The 270 degree laser scanner of the Kompai was used as source for sensor data and the robot was controlled by setting linear and angular speeds. We used a similar setup in previous work (Billing et al., 2011) and here extend earlier tests to include the new features of PSL.

Demonstrations were performed via tele-operation using a joypad, while sensor and motor data was recorded with a temporal resolution of 20 Hz. The dimensionality of the laser scanner was reduced to 20 dimensions using an average filter. Angular and linear speeds were fed directly into PSL. ϵ was set to 0.8 m for laser data and 0.1 m/s for motor data. $\hat{\alpha} = 0.95$ was used for both sensor and motor data and σ^2 was set to 5.

In cases PSL does not find a match and is unable to produce a prediction, a reactive obstacle avoidance was used to control the robot. While PSL normally has full control over the robot, it can run into unknown states for a short periods of time, usually when close to walls and obstacles. The obstacle avoidance can in these cases prevent a collision. The robot may of course still have contact with objects as long as PSL is in control.

Four behaviors were used, each one demonstrated ten times with some variations. Each behavior is described below. Numbered areas are illustrated in Figure 1. The behaviors were intentionally designed to overlap, such that the robot would experience similar situations in parts of several behaviors.

ToTV Started from various locations in the apartment (area 2, 4, 5, and 6) and finished in front of the TV (area 1).

Wake Started close to the bed (area 3) and finished in the corridor (area 5).

ToKitchen Started in the hallway (area 7), made a left turn in the corridor (area 5) followed by a right turn towards the kitchen, finally turning around and stopping in the kitchen (area 2).

Serve Started in the kitchen (area 2), went clockwise around the table, slowly passing by each chair one by one, through area 8, and back to the kitchen. The behavior finished by turning around and stop.

6.1 Evaluation of simultaneous control and recognition

In order to test how well PSL could reproduce the demonstrated behaviors, and generalize, ten test cases were designed.

Case 1 PSL was trained on demonstrations of the ToTV behavior. Tests are made starting from area 3.

Figure 1: The simulated apartment environment used for evaluation. Numbered regions indicate critical areas used as reference for demonstrations and reproduced behaviors (see text).

Case 2 PSL was trained on demonstrations from the ToTV and Wake behaviors. Tests were made starting form area 3.

Case 3 PSL was trained on demonstrations from the ToKitchen behavior. Tests were made starting from area 7.

Case 4 PSL was trained on demonstrations from the Serve behavior. Tests were made starting from area 2.

Case 5, 6, and 7 PSL was trained with demonstrations of all four behaviors. The training data was not separated into different behaviors but trained and represented as a single PSL context. Tests were made starting from area 3 (case 5), area 7 (case 6) and area 2 (case 7).

Case 8, 9, and 10 PSL was trained on demonstrations from all four behaviors. The training data was separated into four different contexts, such that each context represented one behavior. Behavior recognition was used to continuously update the responsibility for each context. Tests were made starting from area 3 (case 8), area 7 (case 9) and area 10 (case 7).

Apart from the demonstrated data, the robot did not get any information of which behavior to execute at a certain time. When trained on more than one behavior, PSL had to recognize the present starting location and use this information to select the appropriate behavior.

6.2 Evaluation of behavior recognition during manual control

In addition to the evaluation described in the previous section, PSL was evaluated as a method for behavior recognition during manual control. This can be seen as an attempt to reproduce our previous results (Billing et al., 2010a) in a more realistic setting.

A single demonstration was made starting from area 7, moving out of the room and turning left towards area 6. After approximately 17 seconds, the robot reaches area 3, turns around and goes back towards area 6, out of the bedroom and reaches area 5 at t = 30 s. The robot continuous with the table to its right, passes area 8 at t = 37 s and makes a right turn around the table towards the kitchen. After 43 seconds of driving, the robot reaches the kitchen, turns around, leaves the kitchen at t = 48 s and makes a second lap around the table. When reaching area 8 for the second time (t = 56 s) the robot makes a left turn towards the TV and parks in front of the TV after a total of 67 seconds of driving.

