UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING AND INFORMATICS

Fuzzy Rule Interpolation-based Q-learning

PhD dissertation

DÁVID VINCZE
MSC IN INFORMATION ENGINEERING

'JÓZSEF HATVANY' DOCTORAL SCHOOL
OF INFORMATION SCIENCE, ENGINEERING AND TECHNOLOGY

ACADEMIC SUPERVISOR:
Dr. habil. Szilveszter Kovács

Miskolc, 2013

Declaration

The author hereby declares that this thesis has not been submitted, either in the same or in different form, to this or to any other university for obtaining Ph.D. degree. The author confirms that the submitted work is his own and the appropriate credit has been given where reference has been made to the work of others.

Nyilatkozat

Alulírott Vincze Dávid kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerinti vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Miskolc, 2013. december 16.

Vincze Dávid

A disszertáció bírálatai és a védésről készült jegyzőkönyv megtekinthető a Miskolci Egyetem Gépészmérnöki és Informatikai Karának Dékáni Hivatalában, valamint a doktori iskola weboldalán az Értekezések menüpont alatt: http://www.hjphd.iit.uni-miskolc.hu

Acknowledgements

This dissertation is based on the results of many years of work which I could not be able to achieve without the support of others.

Foremost I am most greatly thankful to my supervisor, Dr. habil. Szilveszter Kovács for the continuous support, guidance and encouragement all through the past years.

Also I am thankful to Dr. habil. László Kovács, head of the department, for supporting my research and allowing time and resources in order to complete this work.

Moreover I would like to thank Prof. Péter Korondi for the great support and advices regarding my work.

I am indebted to many of my colleagues at the Department of Information Technology for helping me in organizing the events connected to the doctoral procedure.

List of abbreviations

3DoF – 3 Degrees of Freedom
AI – Artificial Intelligence
BBC – Behaviour-Based Control
COG – Center of Gravity
CORBA – Common Object Request Broker Architecture
CNF – Convex and Normal Fuzzy set
CRI – Compositional Rule of Inference
CRF – Conservation of Relative Fuzziness
FA – Fuzzy Automaton
FERI – Fundamental Equation of the fuzzy Rule Interpolation
FFA – Fuzzy Finite-state Automaton
FIS – Fuzzy Inference System
FIVE – Fuzzy rule Interpolation based on Vague Environment
FLC – Fuzzy Logic Controller
FQ-learning – Fuzzy Q-learning
FRI – Fuzzy Rule Interpolation
FRIPOC – Fuzzy Rule Interpolation based on POlar Cuts
FRIQ-learning – Fuzzy Rule Interpolation-based Q-learning
GM – General Methodology
HRI – Human-Robot Interaction
ICE – Internet Communication Engine
IP – Internet Protocol
IMUL – Interpolation technique for MULtidimensional input spaces
KH – Kóczy-Hirota (FRI method)
LESFRI – LEast Squares Fuzzy Rule Interpolation
MACI – Modified Alpha-Cut based Interpolation
MATLAB – MATrix LABoratory
SST – Strange Situation Test
VE – Vague Environment
VEIN – Vague Environment-based two-step fuzzy rule INterpolation
VESI – Vague Environment-based Set Interpolation
VirCA – Virtual Collaboration Arena
VKK – Vass, Kalmár and Kóczy (FRI method)
RL – Reinforcement Learning
RTM – Robot Technology Middleware
UDP – User Datagram Protocol

List of figures

List of tables

Table of contents

I Introduction

In modern societies the topic of Computational Intelligence (or Artificial Intelligence (AI) for non-technical people) is well known, but of course to varying degrees. Machines are already present long enough to be understood by everyone at some level depending on the individual. Probably for an average person it is plausible to compare machine intelligence to human intelligence; one possibly expects human or human like behaviour from machines. To be more specific, the expected level of intelligence could depend on the form of existence, just to think of e.g. vending machine vs. humanoid robot. Modeling the way of complex human thinking and complex behaviour sets is currently still far from becoming a reality. Also, developing human like (regarding the outfit) robots can be a dead-end considering the 'uncanny valley' effect [47]. A solution for these problems can be the development of behaviour models and robots inspired by ethological studies, where the aim is not to mimic a human, not to copy human behaviour but to model an existing relationship between human and animal. One way to construct a working model is to manually implement behaviour components and strategies based on expert knowledge exactly defining how to achieve different tasks, and what to do in specific situations.

There can be cases when the exact principles of operation are unknown, only the expected results are known. Also there can be scenarios in real life when only the goal to be achieved is specified without giving an exact (step-by-step) method or without defining the correlations of actions and states (e.g. a method is not available). This latter matches the basic idea of reinforcement learning, where the system learns to achieve the goal from scratch without initial knowledge, based only on rewards and punishments given by the environment in a trial-and-error style. In possession of a function which can be used to calculate appropriate rewards, a possible model can be constructed for the specific problem. From the constructed model the incorporated knowledge can be extracted and reused in other (static) systems. A drawback of reinforcement learning methods is that they are ineffective in the case when the dimension count of the possible states and actions are relatively high. This means that applying reinforcement learning for the construction of complex behaviour sets is practically hardly an option.

In this work I introduce novel reinforcement learning methods based on fuzzy inference systems. These new methods can work with only the cardinal correlations supplied (generated), hence there is no need to store derivable or unneeded correlations unlike in traditional fuzzy inference based systems. Furthermore the new methods can automatically construct the required knowledge base in the form of fuzzy rules based on the reward calculation function.

Also I present a novel model for the realization of human-robot interaction inspired by ethological studies along with a framework which operates the model in real-time. In this model the fundamental behaviour components of an agent in a standardized ethological test is realized with a fuzzy rule interpolation based fuzzy automaton.

1.1 Aims of research

One aim of the research was to develop a fuzzy inference based reinforcement learning method which allows the usage of sparse fuzzy rule bases. Secondly to extend this method with the capability of automatic fuzzy rule base construction from scratch.

Furthermore to optimize the incorporated fuzzy rule interpolation method called FIVE, especially for the developed fuzzy inference based reinforcement learning method. First this requires an analysis of the implementation of the FIVE fuzzy rule interpolation method to identify the possible optimizations. Based on the results general implementation optimizations can be suggested, further studying the FIVE fuzzy rule interpolation technique structural and method specific improvements could be made.

Another, somewhat distinct goal was to construct an ethologically inspired system based on fuzzy automata incorporating fuzzy rule interpolation. This system should properly model an already existing standardized ethological test procedure, operated through a corresponding framework.

1.2 Dissertation guide

After the introduction, the second chapter gives an overview on fuzzy set theory and on fuzzy control systems, and an extensive introduction to fuzzy rule interpolation.

In the third chapter a novel method named Fuzzy Rule Interpolation based Q-learning (FRIQ-learning) and its extensions are presented in details. The method itself is explained and demonstrated with an application example. Furthermore this new method, the FRIQ-learning is extended with automatic rule base generation capabilities. First an incremental rule base construction method is introduced, and then another method is presented for reducing the size of the previously incrementally generated rule base. As with the pure FRIQ-learning method, the incremental construction and decremental reduction methods are also demonstrated and evaluated via extended application examples.

The fourth chapter analyses the incorporated fuzzy rule interpolation method called FIVE from the viewpoint of optimization possibilities. After that some techniques are described which can make the FIVE fuzzy rule interpolation method approximately ten times faster with only a neglectable loss in the functionality of the method. Some of the suggested optimizations are general implementation or structural modifications, but one of the optimization strategies was especially developed to be used in conjunction with the FRIQ-learning method presented in the third chapter.

An ethologically inspired model realized with fuzzy rule interpolation based fuzzy automaton is presented in the fifth chapter. The fuzzy automaton models a standardized ethological test, which is an adaptation of the Ainsworth test for dogs [63]. The details of this test are described as well as the sophisticated operating framework with its modules are presented. Some details of the operating rule bases are shown via exact sample rules and example simulation runs. Finally several interfaces (software and hardware) which were developed for this simulation are presented.

II Fuzzy systems and fuzzy rule interpolation

This chapter first gives a short overview on fuzzy set theory and fuzzy systems, then the fundamentals of fuzzy rule interpolation and some of the methods are presented. A specific fuzzy rule interpolation method, the FIVE fuzzy rule interpolation method is described in details, which is incorporated in the work presented further on.

2.1 Fuzzy sets and systems

In the way of human thinking some terms lack of having exact boundaries, e.g. in case of defining the terms for the age of a person one can say young, old, etc. These terms can hardly be described as expected by using simple mathematical tools. E.g. it is obvious to say young for someone who is 18 years old, and old for someone who is aged 70, but in case someone is 35, it is not straightforward whether the person is young or old. Many other examples could be mentioned following the previous scheme, like small, large, low, high, few, lot, etc. Fuzzy set theory introduced by Zadeh in [71] gives a possible way of mathematical description of such terms. In contrast to conventional sets, where values are whether members or not members of any of the sets, values in fuzzy sets have a certain degree of membership, meaning the values can be just partly members of sets. In [71] a fuzzy set is defined as: a fuzzy set is a continuum of membership grades.

A fuzzy set can be defined by defining its membership function, which determines which values are members, and to what degree in the interval [0, 1]. This membership function fully characterizes a fuzzy set. Theoretically a membership function can be of any kind, but in practice usually some special shapes are being used, because those can be generated and stored much efficiently than an arbitrary shape, e.g. triangle, trapezoidal, bell, sigmoid, Gaussian, etc. See Fig. 1. generated by the ‘mfdemo’ demonstration script in MATLAB [73] for some of the commonly used fuzzy membership functions.

An example definition of a simple fuzzy set defining the term ‘young’ can be written as the following:

$$young(x) = \begin{cases} 1, & if\ age(x) \leq 20 \\ (\ 30 - age(x)\ ) / 10, & if\ 20 < age(x) \leq 30 \\ 0, & if\ age(x) > 30 \end{cases} \quad (1)$$

The interesting part in this case is when the age is between 20 and 30, where the degree of membership is neither 0 nor 1. Between 20 and 30 the degree of membership linearly decreases.

Another simple way of defining fuzzy sets is to define its ‘breakpoints’, e.g. a trapezoidal set can be defined like (*membership* at *value*): *0 at 60, 1 at 80, 1 at 80, 0 at 100*. This is interpreted as: increase the degree of membership from 0 to 1 between values 60 and 80, decrease the degree of membership from 1 to 0 between values 80 and 100.

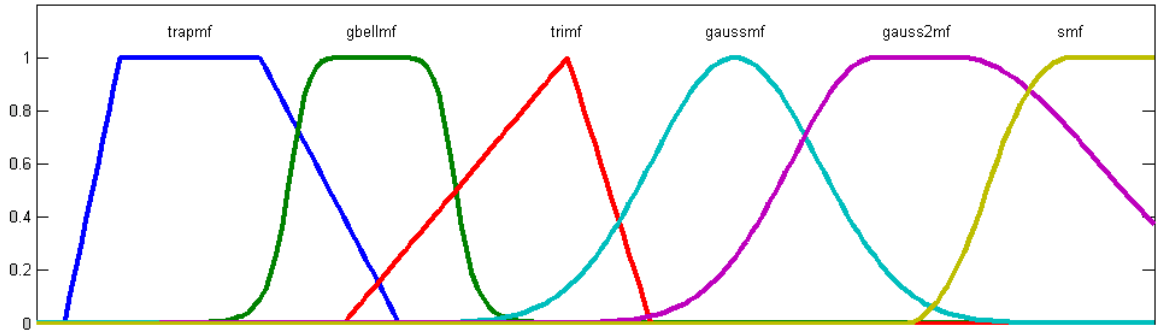

Fig. 1. Examples of fuzzy membership functions ('mfdemo' from MATLAB [73])

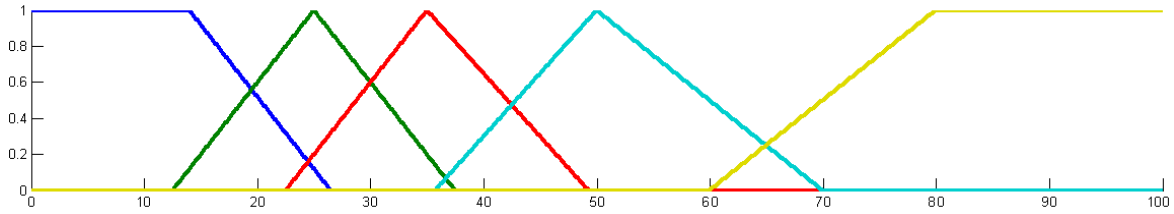

Fig. 2. An example of triangle shaped fuzzy sets defined for the linguistic variable *age*

Or another way can be to define a triangle shaped set is to define its core (where the membership function is 1) and the steepness of both its sides (see 'trimf' in Fig. 1. and Fig. 2. for examples of this kind of fuzzy sets).

