in Proceedings of IEEE Int. Conf. on Fuzzy Systems, 1997.

Unsupervised Learning of Control Surfaces Based on B-Spline Models

Jianwei Zhang, Khac Van Le and Alois Knoll
Technical Computer Science, Faculty of Technology,
University of Bielefeld, 33501 Bielefeld, Germany

Abstract

Based on our earlier work on construction of fuzzy controllers with B-spline models, we propose an automatical learning approach for generating control vertices of such a type of fuzzy controller. For supervised learning, we point out that rapid convergence of this learning procedure can be guaranteed, which is confirmed by diverse examples of approximating non-linear functions and interpolating training data. For unsupervised learning, we employ a type of state evaluation functions which can be found for a large amount of control problems. Using such an evaluation function, a learning algorithm is devised which modifies the local control action efficiently to guide the system to the desired state. Implementations with the cart-pole balancing and a sensor-based mobile robot validate this learning approach.

1. Introduction

Fuzzy logic systems are now widely accepted as a general mechanism for modelling non-linear processes or systems without mathematic model. However, the flexibility of a fuzzy controller model brings also the difficulty to formulate a systematic design methodology. In a lot of applications "trial-and-error" is still used to tune the controller parameters whose number can be quite large. Therefore the following aspects are crucial for achieving a wider range of applications of fuzzy controllers in industry: a). the guidelines of selecting membership function types, inference/defuzzification methods; and b). an automatic procedure to determine controller parameters for achieving the optimal control effect.

Recently, learning of fuzzy controllers has become an important research topic. Among them, neuro-fuzzy methods provide some important results, especially in supervised learning, [4], [2]. Supervised learning can be applied in modelling like function approximation, forecasting using data interpolation and/or extrapolation, etc. Unfortunately, in most control problems, only rough, incomplete training data can be prepared or the input/output data are even not available at all. To apply fuzzy control in a mobile robot, for example, [3] uses a supervised learning approach to train a fuzzy controller for collision-avoidance. The training data are generated by the Braitenberg method. Obviously, the "teacher" should do well enough for the "student" to learn and the "student" cannot behave better than the "teacher" after being trained.

Unsupervised learning is more interesting and more challenging for both neural networks and fuzzy logic systems. Some advances have been made in *reinforcement learning*. Q-learning proposed in [6] is a representative method. In [1] the reinforcement learning is introduced to fuzzy controllers. Efficient and on-line learning methods for general unsupervised learning of fuzzy controllers still need further investigation.

We propose an approach which can systematically build the fuzzy sets for linguistic terms of the IF-part while the fuzzy sets of the THEN-part can be adapted through learning. The model of linguistic terms in our approach is based on B-spline basis functions, a special set of piecewise polynomial curves. Since modification of control vertices mainly changes of the control surface, learning of the controller is transformed to learning of positions of fuzzy-singletons of output variables.

In our earlier work [7], we presented the basic idea of employing B-spline basis functions of different orders to model linguistic terms, which is briefly summarised in section 2. The learning model of this type of controller devised for supervised learning is reviewed in section 3.1. Inspired by the learning function of the supervised case, an approach for unsupervised learning using a type of evaluation functions is described in section 3.2-3.4. The implementations of two control examples are demonstrated in section 4 and 5. Section 6 draws some conclusions.

2. B-Spline Fuzzy Controllers

In [5], the conventional eight types of continuous parametric membership functions (MFs) are summarised and compared in the examples of function approximation (supervised learning). However, the B-spline basis functions provide another kind of fuzzy set representation which possess some special properties.

2.1. B-spline basis function as MFs

Several main types of B-spline basis functions are illustrated in [7]. The basis function of order 1 can be interpreted as for modelling crisp sets, while the type of order 2 as for defining the commonly used triangular fuzzy sets. If the order of the B-spline basis functions is larger than 2, each fuzzy set defined is no longer normalised, but the "partition of unity" is valid for all the values within the universe of discourse of a linguistic variable. The overlapping of more than two fuzzy sets can be too complex for manual specification, but we argue that these fuzzy sets are naturally defined, i.e. given the partition points (knots) of an input variable, they can be automatically computed with the recursive representation; and better performance in smoothness and modelling ability can be achieved.

