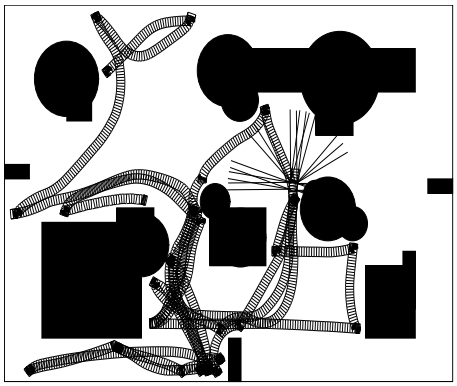

Figure 2: start of training

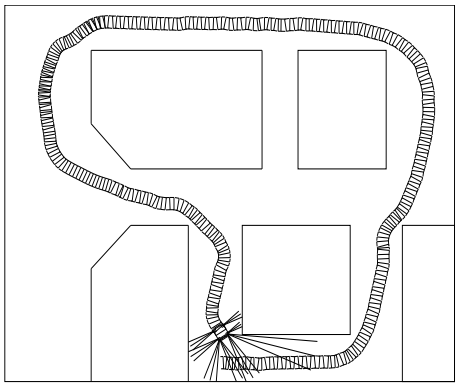

Figure 3: end of training, test environment

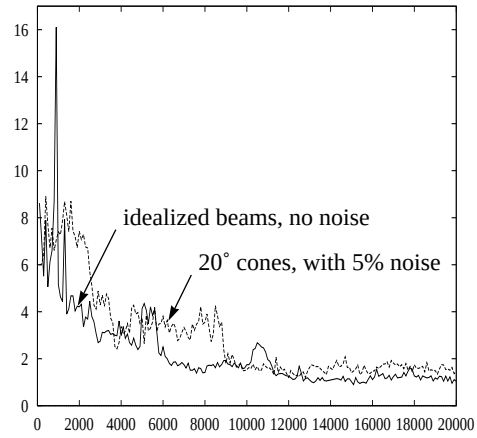

Figure 4: TD-error versus #trials

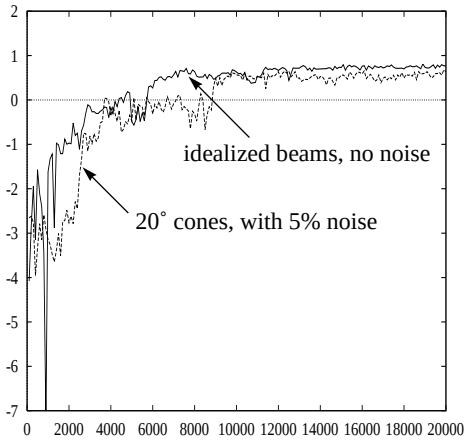

Figure 5: Reinforcement versus #trials

References

1. I. Ahrns: Ultraschallbasierte Navigation und adaptive Hindernisvermeidung eines autonomen mobilen Roboters, Master Thesis, Inst. f. Inf. u. Prakt. Math., CAU zu Kiel (1996)
2. I. Ahrns, J. Bruske and G. Sommer: On-line Learning with Dynamic Cell Structures, ICANN'95 (1995) 141-146
3. H.R. Beom and H.S. Cho: A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning, IEEE Trans. Systems, Man and Cybernetics, Vol. 25, No. 3 (1995) 464-477
4. H. R. Berenji and P. Kheda: Learning and Tuning Fuzzy Logic Controllers Through Reinforcements, IEEE Transactions on Neural Networks, Vol. 3, No. 5 (1992) 724-740
5. J. Bruske and G. Sommer: Dynamic Cell Structure learns Perfectly Topology Preserving Map, Neural Computation, Vol. 7, No. 4 (1995) 845-865
6. B. Fritzke: Growing cell structures - a self-organizing network for unsupervised and supervised learning, Neural Networks, Vol. 7, No. 9 (1995) 1441-1460
7. B. Kroese and J. Dam: Adaptive state space quantization for reinforcement learning of collision-free navigation, Proc. of the Int. Conf. on Intel. Robots and Systems (1992) 1327-1333
8. C. Lin and C. S. Lee: Neural-Network-Based Fuzzy Logic Control and Decision Systems, IEEE Transactions on computers, Vol. 40 (1991) 1320-1336
9. Thomas Martinetz and Klaus Schulten: Topology Representing Networks, Neural Networks, Vol 7 (1994) 505-522
10. D. Nauck, F. Klawonn and R. Kruse: Neuronale Netze und Fuzzy-Systeme, Vieweg Verlag (1994)
11. T. Prescott and J. Mayhew: Obstacle Avoidance through Reinforcement Learning, NIPS (1992) 523-530
12. P. Reignier: Fuzzy logic techniques for mobile robot obstacle avoidance, Robotics and Autonomous Systems, Vol. 12 (1994) 143-153
13. R. S. Sutton: Learning to Predict by the Methods of Temporal Differences, Machine Learning, No. 1 (1988) 9-44
14. R.J. Williams: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning, Machine Learning, No. 8 (1992) 229-256

