ELSEVIER

Information Sciences 142 (2002) 227–248

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Extracting compact fuzzy rules based on adaptive data approximation using B-splines

J. Zhang *, S. Köper, A. Knoll

Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany

Abstract

We discuss the importance of making a fuzzy controller human interpretable and give an overview of the existing models and structures for that purpose. We then summarise our approach to designing fuzzy controllers based on the B-spline model by learning. By using an optimal partition algorithm and linguistic modificators like “between”, “at most”, “at least”, etc., the rule base can be reduced to the minimum. This helps to avoid the over-fitting problem and improves the interpretability of the model. We tested the controller on different benchmark problems and achieved a rule compression ratio of up to 71%. © 2002 Published by Elsevier Science Inc.

Keywords: Rule extraction; Neuro-fuzzy system; Genetic algorithms (GAs); Interpretability

1. Introduction

Until recently, a large part of current science and technology was based on analytical methods which are usually specialised for pre-defined domains. However, for a human-being it is time-consuming to get acquainted with a model, to devise a model, even difficult to explain a model clearly to someone

* Corresponding author. Tel.: +49-521-106-2951; fax: +49-521-106-6440.
E-mail addresses: zhang@techfak.uni-bielefeld.de (J. Zhang), skoeper@techfak.uni-bielefeld.de (S. Köper), knoll@techfak.uni-bielefeld.de (A. Knoll).
http://www.techfak.uni-bielefeld.de/techfak/ags/ti/.

0020-0255/02/$ - see front matter © 2002 Published by Elsevier Science Inc.
PII: S0020-0255(02)00167-6

else. On this problem, physicist Richard P. Feynman mentioned "the way we have to describe nature is generally incomprehensible to us". Nevertheless Albert Einstein believed "it should be possible to explain the laws of physics to a barmaid".

Among the mechanisms to interpret the nature of a process like equations, tables, flow charts, stories, etc., fuzzy linguistic rules and relation descriptions are easy to understand. For building models with training data, certain types of fuzzy rule systems like the B-spline model [13] have been developed, which can approximate any low-dimensional to middle-dimensional input–output functions.

There are good reasons for making such a controller model symbolically interpretable:

- Linguistic modelling provides a way of transferring skills from one human expert to other non-experts or to an artificial system. The transferred rules and human knowledge can be used to accelerate the model-building and to patch the model in the case of data deficiencies.
- Automatic learning of transparent models makes the analysis, validation and supervision in the model development easier. This way, a large part of design expenses can be shifted from humans to the computer.
- Transparent models will have wide applications in decision-support systems. In the next years, most of the control of complex systems will still be semi-autonomous. Moreover, "human-in-the-loop" is based on compact and summarising descriptions of a system model.

New control and modelling approaches will make good use of the rapid increase of computation power and memory brought about by recent computer technology. Advances have been made in the theory and applications of neural networks and fuzzy control approach to the control of complex systems which cannot be adequately modelled by differential equations. The fuzzy rule description of a system has the advantage over neural networks that it is not just a "black-box". Neuro-fuzzy models integrate automatic feature extraction and learning of membership functions (MFs) into a fuzzy control structure. Together with optimisation algorithms, which deal with local minima, semi-automatic procedures can be designed for constructing non-linear models and controllers which can be understood by human users.

2. State of the art

2.1. Structure of interpretable models

Variants of fuzzy rule systems are: additive, multiplicative and hierarchical. Figs. 1 and 2 illustrate these structures. The solution with hierarchical structure assumes that the input information can be classified into groups [12]. Within each group the inputs determine an intermediate variable, they can be

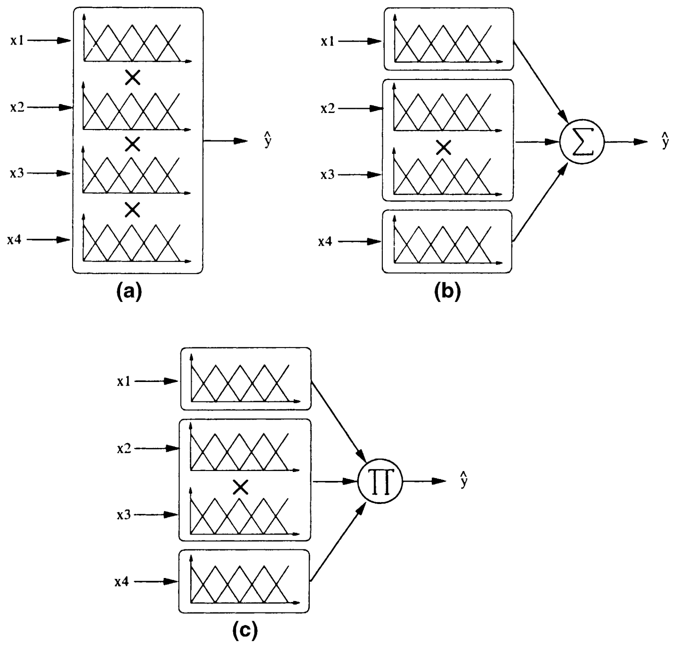

Fig. 1. Fuzzy system with four inputs and each with five linguistic terms: (a) complete fuzzy system (5^4 rules); (b) additive fuzzy system (35 rules); (c) multiplicative fuzzy system (35 rules).

decoupled from inputs of other groups. To realise such a grouping, heuristics based on the fusion of physical sensors usually have to be applied.

As application areas grow, the *systematic* design of an optimal fuzzy controller becomes more and more important. Classical fuzzy controllers of the Mamdani type [7] are based on the idea of directly using symbolic rules for diverse control tasks. Another important type of fuzzy controllers is based on the Takagi–Sugeno–Kang (TSK) model [10]. Recently, TSK type fuzzy controllers have been used for function approximation and supervised learning [11]. However, it is pointed out that the TSK model is a black-box based on a multi-local-model. In the following section, we describe an approach that can build MFs for linguistic terms of the IF-part systematically, then optimise them by genetic algorithms (GAs). Our approach is based on the B-spline model which can be classified as a zero-order TSK model. However, we define linguistic terms for input variables with B-spline basis functions and for output variables with singletons. Such a method requires fewer parameters by adding MFs than other set functions such as trapezoid, Gaussian function, etc. The output computation is very simple, and the interpolation process is transparent. We have also achieved good approximation capabilities and rapid convergence of the B-spline controllers.