Opposed to mathematics, where variables have numerical values, in fuzzy systems non-numeric linguistic variables are commonly used to express facts (like *age* in the previous example). Linguistic variables have their possible linguistic values predefined, e.g. *young, old* for the linguistic variable of *age*. A possible example for the linguistic variable *age* is shown in the followings:

```
age = { child, young, middle-aged, old, very old }
child = { 14, 0, 8 }
young = { 25, 8, 8 }
middle-aged = { 35, 8, 7 }
old = { 50, 7, 5 }
very old = { 80, 5, 0 } ,
```

where age is the linguistic variable and *child, young, middle-aged, old, very old* are linguistic values. The generated fuzzy sets based on this description are shown in Fig. 2. The corresponding fuzzy sets in this example are defined as triangle shaped sets, with their core and their steepness on the left and right sides respectively (steepness value 0 can be used to form trapezoidal fuzzy sets). Practically the linguistic values are referring to fuzzy sets. Linguistic values together form a linguistic term set, which correspond to the linguistic variables.

Operations on fuzzy sets can also be performed. Fuzzy triangular norms (t-norm) are generalized operations on fuzzy sets. The basic fuzzy t-norm corresponds to the intersection

set operation, and the basic fuzzy t-conorm (also called s-norm) corresponds to the union set operation. The simplest t-norm is the minimum function (the greatest t-norm), and maximum is the simplest t-conorm (the least t-conorm) [71], also known as s-norm. Several other t-norms and t-conorms exist [11], e.g.: product and probabilistic norm, Łukasiewicz t-norm and t-conorm, drastic product and sum, nilpotent minimum and maximum, etc. A good overview on triangular norms can be found in [45].

Using fuzzy sets, linguistic variables, terms and by defining the relationship between terms fuzzy inference systems (FIS) can be constructed. In a FIS, the form of knowledge representation is in the form of fuzzy rules. Fuzzy rules can be defined in the form of IF-THEN rules. General format of fuzzy rules is the following (predicate then conclusion):

$$\text{IF } x \text{ is } A \text{ THEN } y \text{ is } B \tag{2}$$

The predicate part is also called antecedent part, and the conclusion is also referred as the consequent. For more details and example see [72].

Having the defined linguistic variables, terms, based on the fuzzy rules (fuzzy rule base) a FIS can determine the conclusion for the different observations gathered from the environment (inputs). Various fuzzy reasoning methods exist for the calculation of the conclusion, e.g. the classical Zadeh-Mamdani-Larsen Compositional Rule of Inference (CRI) [72], [44], [42] or the Takagi - Sugeno fuzzy inference [55], [57]. The mentioned fuzzy inference systems must have a full coverage of rules for all the possible observations, so the number of rules in a FIS depends on the number of the used antecedents and linguistic terms.

It is obvious that the observations cannot be directly used in a fuzzy inference system, because a FIS expects fuzzy values as inputs, therefore the inputs have to be processed first. The process of converting a crisp input value to a fuzzy value is commonly referred to as ’fuzzification’. The fuzzy inference engine then calculates a conclusion based on the supplied fuzzy rule base and its data base which contains the membership function definitions. The resulting fuzzy set is then defuzzified (‘defuzzification’ is the process, when the output fuzzy value is converted into a crisp value, which can be then used outside a fuzzy system) and fed back into the system (the process under control). A fuzzy controller is a complete system (see Fig. 3.), which includes all the components to make a fuzzy inference system with real inputs and real outputs. There are fuzzy reasoning methods, where the output of the inference engine is directly a crisp value, hence there is no need for defuzzification. E.g. the zero order Takagi-Sugeno fuzzy inference method [57], where the conclusion is a singleton fuzzy set (constant), therefore it can be used directly without defuzzification. Another way of eliminating the defuzzification step is to use Direct Fuzzy Inference [41], where only real (crisp) data is used both on the input and on the output side. For more general details on the structure of fuzzy controllers see [23] and [29].

The term ’fuzzification’ is under dispute (M. Sugeno), because ‘fuzzification’ could suggest that fuzzy control systems work by a kind of transformation where ‘defuzzification’ is the inverse of the process.

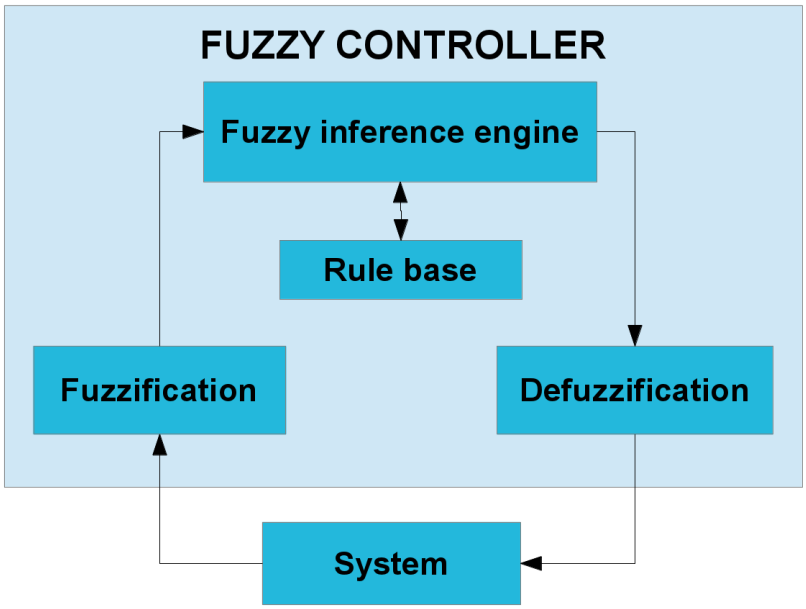

Fig. 3. Block diagram of a fuzzy controller

In the following subchapter the topic of fuzzy rule interpolation will be presented in details. Fuzzy rule interpolation methods allow fuzzy inference systems to work without having a complete rule base. Superseding the need for complete rule bases new possibilities arise in fuzzy controllers.

2.2 Fuzzy rule interpolation

In classical fuzzy reasoning methods (e.g. the Zadeh-Mamdani-Larsen compositional rule of inference [72], [44], [42]), it is obvious that having covering (complete) rule bases is a must.

A traditional fuzzy-rule-based system requires a complete rule base with all of the possible rules set, even though lots of these rules are unimportant from the standpoint of the actual application. A fuzzy rule base is called sparse or incomplete if an observation that does not hit any of the rules in the rule base may exist. Accordingly, there can be observations for which no conclusion can be reached with traditional fuzzy reasoning techniques.

Fuzzy Rule Interpolation (FRI) methods are efficient knowledge-representation structures with relatively few rules. In spite of their good knowledge representation efficiency, their high computational demand makes the FRI methods hardly suitable for embedded real-time applications, for which short reasoning time has a high importance.

On the other hand, in many embedded control application areas, having no conclusion is an avoidable situation. There are some traditional workarounds for such situations in the literature, e.g., applying the last real conclusion instead of the missing one, but these can have some unpredictable side effects. One real solution for the sparse rule base is the application of fuzzy rule interpolation (FRI) methods. In this case, the derivable rules are intentionally missing from the rule base, as FRI methods are capable of providing

reasonable (interpolated) conclusions even if none of the existing rules fire under the current observation. The rule base of an FRI system is not necessarily complete, so it can contain the most significant fuzzy rules alone without risking the chance of having no conclusion for some of the observations. In this case, since there is an efficient knowledge representation, a considerable amount of unnecessary work can be avoided during the rule base creation.

One of the first FRI techniques was published by Kóczy and Hirota [25]. It is usually referred as the KH method. It is applicable to convex and normal fuzzy (CNF) sets. It determines the conclusion by its α-cuts in such a way that the ratio of distances between the conclusion and the consequents should be identical with the ones between the observation and the antecedents for all important α-cuts. Later this method was named as the Fundamental Equation of the fuzzy Rule Interpolation (FERI) [3], which is:

$$d(A^*, A_1) : d(A^*, A_2) = d(B^*, B_1) : d(B^*, B_2) , \tag{3}$$

where A_1 and A_2 are the antecedent sets of the surround rules, A^* is the observation, B_1 and B_2 are the consequent sets of the surround rules and B^* is the approximated conclusion. The applied formula can be solved for B^* for relevant α-cuts after decomposition.

It is shown in, e.g. in [27], [28] that the conclusion of the KH method is not always directly interpretable as fuzzy set. This drawback motivated many alternative solutions. A modification was proposed by Vass, Kalmár and Kóczy [65] (VKK method), where the conclusion is computed based on the distance of the centre points and the widths of the α-cuts, instead of lower and upper distances. VKK method decreases the applicability limit of KH method, but does not eliminate it completely. The technique cannot be applied if any of the antecedent sets is singleton (the width of the antecedent's support must be nonzero). In spite of the disadvantages, KH is popular because its simplicity that infers its advantageous complexity properties. It was generalized in several ways. Among them the stabilized KH interpolator is emerged, as it is proved to hold the universal approximation property [62], [59]. This method takes into account all flaking rules of an observation in the calculation of the conclusion in extent to the inverse of the distance of antecedents and observation. The universal approximation property holds if the distance function is raised to the power of the input's dimension.

Another modification of KH is the modified alpha-cut based interpolation (MACI) method [60], which alleviates completely the abnormality problem. MACI's main idea is the following: it transforms fuzzy sets of the input and output universes to such a space where abnormality is excluded, then computes the conclusion there, which is finally transformed back to the original space. MACI uses vector representation of fuzzy sets and originally applicable to CNF sets [70]. These latter conditions (CNF sets) can be relaxed, but it increases the computational need of the method considerably [61]. MACI is one of the most applied FRI methods [69], since it preserves advantageous computational and approximate nature of KH, while it excludes its abnormality.

Another fuzzy interpolation technique was proposed by Kóczy et al. [26]. It is called conservation of 'relative fuzziness' (CRF) method, which notion means that the left (right) fuzziness of the approximated conclusion in proportion to the flanking fuzziness of the

neighboring consequent should be the same as the (left) right fuzziness of the observation in proportion to the flanking fuzziness of the neighboring antecedent. The technique is applicable to CNF sets. An improved fuzzy interpolation technique for multidimensional input spaces (IMUL) was proposed in [68], and described in details in [69]. IMUL applies a combination of CRF and MACI methods, and mixes advantages of both. The core of the conclusion is determined by MACI method, while its flanks by CRF. The main advantages of this method are its applicability for multi-dimensional problems and its relative simplicity.