Finally, the centers $\bar{c}^i$ of the bmu and its topological neighbors are updated according to an error modulated Kohonen rule as described in [2]:

$$\Delta\bar{c}^i = \varepsilon_{modulated} err_{VQ} \text{ with } err_{VQ} = \bar{s} - \bar{c}^i. \quad (10)$$

The error we use for modulation is the TD-error which serves as the resource for our DCS network. The lateral connection structure of the DCS is adapted as well, again refer to [2] for details.

A new neural unit (rule) is inserted whenever the distance to the current best matching unit is too large or the resource value of a unit exceeds a threshold. At most N_{max} units are inserted. Using the TD-error as a resource value and thereby as a criterion for rule insertion serves to disambiguate regions of the input space where similar motor actions result in different rewards.

4 Experiments

In order to test the applicability of our learning controller to collision avoidance with the TRC Labmate the simulated Labmate was placed in the training environment depicted in Figure 2. The Labmate was then allowed to drive around until either the distance to an obstacle dropped below 20cm or 200 time steps elapsed, ending a trial. In the former case, an orientation behavior is triggered which causes the Labmate to rotate until the front sensors indicate free space. In the latter case the Labmate is stopped and rotated for a random angle (to prevent it from staying on a closed trajectory all the time). Since we want to test the performance of the controller independent of incorporated prior knowledge the controller started with only one fuzzy rule:

$$\text{if } (s_1 \approx 5000) \wedge \ldots \wedge (s_8 \approx 5000) \text{ then } (\bar{\mu}_v \approx v_0) \wedge (\bar{\mu}_\omega \approx 0) \wedge \left(\bar{\sigma}_v \approx \sigma_v^0\right) \wedge \left(\bar{\sigma}_\omega \approx \sigma_\omega^0\right), \quad (11)$$

stating that if all sensor readings are about 5m the Labmate should drive forward (zero angular velocity $\bar{\mu}_\omega$) with velocity v_0. The certainty values for forward velocity and angular velocity ($\bar{\sigma}_v$, $\bar{\sigma}_\omega$) were set to small initial values (σ_v^0, σ_ω^0).

As immediate reinforcement $r(t)$ we used the difference between evaluations of two succeeding situations, $r(t) = \Psi(\bar{s}_t) - \Psi(\bar{s}_{t-1})$, with $\Psi: \Re^8 \rightarrow \Re$ the evaluation function of a sensory situation. In addition the Labmate was given a high negative reinforcement signal if it had approached an obstacle within less than 20cm[2].

For a typical run Figure 2 shows the Labmate at the beginning of training in the training environment. Figure 3 shows collision free navigation of the Labmate in a test environment after training phase. End of training is indicated by the averaged TD-error approaching a minimum, the averaged reinforcement approaching its maximum and - of course - avoidance of collisions. Plots for the TD-error and the reinforcement (both averaged over 100 trials) are depicted in Figure 2 and Figure 5. In our experiments training took between 1000 and 10000 trials, taking (on average) a longer time when sonar sensors with a characteristic beam width of 20˚ and 5% noise were simulated than simulating idealized sensors (0˚ beam width) without noise. However, the difference between these two types of simulated sensors turned out to be surprisingly small. At most $N_{max} = 100$ neural units (rules) have been utilized. No effort has been spent on parameter optimization.