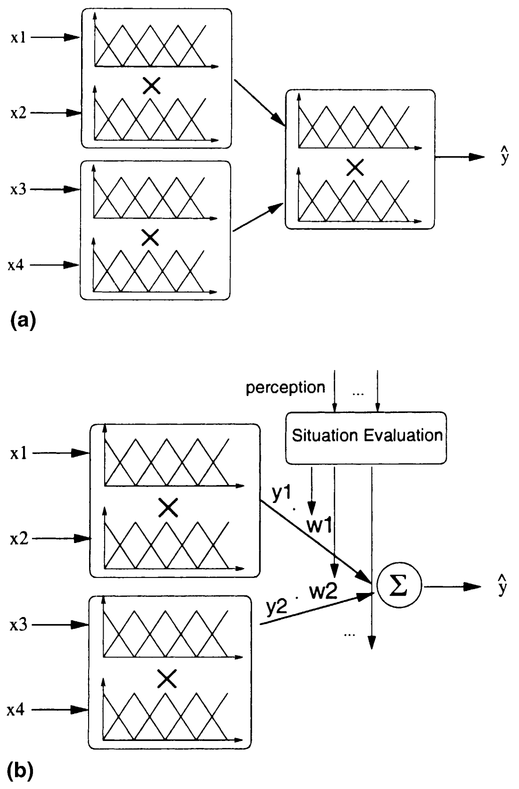

Fig. 2. Hierarchical fuzzy systems: (a) hierarchical fuzzy system with four inputs, each with five linguistic terms – resulting in 75 rules; (b) behaviour blending using a two-step hierarchy. "Situation Evaluation" uses rules to determine the weight of each monolithic controller [14].

3. Constructing a fuzzy rule system with B-splines

3.1. Basic concept

Our B-spline model provides an ideal implementation of "Cerebellar Model Articulation Controller" (CMAC) proposed by Albus [1]. The CMAC model provides a neural interpretation of the B-spline model. B-spline models employ piecewise polynomials as MFs.

3.1.1. Definition of univariate B-splines

The B-splines are employed to specify the linguistic terms. To compute B-splines, the universe of discourse of each input is divided into a number of intervals, where each interval is delimited by breakpoints called *knots*. In our control and modelling applications, knots are chosen to be different from each

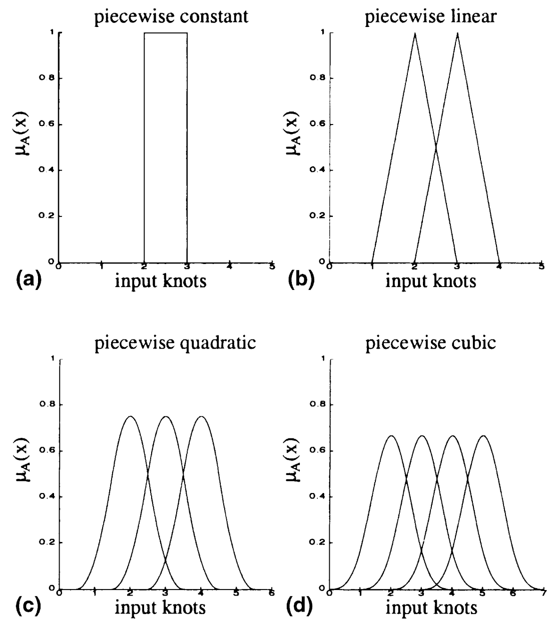

Fig. 3. Univariate B-splines of order 1–4: (a) order 1; (b) order 2; (c) order 3; (d) order 4.

other (periodical model). Visually, the selection of order of the B-splines, denoted by k, determines the appearance and position of each B-spline (Fig. 3).

Assume x is a general input variable of a system that is defined on the universe of discourse $[x_1, x_m]$. Given a sequence of ordered knots: $x_1, x_2, \ldots$, the ith B-spline $N_{i,k}$ of order k (degree $k-1$) is recursively defined as follows:

$$N_{i,k}(x) = \begin{cases} \begin{cases} 1 & \text{for } x \in [x_i, x_{i+1}) \\ 0 & \text{otherwise} \end{cases} & \text{if } k = 1, \\ \frac{x - x_i}{x_{i+k-1} - x_i} N_{i,k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(x) & \text{if } k > 1, \end{cases} \tag{1}$$

where $1 \leqslant i \leqslant m - k$. m knots $x_1, \ldots, x_m$ form $l = m - k$ B-splines (for an example see Fig. 4).

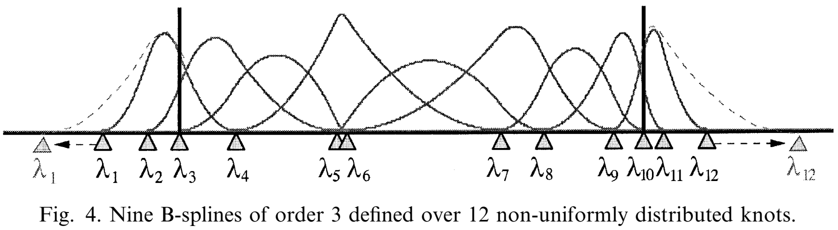

Fig. 4. Nine B-splines of order 3 defined over 12 non-uniformly distributed knots.

3.1.2. Properties

Recursive definition is one basic property of B-splines, which enables the generation of basis functions of arbitrary orders with the incremental smoothness for a given set of knots. The other important properties of B-splines with respect to modeling and control are:

Partition of unity: $\sum_{i=0}^{l} N_{i,k}(x) = 1$.

Positivity: $N_{i,k}(x) \geqslant 0$ for all x.

Local support: $N_{i,k}(x) = 0$ for $x \notin [x_i, x_{i+k}]$.

C^{k-2} *continuity*: If the knots $\{x_i\}$ are pairwise different from each other, then $N_{i,k}(x) \in C^{k-2}$, i.e., $N_{i,k}(x)$ is $(k-2)$ times continuously differentiable.

Fig. 4 shows the non-uniform B-splines which are computed with unevenly distributed knots. To compactly model and control a system, it is desirable that the minimal number of basis functions be used. Non-uniform B-splines are the most important model in these applications. They coincide with the psychological investigations that human linguistic terms are often non-symmetrical.