Conceptually different approaches were proposed by Baranyi *et al* [3] based on the relation and on the semantic and inter-relational features of the fuzzy sets. The family of these methods applies 'General Methodology' (GM); this notation also reflects to the feature that these methods are able to process arbitrary shaped fuzzy sets. The basic concept is to calculate the reference point of the conclusion based on the ratio of the distances between the reference points of the observation and the antecedents. Then, a single rule reasoning method (revision function) is applied to determine the final fuzzy conclusion based on the similarity of the fuzzy observation and an 'interpolated' observation. Several methods follow the two-step idea of the GM. For example FRIPOC (Fuzzy Rule Interpolation based on Polar Cuts) by Johanyák and Kovács [16] extends the range of the applicable membership functions types by introducing the concept of polar cuts using a polar coordinate system, LESFRI by Johanyák and Kovács in [17] and [18] preserves the characteristic shape type of the antecedent and consequent partitions by applying the method of least squares, and VEIN [19] solves the task of rule interpolation in the vague environment by the help of the set interpolation method VESI proposed by Johanyák in [20].

Most of the FRI methods share the burden of high computational demand, e.g. the task of searching for the two closest rules surrounding the actual observation, and calculating the conclusion at least in some characteristic α-cuts. Moreover, in some methods, the interpretability of the fuzzy conclusion gained is not straightforward [27]. There have been a lot of efforts to rectify the interpretability of the interpolated fuzzy conclusion [60]. In [3], Baranyi et al. give a comprehensive overview of recent FRI methods. Beyond these problems, some of the FRI methods are originally defined for one dimensional input space and need special extension for the multidimensional case (e.g. [13] and [14]). In [69], Wong *et al.* give a comparative overview of the FRI methods capable of multidimensional input space. In [13], Jenei introduces an axiomatic treatment of the FRI methods.

The high computational demand, mainly from the search for the two closest surrounding rules to an arbitrary observation in the multidimensional antecedent space, makes many of these methods hardly suitable for real-time applications. Some FRI methods, (e.g. the method introduced by Jenei et al. in [14], FRIPOC [16], LESFRI [17], and VEIN [19]), eliminate the search for the two closest surrounding rules by taking all the rules into consideration, hence speeding up the reasoning process. On the other hand, keeping the goal of constructing a fuzzy conclusion and not simply speeding up the reasoning process; they still require some additional (or repeated) computational steps for the elements of the level set (or at least some relevant α levels). An application-oriented aspect of the FRI, the low computational and resource demand is emerging in the concept of Fuzzy Interpolation

based on Vague Environment (FIVE). In the following, the implementation details of this method will be studied.

2.3 The FIVE fuzzy rule interpolation method

The FIVE (Fuzzy rule Interpolation based on Vague Environment) method was originally introduced in [31], [32] and [33] and it was developed to fulfill the speed requirements of direct fuzzy control. This case the conclusions of the fuzzy controller are applied directly as control actions in a real-time system, so the concept of the FIVE method is an application oriented aspect of the FRI techniques. Most of the control applications serves crisp observations and requires crisp conclusions from the controller. Adopting the idea of the vague environment (VE) [24], FIVE can handle the antecedent and consequent fuzzy partitions of the fuzzy rule base by scaling functions [24] therefore it can turn the task of fuzzy interpolation to crisp interpolation. The idea of a vague environment is based on the similarity or in other words the indistinguishability of elements. In a vague environment the fuzzy membership function $\mu_A(x)$ is indicating the level of similarity of x to a specific element a which is a representative or prototypical element of the fuzzy set $\mu_A(x)$, or it can be interpreted as the degree to which x is indistinguishable from a [24] (see e.g. on. Fig. 4.).

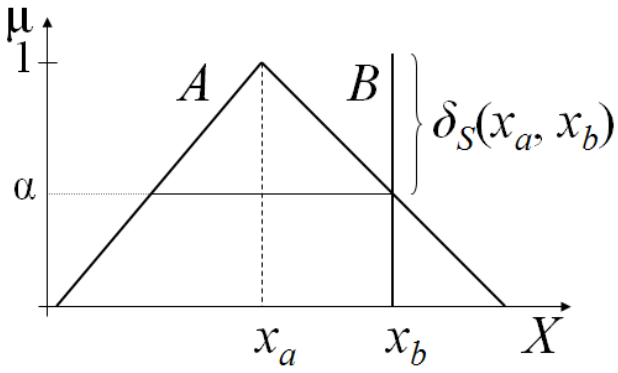

Fig. 4. The α-cuts of $\mu_A(x)$ contain the elements that are (1−α)-indistinguishable from a, where A is a fuzzy set and B is a singleton fuzzy set.

Two values in a vague environment are ε-distinguishable if their distance is greater than ε, where the distances are weighted distances. The weighting factor or function is called scaling function [24] :

$$\delta_s(x_a, x_b) = \left| \int_{x_b}^{x_a} s(x)dx \right| \le \varepsilon \ , \tag{4}$$

where $\delta_s(x_a, x_b)$ is the scaled distance of the values x_a, x_b and $s(x)$ is the scaling function on X.

The scaling function serves the purpose of describing the shapes of the fuzzy sets in the partition. After determining the vague environment of both the antecedent and consequent part universes (the scaling function or at least the approximate scaling function [31], [33]), every member set of the fuzzy partition can be characterized by points in that vague

environment (for example see the exact scaling function *s* in Fig. 5., and also see the approximated scaling function s shown in Fig. 6.). Trapezoidal shaped fuzzy sets can also be constructed from two triangle shaped sets, both having zero steepness on the flanking sides (see e.g. Fig. 7.).

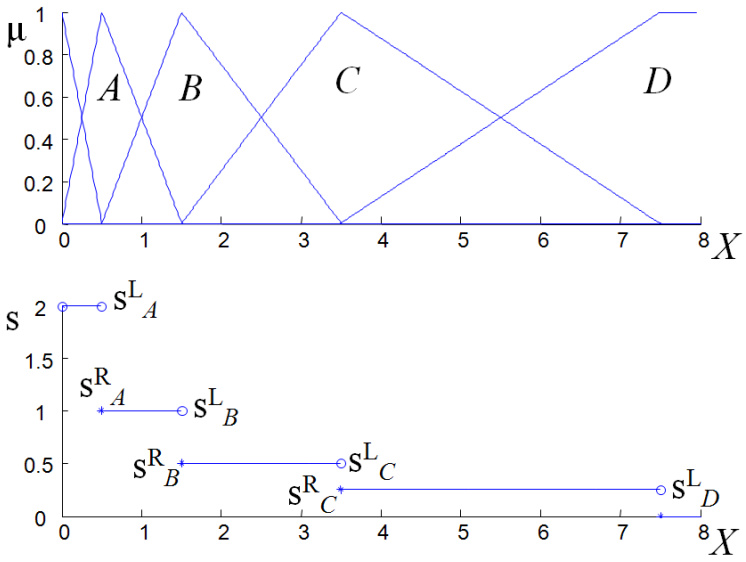

Fig. 5. A Ruspini fuzzy partition (fuzzy sets A-Δ) and its scaling function s(x) on the universe of discourse X

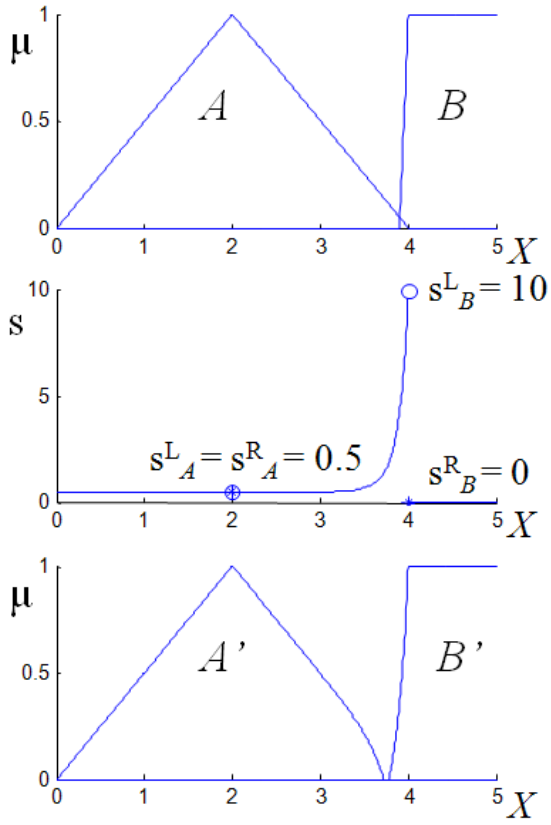

Fig. 6. A fuzzy partition (*A*, *B*), its approximate scaling function s(*x*), and the corresponding approximated partition (*A*', *B*') on the universe of discourse *X*

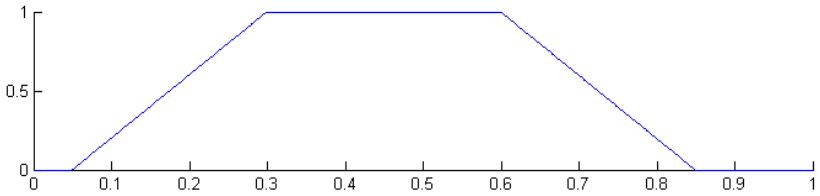

Fig. 7. A trapezoidal membership function composed from two triangle shaped membership functions

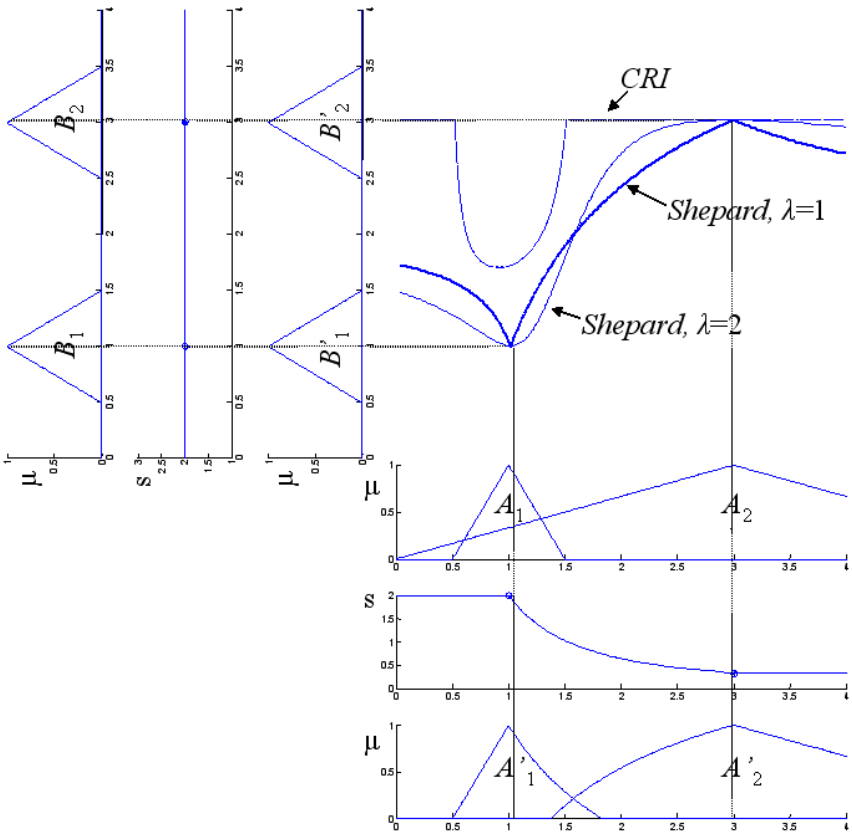

Fig. 8. Interpolation of two fuzzy rules (R_i: $A_i \rightarrow B_i$), by the Shepard operator based *FIVE*, and for comparison the min-max *CRI* with COG defuzzification

Fig. 8. presents an example of a one dimensional antecedent and consequent system with two fuzzy rules. Therefore if the observation is a singleton, any crisp interpolation, extrapolation, or regression method can be adapted very simply for FRI [31], [33]. In the method FIVE because of its simple multidimensional applicability, the Shepard operator based interpolation (first introduced in [54]) was adapted (see e.g. on Fig. 8.).