5 Discussion

We have presented an integrated architecture for neuro fuzzy control based on a DCS network for learning from delayed reinforcement and applied it to the task of learning reactive collision avoidance for a simulated TRC Labmate. Conditions were unusually hard using unprocessed readings from eight sonar sensors as an input and trying to learn a continuous forward and angular velocity. Our experiments confirm that the controller is indeed able to learn collision avoidance from reinforcement and, furthermore, indicate that the controller in spite of remaining plasticity behaves stable, i.e. we did not observe any breakdowns in avoidance performance. However, they also underpin that learning from reinforcement is usually a very slow process, although tuning of learning parameters may help to improve performance. Hence the necessity for augmentative mechanisms to support pure reinforcement learning. These may comprise extensions of the reinforcement learning model (e.g. by utilizing action models), additionally learning mechanisms (e.g. some self-supervised or supervised learning) and, of course, incorporation of prior knowledge. The latter has been omitted in the reported experiments to better demonstrate the learning capabilities of the controller. As is evident from Figure 1, the architecture is open for a supervised learning component as well. In this case the supervised training error can replace/ augment the TD-error signal backpropagated through the ASE.

Future work of the authors will be concerned with testing the adaptive controller on the real Labmate, starting from a set of carefully designed fuzzy logic rules. Furthermore, we will provide components for supervised and self-supervised learning.

---

2. Due to lack of space the reader is referred to [1] for further details, including parameter values and the form of the evaluation function.

3 Learning Architecture & Algorithms

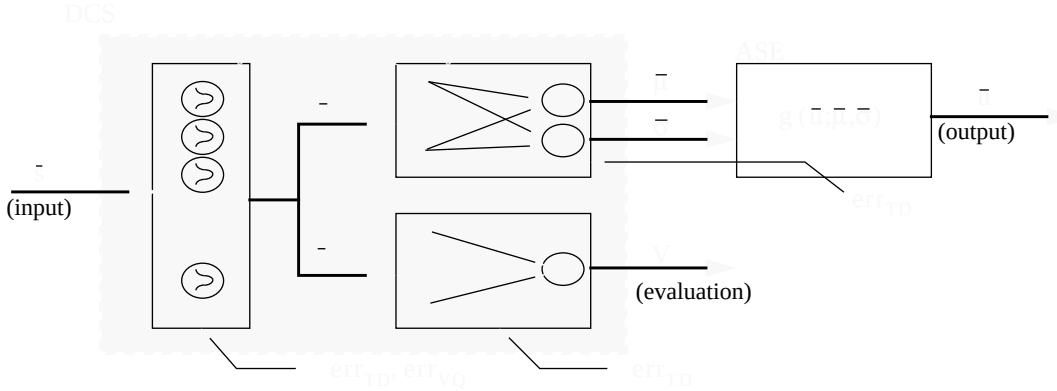

Figure 1: Controller architecture

As depicted in Figure 1 the controller is realized by a single DCS network with an additional stochastic associative search element (ASE) for REINFORCEment learning. The calculation of the control vector $\bar{u}$ given the sensory input $\bar{s}$ proceeds as follows:

First, the input vector $\bar{s}$ is transformed to an activity vector $\bar{a}$ representing the normalized activations (degrees of fulfillment) of the rbf units (rules) with centers $\bar{c}_i$ and uniform width d:

$$\bar{a}_i = \frac{rbf_i(\bar{s})}{\sum_{j \in Nh(bmu_{\bar{s}})} rbf_j(\bar{s})} \text{ with } rbf_i(\bar{s}) = \exp\left(\frac{-(\bar{s}-\bar{c}^i)^2}{d^2}\right) \forall i \in Nh(bmu_{\bar{s}}) , \quad (5)$$

where $Nh(bmu_{\bar{s}})$ denotes the best matching unit (rule), $bmu_{\bar{s}}$, and its direct topological neighbors w.r.t the lateral connection structure of the DCS network.