3.2. Lattice

In our previous work, we compared splines and a fuzzy controller with *single-input–single-output* (SISO) structures [15]. In the following, the MISO model is considered.

Fig. 5 illustrates the partition of a two-dimensional B-spline model with 8 MFs on each uniformly subdivided input interval and the activated B-splines

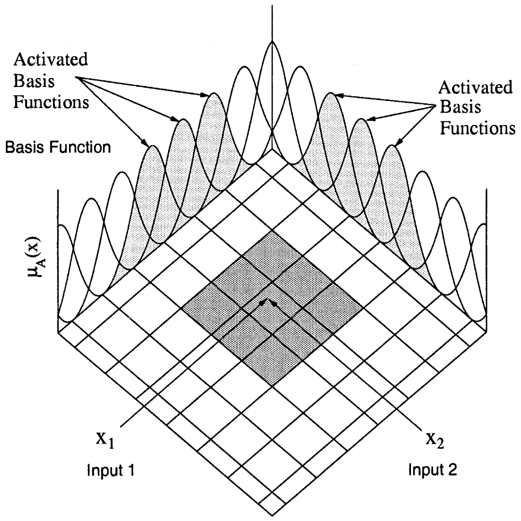

Fig. 5. A two-dimensional illustration of the B-spline model.

(slightly shaded) for a given input. Since learning one new part of the input space affects only a given number of system response values (darkly shaded area of Fig. 5), fast on-line learning can be implemented. Due to these advantages, B-spline models are proposed to be applied to control and modelling systems [13]. By using the B-spline model, the approximation ability is only limited by the number of knots distributed over the input intervals. Regarding that most observed data contain certain noises, the problem of over-fitting may occur. GA-optimised B-spline models are a promising approach to find sparse models, which are able to bridge the gap between high bias and high variance of a model.

Each n-dimensional rectangle $(n > 1)$ of the lattice is covered by the jth multivariate B-spline $\mathbf{N}_j(x)$ which is formed by taking the tensor product of n univariate B-splines:

$$\mathbf{N}_j(x) = \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j). \tag{2}$$

Therefore, the shape of each B-spline, and thus the shape of multivariate ones (Fig. 6), is implicitly set by their order and their given knot distribution.

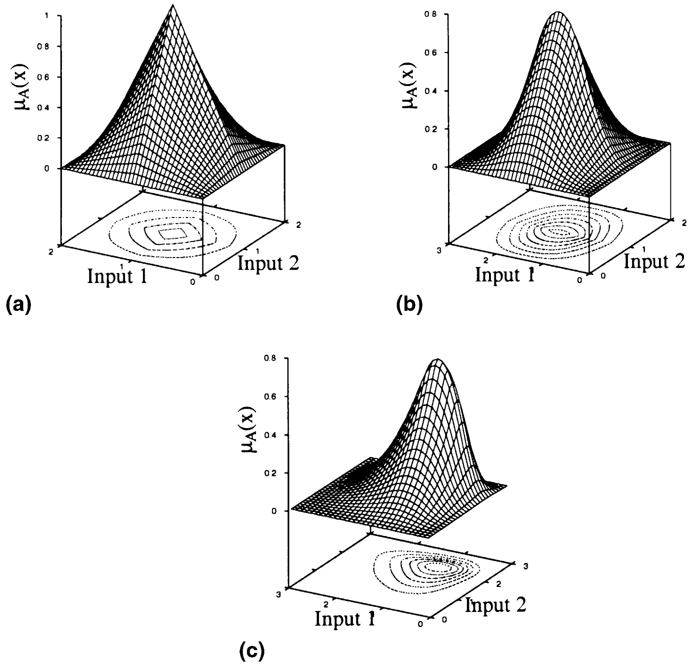

Fig. 6. Bivariate B-splines formed by the tensor product of two univariate B-splines: (a) tensor product of two univariate B-splines of order 2; (b) tensor product of one univariate B-spline of order 3 and one of order 2; (c) tensor product of two univariate B-splines of order 3.

3.3. Use of B-splines in a fuzzy inference system

Under the following conditions, the computation of the output of such a fuzzy rule system is equivalent to that of a *general B-spline hypersurface* [13]:

(1) periodical B-spline basis functions are used as MFs for inputs,
(2) fuzzy-singletons (called *control vertices*) are used as MFs for outputs,
(3) "product" is used as fuzzy conjunction,
(4) "weighted means" is used as defuzzification method,
(5) "virtual linguistic terms" are added at both ends of each input variable, and
(6) the rule base for the "virtual linguistic terms" is extended by copying the output values of the "nearest" neighbourhood.

Generally, if we consider a MISO system with n inputs $x_1, x_2, \ldots, x_n$, a rule $(i_1, i_2, \ldots, i_n)$ with the n conjunctive terms in the premise is given in the following form:

IF $(x_1$ is $N^1_{i_1,k_1})$ and $(x_2$ is $N^2_{i_2,k_2})$ and ... and $(x_n$ is $N^n_{i_n,k_n})$
THEN y is $c_{i_1,i_2,\ldots,i_n}$

where

- n: the number of the input dimensions,
- x_j: the jth input $(j = 1, \ldots, n)$,
- k_j: the order of the B-splines used for x_j,
- $N^j_{i_j,k_j}$: the ith linguistic term of x_j defined by univariate B-splines,
- $i_j = 1, \ldots, l_j$: represents the index of the linguistic term of x_j (l_j denotes the number of B-splines), and
- $c_{i_1,i_2,\ldots,i_n}$: the coefficients of the rule $(i_1, i_2, \ldots, i_n)$, called *control vertices* in the following. [1]

Then the output y of a MISO fuzzy model is:

$$y = \frac{\sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \left(c_{i_1,\ldots,i_n} \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j) \right)}{\sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \left(\prod_{j=1}^{n} N^j_{i_j,k_j}(x_j) \right)} \tag{3}$$

$$= \sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \left(c_{i_1,\ldots,i_n} \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j) \right). \tag{4}$$

This *general NUBS hypersurface* possesses the following properties:

- If the B-splines of order $k_1, k_2, \ldots, k_n$ (>2) are employed to specify the linguistic terms of the input variables $x_1, x_2, \ldots, x_n$, it can be guaranteed that the output variable y is $(k_j - 2)$ times continuously differentiable with respect to the input variables $x_j, j = 1, \ldots, n$.