7 Hypothesis

Since PSL represents behaviors as a semi-reactive controller, no specific coordination is required in order to merge two demonstrated behaviors. One example

when this is useful is when an existing behavior is to be extended to work in a partly new environment. In order to evaluate this aspect of PSL, test cases 1 and 2 were designed. Since no demonstrations in the ToTV behavior was made starting from the bed (area 3), test case 1 is expected to be a difficult task to reproduce. In test case 2, the original demonstration set is however combined with the demonstrated Wake behavior, showing how to exit the bedroom. The performance of test case 2 is therefore expected to be significantly higher than for case 1.

While the problem of knowledge interference (Section 5) may occur both within and between behaviors, the risk clearly increases when the robot is thought to act differently in several similar situations. Billing et al. (2010b) successfully taught a Khepera robot three partial behaviors, but when they were combined into a complete behavior, none of the partial behaviors could be reproduced correctly. In the present work, we aim to reproduce this scenario in a more realistic setting. Test cases 2, 3, and 4 represent three fairly simple behaviors that PSL should be able to reproduce when thought separately. However, since the three behaviors have significant overlaps, knowledge of one behavior may interfere with knowledge of another, and the performance of is therefore expected to decrease when all behaviors are trained together (case 5, 6, and 7).

Test cases 8, 9, and 10 were designed to evaluate the effect of behavior recognition during execution of the three different behaviors. If the behavior recognition system works as intended, the performance of case 8, 9, and 10 should be significantly higher than for cases 5, 6, and 7, respectively.

8 Results

Results for all ten test cases are summarized in Table 1. In test case 1, the robot were only able to exit the bedroom in 12 out of 20 runs, but reached the TV (area 1) every time it exited the bedroom. In test case 6, the robot reached the kitchen in 12 out of 20 runs, but took the right way only twice. During the other 10 runs, the robot went around the table as demonstrated in the Serve behavior, and in this way reached the kitchen. Similarly, in test case 9, the robot reached the kitchen in 15 out of 20 runs, and took the demonstrated path to the kitchen 13 times.

Figure 2 displays the responsibility signals from one execution of test case 8. The robot starts from area 3 (see Figure 1) with initially equal responsibilities for all four behaviors. The behavior recognition system quickly recognizes the present situation as a Wake behavior and the robot consequently starts to execute that behavior. The robot reaches area 5 after approximately 10 seconds. When continuing through the corridor, the Wake behavior no longer matches present circumstances causing a shift to the ToTV behavior. The robot also passed by the corridor during the ToKitchen behavior, making that behavior a possible candidate. Since the ToTV behavior had some demonstrations also from area 6, it receives increased confidence earlier than ToKitchen. As a result,

Test case	Reached TV	Reached Kitchen	Fail
1: ToTV	12	0	8
2: ToTV + Wake	18	0	2
3: ToKitchen	0	19	1
4: Serve	0	19	1
5: All in one context	1	13	6
6: All in one context	5	2 (12)	3
7: All in one context	5	14	1
8: All in separate contexts	15	3	2
9: All in separate contexts	3	13 (15)	2
10: All in separate contexts	0	17	3

Table 1: Results for the ten test cases.

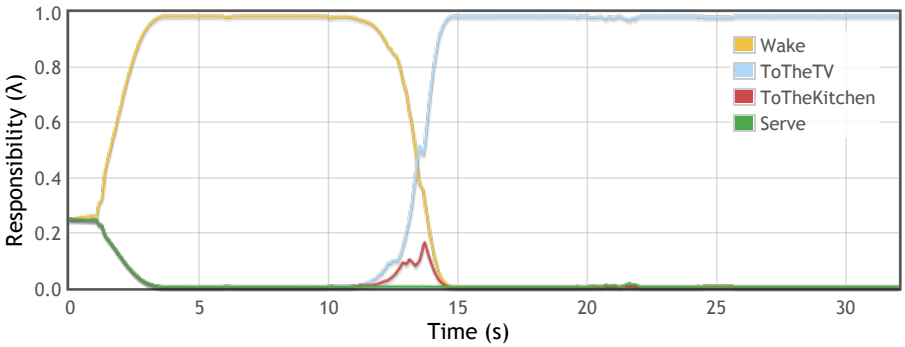

Figure 2: Typical responsibility signal from one execution of test case 8.