2.2. Fuzzy controller as interpolator

Under the following conditions: a). periodical B-spline basis functions as MFs for inputs, b). fuzzy-singletons (called *control vertices*) as MFs for outputs, c). "product" as fuzzy-conjunctions, d). "centroid" as defuzzification method, e). adding "virtual linguistic terms" at both ends of each input variables, and f). extending the rule base for the "virtual linguistic terms" by copying the output values of the "nearest" neighbourhood, the computation of the output of such a fuzzy controller is equivalent to that of a general B-spline *hyper-surface*. Generally, if we consider a MISO system with n inputs $x_1, x_2, \dots, x_n$, rules with the n conjunctive terms in the premise are given in the following form: $\{Rule(i_1, i_2, \dots, i_n)$: IF ($x_1$ is $X^1_{i_1,k_1}$) and (x_2 is $X^2_{i_2,k_2}$) and ... and (x_n is $X^n_{i_n,k_n}$) THEN y is $Y_{i_1 i_2 \dots i_n}\}$, where x_j is the j-th input ($j = 1, \dots, n$), k_j is the order of the B-spline basis functions used for x_j, $X^j_{i_j,k_j}$ is the i_j-th linguistic term of x_j defined by B-spline basis functions, $i_j = 1, \dots, m_j$ represents the index of the linguistic term of x_j and $Y_{i_1 i_2 \dots i_n}$ denotes the control vertex of $Rule(i_1, i_2, \dots, i_n)$.

Then the output y of a MISO fuzzy controller is:

$$\begin{aligned} y &= \frac{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} (Y_{i_1,\dots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j))}{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)} \\ &= \sum_{i_1=1}^{m_1} \cdots \sum_{i_m=1}^{m_n} (Y_{i_1,\dots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)) \end{aligned}$$

C^{k_j-2}-continuity of all the outputs with respect to each input variable x_j can be achieved if B-spline basis functions of order k_j are used for fuzzy partitioning x_j.

2.3. Characteristics

Computing parameters of such a B-spline fuzzy system is divided into two steps: for the IF-part and for the THEN-part.

Characteristics of modelling the IF-part are:

- Based on the partitioning points of the input space and the maximal point distribution of the control surface if known, the fuzzy sets can be automatically computed.
- These fuzzy sets can be further adapted during the generation of the whole system. We developed an algorithm for adapting the knots of the IF-part, which is a modified algorithm for self-organising neural networks. Nevertheless, if sufficient B-functions are used for the inputs, the local modification of the knots has only negligible influence on the control surface. Therefore, in the following, we concentrate on the control vertices of the THEN-part which can be automatically achieved through a learning procedure.

Characteristics for generating the THEN-part are:

- For supervised learning, the differentials of the squared error with respect to control vertices are convex functions. Therefore, rapid convergence for supervised learning is guaranteed.
- The control surface changes locally while control vertices are modified. Based on this feature, the control vertices can be optimised gradually, area-by-area. This feature is especially important for a multi-dimensional control hyper-surface with a large amount parameters.

3. Learning Approach

3.1. Supervised learning

Supervised learning assumes that a "teacher" provides all the desired system output for each input data. Based on the complete set of these input/output vectors, B-spline type fuzzy controllers can be trained very rapidly.

Assume $\{(\mathbf{X}, y_d)\}$ is a set of training data, where $\mathbf{X} = (x_1, x_2, \dots, x_n)$ is the input vector and y_d is the desired output for $\mathbf{X}$. The squared error is considered:

$$E = \frac{1}{2}(y - y_d)^2, \tag{1}$$

where y is the current output value during training.

The parameters to be found are $Y_{i_1,i_2,\dots,i_n}$, which make the error in (1) as small as possible, i.e.

$$E = \frac{1}{2}(y - y_d)^2 \equiv \text{MIN}. \tag{2}$$

Each control vertex $Y_{i_1,\dots,i_n}$ can be modified by using the gradient descent method:

$$\begin{aligned} \Delta Y_{i_1,\dots,i_n} &= -\epsilon \frac{\partial E}{\partial Y_{i_1,\dots,i_n}} &(3) \\ &= -\epsilon (y - y_d) \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j) &(4) \end{aligned}$$

where $0 < \epsilon \leq 1$.