[1] Corresponding to *de Boor points* in CAGD.

- If the input space is partitioned fine enough and at the correct positions, the interpolation with the B-spline hypersurface can reach a given precision.

The transformation from (4) to a rule base is conceptually important because it provides a construction method for data approximation using fuzzy rule systems. The resulting advantage over the pure B-spline interpolation lies mainly in its linguistic modeling ability: interpolation data can be prepared by using natural language with the help of expert knowledge. Furthermore, the interpolation procedure becomes transparent because it can also be interpreted in fuzzy logic "IF–THEN" form.

3.4. Generating the THEN-part

Determining optimal coefficients of a B-spline model with fixed knot vector is generally simpler than finding the optimal knot vectors. Applying methods of B-spline theory in Computer Aided Graphic Design (CAGD), the coefficients can be estimated by solving an over-determined linear system. Another method based on gradient descent can also be used. Although being an iterative solution, this learning method is conceptually easier to understand. In the gradient descent approach, fuzzy singletons represented by control vertices can be initialised with the values acquired from expert knowledge. These parameters will be fine-tuned by a learning algorithm.

In the following, we show that for supervised learning the squared errors with respect to the control vertices are convex functions. Therefore, rapid convergence for supervised learning is guaranteed. The control space changes locally due to the "local support" property of B-splines while the control vertices are modified. Based on this feature, the control vertices can be optimised gradually, area-by-area.

Assume that $(x_1^r, \ldots, x_n^r, y_{\text{desired}}^r)$ is a set of training data. The output value computed by a B-spline model is denoted with y_{computed}^r. By defining the mean square error (MSE) criterion as

$$E = \frac{1}{2} \cdot (y_{\text{computed}} - y_{\text{desired}})^2 \equiv \text{MIN}, \tag{5}$$

the derivation of each coefficient $c_{i_1,\ldots,i_k}$ is

$$\Delta c_{i_1,\ldots,i_k} = -\epsilon \frac{\partial E}{\partial C_{i_1,\ldots,i_k}} = -\epsilon (y_{\text{computed}}^r - y_{\text{desired}}^r) \cdot \prod_{j=1}^{n} N_{i_j,n_j}(x_{i_j}^r), \tag{6}$$

where ϵ denotes the *learning rate*. Since the second partial derivation of $c_1^r, \ldots, c_k^r$ is always positive, the error function (5) is convex in the weight space. If the inputs $(x_1^r, \ldots, x_n^r)$ are *linearly independent*, due to the convexity of the MSE performance surface, there exists only one global minimum and no local minima. On the other hand, if the auto-correlation matrix is singular, which

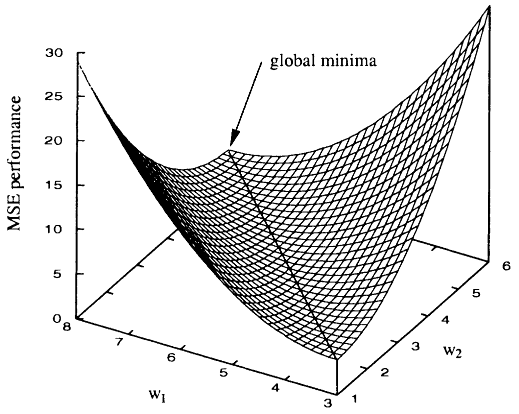

Fig. 7. A singular MSE performance surface.

occurs when the inputs x_r are *linearly dependent*, there exists an infinite number of global minima (Fig. 7), but still no local minima in the weight space.

3.5. Adaptive modelling of the IF-part

Based on the granularity of the input space and the distribution of extrema in the control space (if known), the fuzzy sets can be initialised using the recursive or explicit computation of B-splines. These fuzzy sets based on non-uniform B-splines can be further adapted during the generation of the whole system by using GAs.

By freeing the knots, the task of finding control vertices and accurate knot vectors to fit training data becomes a non-linear minimisation problem. To solve this problem we follow a strategy of problem splitting. We first consider the underlying model $\delta(\lambda)$ of the controller and then compute the control vertices to minimise $\delta(c)$. Instead of using constrained least-square methods (constrained because of avoiding to "ride" on the gradient edge of coincident knots [5]), the knots are estimated by using GAs. GAs are both theoretically and empirically proven to provide the means for efficient search, even in complex spaces [3]. Therefore, each individual, e.g. each B-spline fuzzy system with its special knot distribution, represents one point in the search space.

3.5.1. The genetic algorithms

We applied the basic GA introduced by Holland [4] with some modifications as follows (Fig. 8):

- We used *gray coding* instead of standard binary code for representing coded chromosomes, a common modification.

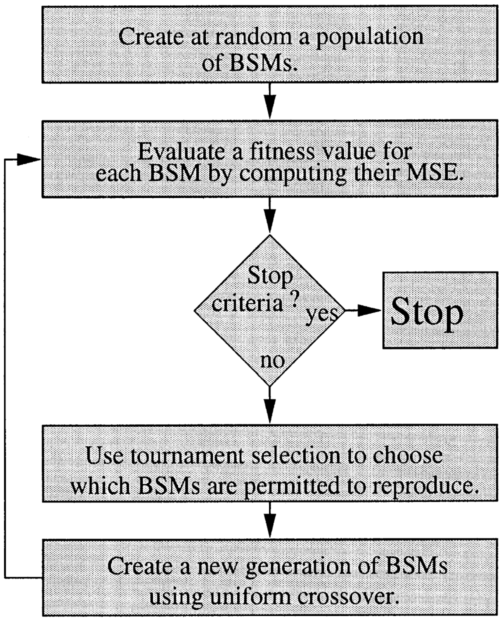
Create at random a population of BSMs.
Evaluate a fitness value for each BSM by computing their MSE.
Stop criteria ?
yes
no
Stop
Use tournament selection to choose which BSMs are permitted to reproduce.
Create a new generation of BSMs using uniform crossover.

Fig. 8. Flowchart of the applied GA (BSM: B-spline model).

