Learning to Ground Fact Symbols in Behavior-Based Robots

Joachim Hertzberg, Herbert Jaeger, Frank Schönherr[1]

Abstract. A robot running a hybrid control system (its architecture comprising a deliberative and a reactive part) must permanently update its symbolic situation model to allow its ongoing deliberation to operate. Previous work has shown that this update can be improved by using, possibly among other sources, the robot's sensor information as filtered through recent activation value histories of robot behaviors. In that work, characteristic patterns in groups of behavior activation values are used to define *chronicles*, which allow true facts about the current situation to be hypothesized. Chronicle definitions are hand-crafted as part of the domain modeling.

In this paper, we demonstrate that analogs of chronicle definitions can be *learned*. We use an echo state network, which is a particular type of recurrent neural network. To train it, the same activation value data are used as before in chronicle definitions. The training process is fast. The detection process is cheap to run on-line on board the robot. The new method is demonstrated on data from a robot simulator. It provides the robot programmer an alternative tool for getting recent symbolic situation fact hypotheses.

1 BACKGROUND

Typically, a state-of-the-art autonomous mobile robot has a *hybrid* control architecture, i.e., its control system includes a reactive (aka. functional) and a deliberative layer, and possibly more layers to connect the two [2]. Hybridity is needed to integrate the overall robot action information of greatly different granularities and time scales. To achieve timely response to rapid, asynchronous change in the robot environment, fast reaction is needed, typically based on raw sensor data—collision avoidance is the standard example. To achieve long-term goal-directed action in incompletely known environments, deliberation is needed, typically based on symbolic concepts describing the current situation. Task rescheduling due to a new high-priority user request is an example.

Hybrid robot control architectures—be they organized in the "classical" three layers like [5, 1] and many more, or in two layers like [11, 17]—must organize two directions of information flow to make the overall robot action coherent: "Downward" from abstract tasks to concrete physical behavior, and an "upward" flow of fine-grained data about the physical world to abstract symbols for deliberation. This is an instance of the *symbol grounding* problem [10]; more recently, *object anchoring* [7] has stirred some interest.

This paper deals with the "upward" information flow: what the reactive control layer tells the deliberative one about the world. In most robot control architectures, such as [5, 1, 17], this data is used as status information about the state of executing the recent abstract task on a more concrete layer, providing exception messages, termination conditions, and the like. While this is crucial information, more can be done. [15] presents an approach to derive hypotheses about facts about the recent situation *external to the robot* by analyzing the histories of behavior activation values. Certain patterns in these time series are matched against given *chronicle* definitions in the spirit of [9]; a summary will be given below in Sec. 3. Behavior activation values provide a very particular way of aggregating sensor values, namely, as is necessary for permanently calculating the degree of activity for each and every behavior on the reactive layer. For fact derivation, they are available for free in the sense that they have to be calculated anyway in the robot control.

In [15] chronicles are hand-crafted, based on the domain modeler's understanding of how different robot behaviors interact with each other and with the physical environment. In this paper, we attempt to arrive at the same type of hypothesis about symbolic facts true in the recent situation, based on the same behavior activation data plus other sensor and control data traces, but under a black-box view towards the robot control system: A naive user should be able to train the fact hypothesis process by giving few positive and negative examples for the respective facts to the robot acting in its environment, and the robot should then classify well situations according to the validity of the trained fact, as far as it is mirrored in its data.

Obviously, this is not easy due to the potentially huge amount of data in the streams of sensor and control values flooding a robot at any time. As will become clear in the paper, a particular type of classification learning method, echo state networks [12], in combination with the sensor data filtering as through the behavior activation value histories, makes the task feasible.

The rest of this paper is organized as follows. Sec. 2 describes the hybrid robot control architecture that we are using: the DD&P architecture. Note, however, that the learning method proposed here does in no way require DD&P; other behavior-based robot control schemes would work analogously. Sec. 3 recapitulates from [15] the principle and the technique to hypothesize situation facts from behavior activation value histories by using hand-coded chronicle definitions. Sec. 4 sketches echo state networks [12, 13], which represent the powerful learning framework of our approach. Sec. 5 puts it all together, describing how echo state networks can be used to arrive at analogs of chronicles. Sec. 6 presents experiments with the learning approach, based on training and test traces gained from a robot simulator. Sec. 7 concludes.