Second, a prototypical action vector $\bar{\mu}$, the certainty vector $\bar{\sigma}$ and the predicted cumulative reinforcement V are calculated by a weighted sum of contributing vectors (consequent functions) attached to the rbf units:

$$\bar{\mu} = \sum_{j \in Nh(bmu_{\bar{s}})} \bar{a}_j \bar{\mu}^j , \; \bar{\sigma} = \sum_{j \in Nh(bmu_{\bar{s}})} \bar{a}_j \bar{\sigma}^j \text{ and } V = \sum_{j \in Nh(bmu_{\bar{s}})} \bar{a}_j V_j . \quad (6)$$

Finally the ASE draws the actual action vector $\bar{u}$ according the probability density function $g(\bar{u};\bar{\mu},\bar{\sigma})$, the components of g being Gaussians:

$$Prob(\bar{u}_i) = g_i(\bar{u};\bar{\mu},\bar{\sigma}) = N(\bar{\mu}_i,\bar{\sigma}_i) . \quad (7)$$

On-line adaptation is performed w.r.t. the contributing prototypical action vectors $\bar{\mu}^j$, certainty vectors $\bar{\sigma}^j$ and evaluation values V_j attached to the rbf units (consequent - part) as well as to the centers $\bar{c}^j$ of the rbf units (antecedent - part).

The evaluation values V_j are updated using a TD(1) rule:

$$\Delta V_j = \beta \, err_{TD}(t) \sum_{\tau=0}^{t} \gamma^{t-\tau} \frac{\partial}{\partial V_j} V(\bar{s}_\tau) , \quad (8)$$

where $err_{TD}(t) = r(t) - b(t)$ denotes the current temporal difference error with $r(t)$ the reinforcement signal and $b(t) = V(\bar{s}_{t-1}) - \gamma V(\bar{s}_t)$ an adaptive baseline.

Prototypical action vectors $\bar{\mu}^j$ and certainty vectors $\bar{\sigma}^j$ are adapted using a REINFORCE gradient descent:

$$\Delta \bar{\mu}_i^j = \beta_\mu (r-b) \frac{\partial}{\partial \bar{\mu}_i^j} \ln(g_i(\bar{u};\bar{\mu},\bar{\sigma})) \text{ and } \Delta \bar{\sigma}_i^j = \beta_\sigma (r-b) \frac{\partial}{\partial \bar{\sigma}_i^j} \ln(g_i(\bar{u};\bar{\mu},\bar{\sigma})) . \quad (9)$$

The REINFORCE framework [14] states that equation (9) implements a gradient descent on the expected reinforcement (at least for a constant baseline b). When the algorithm converges towards a local maximum of the reinforcement the $\bar{\sigma}^j$ will decrease to small values, narrowing the range of stochastic search. Hence the term certainty values: If we pre-structure the network with fuzzy rules we can specify the search range for the conclusion of this rule by specifying its $\bar{\sigma}^j$ vector. Values close to zero result in non-changing consequents (fixed rules). On the other hand, if we analyze the network at consecutive time steps, non decreasing components of $\bar{\sigma}^j$ indicate convergence to (certainty about) the corresponding prototypical action.

2.1 Normalized Radial Basis Function (NRBF) Networks

Normalized Radial Basis Function (NRBF) Networks are function approximators and calculate their output as[1]

$$\bar{y}(\bar{x}) = \frac{\sum_{i=1}^{K} \bar{o}^i h_i(\|\bar{x} - \bar{c}^i\|)}{\sum_{i=1}^{K} h_i(\|\bar{x} - \bar{c}^i\|)}, \quad (1)$$

where $h_i(\|\bar{x} - \bar{c}^i\|)$ denotes a radial basis function with center $\bar{c}^i$. The vectors $\bar{o}^i$ can be thought of as output weight vectors attached to each "hidden" rbf unit. Usually the $h_i: \Re^+ \to \Re$ are strictly monotonically decreasing functions with $h_i(0) = 1.0$ and $h_i(\infty) = 0$, most often implemented by Gaussians.