- To bypass the undesirable effect of the increasing probability along the descendent chromosome-string to receive a changed allele (bit) (thus the conjointly heredity of genes decreases when the distance of their position increases) while using n-point crossover, we used *uniform crossover*. This kind of crossover has no positional and a high distributional bias, so that a high blending rate between participant chromosomes is granted. This leads to an algorithm producing permanently solutions which explore new locations by bridging even great distances of the search space.
- Instead of using fitness-proportional selection it is advantageous to use *tournament selection*. This selection schema draws ξ individuals $(2 \leqslant \xi \leqslant \mu)$ with a probability $1/\mu$ from the current population and copies the individual with the best fitness into the mating pool. Besides saving computational power as a result of no need to sort the population (as in ranking based selection schemes), it is easier to bias the *takeover time*.

3.5.2. Chromosome encoding for knot placement

To minimise $\delta(\lambda)$ each individual consists of n knot vectors, where n is the problem dimension. Each encoded knot vector consists of 32 knots and a so-called *activation string* of 32 bit length. Which knots are in use to define the current model is encoded through the activation string. Activated knots are represented by 1, and inactivated knots are represented by 0.

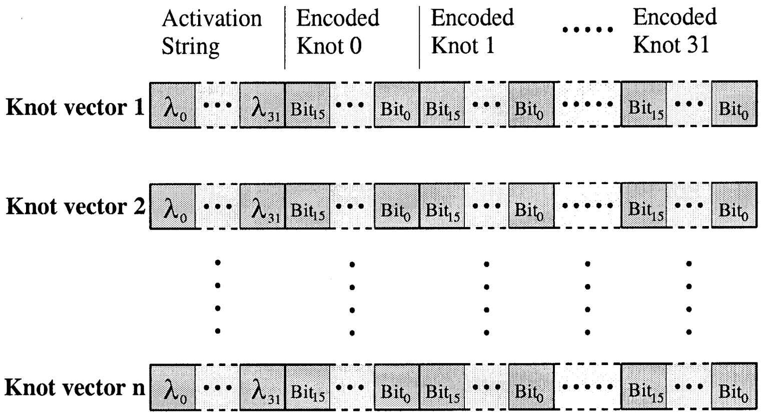

Fig. 9. Encoded B-spline model.

Every knot is encoded by 16 bit (see Fig. 9) and therefore each knot can be placed on its respective input interval $[a, b]$ with an accuracy of $1/(2^{16}) \times (b-a)$. The fitness values for each individual are simply computed by determining the control vertices of the controller. Using these control vertices the MSE is evaluated and the fitness for one individual is set equal to $1/\mathrm{MSE}$.

4. Optimal number of MFs

An easy way to reduce the number of fuzzy rules is to minimise the number of linguistic terms. However, the approximation error increases when decreasing the number of MFs. We studied the problem how many linguistic terms are needed to minimise both the number of linguistic terms *and* the approximation error.

We analysed the approximation of 12 functions [8] with several fuzzy controllers. Six functions have one input, three have two inputs and three have three input dimensions. For approximating the functions, several fuzzy controllers with different numbers of linguistic terms were trained with the algorithms described in Section 3. Then the approximation errors were analysed.

An example shows the approximation results of function g_2. This function is defined in (Eq. (7)):

$$\begin{aligned} g_2(x,y) = f_4(x)\cdot 2\Big(&\exp(-((y-0.1)/0.25))^2 \\ &-0.8\exp(-((y+0.75)/0.15))^2 \\ &-0.4\exp(-((y-0.8)/0.1))^2\Big) \text{ with } -1 \leqslant x,\ y \leqslant 1. \end{aligned} \tag{7}$$

Fig. 10 shows the relationship between the number of linguistic terms and the MSE of the approximation. The approximation error function has a typical

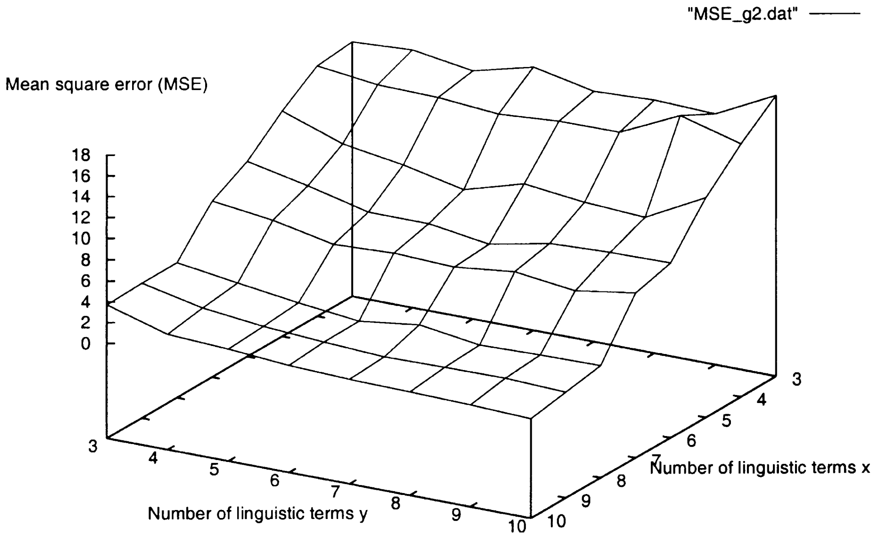

Fig. 10. Relationship between the number of linguistic terms and the MSE of the B-spline fuzzy system (function $g_2(x,y)$).

shape. While increasing the number of linguistic terms the error first decreases fast, later on it decreases more slowly. A good compromise between a minimum number of linguistic terms and a minimum approximation error is using eight terms for input x and three terms for input y. More terms would not improve the approximation substantially, fewer terms would increase the error sharply.

Fig. 11 shows function g_2. Its projections on the x- and y-axis in Figs. 12 and 13 show why we get these numbers for the optimal number of linguistic terms. The extrema of the function lie on a grid. If the extrema are projected on the axes, several extrema will cover each other. On the x-axis we find eight groups of extrema and on the y-axis three groups that are just the same numbers we found as the optimal number of linguistic terms.

Furthermore, it is interesting to observe how the GAs arrange the MFs on the input axes (Figs. 14 and 15). The MFs cover nearly all the groups of extrema we found in Figs. 12 and 13. They lie on the same grid as the function's extrema. Lastly we point out that a human would describe this function with fuzzy rules in the same way. For each extrema he would use one fuzzy rule. For the other functions we found exactly the same results.