2 BEHAVIOR-BASED ROBOT CONTROL IN THE DD&P ARCHITECTURE

DD&P [11] is the robot control architecture used here. The learning method described later may be embedded into other architectures,

[1] Fraunhofer-AIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany. email: hertzberg|herbert.jaeger|schoenherr@ais.fraunhofer.de

provided that their reactive layer produces time series of behavior activation values. As DD&P is the background rather than a feature for the contents of this paper, we sketch here abstractly its most essential points, emphasizing its use of activation values in the reactive layer, as they play a key role for the learning method.

DD&P is a two-layer hybrid architecture using Dual Dynamics (DD, [14]) to formulate the reactive layer in a behavior-based way. DD is also used stand-alone, e.g., in the RoboCup team of [6]. DD&P adds to DD a deliberative robot control layer. For this paper, the DD layer is of concern. The following sketch borrows from [15].

A DD controller consists of two kinds of behaviors: low-level behaviors (LLBs), which are directly connected to the robot actuators, and higher-level behaviors (HLBs), which are connected to LLBs and/or HLBs. Each LLB implements two distinct functions: a target function and an activation function. The target function for the behavior b provides the reference t_b for the robot actuators (*"what to do"*) as

$$t_b = f_b(s^T, \alpha^T_{LLB}), \tag{1}$$

where f_b is a nonlinear vector function with one component for each actuator variable, s^T is the vector of all inputs from sensors or sensor-filters (aggregated sensor values) and α^T_{LLB} is the vector of activation values of the LLBs.

The LLB activation function *modulates* the output of the target function. It provides a value in [0,1], describing the degree how the behavior attempts to influence the robot actuators (not, fully, and all degrees in between). The *variation* of the *activation value* $\alpha_{b,LLB}$ of the LLB b it is computed as the following nonlinear function:

$$\dot{\alpha}_{b,LLB} = g_b(\alpha_{b,LLB}, OnF_b, OffF_b, OCT_b) \tag{2}$$

OCT_b is a term allowing the deliberative layer in DD&P to influence the activation values, which is beyond the scope of this paper (see [11]). OnF_b ($OffF_b$, resp.), the "on-forces" ("off-forces"), is a scalar variable summing up all conditions that recommend (discommend) activating the respective behavior. They are defined as nonlinear functions of s^T, α^T_{LLB}, and α^T_{HLB}.

The HLBs implement only the activation function. They are allowed to modulate only the LLBs or other HLBs on the same or *lower* level. In our case, the change of activation values for the HLBs $\dot{\alpha}_{b,HLB}$ are computed in the same manner as in Eq. 2.

One reason for updating behavior activation in the form of Eq. 2 is this. Referring to the previous activation value $\dot{\alpha}_b$ incorporates a memory of the previous evolution, which can be overwritten in case of sudden and relevant changes in the environment, but normally prevents activation values from spiking or oscillating with high frequency. At the same time, this form of the activation function provides some *low-pass filtering* capabilities, deleting sensor noise or oscillating sensor readings.

In this variant of DD, behaviors are arbitrated using the activation values. Each behavior can influence (excite or inhibit) every other behavior on the same or lower level. The interaction between behaviors is defined by the variables OnF and $OffF$.

The output or *reference vector* r of the BBS for the robot actuators is generated by summing all LLB outputs by a *mixer*:

$$r = \kappa \sum_b \alpha_{b,LLB} t_b$$

where $\kappa = (1/\sum_b \alpha_{b,LLB})$ is a normalizing factor.

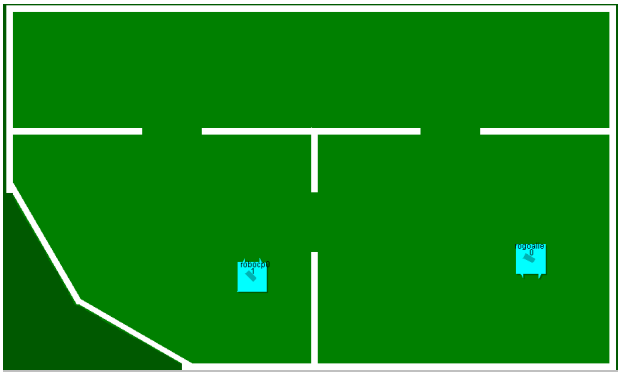

Figure 1. Test arena for the examples and experiments in the text. The second robot adds dynamical effects to the environment.