ToTV takes control of the robot as the responsibility of the Wake behavior decreases, quickly suppressing ToKitchen. After approximately 20 seconds, when the robot is passing by area 8, both ToTV and Serve have small prediction errors, causing slight fluctuations of the responsibilities. The high prior for ToTV will however cause the system to remain with that behavior. Finally, after approximately 30 seconds, the robot parks in front of the TV (area 1). Figure 3 displays the results from the evaluation of behavior recognition during manual control (Section 6.2).

9 Discussion

On the whole, results presented in Section 8 match our expectations (Section 7). Without the use of behavior recognition (case 5, 6 and 7), the robot successfully reproduces the demonstrated behavior only in 17 out of 60 trails. This is a clear case of what we call knowledge interference (Section 5) since the same set of demonstrations, when divided up in different behaviors (case 2, 3 and 4),

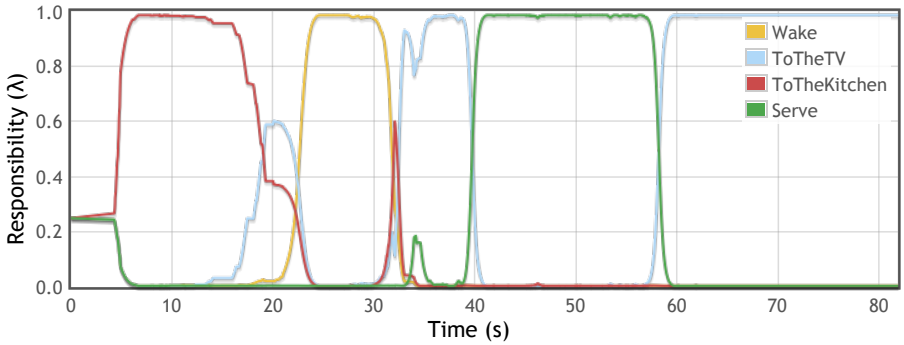

Figure 3: Responsibility signal during manual control of the robot (Section 6.2).

resulted in successful reproductions in almost all trails. This problem is reduced when behavior recognition is active (case 8, 9 and 10), increasing the correctly reproduced trails to 45 out of 60.

The responsibility signals computed by the behavior recognition system (figures 2 and 3) is much more stable than previous evaluations have shown (Billing et al., 2010a). We believe that this can partly be explained by the large dimensionality of the laser scanner data, compared to the infrared proximity sensors of the Khepera robot used in previous experiments. Another reason may be that the fuzzy version of PSL used in this work produces smaller, and more reliable, prediction errors than the discrete version of PSL previously used. See Billing et al. (2011) for a comparison.

The results from behavior recognition during manual control (Figure 3) indicate that the responsibility signal serves as an extra memory for PSL. During t = 30 to 37s and t = 48 to 56s, the robot drives around the table in a similar way as during both ToTV and Serve behaviors (see Section 6.2 for details). The first episode is interpreted as a ToTV behavior while the Serve behavior receives the highest responsibility during the second episode. The reason for this difference is that the responsibility of each context is updated based on its prior responsibility (Equation 11), allowing the responsibilities to remain unchanged as long as the most active context does not produce significantly larger prediction errors than any other context. While this property may be a disadvantage in some situations, we argue that it serves as a powerful way to activate relevant parts of the PSL knowledge base based on information much further back in time than PSL can represent using hypotheses alone. At the same time, the reactive properties of the system remains. Even though the responsibilities for the ToTV and Serve behaviors remain stable during the manual demonstration, the system is able to reevaluate this interpretation as soon as the demonstrated behavior is diverging from its expected path (e.g., t = 37 and t = 56).

One of the weak parts of the present approach is that it still relies on a

human to divide the demonstrated data in a way that reduces knowledge interference. We see no reason to believe that the human's interpretation of what is one behavior, and what is another, perfectly corresponds to the robot's need to categorize experiences. As a result, there may be other ways to divide the set of demonstrations used in this work (a total of 40 demonstrations) such that the results would be significantly better, or significantly worse. One way to get away from this problem may be to identify contexts (Equation 4) automatically, for example based on an entropy measure. While hypotheses frequently selected in sequence should go into the same context, each context should keep the amount of conflicting hypotheses low. This could possibly be formulated as an optimization problem where the total entropy over all contexts is to be minimized. Exploring this possibility to automatically identify contexts that potentially could be interpreted as behaviors from a human's point of view is part of future work.