5. Optimal partition algorithm

If an application demands a high precision, a lot of fuzzy rules will be necessary. This large number of rules will have a negative effect on the inter-

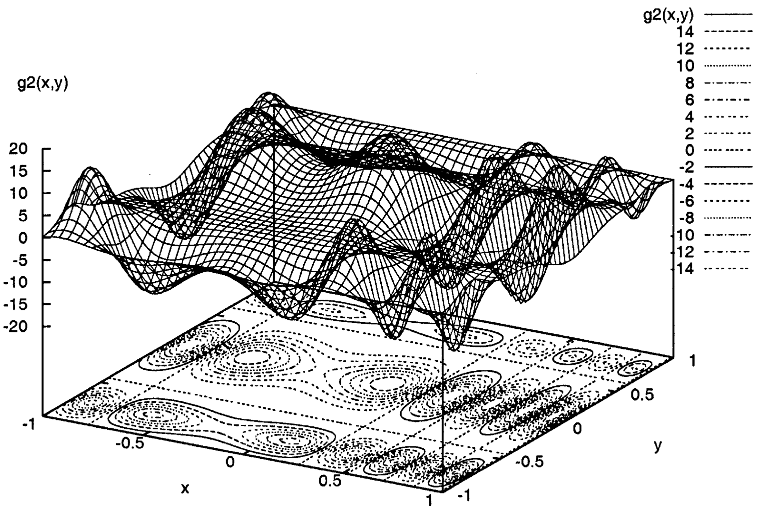

Fig. 11. The function g_2.

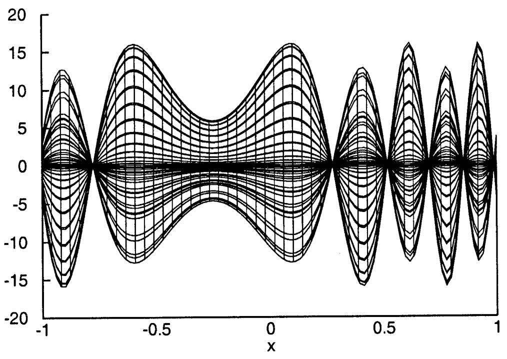

Fig. 12. Projection of g_2 to the x-axis.

pretability of the model. This section describes an optimal partition algorithm, which helps to explain how a system with a large number of rules can be best interpreted.

5.1. Basic idea

A fuzzy controller divides the universe of inputs and outputs in fuzzy sets. These fuzzy sets are arranged e.g. in "small", "middle" and "large". This information is useful to reduce the size of the fuzzy rule base. Table 1 shows a simple example of a fuzzy controller with two inputs [2]. Each input is covered by five fuzzy sets ("NB", "NS", "Z", "PS", "PB").

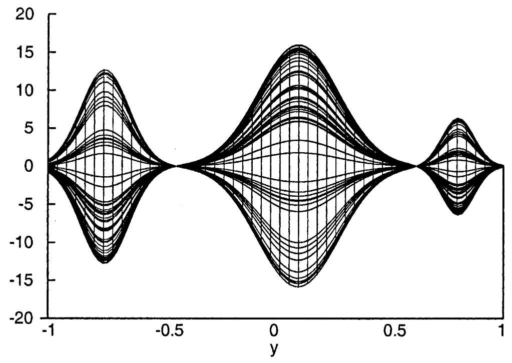

Fig. 13. Projection of g_2 to the y-axis.

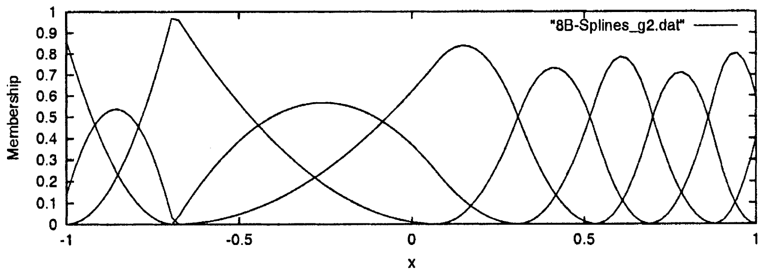

Fig. 14. Arrangement of eight B-spline MFs on the x-axis.

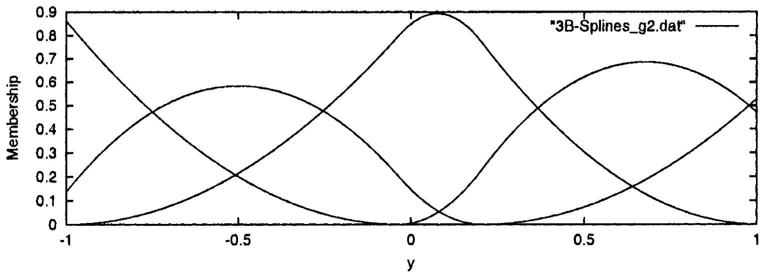

Fig. 15. Arrangement of three B-spline MFs on the y-axis.

A simple way to reduce this rule base is to combine neighbouring rules with the same consequences, e.g. the three rules

IF e IS '*NB*' AND $\dot{e}$ IS '*NB*' THEN f IS '*NB*'
IF e IS '*NS*' AND $\dot{e}$ IS '*NB*' THEN f IS '*NB*'
IF e IS '*Z*' AND $\dot{e}$ IS '*NB*' THEN f IS '*NB*'

can be replaced by the rule

Table 1
The rule table of an example controller

$e \backslash \dot{e}$	NB	NS	Z	PS	PB
NB	NB	NB	NB	NS	Z
NS	NB	NB	NS	Z	PS
Z	NB	NS	Z	PS	PB
PS	NS	Z	PS	PB	PB
PB	Z	PS	PB	PB	PB

IF e IS '*at most Z*' AND $\dot{e}$ IS '*NB*' THEN f IS '*NB*'.
Another possibility is to replace the four rules
IF e IS '*NB*' AND $\dot{e}$ IS '*NB*' THEN f IS '*NB*'
IF e IS '*NS*' AND $\dot{e}$ IS '*NB*' THEN f IS '*NB*'
IF e IS '*NB*' AND $\dot{e}$ IS '*NS*' THEN f IS '*NB*'
IF e IS '*NS*' AND $\dot{e}$ IS '*NS*' THEN f IS '*NB*'
by the rule
IF e IS '*at most NS*' AND $\dot{e}$ IS '*at most NS*' THEN f IS '*NB*'.
In this example the modifier "at most" is used to combine several rules into one rule. Other modifiers are "at least", and "between". For a definition of these modifiers see [2].