3 GROUNDING FACT HYPOTHESES IN BEHAVIOR ACTIVATION VALUES

The arena for all examples in the rest of this paper is shown in Fig. 1. It consists of a narrow corridor (upper part) and two rooms connected by a door. The robot to be controlled travels around in the arena, possibly facing another robot or other obstacles. The sensors of the simulated robot are simple office-navigation robot standard (various distance sensors). A simulated laser scanner is available, but not used among the raw sensor values for learning.

We have here the following LLBs: AvoidColl, FollowLeftWall, FollowRightWall, GoThruDoor, TurnToDoor, and Wander. As HLBs, we have CloseToDoor, InCorridor, and TimeOut. They have all the function suggested by their names, such as AvoidColl avoiding a collision. But note that the mentioning of Door in behavior names is just mnemonic for the domain modeler. The robot does not know about doors, and it will react to any narrow passage looking like a door (such as a narrow pathway between obstacles) with rising activation of the respective behaviors.

[15] presents an approach to derive hypotheses about situation facts from time-series or *histories* of DD&P activation values. We sketch this approach to compare it later with the new learning approach developed below (Sec. 5).

The histories form curves like the ones shown in Fig.2. In these curves, certain patterns re-occur for single behaviors within intervals of time, such as a value being more or less constantly high or low, and values going up from low to high or vice versa. [15] uses these isolated patterns as building blocks for fact extraction, the so-called *qualitative activations*. In this section, we consider four of them: rising/falling edge, high and low, symbolized by predicates $\Uparrow$e, $\Downarrow$e, Hi, and Lo, resp. In general, more qualitative activations may be of interest, such as a value staying in a medium range over some time. All exact formal definition can be found in [15][Sec. 5].

The key idea for extracting facts from activation histories is to consider patterns of qualitative activations of several behaviors that occur within the same interval of time. These patterns are called *activation gestalts* or briefly AG. For example, $AG(\{\Uparrow\text{e(GoThruD.)}, \Downarrow\text{e(TurnToD.)}, \text{Hi(CloseToD.)}\})[t_1, t_2]$ means loosely speaking that the behavior GoThruDoor must get activated, TurnToDoor get deactivated and CloseToDoor stay active within $[t1, t2]$. The activation values for all other behaviors not mentioned in this AG are unconstrained in this time interval. Grouping together AGs for sequences of time intervals results in the final def-

inition: a *chronicle*. Informally, such a chronicle is a sequence of AGs that define events for a sequence of time intervals of previously unknown duration. A ground fact is extracted from the activation history of a BBS as a *hypothesis* at time t if its *defining chronicle* matches the respective activation histories ending at t.

We give as a graphical example a chronicle allowing to extract the fact InRoom that the robot is in a room (rather than in the corridor), such as the one left of the wall in Fig. 1. The grey shaded parts in Fig. 2 highlight the components from the respective defining chronicle, with the t_i separating the consecutive activation gestalts. The stream of activation values is permanently checked against all defining chronicles at robot run time to extract fact hypotheses. [15] argues that this kind of online chronicle recognition is computationally cheap. [15] assumes that chronicles get hand-crafted as part of the domain modeling together with programming the robot behaviors.

Some general remarks about the use of activation values for fact extraction. First, the activation values of our DD control system (Sec. 2) are well-suited for the task in that their formal background in dynamical system theory provides both the motivation and the mathematical inventory to make them change smoothly over time. This typical smoothness is handy for defining *qualitative activations* and serve as a stable basis for chronicle recognition. Second, activation values are calculated anyway in the BBS and hence can be used for free for fact extraction. Third, activation values are by their definition event-based and noise-filtered. This reduces drastically the necessary effort for reliable fact extraction.

To describe our new learning approach for fact hypothesis, which is alternative or complementary to the approach of [15] just sketched, we first present its learning method.

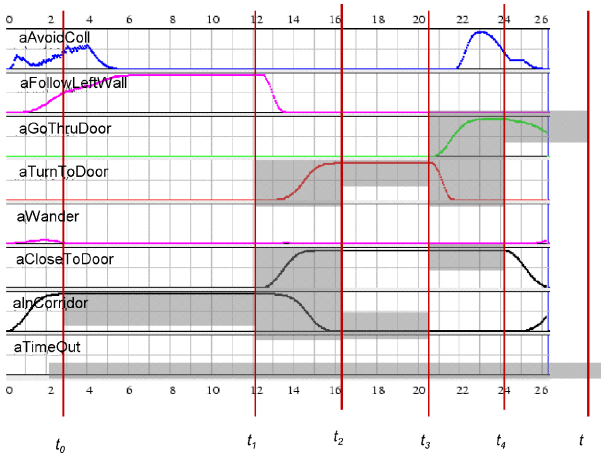

Figure 2. Activation value history plot overlayed with chronicle definition (grey shading), giving strong evidence that the robot is inside a room rather than in the corridor.