To be more precise, if there are singleton rule consequents (c_k), the fuzzy rules R_k have the following form:

$$\textbf{If } x_1 = A_{k,1} \textbf{ And } x_2 = A_{k,2} \textbf{ And } \ldots \textbf{ And } x_m = A_{k,m} \textbf{ Then } y = c_k \qquad (5)$$

Adapting the VE concept and the scaling-function-based similarity calculation to the Shepard-operator-based interpolation, the conclusion of the interpolative fuzzy reasoning can be obtained as [34]:

$$y(\mathbf{x}) = \begin{cases} c_k & \text{if } x = a_k \text{ for some } k, \\ \left(\sum_{k=1}^{r} c_k / \delta_{s,k}^{\lambda} \right) \Big/ \left(\sum_{k=1}^{r} 1 / \delta_{s,k}^{\lambda} \right) & \text{otherwise.} \end{cases} \quad (6)$$

where $\lambda > 0$ and $\delta_{s,k}$ are scaled distances:

$$\delta_{s,k} = \delta_s(\mathbf{a}_k, \mathbf{x}) = \left[\sum_{i=1}^{m} \left(\int_{a_{k,i}}^{x_i} s_{X_i}(x_i) dx_i \right)^2 \right]^{1/2}, \quad (7)$$

and s_{X_i} is the i^{th} scaling function of the m dimensional antecedent universe, $\mathbf{x}$ is the m dimensional crisp observation, and $\mathbf{a}_k$ are the cores of the m dimensional fuzzy rule antecedents $\boldsymbol{A}_k$.

The consequent and antecedent sides of the vague environment can be also precalculated and cached, which provides the promptness of the method. Beyond the simplicity and therefore the high reasoning speed, the original FIVE method has obvious drawbacks: the lack of the fuzziness on the observation and on the conclusion side.

The explanation is that this deficiency is inherited from the nature of the applied vague environment, which describes the indistinguishability of two points and therefore the similarity of a fuzzy set and a singleton only. The lack of the fuzziness on the conclusion side has a small influence on common applications where the next step after the fuzzy reasoning is the defuzzification. On the other hand, the lack of the fuzziness on the observation side can restrict applicability of the method. Moreover an extension of the original FIVE method was suggested in [34], where the question of the fuzzy observation is handled by merging vague environments of the antecedent universes and the fuzzy observation. An implementation of FRI FIVE as a component of the FRI MATLAB Toolbox [15] can be downloaded from [75].

III Fuzzy Rule Interpolation-based Q-learning

This chapter presents a novel reinforcement learning method based on fuzzy rule interpolation. First, reinforcement learning and one popular learning algorithm, the Q-learning is presented briefly before the discussion of the developed FRI-based Q-learning algorithm (introduced in: [82], [81], [84] and [90]).

3.1 Reinforcement learning

The strength of reinforcement learning lies in the fact that it does not specify how to solve a particular problem, instead the final goal is defined. So it focuses on what to do not how to do. The primary ideas of reinforcement learning techniques (dynamical system state and the idea of 'optimal return' / 'value' function) are inherited from optimal control, Markov decision processes and dynamic programming [4]. Reinforcement learning techniques are a kind of trial-and-error style techniques adapting to dynamic environment via incremental iterations, without the need for training the system with pre-fabricated examples like with supervised learning methods.

The basic components of reinforcement learning are: states, actions, rewards, and policy. The 'states' are for describing the possible values of the state variables of an agent. The 'actions' refers to a set of the possible actions the agent can choose from. The 'rewards' supply the goodness value of the chosen action taking the agent to a new state. The 'policy' is the description of the agent behaviour, in the form of mapping between the agent states and the corresponding suitable actions. It determines which action should be chosen by the agent in the various possible states for achieving the highest possible reward.

Finding an optimal policy by building the state-value- or action-value-function is a common goal of the reinforcement learning strategies [56]. The state-value-function $V^{\pi}(s)$, is a function of the expected return (a function of the cumulative reinforcements), related to a given state $s \in S$ as a starting point, following a given π policy. These rewards (reinforcements) are the expression of the desired goal of the learning agent as a kind of evaluation following the previous action (in spite of the instructive manner of error feedback based approximation techniques, like the gradient descent optimization). The action-value function $Q^{\pi}(s,a)$ is a function of the expected return, in case of taking action $a \in A_s$ in a given state s, and then following a given policy π. Having the action-value-function, the optimal (greedy) policy, which always takes the optimal (the greatest estimated value) action in every state, can be constructed as [56]:

$$\pi(s) = \arg\max_{a \in A_s} Q^{\pi}(s,a) \tag{8}$$

Namely for estimating the optimal policy, the action-value function $Q^{\pi}(s,a)$ is needed to be approximated. The function is learned based on rewards or punishments (negative rewards) given by the environment on the goodness of the selected action in the given observable state. Having a complex task to adapt, both the number of possible states and the

number of the possible actions could be an extremely high value. This implies the problem that it takes a considerable amount of computing resources and computation time, which is a significant drawback of reinforcement learning; however there are some cases where a distributed approach with continuous reward functions can reduce these resource needs [21]. Generally reinforcement learning methods can lead to results in practically acceptable time only in relatively small state and action spaces.

A common problem regarding the construction of the optimal policy is finding an acceptable balance between exploration and exploitation. For achieving high rewards it is straightforward to use those actions, which were proved to be highly efficient before. Following this principle rewards could be high, but the search for other possible better actions is omitted. A way to increase exploration is to use delayed rewards. In this case rewards are given only after completing several actions taking through several states. This could result in a final reward which is higher than the actual policy would suggest using immediate rewards.

For solving reinforcement learning problems (finding the optimal policy) a commonly used algorithm is the Temporal Difference learning (TD-learning) [56], which does not need to have a model of the environment. TD-learning can estimate the value updates without having the value of the final reward (without the agent reaching the end condition), when the final reward is available the value updates are backpropagated (a neural network can be used for backpropagation, like in the case with TD-Gammon [58]). The most widely used variations of the TD-learning are Q-learning [67] and SARSA (originally entitled 'modified Q-learning') [53]. Q-learning is an off-policy, SARSA is an on-policy TD-learning variation. On-policy algorithms must follow their policy therefore those can only learn from the current policy being learned. In contrast, off-policy strategies can profit from actions selected not conforming to the currently learned policy. Hence off-policy strategies are more versatile from the viewpoint of exploration.

Furthermore with the introduction of fuzzy models, the discrete Q-learning can be extended to continuous state and action space, which in case of suitably chosen states can lead to the reduction of the size of the state-action space [35].

3.2 Q-learning and Fuzzy Q-learning

As Q-learning [67] is considered to be the most popular reinforcement learning method [52], it is plausible to choose it for further study and extension possibilities. The purpose of the Q-learning is finding the fixed-point solution Q of the Bellman Equation [4] through iteration. In discrete environment Q-Learning [66], the action-value-function is approximated by the following iteration:

$$Q_{i,u} \approx \tilde{Q}_{i,u}^{k+1} = \tilde{Q}_{i,u}^{k} + \Delta\tilde{Q}_{i,u}^{k+1} = \tilde{Q}_{i,u}^{k} + \alpha_{i,u}^{k} \cdot \left(g_{i,u,j} + \gamma \cdot \max_{v \in U} \tilde{Q}_{j,v}^{k+1} - \tilde{Q}_{i,u}^{k} \right) \quad (9)$$

$\forall i \in I, \forall u \in U$, where $\tilde{Q}_{i,u}^{k+1}$ is the *k+1* iteration of the action-value taking the u^{th} action A_u in the i^{th} state S_i , S_j is the new (j^{th}) observed state, $g_{i,u,j}$ is the observed reward completing

the $S_i \to S_j$ state-transition, γ is the discount factor and $\alpha_{i,u}^k \in [0,1]$ is the step size parameter (which can change during the iteration steps), *I* is the set of the discrete possible states and *U* is the set of the discrete possible actions. Many solutions exist [2], [6], [8], [12] for applying this iteration to continuous environment by adopting fuzzy inference (Fuzzy Q-Learning). Traditionally the simplest FQ-Learning, the 0-order Takagi-Sugeno Fuzzy Inference model is the most common. Therefore in this work this one is studied (a slightly modified, simplified version of the Fuzzy Q-Learning introduced in [2] and [12]). In this case, for characterizing the value function *Q(s,a)* in continuous state-action space, the order-0 Takagi-Sugeno Fuzzy Inference System approximation $\tilde{Q}(s,a)$ is adapted in the following manner:

If *s* **is** S_i **And** *a* **is** A_u **Then** $\tilde{Q}(s,a) = Q_{i,u}$, $i \in I, u \in U$, (10)

where S_i is the label of the i^{th} membership function of the *n* dimensional state space, A_u is the label of the u^{th} membership function of the one dimensional action space, $Q_{i,u}$ is the singleton conclusion and $\tilde{Q}(s,a)$ is the approximated continuous state-action-value function. Having the approximated state-action-value function $\tilde{Q}(s,a)$, the optimal policy can be constructed by function (8). Setting up the antecedent fuzzy partitions to be Ruspini partitions, the order-0 Takagi-Sugeno fuzzy inference forms the following approximation function:

$$\tilde{Q}(s,a) = \sum_{i_1,i_2,\ldots,i_N,u}^{I_1,I_2,\ldots,I_N,U} \prod_{n=1}^{N} \mu_{i_n,n}(s_n) \cdot \mu_u(a) \cdot q_{i_1 i_2 \ldots i_N u} \tag{11}$$

where $\tilde{Q}(s,a)$ is the approximated state-action-value function, $\mu_{i_n,n}(s_n)$ is the membership value of the i_n^{th} state antecedent fuzzy set at the n^{th} dimension of the *N* dimensional state antecedent universe at the state observation s_n, $\mu_u(a)$ is the membership value of the u^{th} action antecedent fuzzy set of the one dimensional action antecedent universe at the action selection *a*, $q_{i_1 i_2 \ldots i_N u}$ is the value of the singleton conclusion of the $i_1, i_2, \ldots, i_N, u$ -th fuzzy rule. Applying the approximation formula of the Q-learning (9) for adjusting the singleton conclusions in (11), leads to the following function:

$$\begin{aligned} q_{i_1 i_2 \ldots i_N u}^{k+1} &= q_{i_1 i_2 \ldots i_N u}^{k} + \prod_{n=1}^{N} \mu_{i_n,n}(s_n) \cdot \mu_u(a) \cdot \Delta \tilde{Q}_{i,u}^{k+1} = \\ &= q_{i_1 i_2 \ldots i_N u}^{k} + \prod_{n=1}^{N} \mu_{i_n,n}(s_n) \cdot \mu_u(a) \cdot \alpha_{i,u}^{k} \cdot \left(g_{i,u,j} + \gamma \cdot \max_{v \in U} \tilde{Q}_{j,v}^{k+1} - \tilde{Q}_{i,u}^{k} \right) \end{aligned} \tag{12}$$

where $q_{i_1 i_2 .. i_N u}^{k+1}$ is the *k+1* iteration of the singleton conclusion of the $i_1 i_2 ... i_N u$ th fuzzy rule taking action A_u in state S_i, S_j is the new observed state, $g_{i,u,j}$ is the observed reward completing the $S_i \to S_j$ state-transition, γ is the discount factor and $\alpha_{i,u}^k \in [0,1]$ is the step size parameter. The $\mu_{i_n,n}(s_n) \cdot \mu_u(a)$ is the partial derivative of the conclusion of the 0-order Takagi-Sugeno fuzzy inference $\tilde{Q}(s,a)$ with respect to the fuzzy rule consequents $q_{u,i}$ according to (11), required for the applied steepest-descent optimization method. The $\tilde{Q}_{j,v}^{k+1}$ and $\tilde{Q}_{i,u}^k$ action-values can be approximated by equation (11).