The gradient descent method can guarantee that the learning algorithm converges to the global minimum of the error function since the second partial differentiation with respect to $Y_{i_1,i_2,\ldots,i_n}$ is always positive:

$$\frac{\partial^2 E}{\partial^2 Y_{i_1,\ldots,i_n}} = \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j) \geq 0. \quad (5)$$

This means the error function (1) is convex in the space $Y_{i_1,i_2,\ldots,i_n}$ and possesses therefore only one (global) minimum.

We compared the test functions used in [4]. The results show that B-spline fuzzy systems can achieve the same modelling effect as done by the ANFIS (Adaptive-Network-Based Fuzzy Inference System) method, but B-spline fuzzy systems converge faster in most cases.

3.2. Inspiration by supervised learning

In the above examples, a fuzzy system can learn under supervision. Such a learning process needs a teacher, i.e. for each input vector, the desired output should be known. Then the fuzzy controller attempts to interpolate these input/output vectors to provide a continuous (hyper-)surface for the whole control space. Unfortunately, in reality, it is not always simple to find the goal function of the output for a complex system. An unsupervised learning approach should therefore be developed.

We first discuss a control system with $(x_1, x_2, \ldots, x_n)$ as input and y as output. Let us rewrite the modification of the control vertices for supervised learning:

$$\Delta Y_{i_1,\ldots,i_n} = -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} = -\epsilon(y - y_d) \cdot \prod_{j=1}^{n} X_{i_j,k_j}(x_j)$$
$$= sign(y_d - y)\, \epsilon \cdot |y_d - y| \cdot \prod_{j=1}^{n} X_{i_j,k_j}(x_j) \quad (6)$$

We interpret (6) as the following: $sign(y_d - y)$ indicates the direction of the modification of $Y_{i_1,\ldots,i_n}$ in each learning step, while the product $\epsilon \cdot |y_d - y| \cdot \prod_{j=1}^{n} X_{i_j,k_j}(x_j)$ determines the magnitude of the modification.

3.3. Evaluation function for unsupervised learning

In unsupervised learning, it is usually possible to define an "evaluation function" if the desired data of the output are unknown. Such an evaluation function should describe how "good" the current system state $((x_1, x_2, \ldots, x_n), y)$ is. For each input vector, an output is generated, with which the system transits to another state. The new state is compared with the old one; an adaptation of the control vertices is performed if necessary.

Assume the evaluation function, denoted by $F(\cdot)$, is monotonic, i.e. if state A is better than state B, then $F(A) \geq F(B)$. The adaptation of the control vertices can be performed with a similar representation as in supervised learning.

Let us reconsider the modification of the control vertices through the equation (6). State A transits to B by the output y. The desired state is A_d. We consider those systems, for which a function $F(\cdot)$ can be found which fulfills the following condition:

Assume A is the current state and y_1 an arbitrary output. With y_1 the system transits to the state B_1. If another output y_2 fulfills[1] $y_2 \cdot y_1 \leq 0$, and with y_2 the system transits to B_2, the following relation of the evaluation functions is valid: $(F(B_1) - F(A)) \cdot (F(B_2) - F(A)) \leq 0$.

Intuitively interpreted, given two system outputs with different signs, the system states measured by such an evaluation function change from the current state (A) away in two directions ("better" or "worse").

3.4. Modifying control vertices for a single output

At the moment t the system has the state A. The ideal state of the moment $t+1$ would be A_d. Assume that this state is achieved through the controller output y_d which is unfortunately unknown. What really happens is that with the controller output y generated at the moment t, the system transits to the state B.

The important hints for finding the correct output y can be acquired if the state B is compared with A_d. Inspired by the modification of control vertices for supervised learning in (6), we divide the learning into two parts: changing direction and changing rate.

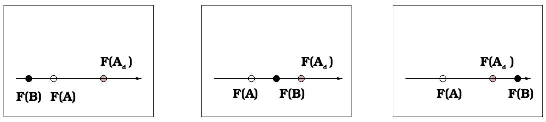
Firstly, the system should find out by itself in which direction the control vertices will be changed since there are normally no explicit representation between the evaluation function and the controller output. Considering the state transition from A to B, the constellation of A, B and A_d results in the following three cases, Fig. 1. The fourth transition type ($F(B) \equiv F(A_d)$) is not shown here since in this case the system does not need to take any action.