4 ECHO STATE NETWORKS

Extracting non-trivial symbolic facts from sensor/motor/activation time series requires a filtering mechanism with some short-term memory "capabilities", because the time series – e.g. events – need to be integrated over some time. A potentially very powerful class of filters with substantial memory capacity is recurrent neural networks (RNNs). However, RNNs are rarely considered as a practical tool because known training algorithms [3] are difficult to use, converge slowly, and have principal limitations in achieving substantial memory spans. *Echo state networks* are a novel approach to RNN training which overcomes these and other known difficulties, making RNNs a practical alternative. Here we can only sketch the basic principle of this approach; for details see [12].

Under certain conditions, the activation state $\mathbf{x}(n) = (x_i(n))_{i=1,\ldots,N}$ of an N-unit RNN is a function of the input history $\mathbf{u}(n), \mathbf{u}(n-1), \ldots$ presented to the network. I.e., there exists an *echo function* $\mathbf{E} = (e_i(n))_{i=1,\ldots,N}$ such that $x_i(n) = e_i(\mathbf{u}(n), \mathbf{u}(n-1), \ldots)$. Metaphorically, the state $\mathbf{x}(n)$ can be understood as an "echo" of the input history.

The task of "filtering" an input sequence $\mathbf{u}(n), \mathbf{u}(n-1), \ldots$ such that an interesting output is generated by the filter, formally amounts to implementing a filter function $F(\mathbf{u}(n), \mathbf{u}(n-1), \ldots)$. Let us consider the case of a single output variable. We have then a network architecture where a "reservoir" RNN is combined with a single output unit. The desired output variable F is the activation of this output unit. The basic idea of the echo state approach is to combine F from the excited dynamics available at the RNN units, i.e. to put $F(\mathbf{u}(n), \mathbf{u}(n-1), \ldots) \approx \sum_i w_i x_i(n) = \sum_i w_i e_i(\mathbf{u}(n), \mathbf{u}(n-1), \ldots)$.

The training task amounts to finding weights w_i for the connections leading from the "reservoir" network to the output unit, such that the mean square error of the network output (compared to the correct teacher output) is minimal. This is a supervised training scheme. Since no cyclic interdependencies exist between the w_i, this task can be computed by a simple linear regression.

To sum up, conventional approaches to RNN training attempt to adjust *all* weights within the RNN. Because of cyclic dependencies, this is only feasible with *small* networks (order of 20 units). By contrast, in the echo state approach, a *large* RNN is used (order of hundreds of units). This network is not modified during learning. It serves as a "reservoir" of a rich collection of excitable dynamics, which is "tapped" via the output weights w_i, which are determined in a constructive training algorithm by minimizing the mean square error of network output vs. teacher signal over the training data. *Only* the RNN-to-output connections are adjusted.

The echo state approach has been applied to a large variety of tasks, many of which were inaccessible to RNN modeling before. A highlight is echo state prediction models for chaotic systems, which (for the Mackey-Glass benchmark system) outperform existing black-box modeling techniques by two orders of magnitude [12].

5 TRAINING ECHO STATE NETWORKS TO HYPOTHESIZE SITUATION FACTS

We want to derive assertions of a fact φ from sensor/motor/activation input histories $\mathbf{u}(n), \mathbf{u}(n-1), \ldots$, where the latter are (possibly high-dimensional) traces of various sensor etc. variables available to the robot. Coding the truth/falsity of φ by values 1 and 0, we wish to realize $F(\mathbf{u}(n), \mathbf{u}(n-1), \ldots)$ as a 0-1-valued indicator function that returns 1 when φ is perceived to hold, else 0.

To realize F by an echo state network, one first has to generate training data. We used the simulated environment of Sec. 3 (Fig. 1). Various variables available to the robot were sampled: raw sensor data (e.g., infrared sensor readings, together 10 channels), processed sensor data (e.g., robot heading, 3 channels), motor measurement data (robot speed and angular velocity readings, 3 channels), and activations of the nine LLBs and HLBs of the behavior-based control system of Sec. 3 (altogether 9 channels). The sampling rate was 4 Hz in simulated time.