3.3 The FIVE FRI-based Q-learning

Replacing the zero-order Takagi-Sugeno fuzzy model of the FQ-learning with the FIVE model leads to the proposed FRIQ-learning method.

Introducing the FIVE FRI in Q-learning gives the possibility of omitting rules (action-state values) from the rule base and gaining the potentiality of applying the proposed method in higher state dimensions with a reduced rule-base sized action-state space. An example for effective rule base reduction by FRI FIVE in a given situation is introduced in [36].

The FIVE FRI based fuzzy model in case of singleton rule consequents [34] can be expressed by the following formula:

$$\tilde{Q}(s,a) = \begin{cases} q_{i_1 i_2 .. i_N u} & \text{if } x = a_k \text{ for some } k, \\ \left(\sum_{k=1}^{r} q_{i_1 i_2 .. i_N u} \Big/ \delta_{s,k}^{\lambda} \right) \Big/ \left(\sum_{k=1}^{r} 1 \Big/ \delta_{s,k}^{\lambda} \right) & \text{otherwise.} \end{cases} \quad (13)$$

where the fuzzy rules R_k have the form:

If $x_1 = A_{k,1}$ **And** $x_2 = A_{k,2}$ **And** … **And** $x_m = A_{k,m}$ **Then** $y = c_k$, (14)

$\delta_{s,k}$ is the scaled distance:

$$\delta_{s,k} = \delta_s(\mathbf{a}_k, \mathbf{x}) = \left[\sum\nolimits_{i=1}^{m} \left(\int_{a_{k,i}}^{x_i} s_{X_i}(x_i) dx_i \right)^2 \right]^{1/2}, \quad (15)$$

and s_{X_i} is the i^{th} scaling function of the *m* dimensional antecedent universe, **x** is the *m* dimensional crisp observation and $\mathbf{a}_k$ are the cores of the *m* dimensional fuzzy rule antecedents $\boldsymbol{A}_k$.

Applying the FIVE FRI method with singleton rule consequents (13) to be the model of the state-action-value function, we get:

$$\tilde{Q}(s,a)=\begin{cases} q_{i_1 i_2 \ldots i_N u} & \text{if x = } a_k \text{ for some } k, \\ \sum_{I_1,I_2,\ldots,I_N,U} \prod_{n=1}^{N}\left(1/\delta_{s,k}^{\lambda}\right)/\left(\sum_{k=1}^{r} 1/\delta_{s,k}^{\lambda}\right)\cdot q_{i_1 i_2 \ldots i_N u} & \text{otherwise.} \end{cases} \quad (16)$$

where $\tilde{Q}(s,a)$ is the approximated state-action-value function.

The partial derivative of the model consequent $\tilde{Q}(s,a)$ with respect to the fuzzy rule consequents $q_{u,i}$, required for the applied fuzzy Q-learning method (12) in case of the FIVE FRI model from (16) can be expressed by the following formula (according to [39]):

$$\frac{\partial \tilde{Q}(s,a)}{\partial q_{i_1 i_2 \ldots i_N u}}=\begin{cases} 1 & \text{if x = } a_k \text{ for some } k, \\ \left(1/\delta_{s,k}^{\lambda}\right)/\left(\sum_{k=1}^{r} 1/\delta_{s,k}^{\lambda}\right) & \text{otherwise} \end{cases} \quad (17)$$

where $q_{u,i}$ is the constant rule consequent of the k^{th} fuzzy rule, $\delta_{s,k}$ is the scaled distance in the vague environment of the observation, and the k^{th} fuzzy rule antecedent, λ is a parameter of Shepard interpolation (in case of the stable multidimensional extension of the Shepard interpolation it equals to the number of antecedents according to [62]), x is the actual observation, r is the number of the rules.

Replacing the partial derivative of the conclusion of the 0-order Takagi-Sugeno fuzzy inference (12) with the partial derivative of the conclusion of FIVE (17) with respect to the fuzzy rule consequents $q_{u,i}$ leads to the following equation for the Q-Learning action-value-function iteration:

$$\begin{aligned} &\text{if } \mathbf{x}=\mathbf{a}_k \text{ for some } k: \\ &q_{i_1 i_2 \ldots i_N u}^{k+1}=q_{i_1 i_2 \ldots i_N u}^{k}+\Delta\tilde{Q}_{i,u}^{k+1}= \\ &=q_{i_1 i_2 \ldots i_N u}^{k}+\alpha_{i,u}^{k}\cdot\left(g_{i,u,j}+\gamma\cdot\max_{v\in U}\tilde{Q}_{j,v}^{k+1}-\tilde{Q}_{i,u}^{k}\right) \\ &\text{otherwise}: \\ &q_{i_1 i_2 \ldots i_N u}^{k+1}=q_{i_1 i_2 \ldots i_N u}^{k}+\prod_{n=1}^{N}\left(1/\delta_{s,k}^{\lambda}\right)/\left(\sum_{k=1}^{r} 1/\delta_{s,k}^{\lambda}\right)\cdot\Delta\tilde{Q}_{i,u}^{k+1}= \\ &=q_{i_1 i_2 \ldots i_N u}^{k}+\prod_{n=1}^{N}\left(1/\delta_{s,k}^{\lambda}\right)/\left(\sum_{k=1}^{r} 1/\delta_{s,k}^{\lambda}\right)\cdot\alpha_{i,u}^{k}\cdot\left(g_{i,u,j}+\gamma\cdot\max_{v\in U}\tilde{Q}_{j,v}^{k+1}-\tilde{Q}_{i,u}^{k}\right) \end{aligned} \quad (18)$$

where $q_{i_1 i_2 .. i_N u}^{k+1}$ is the *k+1* iteration of the singleton conclusion of the $i_1 i_2 ... i_N u$ th fuzzy rule taking action A_u in state S_i, S_j is the new observed state, $g_{i,u,j}$ is the observed reward completing the $S_i \to S_j$ state-transition, γ is the discount factor and $\alpha_{i,u}^k \in [0,1]$ is the step size parameter.

As in the previous chapter the $\widetilde{Q}_{j,v}^{k+1}$ and $\widetilde{Q}_{i,u}^{k}$ action-values can be approximated by equation (18), which now uses the FIVE FRI model.

3.3.1 Application example for presenting FIVE-based Q-learning

As a benchmark for the proposed FRI based Q-learning method, the well known cart-pole (reversed pendulum) problem is chosen as an application example in this work. An implementation by José Antonio Martin H. which uses SARSA [53] (a Q-learning method) in discrete space is freely available from [74]. In order to make the comparison easier, the application example introduced in [74] was extended to adapt the FIVE FRI model.

For easier comparability purposes in the application example, the discrete Q-learning, and the proposed FRIQ-learning had the same state-action space resolution. In the discrete case the resolution means the number of the discrete cases, in the fuzzy model case, these are the cores of the fuzzy sets in the antecedent fuzzy partitions.

The example program runs through episodes, where an episode means a cart-pole simulation run. The goal of the application is to move the cart to the center position while balancing the pole. Maximum reward is gained when the pole is in vertical position and the cart is on the center position mark. An episode is considered to be successfully finished (gains positive reinforcement in total) if the number of iterations (steps) reaches one thousand while the pole stays up without the cart crashing into the walls. Otherwise the episode is considered to be failed (gains negative reinforcement in total).

Some sample rules from the rule base used for calculating the *q* values can be seen in Table 1. The rule antecedent variables are the following: s_1 – shift of the pendulum, s_2 – velocity of the pendulum, s_3 – angular offset of the pole, s_4 – angular velocity of the pole, *a* – compensation action of the cart. The linguistic terms used in the antecedent parts of the rules are: Negative (N), Zero (Z), Positive (P), the multiples of three degrees in [-12,12] degree interval (N12, N9, N6, N3, Z, P3, P6, P9, P12) and for the actions: from negative to positive in one tenth steps (AN10-AP10, Z). For the easier comparison purposes, the above resolution of the state-action spaces are the same as the resolutions of the discrete Q-learning implementation selected as the reference example (introduced in [74]). The consequents (*q*) are initialized with zero values. These values will be then updated while learning according to (18). The fuzzy rules are defined in the following form:

R_i:
If s_1 = $A_{1,i}$ **and** s_2 = $A_{2,i}$ **and** s_3 = $A_{3,i}$ **and** s_4 = $A_{4,i}$ **and** a = $A_{5,i}$ **Then** q = B_i (19)

Table 1. Sample rules from the 'q calculation' rule base

R#	s_1	s_2	s_3	s_4	a	q
0001	N	N	N12	N	AN10	0
0512	N	Z	N3	N	AN3	0
1024	N	P	P6	N	AP5	0
2048	P	P	N3	P	Z	0
2268	P	P	P12	P	AP10	0

Table 2. Rules which have weights greater than 0.01 in case of a specified observation

R#	s_1	s_2	s_3	s_4	a	w
0013	N	N	N12	N	AP2	0.020319
0054	N	N	N9	N	AP1	0.026501
0055	**N**	**N**	**N9**	**N**	**AP2**	**0.162300**
0056	N	N	N9	N	AP3	0.026501
0097	N	N	N6	N	AP2	0.020319
0432	N	Z	N9	N	AP1	0.010130
0433	N	Z	N9	N	AP2	0.029851
0434	N	Z	N9	N	AP3	0.010130
1147	P	P	N12	N	AP2	0.021033
1188	P	P	N9	N	AP1	0.027540
1189	**P**	**P**	**N9**	**N**	**AP2**	**0.175810**
1190	P	P	N9	N	AP3	0.027540
1231	P	P	N6	N	AP2	0.021033
1566	P	Z	N9	N	AP1	0.010398
1567	P	Z	N9	N	AP2	0.031080
1568	P	Z	N9	N	AP3	0.010398

For testing and validation purposes the randomized action selection (see [74]) (state-action space exploration) was disabled temporarily. With using the same state descriptors with both the original [74] and FRIQ-learning version the results were exactly the same as expected (the state variables hit exactly the cores of the fuzzy sets at the antecedent side, hence the value of the derivative – weight of the rule – was equal to 1).

In continuous environment there are so many states that exploring the whole universe could require tremendous amount of computing power and time. In order to get a useable result in acceptable time but to use the extended capabilities of FRI based Q-learning, one more state value was added to the states [74]. The shift values (s_1) which were originally -1 and 1 are now extended with a new, zero value: -1, 0, 1. The rest of the rule base remained the same as with the two states, hence handling the new 0 value in s_1 is the job of the FRI

based fuzzy inference model. The results of the simulation with the original and the extended s_1 values can be seen on Fig. 9. and Fig. 10. Fig. 9. shows how many iteration steps could the cart survives both with the original two s_1 state values (red) and the extended three s_1 values (green). Fig. 10. shows the cumulated rewards in each iteration cycle (red: two s_1 states, green: three s_1 states). The figures show that both versions learn a suitable state-action-value function, and the one with three s_1 states gives better results at first, but converges slower. Table 2. shows the weights of rule consequent (q) updates in case of a specified observation with the new s_1 state: s_1 = Z, s_2 = N, s_3 = N9, s_4 = N, a = AP2. There are a lot of rules with so small weights that they do not affect the q values considerably, hence only the rules with weights more than 0.01 are shown on Table 2. Except s_1 all of the antecedents hit exactly the cores of the fuzzy sets. The two heaviest rules are typesetted bold, and it can be clearly seen that with using FIVE the rule updates are positive and negative with nearly the same frequencies (requiring the missing zero value).