(a). The system state becomes worse, i.e. the system acts incorrectly. According to the condition in section 3.3 the change direction is $-sign(y)$.

(b). The system acts in the correct direction. The value of the output should be enlarged. The change direction is then $sign(y)$.

(c). This case is the inverse case of (b). The change direction should be $-sign(y)$.

[1] Under the assumption that y possesses symmetric values in negative and positive areas (with the zero in the middle). In other cases, y can be normalised to fulfill this condition.



(a) (b) (c)

Figure 1. The possible system state transitions (a). the system state becomes worse; (b). the system state becomes better, but the change is too slow; (c). the system state becomes better, but the change is too fast.

These three cases can be synthesised by

$$S = sign(F(A) - F(B)) \cdot sign(F(B) - F(A_d)) \cdot sign(y).$$

Now we replace y with $F(B)$ in (6), and y_d with $F(A_d)$. The change of control vertices can finally be written as:

$$\Delta Y_{i_1,\ldots,i_n} = S \cdot \epsilon \cdot |F(B) - F(A_d)| \cdot \prod_{j=1}^{n} X_{i_j,k_j}(x_j). \tag{7}$$

The control vertices are modified using (7) in a goal-oriented way instead of randomly as in Q-learning. Theoretically, the optimum can be achieved rapidly. The following experiments validate this feature.

4. Learning of Cart-Pole Balancing

The balance of a cart-pole system is often used as a common reference example for control algorithms and learning approaches. Therefore, we firstly apply the learning approach in this control task.

4.1. Input/output

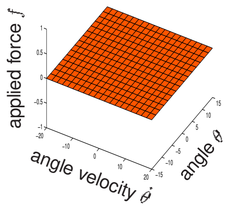
The pendulum possesses an initial state $(\theta, \dot{\theta})$. The control variable is the force f to be exerted, which is able to bring the cart-pole system to the balanced final state $\theta = 0$ and $\dot{\theta} = 0$.

The motion of the pendulum was computed using the *Runge-Kutta* approach with the sample time of 0.01. The inputs of the system are: a). angle: $\theta(^\circ) \in [-15, +15]$ and b). angle velocity: $\dot{\theta}(^\circ/s) \in [-20, +20]$. Each of the two input variables are covered with 7 B-spline basis functions of order 3. The output of the system is the force f to be exerted on the cart.

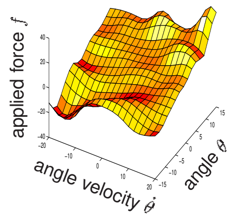
4.2. Learning the control surface

For learning we choose the evaluation function as: $F(A) = F(\theta, \dot{\theta}) \stackrel{def}{=} -|2 \cdot \theta + \dot{\theta}|$, and the relation of the evaluation functions of the desired state A_d and A: $F(A_d) \stackrel{def}{=} 0.5 \cdot F(A)$.

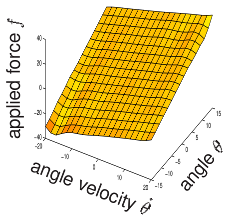
Fig. 2(a), 2(b) and 2(c) depict the force f at the beginning, after 100 learning steps and after 3000 learning steps, respectively. Fig. 2(d) demonstrates the correctness of the learning approach, the evaluation function in this case is not positive and provides zero for the states $(\theta, \dot{\theta})$ which fulfill the following conditions: a). $x_1 \cdot x_2 < 0$ and b). $2 \cdot |x_1| = |x_2|$.



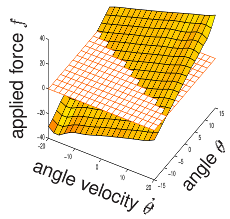
(a) at the beginning



(b) after 100 learning steps



(c) after 3000 learning steps



(d) the best system states (the intersection line)

Figure 2. Applied force f to the cart-pole system.

In other words, these states on the segment $F(\theta, \dot{\theta}) = -|2 \cdot \theta + \dot{\theta}|$ are the best states, which is shown in Fig. 2(d) by the intersection line of the zero-plane and the force f plane.