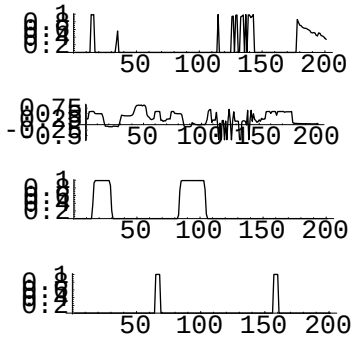

Figure 3. Four channels of the training data. Simulated time shown approx. 50 secs (200 time cycles). First: raw readings of an infrared distance sensor. Second: Right motor velocity. Third: Activation of LLB TurnToDoor. Fourth: Hand-coded fact flag φ_C.

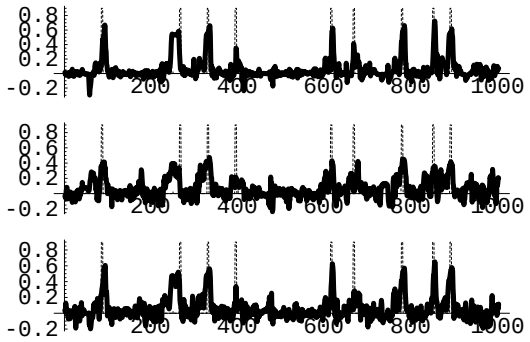

Figure 4. Network output (solid thick line) and teacher (dashed) for the fact φ_D = "robot passes through any door". First graph: behavior activation and motor variables available to network. Second: sensor and motor variables. Third: all variables.

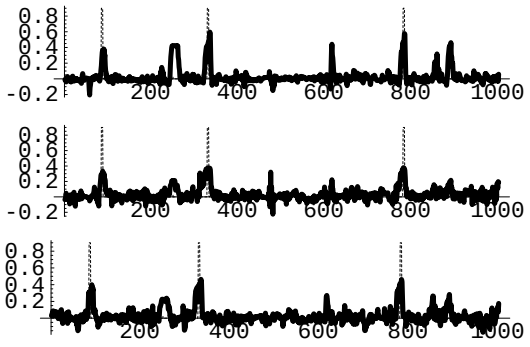

Figure 5. Same as in Fig. 4 for φ_R = "robot passes through door that leads from corridor into a room".

In this article, we consider only facts φ_X that correspond to events X of short duration, which is why we may use the facts and the events signaling the facts interchangeably in the following.[2] Three kinds of facts $\varphi_D, \varphi_R, \varphi_C$ were asserted by hand in three extra channels. They were put from 0 to 1 for one simulated second when events D = "robot passes through any door", R = "robot passes through door that leads from corridor into a room", and C = "robot passes by a 90 degree corner" occurred. Note that R events are a proper subset of D events in the arena in Fig. 1 (and therefore $\varphi_R \subset \varphi_D$).

Fig. 3 shows a portion of the training data. The data was used to train an echo state network with three output units (for $\varphi_D, \varphi_R, \varphi_C$ – note that these output units are trained and operate independently and simultaneously, tapping from the same "reservoir" RNN). The network was a 100 unit RNN, with sparse, random connectivity (5 %) and random weights. The weights were scaled such that the network weight matrix had a spectral radius of $|\lambda_{\max}| = 0.95$, which brings about a long short-term memory span of the RNN of about 20 update cycles, i.e., about 5 seconds simulated time (an explanation of these matters is beyond the scope of this article, see [12, 13] for details). Weights from input units to the network were put at random values with a random connectivity of 30%. The same network was used in all experiments reported below.

6 EXPERIMENTS

From the data collected in the simulation run, roughly the first simulated 15 minutes (4000 update cycles) were used for training and the remaining 5 minutes (1000 cycles) for testing. During the training period, a subset of the sensor/motor/activation variables was fed into the "reservoir" network. In different experiments, different subsets were chosen. The resulting network states were written into memory. At the end of this period, the output weights were computed such that the mean square error $4000^{-1} \sum_{n=1}^{n=4000} (\varphi_X(n) - W_X \mathbf{x}(n))^2$ was minimized. Here, $X = D, R, C$; W_X is the vector of weights leading from the RNN to the corresponding output units; and $\mathbf{x}(n)$ is the network state at cycle n. From a computational point of view, this amounts to computing the pseudo-inverse of the rectangular matrix consisting of the 4000 networks state vectors – a single operation for which many efficient implementations are available (we used the Mathematica routine). The network was then ready for exploitation. For testing, the sensor/motor/activation variables from the remaining 1000 cycles were fed into the network, and the network output was visually compared to the correct (hand-coded) teacher response.