With the introduction of FIVE FRI in Q-learning instead of discrete state-action spaces, continuous spaces could be applied. Applying continuous spaces can lead to better resolution providing more precise description of the state-action pairs. The application example and the numerical results prove that the proposed FRI model based method performs similarly to the discrete solution in case of the same action-state resolution and circumstances. Even if in this case the FRI based Q-learning performs in a similar way to the discrete solution, it holds the potentialities of action-state space (i.e. rule base in FRI case) reduction.

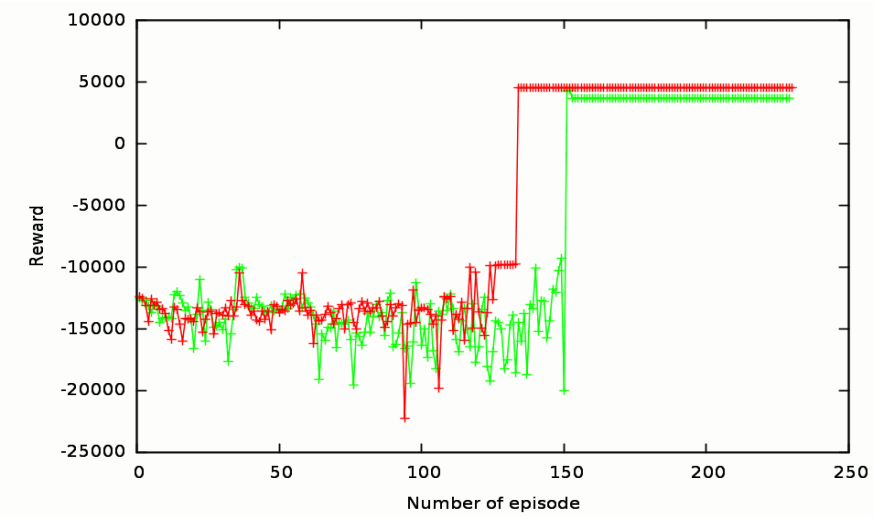

Fig. 9. Steps survived with the original two states (red) and with the extended three states (green).

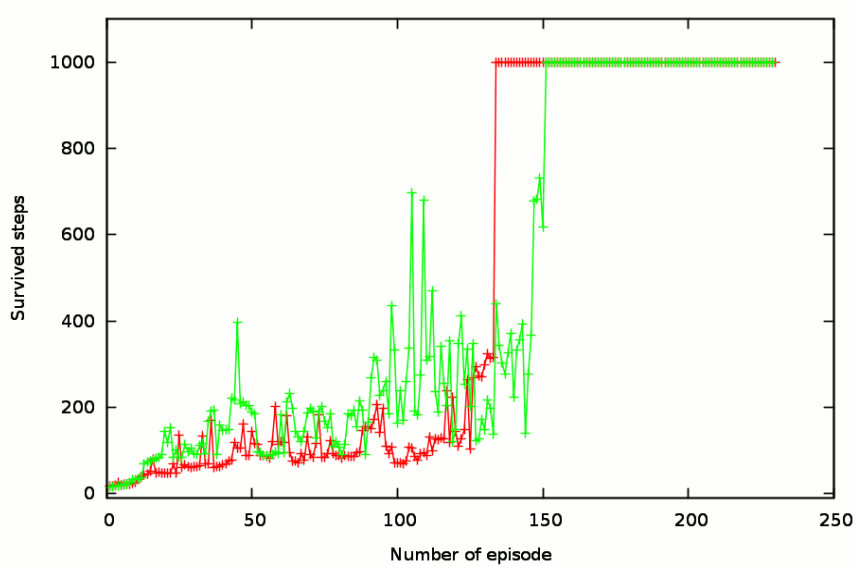

Fig. 10. The cumulative rewards the original two states (red) and with the extended three states (green).

3.4 Incremental rule base creation with FRIQ-learning

Towards achieving the fuzzy rule base size-reduction, an incremental rule base creation strategy is suggested. This method (also introduced in [81] and [84]) simply increases the number of the fuzzy rules by inserting new rules in the required positions (for an example see Fig. 11.). Instead of initially building up a full rule base with the conclusions of the rules (q values) set to a default value, only a minimal sized rule base is created with 2^{N+1} fuzzy rules at the corners of the $N+1$ dimensional antecedent (state-action space) hypercube. Similarly like creating Ruspini partitions with triangular shaped fuzzy sets in all the antecedent universes (see Fig. 11/1.). In cases when the action-value function update (18) is high (e.g. greater than a preset limit ε_Q: $\Delta\tilde{Q} > \varepsilon_Q$), and even the closest existing rule to the actual state is farther than a preset limit ε_s, then a new rule is inserted to the closest possible rule position (see Fig. 11/1.). The possible rule positions are gained by inserting a new state among the existing ones ($s_{k+1}=s_k$, $\forall k > i$, $s_{i+1}=\frac{s_i+s_{i+2}}{2}$, see e.g. in Fig. 11/2.). In case if the update value is relatively low ($\Delta\tilde{Q} \le \varepsilon_Q$), or the actual state-action point is in the vicinity of an already existing fuzzy rule, then the rule base remains unchanged (only the conclusions of the corresponding rules will be updated). The next step is to update the q value, done regarding to the FRIQ-Learning method according to the equation (18) as it was discussed earlier.

Selecting a new rule position could be difficult in case of having many dimensions. To simplify the new rule position calculation it is practical to handle the dimensions separately and choose a position not in the n-dimension space, but in simple vectors for each of the n dimensions. Then use the selected values in the vectors for the position in the n-dimensional space.

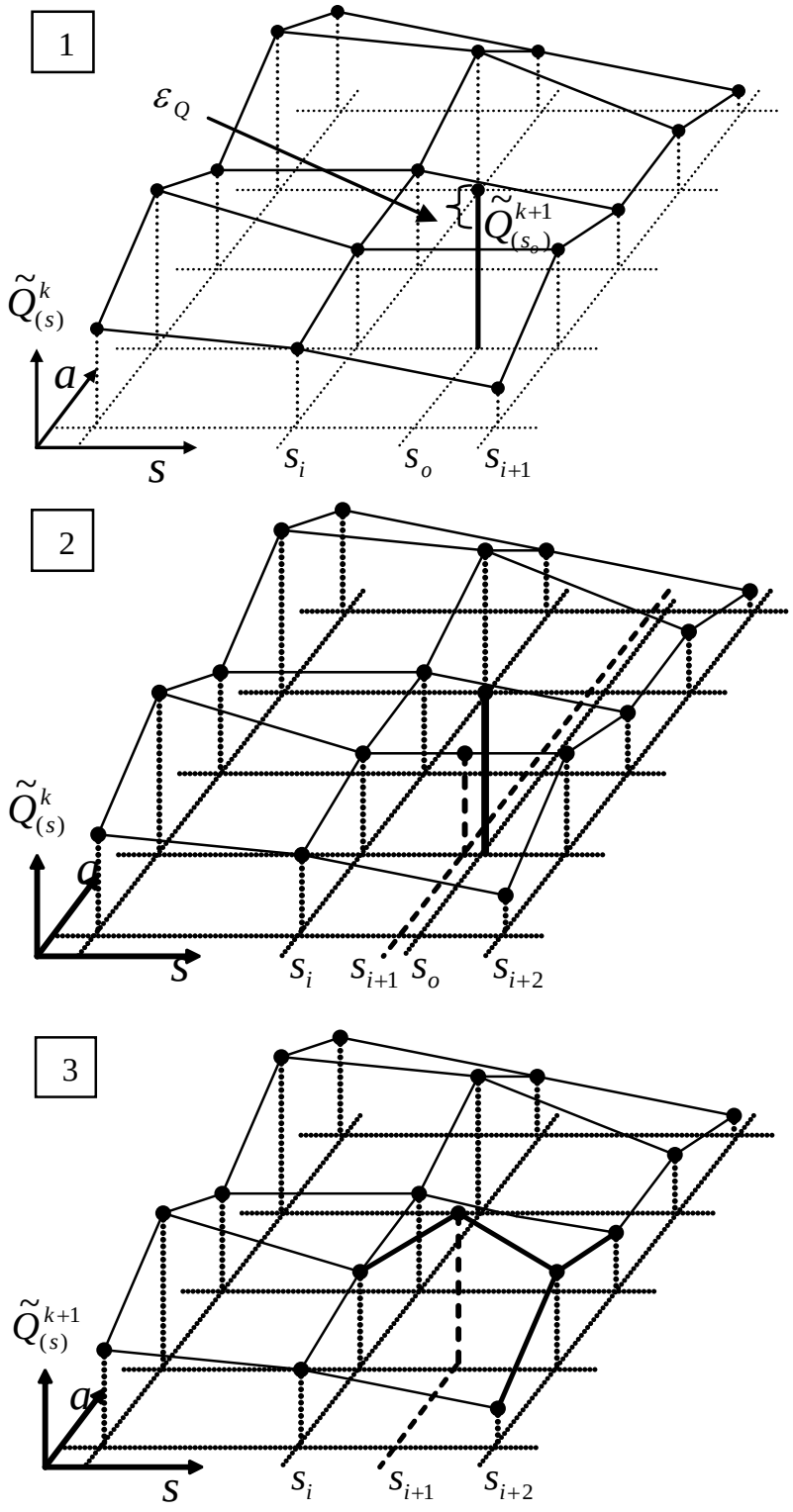

Fig. 11.

1. The next approximation of Q at s_o: $\tilde{Q}^{k+1}_{(s_o)}$.

2. A new fuzzy rule should be inserted at s_{i+1}.

3. Next approximation, with a new rule inserted, and value updated according to (18)

This way the resulting action-value function will be modeled by a sparse rule base which contains only the fuzzy rules which seem to be most relevant in the model. Applying the FIVE FRI method, as stated earlier, allows the usage of sparse rule bases which could result in saving a considerable amount of computational resources and reduced state space.

3.4.1 Application example for the incremental rule base construction

The notorious Q-learning demonstration application, the cart-pole (reversed pendulum) problem is used again for demonstration purposes as a benchmark for the proposed incremental rule-base building FRIQ-learning method. In order to make the benchmarking simple and comparable, the previous application example of FRIQ-learning was modified to make use of the introduced FIVE FRI based rule-base size reduction method. For the easier comparability purposes, in the application example the discrete Q-learning and the proposed reduced rule-base FRIQ-learning had the same resolution of state-action space. Resolution in case of a discrete model means the number of the discrete cases, in the fuzzy model case these are the number of the cores of the fuzzy sets in the antecedent fuzzy partitions.

The rules of the initial rule base used for calculating the q values (set to zero initially) are shown in Table 3. The rule antecedent variables are the following: s_1 – shift of the pendulum, s_2 – velocity of the pendulum, s_3 – angular offset of the pole, s_4 – angular velocity of the pole, a – compensation action of the cart. Five antecedent variables in total, hence the 32 rules (corners of a 5-dimensional hypercube). The linguistic terms used in the antecedent parts of the rules are: Negative (N), Zero (Z), Positive (P), the multiples of three degrees in [-12,12] degree interval (N12, N9, N6, N3, Z, P3, P6, P9, P12) and for the actions: from negative to positive in one tenth steps (AN10-AP10, Z), 21 in total.