There are several possibilities to combine neighbouring rules in a rule table. Not all of these possibilities lead to the minimum of rules. The question is how to find the optimal partition of the rule base. One step to answer this question is to find the common features of connectable rules. There are three of them:
(1) the same consequences,
(2) direct neighbourhood, and
(3) the rules have to form rectangular regions in the rule table.

The result of features (1) and (2) is that the search can be restricted to regions with identical consequences. These regions form polygons in the rule table which are composed of squares (one square per rule). These polygons are called rectilinear polygons. If this polygon is a rectangle, then it will be a valid rule (feature (3)). The way to reduce the rule base with two inputs to the minimum is to find a partition of all of these polygons in a minimum number of rectangles. If there are more then two inputs there will be hyperpolygons composed of hypercubes. For this hyperpolygons a partition in hypercuboids is searched.

5.2. Minimum partition of a polygon in rectangles

There are several algorithms to solve the problem of partitioning a polygon in a minimum number of rectangles [6,9]. The main idea is: The difference between a rectilinear polygon and a rectangle is that a complex polygon is not convex. Every convex rectilinear polygon is a rectangle. The goal is to intersect the polygon horizontally and vertically that all concave vertices are removed. Every intersection increases the number of rectangles by one and decreases the number of concave vertices by one if the cut goes through one concave vertice or by two if the cut goes through two concave vertices. The optimal partition will be reached if first a maximum of cuts through two concave vertices and then the cuts through one concave vertice are made.

The optimal partition of a rule base takes the following steps:

(1) Find all connected areas with the same consequences in the rule table. Then a set of polygons is generated.
(2) For all polygons: find the maximum number of cuts through two concave vertices.
(3) Perform these cuts.
(4) Intersect the polygons through the remaining concave vertices.
(5) For all rectangles: find the corresponding fuzzy rules.

Fig. 16 shows all possible partitions of an example polygon.

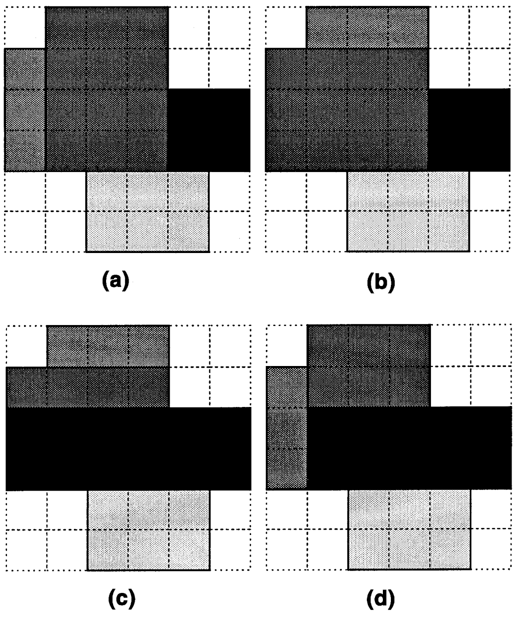

Fig. 16. Minimum partitions of a polygon.

5.3. Applying the algorithm on B-spline fuzzy systems

The described algorithm is easy to apply to fuzzy rule bases whose consequences are real fuzzy sets (like the rules of Mamdani controllers). The consequences of B-spline controller rules are fuzzy singletons. Every singleton is learned by the approach described in Section 3.4 and the possibility that two singletons have the same value is very small. Before the algorithms can be applied to a B-spline rule base the number of possible fuzzy singletons must be reduced, e.g. with a clustering algorithm like the fuzzy *c*-means algorithm.

The second problem is that the singletons will not be the interpolation points of the curve (see Fig. 17) when B-spline systems with order three or higher are used. If the rule base is to be interpreted, it will be reasonable to use the interpolation points. These points can easily be calculated using the established B-spline model.

Before the partition algorithm can be applied to a B-spline rule base, the fuzzy singletons must be replaced by the interpolation points, then the interpolation points must be clustered.

Table 2 shows the original rule table, Table 3 the rules with interpolation points as consequences and Table 4 the resulting rule table with clustered consequences.

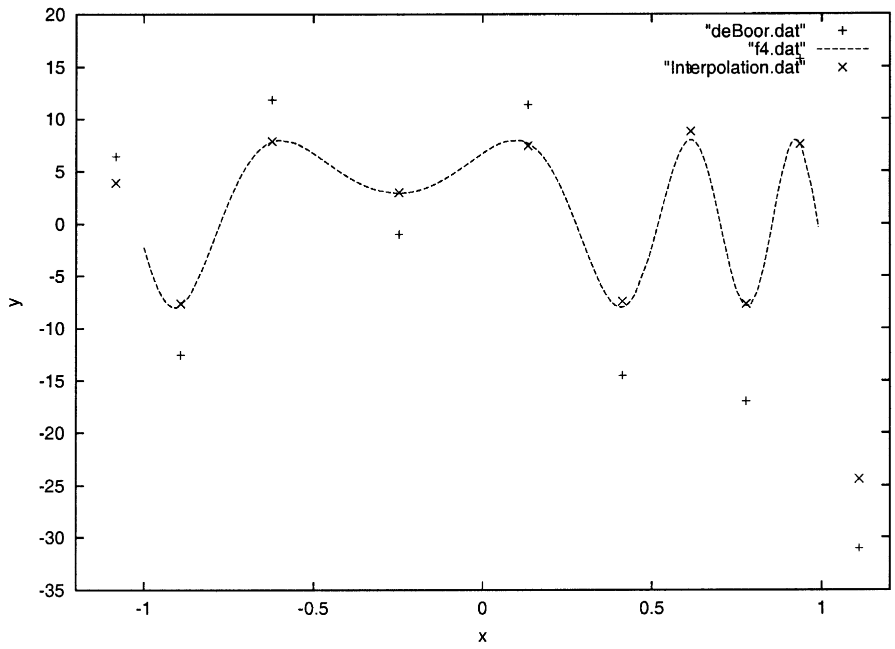

Fig. 17. Function $f_4(x)$ (dashed), control vertices (+), interpolation points (×).