4.3. Validation

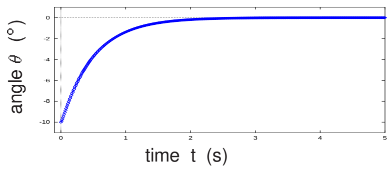
After learning, the cart-pole system can be successfully balanced from any initial states. One of the simulation results can be found in Fig. 3, which depicts the motion profiles of the pendulum from the starting state (θ=-10, $\dot{\theta}$=10). After 2 seconds the pendulum has reached the goal state (θ=0, $\dot{\theta}$=0).

5. Sensor-Based Control of a Mobile Robot

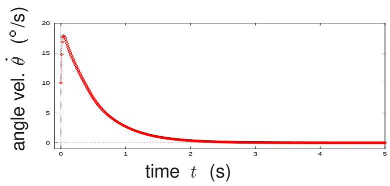
The proposed learning approach was also applied to a real mobile robot system *Khepera*. Our earlier work with *Khepera* was using fuzzy control to integrate planning and control, [8]. In the following, we show how the control vertices of a fuzzy controller can be generated automatically through learning.

5.1. Sensors and control variables

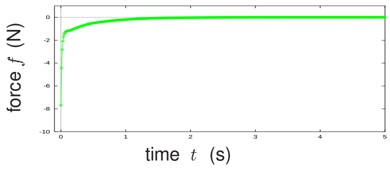
The robot perceives its environment with six infra-red distance sensors, which can be grouped into three vari-



(a) angle



(b) angle velocity



(c) applied force

Figure 3. Input and output profiles of the pendulum starting from the state (-10,10).

ables, Fig. 4. The original sensor values are from 0 to 1023. To reduce the uncertainties, the range of the sensor values is scaled down to the interval $[0, 100]$, using 5 as the increment, i.e. the sensor values as controller inputs are 0, 5, 10, ..., 90, 95, 100. All these inputs are covered with 7 B-spline basis functions as linguistic terms. Since the robot is equipped with only limited on-board computation capability, the order 2 of the basis functions was chosen to make the real-time learning possible.

The two wheels of the robot are controlled independently by two motors. The output of the controller is the steering angle w. A positive value of w steers the robot to the right, and negative value to the left. The velocities of the two wheels can then be computed as: a). for the left wheel: $v + w$ and b). for the right wheel: $v - w$, where v is the current front velocity of the robot.

5.2. Learning to avoid obstacles

Before trying to develop the evaluation function, we first discuss the situations which the robot can possibly face, Fig. 5. In the situations of Fig. 5(b), 5(e), 5(f), the robot should try to keep the difference of *SensL* and *SensR* as small as possible. For the cases shown in Fig. 5(c), 5(d), the robot should try to minimise the sum of all three sensors *SensL*, *SensV* and *SensR*. Fig. 5(g) and 5(h) illustrate

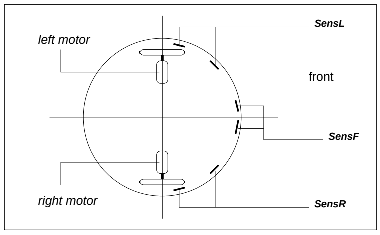
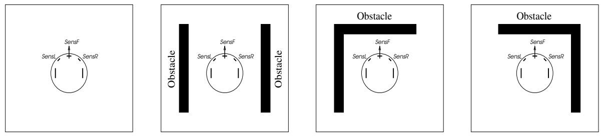
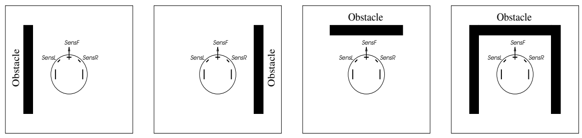


Figure 4. Sensors and motors of the Khepera robot.

two cases, for which no reasonable evaluation function can be found, the robot can simply turn left.



(a) (b) (c) (d)



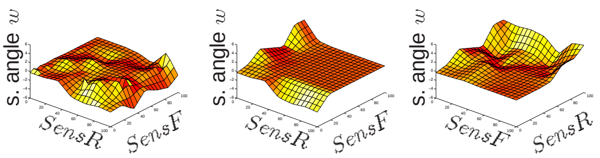
(e) (f) (g) (h)

Figure 5. Possible situations and actions (a): free space, straight forward; (b): in a corridor, straight forward; (c): turn right; (d): turn left; (e): turn right; (f): turn left; (g): turn left; (h): turn left.