For the sake of simpler comparison purposes the above resolution of the state-action spaces are the same as the resolution of the discrete Q-learning implementation selected as the reference example (see [74]). The consequents (q) are initialized with zero values. These values will be then updated while learning according to (18), when the difference of the current and previous q value is considered small. In the opposite case, when the difference is considered large, then the program calculates a new rule position where the new q value should be stored. If the new rule position does not already exits, then it will be inserted into the rule base, otherwise the consequent of the rule and its surrounding rules will be updated like in the case when the q difference is considered small.

Calculation of the new position of a rule is done in the following manner: possible positions are stored separately for each state space and the action state. These vectors are initialized with the possible state/action values. Minimally the start and the end positions of the interval are required, but additional hints can be given to get faster results. When an observation falls close to an existing rule then that existing rule will be updated. Else when the observation falls into the vicinity of the midst of the segment connecting two neighbouring values, a new possible state is registered in the midst of the segment, and that will be the place of the new rule in the corresponding dimension.

Table 3. Rules of the initial Q value calculation rule base

R#	s_1	s_2	s_3	s_4	a	q
01	N	N	N12	N	AN10	0
02	N	N	N12	N	AP10	0
03	N	N	N12	P	AN10	0
04	N	N	N12	P	AP10	0
05	N	N	P12	N	AN10	0
06	N	N	P12	N	AP10	0
07	N	N	P12	P	AN10	0
08	N	N	P12	P	AP10	0
09	N	P	N12	N	AN10	0
10	N	P	N12	N	AP10	0
11	N	P	N12	P	AN10	0
12	N	P	N12	P	AP10	0
13	N	P	P12	N	AN10	0
14	N	P	P12	N	AP10	0
15	N	P	P12	P	AN10	0
16	N	P	P12	P	AP10	0
17	P	N	N12	N	AN10	0
18	P	N	N12	N	AP10	0
19	P	N	N12	P	AN10	0
20	P	N	N12	P	AP10	0
21	P	N	P12	N	AN10	0
22	P	N	P12	N	AP10	0
23	P	N	P12	P	AN10	0
24	P	N	P12	P	AP10	0
25	P	P	N12	N	AN10	0
26	P	P	N12	N	AP10	0
27	P	P	N12	P	AN10	0
28	P	P	N12	P	AP10	0
29	P	P	P12	N	AN10	0
30	P	P	P12	N	AP10	0
31	P	P	P12	P	AN10	0
32	P	P	P12	P	AP10	0

The fuzzy rules are defined in the same form as in (19). The built-in randomized action selection (see [74]) (state-action space exploration) was disabled temporarily for testing and validation reasons.

The comparison of the results of the original discrete method simulation and the FRIQ-learning method using incremental rule-base creation is shown in Fig. 12. and Fig. 13. Fig. 12. shows the number of iteration steps the cart could survive both with the original discrete method (red) and the FRIQ-learning method (green) while Fig. 13. shows the cumulated rewards in each iteration. These figures show that both versions learn a suitable state-action-value function, and the FRIQ-learning version gives better results at first, but converges slower.

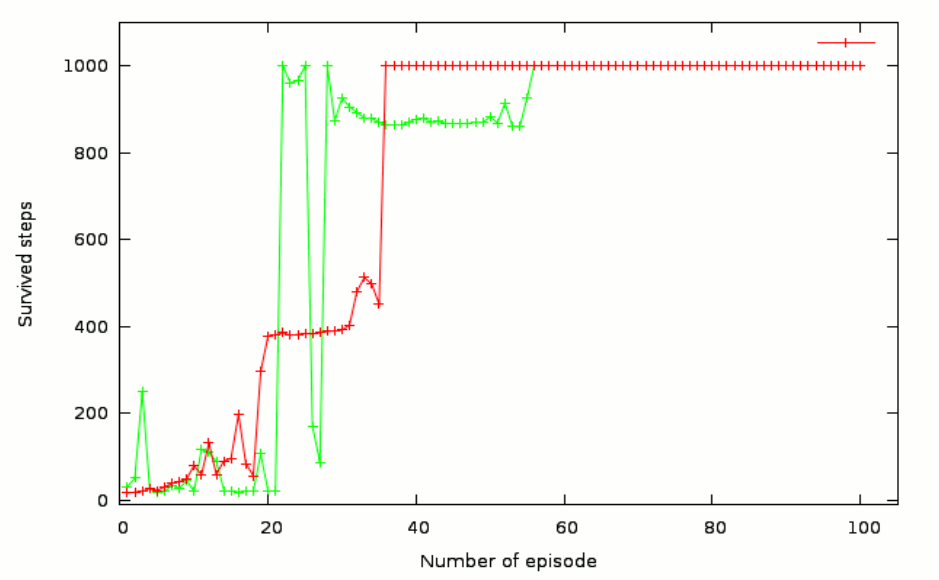

Fig. 12. Survived steps per episode: original discrete method – red line, FRIQ-learning method – green line

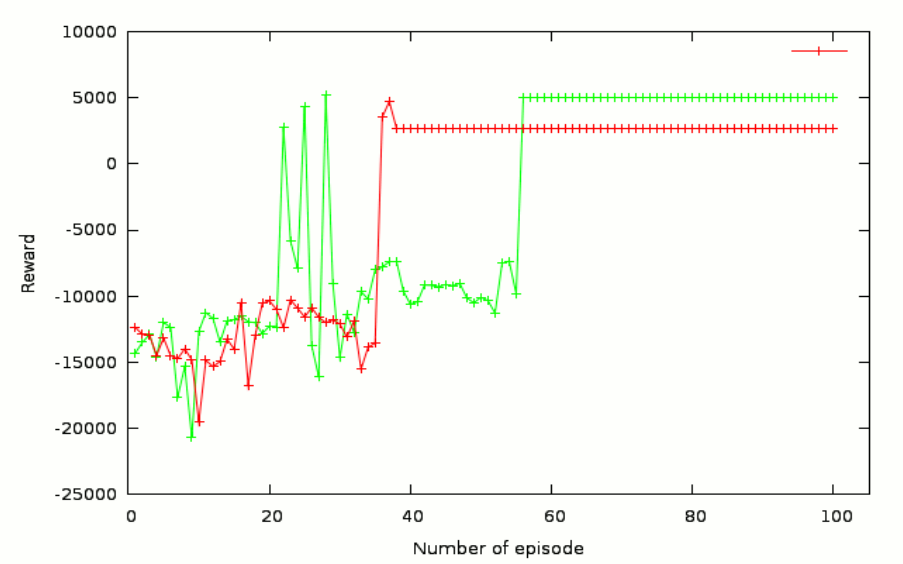

Fig. 13. The cumulative rewards per episode: original discrete method – red line, FRIQ-learning method – green line

The blue curve in Fig. 14. shows the number of fuzzy rules in the rule base. Starting with the initial 32 rules, the size of the rule base quickly rises through the episodes, stabilizing at 182 rules. A higher rate of rule insertion can be observed (magenta coloured curve in Fig. 14.) till the first successful episode, then the insertion rate drops approximately to the third of the previous rate, then stays constant while all the episodes will be successful. For describing the original discrete states 2268 rules would be needed, but the same result can be achieved with only 182 rules.

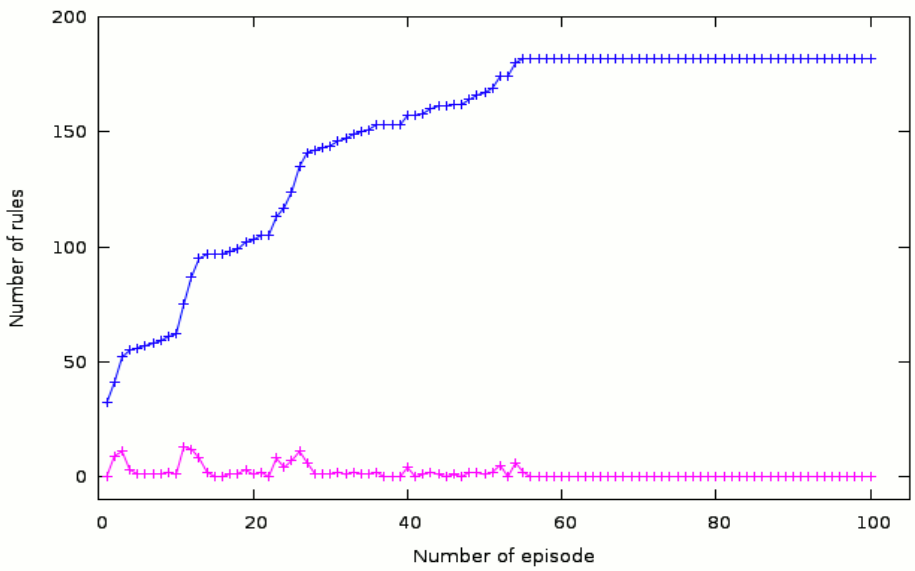

Fig. 14. The number of rules per episode – blue line
The difference in rule numbers per episode – magenta line

3.5 Decremental rule base reduction with FRIQ-learning

The previously completed incrementally constructed rule-base possibly contains rules which were only significant during the construction process itself, but meanwhile their importance lowered in the final rule-base. There can be rules which were superseded by other much 'stronger' or near equal but different rules. Also there can be rules which are redundant (can be calculated by using interpolation – see FRI in Chapter II.) in the finished rule base. It is possible to find and remove these rules from the rule base automatically by using a decremental rule base reduction strategy presented in the followings.

According to the Bellman equation [4] (9) only the highest of the possible Q values for the next $(k+1)^{th}$ iteration step is used in the calculation of the next (currently approximated) Q value:

$$\max_{v \in U} \tilde{Q}_{j,v}^{k+1} \tag{20}$$

In other words this means that high Q values are possibly more significant than lower Q values. This suggest a strategy to try to omit rules from the previously incrementally constructed rule base (e.g. see Fig. 15/1.) which have low Q values as their conclusion.

Following this strategy, the selected rules based on their conclusion value (means the absolute Q value) are omitted one by one, as the whole process is evaluated over and over again. If the rewards given by the environment with the truncated rule base remain the same, or near the same (reward difference is within a preset interval), or maybe higher than the rule is considered to be redundant, therefore it will be removed from the rule base. In the other case when the given reward is considerably lower or the evaluation fails, the rule is considered being a cardinal rule, therefore it has to stay in the rule base (see Fig. 15/2.).

Various thresholds can be used in defining ‘near the same’ depending on the task and requirements. Close matches of the rewards should result in approximately the same steps as were the original incrementally constructed full rule base, when using the final reduced rule base. Accepting relatively greater (depending on the exact reward function), but still valid, differences between the rewards could result in a different step-by-step solution, but the overall task will still be solved. This should be taken into consideration depending on the exact task.

After the reduction process is complete, the final rule base will contain only the most significant rules, in other words this method extracts the cardinal rules which are basically operating the FRI-based system.