Another interesting feature of the architecture presented here is that it provides a clean interface to higher level control. From an engineering point of view, PSL could be seen as a reactive layer, similar to the ones frequently used in hybrid systems, but with the ability to also feed information to the deliberating parts of the system. PSL could consequently constitute both a semi-reactive actuator layer and a sensor layer. Some research in this direction is already taking place, PSL have for example been considered for integration with a high level controller based on semantic networks (Fonooni et al., 2012).

The architecture could also be extended to a hierarchical structure with one instance of PSL running on each layer in the system. The lowest layer would interact with sensors and actuators of the robot. Layers higher up in the hierarchy would interact with the responsibility signals of the layer directly below and in this way affect the behavior of the system as a whole. The temporal resolution of PSL would decrease upwards in the hierarchy, allowing hypotheses on higher levels to extend over longer periods of time. Similarly to the hierarchical architectures discussed in Section 1, information in form of prediction errors would also be propagated as inputs to the layer directly above.

While a hierarchical version of PSL has many strong similarities with both HAMMER (Demiris & Khadhouri, 2006) and HMOSAIC (Haruno et al., 2003) there are also important differences. PSL contexts can share information in a straightforward way, not directly possible with the strictly separated modules proposed by HAMMER and HMOSAIC. We also make a emphasis on modules that are semi-reactive, following the division of labor between layers presented in Section 1. Both these properties are to large extent shared with RNNPB. While a recurrent neural network could be expected to handle dimensionality better than PSL, it may have drawbacks in the sense that it is more difficult to progressively extend the models size in the way PSL does. We hope to be able to conduct a comparison between PSL and RNNPB in future work.

References

Alissandrakis, A., Nehaniv, C. L., & Dautenhahn, K. (2002). Imitation With ALICE: Learning to Imitate Corresponding Actions Across Dissimilar Embodiments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32, 482–496.

Arkin, R. C. (1998). *Behaviour-Based Robotics*. MIT Press.

Atkeson, C. G. & Schaal, S. (1997). Robot learning from demonstration. In D. H. Fisher Jr (Ed.), *Proceedings of the 14th International Conference on Machine Learning*, number 1994 (pp. 12–20). Nashville.

Barsalou, L. W. (2009). Simulation, situated conceptualization, and prediction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521), 1281–1289.

Barsalou, L. W., Simmons, K. W., Barbey, A. K., & Wilson, C. D. (2003). Grounding conceptual knowledge in modality-specific systems. *Trends in Cognitive Sciences*, 7(2), 84–91.

Billard, A. (2001). Learning motor skills by imitation: a biologically inspired robotic model. *Cybernetics and Systems*, 32, 155–193.

Billard, A., Epars, Y., Cheng, G., & Schaal, S. (2003). Discovering imitation strategies through categorization of multi-dimensional data. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, volume 3 (pp. 2398–2403 vol.3). Las Vegas, Nevada.

Billing, E. A. (2009). *Cognition Reversed - Robot Learning from Demonstration*. Licentiate thesis, Umeå University, Department of Computing Science, Umeå, Sweden.

Billing, E. A. (2011). www.cognitionreversed.com.

Billing, E. A. & Hellström, T. (2008). Behavior Recognition for Segmentation of Demonstrated Tasks. In *IEEE SMC International Conference on Distributed Human-Machine Systems* (pp. 228–234). Athens, Greece.

Billing, E. A. & Hellström, T. (2010). A Formalism for Learning from Demonstration. *Paladyn: Journal of Behavioral Robotics*, 1(1), 1–13.

Billing, E. A., Hellström, T., & Janlert, L. E. (2010a). Behavior Recognition for Learning from Demonstration. In *Proceedings of IEEE International Conference on Robotics and Automation* Anchorage, Alaska.

Billing, E. A., Hellström, T., & Janlert, L. E. (2010b). Model-free Learning from Demonstration. In J. Filipe, A. Fred, & B. Sharp (Eds.), *Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART)* (pp. 62–71). Valencia, Spain.

Billing, E. A., Hellström, T., & Janlert, L. E. (2011). Robot Learning from Demonstration using Predictive Sequence Learning. In A. Dutta (Ed.), *Robotic Systems - Applications, Control and Programming (to appear)*. In-Tech.