2.2 Sugeno type fuzzy control

A *Sugeno type fuzzy controller* [10] consists of a set of K linguistic fuzzy rules

$$R_r: \textit{if } \text{antecedent}_r(\bar{x}) \textit{ then } \bar{\eta}^r = \bar{f}^r(\bar{x})$$

and calculates its output according to the defuzzyfication formula

$$\bar{\eta}(\bar{x}) = \frac{\sum_{r=1}^{K} \bar{f}^r(\bar{x}) \tau_r(\bar{x})}{\sum_{r=1}^{K} \tau_r(\bar{x})}, \quad (2)$$

where $\tau_r(\bar{x})$ is the degree of fulfillment of the r.*th* rule.

Obviously, (1) and (2) become identical if we restrict the consequent functions to constant functions, i.e. $\bar{f}^r(\bar{x}) = \bar{o}^r$, the antecedents to fuzzy propositions of the form

$$\text{antecedent}_r: \text{“X is } T_X^{i_r}\text{“}, \quad (3)$$

where $T(X) = \{T_X^1, \ldots, T_X^k\}$ is the term set of variable X with n-dimensional universe of discourse $U(X) = \Re^n$, and the membership functions M_X^i to radial basis functions.

Hence adapting the parameters of an NRBF network (output weight vectors $\bar{o}^i$, centers $\bar{c}^i$ and widths of the basis functions) can be interpreted as fine tuning of the consequent and membership functions of a (restricted) Sugeno type fuzzy controller. Incremental growing of the NRBF network (insertion of new rbf-units) can be interpreted as rule generation for such a controller.

Finally, if the membership functions M_X^i are Gaussians and the fuzzy conjunction is implemented as algebraic product, antecedents may be alternatively written as conjunctions of n propositions

$$\text{antecedent}_r: \text{“}X_1 \text{ is } T_{X_1}^{i_{1r}}\text{“ and ... and “}X_n \text{ is } T_{X_n}^{i_{nr}}\text{“} \quad (4)$$

where the X_i denote linguistic variables with one dimensional universes of discourse and one dimensional Gaussian membership functions given as the marginal possibility distributions of the M_X^j.

2.3 Dynamic Cell Structures (DCS)

Dynamic Cell Structures (DCS) as introduced in [5] denote a class of RBF-based approximation schemes attempting to concurrently learn and utilize perfectly topology preserving feature maps (PTFMs). DCS are a subclass of Martinetz's Topology Representing Networks (TRN) [3] defined to contain any network using competitive Hebbian learning for building PTFMs.

The architectural characteristics of a DCS network are a) one hidden layer of radial basis functions (possibly growing/ shrinking) b) a dynamic lateral connection structure between these units and c) a layer of (usually linear) output units. Training algorithms for DCS adapt the lateral connection structure towards a PTFM by employing a competitive Hebbian learning rule and activate and adapt rbf units in the neighborhood of the current stimulus, where "neighborhood" relates to the simultaneously learned topology.

Hence viewed from the perspective of fuzzy control, normalized DCS represent Sugeno type fuzzy controllers with the additional feature that only the best matching rules are evaluated and adapted. Learning and exploiting the topology of the input space is not only likely to improve control but also speeds up the computation of the control value because only a small number of rules need to be evaluated. See [5] for a more detailed discussion of this topic.

The particular DCS network used in this article is similar to the one introduced in [2] being especially tuned for on-line learning. Incremental growing of the network (generation of new control rules) is based on a local error variable attached to each neural unit (rule) called the resource of this unit, similar to [6].

---

1. In the following we denote vectors by $\bar{x}$, components of a vector by $\bar{x}_i$ and enumerations of vectors by $\bar{x}^i$.