Table 2
Rule table for function $f_4(x)$

IF x IS	LM	VT	T	VS	S	L	VL	H	VH	RM
THEN y IS	6.4	−12.5	11.9	−1.0	11.4	−14.5	14.8	−17.0	15.7	−31.1

Table 3
Rule table for function $f_4(x)$ consequences are interpolation points

IF x IS	LM	VT	T	VS	S	L	VL	H	VH	RM
THEN y IS	3.9	−7.6	7.9	3.0	7.4	−7.4	8.8	−7.7	7.6	−24.4

Table 4
Rule table for function $f_4(x)$ consequences are clustered interpolation points labelled with linguistic identifiers

IF x IS	LM	VT	T	VS	S	L	VL	H	VH	RM
THEN y IS	L	S	VL	L	VL	S	VL	S	VL	VS

Table 5
Rule base for the MPG-problem

Year\weight	VS	S	M	L	VL
VS	M	M	S	S	VS
S	VL	M	S	S	VS
M	VL	M	S	S	M
L	VL	L	M	S	S
VL	VL	L	L	S	VS

5.4. Examples

Two examples show how a fuzzy rule base can be simplified. The first example is the well-known MPG-Problem. [2] We trained a uniform B-spline system with two inputs (year, weight) and five fuzzy sets per input. The fuzzy singletons were replaced by the interpolation points and then clustered in five fuzzy sets. Table 5 shows the rule base with 25 rules.

This rule base can be reduced with our partition algorithm to 12 rules. This is a reduction by 52%. Table 6 shows the reduced rule base.

The second example is the inverse kinematics of a two-joint planar robot. For the angle of the joints, a non-uniform B-spline system was trained. Each input was covered by 11 fuzzy sets. With the partition algorithm, the rule base

[2] The data are available at `ftp`://ics.uci.edu/pub/machine-learning-databases/auto-mpg.

Table 6
Reduced rule base for the MPG-problem

year \ weight	VS	S	M	L	VL
VS	M	M	S	S	VS
S	VL	M	S	S	VS
M	VL	M	S	S	M
L	VL	L	M	S	S
VL	VL	L	L	S	VS

Table 7
Grouped rule base for the inverse kinematics, joint 1

y \ x	LM	VT	T	VS	S	M	L	VL	H	VH	RM
LM	M	M	S	S	S	S	S	S	S	S	M
VT	M	S	S	S	M	M	M	M	M	S	S
T	S	S	S	S	M	S	M	M	M	M	S
VS	S	S	S	S	M	M	M	M	M	M	M
S	S	S	S	S	S	VS	L	L	L	M	M
M	VL	VS	S	S	S	VL	L	L	L	M	M
L	VS	VL	S	VS	L	L	L	L	L	M	M
VL	M	L	L	L	L	L	L	L	L	M	M
H	L	L	L	L	L	L	L	L	L	M	M
VH	M	M	L	L	L	L	L	L	M	M	M
RM	M	M	M	M	M	L	L	M	M	M	M

of 121 rules was reduced to 35 rules. This is a reduction by 71%. Table 7 shows the reduced rule base.

6. Future research topics

Integration of neural networks, fuzzy systems, GAs, chaos theory with the classical probability theory and statistical pattern recognition will play a central role in the future research. We list some important ones:

- Since the system resources are limited, sensory input needs to be selected by factor analysis/synthesis, tracking focus of interests and other statistical methods.
- The sensor capability can be extended by using up-to-date information technologies such as software robots in WWW, linguistic modelling of human perception and sensor fusion so that information which is difficult to measure, incomplete or noisy can be perceived.
- Methods need to be developed for increasing the capability and quality of reinforcement signals and fitness evaluation of the learning system.
- Symbolic sparse coding, granular computing, fuzzy set, rough set need to be integrated to enable the arbitrary transition between digital measurements and concepts.

The model and approaches described in this paper paves the way for layered-learning to build models of a wide range of modeling and control problems.

Acknowledgements

The authors wish to thank Ingo Renners for developing the GA optimisation algorithm and for providing the program of function approximation.

References

[1] J.S. Albus, A new approach to manipulator control: the cerebellar model articulation controller (CMAC), Transactions of ASME, Journal of Dynamic Systems Measurement and Control 97 (1975) 220–227.
[2] U. Bodenhofer, The construction of ordering-based modifiers, in: Fuzzy-Neuro Systems '99, Computational Intelligence, Leipziger Universitätsverlag, Leipzig, 1999, pp. 55–62.
[3] D.E. Goldberg, Genetic Algorithms in Search Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[4] J.H. Holland, Adaption in Neural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[5] D.L.B. Jupp, The 'lethargy' theorem – a property of approximation by γ-polynomials, Journal of Approximation Theory 14 (1975) 204–217.
[6] W. Lipski, Finding a Manhattan path and related problems, Networks 13 (1983) 399–409.
[7] E.H. Mamdani, Twenty years of fuzzy control: experiences gained and lessons learned, in: IEEE International Conference on Fuzzy Systems, 1993, pp. 339–344.
[8] S. Mitaim, B. Kosko, What is the best shape of a fuzzy set in function approximation, in: IEEE International Conference on Fuzzy Systems, 1996, pp. 1237–1243.
[9] V. Soltan, A. Gorpinevich, Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles, Discrete and Computational Geometry 9 (1993) 57–79.
[10] T. Takagi, M. Sugeno, Fuzzy identification of systems and its application to modelling and control, IEEE Transactions on System, Man and Cybernetics SMC-15 (1) (1985) 116–132.
[11] L.-X. Wang, Adaptive Fuzzy Systems and Control, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[12] L.-X. Wang, Universal approximation by hierarchical fuzzy systems, Fuzzy Sets and Systems 93 (1998) 223–230.

[13] J. Zhang, A. Knoll, Constructing fuzzy controllers with B-spline models – principles and applications, International Journal of Intelligent Systems 13 (2/3) (1998) 257–285.

[14] J. Zhang, A. Knoll, Integrating deliberative and reactive strategies via fuzzy modular control, in: A. Saffiotti, D. Driankov (Eds.), Fuzzy Logic Techniques for Autonomous Vehicle Navigation, Springer, Berlin, 2000, pp. 367–387 (Chapter 15).

[15] J. Zhang, J. Raczkowsky, A. Herp, Emulation of spline curves and its application in robot motion control, in: Proceedings of IEEE International Conference on Fuzzy Systems, Orlando, 1994, pp. 831–836.