Here we report three times three experiments. In the first of each group, only motor variables and behavior activations were fed into the network. In the second, only sensor variables (including motor variables) were used. Finally, in a third experiment all available robot variables were exploited. Figure 4 shows the results of the three experiments for the fact φ_D; Figs. 5, 6 for the facts φ_R and φ_C.

A detailed analysis of the findings cannot be attempted here due to space limitations. We can only point out some main observations:

- The learning task was essentially mastered in all 9 conditions in the Fig.s 4–6 with no false negatives and very few false positives.
- With the exception of the φ_D fact, best results were obtained when all available variables were used. φ_D was best filtered by the network that obtained behavior activations and motor variables.
- The tricky distinction between events D and R was mastered (remember R events are a subset of D events).

Let us also mention some observations from similar experiments not presented here:

- A network size of 100 units is roughly appropriate for this task:

[2] "State" facts that hold over a period of time would have to be modeled by recognizing to-be-defined start and end events; for example, the fact InRoom of being in some room would start to hold after φ_R (but not φ_D). Details are out of scope here.

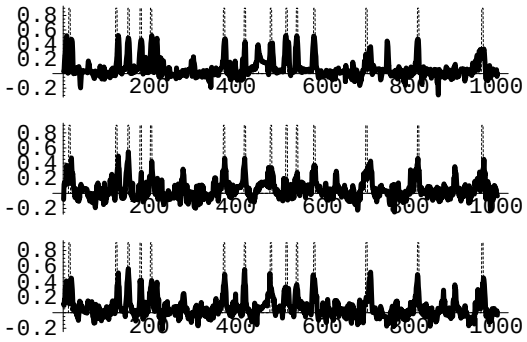

Figure 6. Same as in Fig. 4 for φ_C = "robot passes by a 90 degree corner".

a 400 unit network did not lead to significantly better results, whereas a 25 unit network performed poorly.

- Relevant global scaling network parameters (like the $|\lambda_{\max}|$ mentioned above, see [12] for others) can be scaled in a wide range without significant changes in performance. From the perspective of echo state network theory, the present filtering task appears to be quite simple; "any reasonably-sized network does the job".
- Longer training sequences result in a better signal/noise ratio of the trained network output, but useful networks can be obtained already from very short training sequences (for conspicuous facts, training from a single positive example is sufficient).

The experiment reported here was run off-line on data traces recorded from the simulator. Computation time for the complete experiment (RNN training of 6 simultaneous events including the $3\,\varphi_X$ above plus test data) was 110 sec. for our non-optimized echo state network implementation in Mathematica on a Pentium 500 MHz processor. Much of that computation time was used for producing in Mathematica graphical output like that shown in the Fig.s 4 – 6.

7 DISCUSSION AND CONCLUSIONS

Our main lesson learned from the experiments reported here is that the approach is definitely worth pursuing. As the network computation at execution time runs orders of magnitude faster than the actual robot action and at low computation cost, the fact hypothesis generation can run on-line on board a robot. Learning time (even in our experimental Mathematica implementation) is also very tolerable.

The detection results as visualized in Figs. 4 – 6 are still imperfect in several respects. First, we have no general picture by now, which set of variables yields the best learning result. Using activation values does help; using more variables does not normally help. Second, we have not yet presented the function for filtering from the network outputs the binary fact hypothesis signals. For this paper, we have left it at the qualitative impression that you get from the overlays of raw output and teacher signals as in Figs. 4 – 6. Third, this is simulator data, and although we are confident (owing to the richness of the simulator) that they will scale up well to real robot data, we do not have those yet.

However, the potential of the approach is already obvious. Before, the chronicle-based approach of [15] was available, which is white-box in the sense that the chronicle definitions require the programmer's understanding of the domain model and of the behaviors. The new approach is black-box in the sense that fact hypotheses can be trained without knowing *any* detail of the robot's control system. On the other hand, the chronicle approach is transparent in that chronicle definitions and matches can be inspected and interpreted, whereas the our echo state RNNs are as black-box as any neural network.