Neural Fuzzy Control based on Reinforcement Learning and Dynamic Cell Structures

to appear in: EUFIT'96, Proc. of the 1996 Conference, Aachen

Jörg Bruske, Ingo Ahrns and Gerald Sommer

Lehrstuhl für Kognitive Systeme, Christian Albrechts Universität zu Kiel,
Preußerstr. 1-9, 24105 Kiel
Tel: 0431/560480 Fax: 0431/560481 email: jbr@informatik.uni-kiel.de

Abstract. In this article we introduce the concept of a *hybrid neuro fuzzy controller* based on *Dynamic Cell Structures* (DCS) [5] and *reinforcement learning*. While basically a *Sugeno type controller* this controller additionally learns and exploits the topology of the input space utilizing the DCS. The controller not only *adapts the parameters* of its fuzzy control rules but also *generates new rules*. Our application is *on-line learning* of a *reactive collision avoidance* behavior for an autonomous mobile robot. More particular, the controller employs a *REINFORCE* [14] algorithm in combination with an *Adaptive Heuristic Critique* (AHC) [13] to learn a continuous valued sensory motor mapping for obstacle avoidance with the TRC Labmate from delayed reinforcement. The sensory input consists of eight unprocessed sonar readings, the controller output is the continuous angular and forward velocity of the Labmate. The controller architecture integrates the controller and the AHC within a single network. Utilizing a *REINFORCE* algorithm offers a sound basis for reinforcement learning and allows to attach an additional *certainty value* to each fuzzy rule.

1 Introduction

According to the terminology set forth in [10], hybrid neuro fuzzy controllers (HNFC) denote artificial neural networks (ANN) obtained by "compiling" a fuzzy controller into an ANN. They can then be adapted using training rules initially developed for ANN. Lin and Lee's Neural-Network-Based Fuzzy Logic Control and Decision System [8] is one of the earliest examples of a HNFC using a combination of off-line unsupervised and supervised training for rule generation and parameter adaptation. Berenji's GARIC architecture [4] is perhaps the most well known example of a HNFC trained by reinforcement learning. GARIC adapts the parameters of an Action Selection Network (ASN), which encodes a fuzzy controller, by means of an Action Evaluation Network (AEN) and a Stochastic Action Modifier (SAM). While conceptually similar to GARIC our control architecture integrates the ASN and the AEN within a single DCS network used for adaptive vector quantization of the input space, learning of the required sensory motor mapping and evaluation of the control policy (see Figure 1). Contrary to GARIC our controller is able to generate new control rules in addition to adapting parameters of existing ones. Utilizing a REINFORCE algorithm we avoid the rather questionable heuristics used in GARIC and, moreover, we are able to specify and adapt a certainty value for each control rule. We will introduce DCS for fuzzy control in section 2 and proceed with a detailed description of the controller as well as the learning algorithms in section 3.

Concerning our application, *reactive* collision avoidance of an autonomous robot rests on the hypothesis that collision avoidance can be realized by a simple mapping between sensory data and motor actions, i.e. without involving a controller state. Since a simple mapping from (sensory) input to (motor) output is the essence of most conventional feed forward ANNs and fuzzy logic controllers their application to reactive navigation tasks is self evident and has been tackled by a number of researchers before using either supervised learning or reinforcement learning to train an ANN, e.g. [7], [11] or to refine a (neuro-) fuzzy controller, e.g. [3]. Fuzzy logic controllers without adaptability have been suggested as well, e.g. [12]. As is evident from the continuing research effort on this field, reactive navigation and collision avoidance in particular is far from being a closed chapter in robotics.

Different to e.g. [7] we associate a continuous sensory input with continuous motor actions, i.e. we learn an appropriate forward velocity and angular velocity, not only to turn either left or right with a fixed velocity. As input we use the unprocessed readings of eight noisy sonar sensors which yields learning more difficult than e.g. the three laser readings used in [11]. Finally, we tried to avoid over-simplified simulations by simulating the geometric, dynamic and sensory characteristics of our TRC Labmate. In a number of experiments, [1], this simulator has proved to be in good accordance with the real Labmate.

2 Radial Basis Functions, Sugeno type Fuzzy Control and Dynamic Cell Structures

In this section we briefly introduce and relate Radial Basis Functions, Sugeno type Fuzzy Control and Dynamic Cell Structures. It turns out that Dynamic Cell Structures can be regarded as restricted Sugeno type Fuzzy controllers with additional learning and exploitation of the topology of the input space.