The evaluation function F can be summarised as follows:

- $F(SensL, SensF, SensR) = -(SensL + SensF + SensR)$, if $SensF$ is big (Fig. 5(c) and 5(d)).
- $F(SensL, SensF, SensR) = -|SensL - SensR|$, if $SensF$ is small, $SensL$ or $SensR$ is not zero (Fig. 5(b), 5(e) and 5(f)).
- $F(SensL, SensF, SensR) = 0$, if no obstacles present (Fig. 5(a)).
- Otherwise for Fig. 5(g) and 5(h): simply turn left.

As for the cart-pole control, the evaluation function of the desired state is defined by $F(A_d) \stackrel{def}{=} 0.5 \cdot F(A)$. During the learning process, the change of the evaluation function from one state to the next has a big negative value if the robot has falsely reacted; it is slightly positive if the robot steers correctly but not strong enough. Fig. 6 shows the learning results after 1000 learning steps.



(a) w to $SensL = 0$, $SensF$, $SensR$

(b) w to $SensL$, $SensF = 0$, $SensR$

(c) w to $SensL$, $SensF$, $SensR = 0$

Figure 6. Output w after 1000 learning steps.

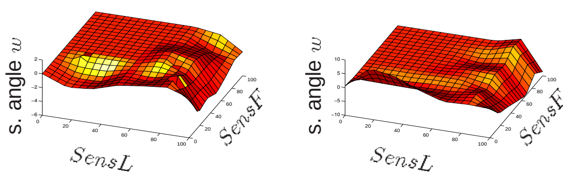
To demonstrate that the robot has learned correctly with the above approach, we perform the following experiment. We change the roles of the wheels, i.e. let the wheels controlled by: a). for the left wheel: $v - w$ and b). for the right wheel: $v + w$. The results shown is now inverted in comparison with Fig. 6 - exactly what we expected.

5.3. Learning to track contours

In this behaviour, the robot does not move way from an obstacle, but tries to "keep the obstacle in the right eye". The robot reacts only if it sees something in its "right eye" or in its "front eye". The evaluation function depends on the last and new sensor values, too.

- $F(SensF, SensR) = (SensF + |SensR - 80|)$, if the robot sees something in the front or on the right,
- zero, otherwise.

After learning, the robot can track the contour of an arbitrary object. The learning results are shown in Fig. 7.



(a) After 100 learning steps

(b) After 1000 learning steps

Figure 7. The output w.

6. Conclusions

The B-spline fuzzy controller is a bridge between a numerical interpolation model and a symbolic rule system. If the rule table is complete, then by adding certain more marginal rules, the smoothness of the controller output can be achieved by selecting the proper order of basis functions. B-spline fuzzy controllers are exact, i.e. no information is lost after the defuzzification. If a user specifies the granularity for partitioning the input space, the control vertices of the output variables which mainly determine the shape of the control surface can be learned.

Although the number of control vertices to be optimised can be quite large in a multi-dimensional problem, learning of such a fuzzy controller is efficient due to the local influence of control vertices on the global control surface. Supervised learning process converges rapidly thanks to the one-minimum property of the learning function. If a suitable evaluation function as shown in section 3.3 for a system for unsupervised learning is designed, the learning algorithm also shows a good convergence property for systems with a single output. Learning, e.g. for mobile robots and manipulator control, can be performed on-line. In the future, we will investigate problems with a large number of inputs and/or multiple outputs based on the learning B-spline fuzzy controllers.

References

[1] H. R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 5(3), 1992.

[2] M. Brown and C. J. Harris. *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall, 1994.

[3] J. Godjevac and N. Steel. Adaptive fuzzy controller for robot navigation. In *IEEE International Conference on Fuzzy Systems*, 1996.

[4] J.-S. R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on System, Man and Cybernetics*, 23(3):665–685, 1993.

[5] S. Mitaim and B. Kosko. What is the best shape of a fuzzy set in function approximation. In *IEEE International Conference on Fuzzy Systems*, 1996.

[6] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD. Thesis, King's College, University of Cambridge, May, 1989.

[7] J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models. In *IEEE International Conference on Fuzzy Systems*, 1996.

[8] J. Zhang, F. Wille, and A. Knoll. Modular design of fuzzy controller integrating deliberative and reactive strategies. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.