There are other approaches dealing with learning in the autonomous robot control context. Some kind of "extended" short-term memory is needed as a prerequisite to handle non-trivial time related dependencies between sensor readings and the world. In [4, 8] this is similar to our RNN based approach, but direct sensory-motor-connection are learned which are strongly related to the concrete environment, the particular behavior system and the current robot task. Making knowledge "explicit", like in the symbol grounding case, overcomes most of such deficiencies. We use symbol grounding in a very general way, [16] uses it to find a consensus for a communication between a group of robots perceiving the same physical object.

It is a main part of our future research to examine in practice the complementarity between chronicle-based and echo state RNN-based fact hypotheses. Our goal is to use them both, complementing their results with the respective other ones whenever possible. If grounding symbolic facts in robot sensor data is a challenge, then this seems like a good strategy.

REFERENCES

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, 'An architecture for autonomy', *Intl. J. of Robotics Res.*, **17**(4), 315–337, (1998).
[2] R. Arkin, *Behavior-Based Robotics*, MIT Press, Cambridge, MA, 1998.
[3] A.F. Atiya and A.G. Parlos, 'New results on recurrent network training: Unifying the algorithms and accelerating convergence', *IEEE Trans. Neural Networks*, **11**(3), 697–709, (2000).
[4] B. Bakker, 'Reinforcement learning with long short-term memory', *Advances in Neural Information Processing Syst.*, **14**, to appear, (2002).
[5] P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack, 'Experiences with an architecture for intelligent, reactive agents', *J. Expt. Theor. Artif. Intell.*, **9**, 237–256, (1997).
[6] A. Bredenfeld, Th. Christaller, J. Hermes, G. Indiveri, H. Jaeger, H.-U. Kobialka, P. Plöger, P. Schoell, and A. Siegberg, 'GMD-Robots', in *RoboCup 2000*, ed., P. Stone, 579–582, Springer (LNCS), (2001).
[7] S. Coradeschi and A. Saffiotti, eds. *Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems: Papers from the 2001 AAAI Fall Symposium*, Technical Report FS-01-01. AAAI Press, 2001.
[8] P. Gaussier, A. Revel, C. Joulain, and S. Zrehen, 'Living in a partially structured environment: How to bypass the limitations of classical reinforcement techniques', *Robotics and Autonomous Systems*, **20**, (1997).
[9] M. Ghallab, 'On chronicles: Representation, on-line recognition and learning', in *Proc. Principles of Knowledge Representation and Reasoning (KR-96)*, eds., Aiello, Doyle, and Shapiro, pp. 597–606, (1996).
[10] S. Harnad, 'The symbol grounding problem', *Physica D*, **42**, 335–346, (1990).
[11] J. Hertzberg, H. Jaeger, U. Zimmer, and Ph. Morignot, 'A framework for plan execution in behavior-based robots', in *Proc. 1998 IEEE Int. Symp. on Intell. Control (ISIC-98)*, pp. 8–13, (September 1998).
[12] H. Jaeger, 'The echo state approach to analysing and training recurrent neural networks', Report 148, GMD, Sankt Augustin, Germany, (2001).
[13] H. Jaeger, 'Short term memory in echo-state networks', Report 152, GMD, Sankt Augustin, Germany, (2001).
[14] H. Jaeger and Th. Christaller, 'Dual dynamics: Designing behavior systems for autonomous robots', in *Proc. Int. Symp. Artificial Life and Robotics (AROB'97)*, eds., Fujimura and Sugisaka, pp. 76–79, (1997).
[15] F. Schönherr, M. Cistelecan, J. Hertzberg, and Th. Christaller, 'Extracting situation facts from activation value histories in behavior-based robots', in *KI-2001: Advances in Artificial Intelligence (Joint German/Austrian Conference on AI, Proceedings)*, eds., F. Baader, G. Brewka, and T. Eiter, pp. 305–319. Springer (LNAI 2174), (2001).
[16] P. Vogt, 'Symbol grounding in communicative mobile robots', in *Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems: Papers from the 2001 AAAI Fall Symposium*, eds., S. Coradeschi and A. Saffiotti, Tech. Report FS-01-01, pp. 87–94. AAAI Press, (2001).
[17] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, 'The CLARAty architecture for robotic autonomy', in *Proc. 2001 IEEE Aerospace Conference, Big Sky, Montana*, (March 2001).