Brass, M., Bekkering, H., Wohlschläger, A., & Prinz, W. (2000). Compatibility between observed and executed finger movements: comparing symbolic, spatial, and imitative cues. *Brain and cognition*, 44(2), 124–43.

Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1), 14–23.

Brooks, R. A. (1991). New Approaches to Robotics. *Science*, 253(13), 1227–1232.

Byrne, R. W. & Russon, A. E. (1998). Learning by Imitation: A Hierarchical Approach. *The Journal of Behavioral and Brain Sciences*, 16(3).

Demiris, J. & Hayes, G. R. (2002). Imitation as a dual-route process featuring predictive and learning components: A biologically plausible computational model. In K. Dautenhahn & C. L. Nehaniv (Eds.), *Imitation in animals and artifacts* (pp. 327–361). Cambridge, MA, USA: MIT Press.

Demiris, Y. & Johnson, M. (2003). Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connection Science*, 15(4), 231–243.

Demiris, Y. & Khadhouri, B. (2006). Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54(5), 361–369.

Fonooni, B., Hellström, T., & Janlert, L. E. (2012). Learning High-Level Behaviors from Demonstration through Semantic Networks. In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence (ICAART) (to appear)* Vilamoura, Algarve, Portugal.

Fullér, R. (1999). *Neural Fuzzy Systems*. Physica-Verlag GmbH & Co.

Gallese, V., Fadiga, L., Fogassi, L., & Rizzolatti, G. (1996). Action recognition in the premotor cortex. *Brain*, 119(2), 593–609.

George, D. (2008). *How the Brain might work: A Hierarchical and Temporal Model for Learning and Recognition*. Phd thesis, Stanford University.

Harnad, S. (1990). The Symbol Grounding Problem. *Physica*, D(42), 335–346.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10), 2201–2220.

Haruno, M., Wolpert, D. M., & Kawato, M. (2003). Hierarchical MOSAIC for movement generation. In *International Congress Series 1250* (pp. 575– 590).: Elsevier Science B.V.

Ho, Y. C. & Pepyne, D. L. (2002). Simple Explanation of the No-Free-Lunch Theorem and its Implications. *Journal of Optimization Theory and Applications*, 115(3), 549–570.

Klir, G. J. & Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall.

Matarić, M. J. (1997). Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3), 323–336.

Matarić, M. J. & Marjanovic, M. J. (1993). Synthesizing Complex Behaviors by Composing Simple Primitives. In *Proceedings of the European Conference on Artificial Life*, volume 2 (pp. 698–707). Brussels, Belgium.

Microsoft (2011). Microsoft Robotic Developer Studio, www.microsoft.com/robotics.

Rizzolatti, G., Camarda, R., Fogassi, L., Gentilucci, M., Luppino, G., & Matelli, M. (1988). Functional organization of inferior area 6 in the macaque monkey. II. Area F5 and the control of distal movements. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 71(3), 491–507.

Rizzolatti, G. & Craighero, L. (2004). The Mirror-Neuron System. *Annual Review of Neuroscience*, 27, 169–192.

Robosoft (2011). Kompai Robot, www.robosoft.com.

Rohrer, B. (2007). S-Learning: A Biomimetic Algorithm for Learning, Memory, and Control in Robots. In *Proceedings of the 3rd International IEEE/EMBS Conference on Neural Engineering* (pp. 148 – 151). Kohala Coast, Hawaii.

Rohrer, B., Bernard, M., Morrow, J. D., Rothganger, F., & Xavier, P. (2009). Model-free Learning and Control in a Mobile Robot. In *Proceedings of the Fifth International Conference on Natural Computation* (pp. 566–572). Tianjin, China.

Rohrer, B. & Hulet, S. (2006). *BECCA - A Brain Emulating Cognition and Control Architecture*. Technical report, Cybernetic Systems Integration Department, Sandria National Laboratories, Alberquerque, NM, USA.

Tani, J., Ito, M., & Sugita, Y. (2004). Self-Organization of Distributedly Represented Multiple Behavior Schemata in a Mirror System : Reviews of Robot Experiments Using RNNPB. *Neural Networks*, 17, 1273–1289.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.

Ziemke, T., Jirenhed, D. A., & Hesslow, G. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68, 85–104.