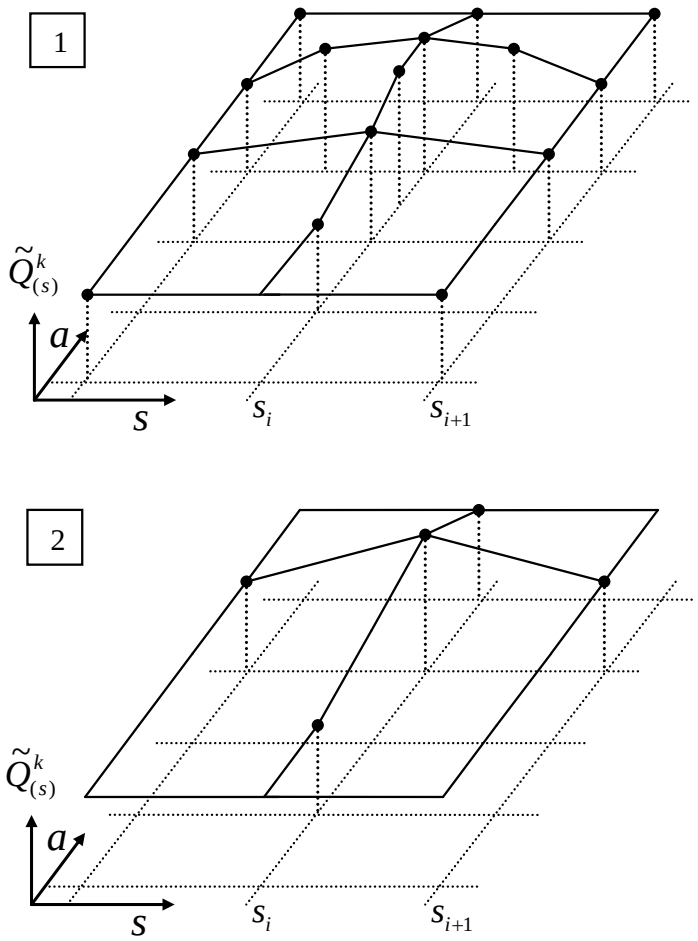

Fig. 15.

1. The incrementally constructed rule-base used as starting point
2. The final decrementally reduced rule-base providing the same (approximately) results

In the followings the already introduced cart-pole example will be used to demonstrate the presented rule base reduction strategy to gather a still functional but minimal size, truncated rule base.

3.5.1 Application example for presenting the decremental reduction

As seen at the earlier application examples, the cart-pole application is used for the demonstration of this extension also. More details on the cart-pole simulation application can found in the previous application example demonstration subchapters. This time the cart-pole demo reads the previously incrementally constructed rule base as a starting rule base. Then the rule with the lowest Q value will be removed temporarily from the rule base. With this truncated rule base a whole episode is evaluated. If the episode is considered successful with the removed rule, then the rule is removed permanently, otherwise the rule is inserted back into the rule base. This strategy is applied until every rule is checked for possible removal (see Fig. 16., Fig. 17. and Fig. 18. – it can be easily recognized on the curve where the rules were not omitted).

The starting rule base, as the result of the incremental rule base construction method as seen in the previous subchapter, consists of 182 rules. This means 182 episode runs and possibly one rule fewer in each and every episode run, which means that the computational resource needs are possibly (when the rule is sentenced to be removed) decreasing with every episode. In this demonstration, after all the rules are checked and the smallest possible rule base is found only 6 rules remain permanently in the final rule base (see Table 4.).

Two different strategies were used in this example application for deciding whether the episode was successful or not. In the first version, the rewards were strictly checked to be the same as the previous episode, which means the rewards have to be the same as they were with the incrementally constructed rule base (Fig. 16., Fig. 17., Fig. 18. and Table 4. refers to this version). In the second version matching of the rewards is not strict, the rewards do not have to match exactly, it is enough for the rewards to be positive, meaning that the episode was successful. Using this second strategy the final reduced rule base contains 5 rules (see Table 5.), whereof 4 of the rules are the same compared to the first reduced rule base (see the bold rows in Table 4. and Table 5.).

Table 4. Rules in the rule-base after performing the decremental reduction when an exact reward match is mandatory

R#	s_1	s_2	s_3	s_4	a	q
1	**P**	**Z**	**Z**	**P**	**AP10**	**1907.33**
2	**P**	**Z**	**N3**	**N**	**AN10**	**1898.73**
3	**P**	**Z**	**Z**	**N**	**AN8**	**1904.22**
4	P	Z	N3	P	AP8	1899.27
5	N	Z	N12	N	AP10	-5251.65
6	**P**	**P**	**Z**	**N**	**AN8**	**-3100.5**

Table 5. Rules in the rule-base after performing the decremental reduction when an exact reward match is not mandatory

R#	s_1	s_2	s_3	s_4	a	q
1	**P**	**Z**	**Z**	**P**	**AP10**	**1907.33**
2	**P**	**Z**	**N3**	**N**	**AN10**	**1898.73**
3	**P**	**Z**	**Z**	**N**	**AN8**	**1904.22**
4	**P**	**P**	**Z**	**N**	**AN8**	**-3100.5**
5	P	Z	P12	P	AP6	-6446.87

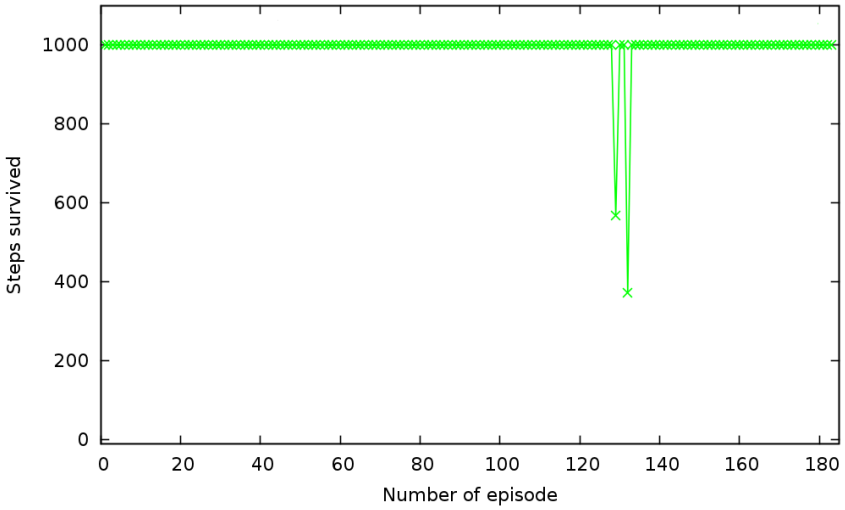

Fig. 16. Survived steps per episode while reducing the rule-base

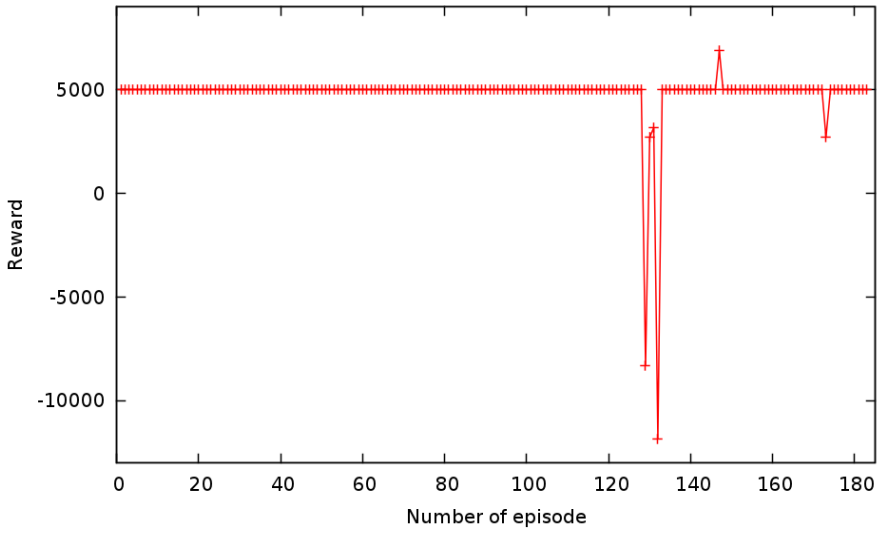

Fig. 17. The cumulative rewards per episode while reducing the rule-base

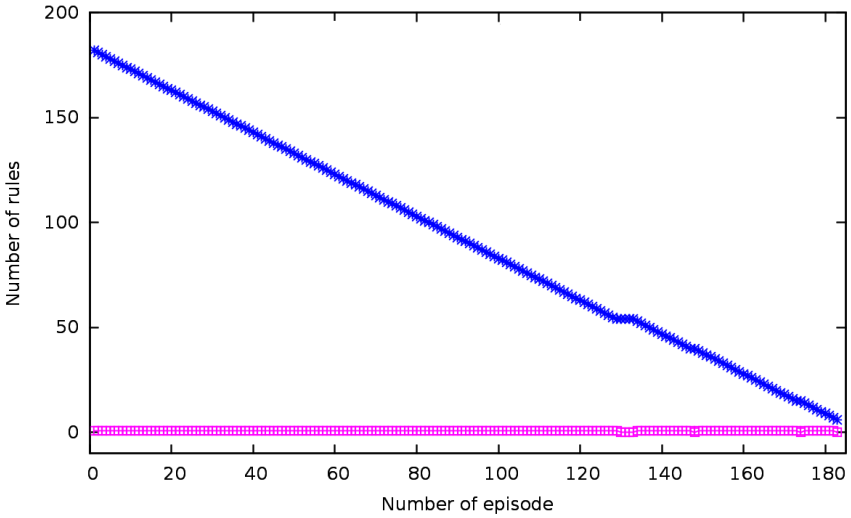

Fig. 18. The number of rules per episode – blue line
The difference in rule numbers per episode – magenta line

This huge drop in the number of rules is significant compared to the original FRIQ-learning application example, where the rule base consisted of 2268 rules. Also this means that not only the knowledge representation becomes directly human-readable, but the amount of computational resources required for processing is greatly reduced.

3.6 Summary

With the introduction of the FRIQ-learning, continuous spaces can be applied instead of the originally discrete state-action spaces. FRIQ-learning also allows the application of sparse fuzzy rule bases, which means that rules which are considered unimportant can be left out from the rule base.

A possible method is introduced for automatically creating a reduced size rule base. The targeted reduced rule base size is first achieved by the incremental creation of an intentionally sparse fuzzy rule base. The fuzzy rule base is incrementally built up from scratch and will contain only the rules which seem to be most relevant in the model, meanwhile the existing rules are also updated when required. This way the real advantages of the FIVE based FRIQ-learning method could be exploited: reducing the size of the fuzzy rule base has the benefits not only in decreasing the computing resource requirements, but having less rules (optimizable parameters), it also speeds up the convergence of the FRIQ-learning. The application example shows the results of the method, instead of 2268 rules, 182 is sufficient for achieving the same task, which is a significant difference. A smaller rule base considerably speeds up the FRIQ-learning process itself and also the whole application.

Additionally a decremental reduction strategy has been developed, which further reduces the size of the previously incrementally constructed rule base. The application example of the method clearly shows the benefit of using such a strategy, instead of the 2268 and 182 rules only 5 rules were enough in a certain case.

These novel methods described above (see [82], [84], [90], [81] for in-depth details) composed as theses are the following:

Thesis I.: The FIVE based fuzzy rule interpolation (FRI) model is suitable for describing the Q function of the Q-learning method (FRIQ-learning). I concluded that describing the Q function with the FIVE based FRI model results in a possible continuous space extension of the action-state space, where the original Q-learning algorithm was defined in discrete action-state space. Furthermore I concluded, that describing the Q function with the FIVE based FRI model allows the omission of some (redundant) states, that is, the simplification of the model. Also, the FIVE FRI method can be specifically optimized for the proposed FRIQ-learning method.

Thesis II.: In case of FRIQ-learning the cardinal rules, also the number of rules describing the action-state space (Q function) can be determined automatically (in an incremental/decremental fashion) in run-time with the appropriate evaluation of the reward function starting from an automatically generated base rule base. I concluded that the state-transition rule-base of the operating FRI-based fuzzy automaton can be extracted from the rule-based action-state space model created this way.

IV Optimization of the FIVE FRI method

Through this chapter the MATLAB implementation and structure of the FIVE FRI method [75] will be studied in more details, with the aim of identifying optimization possibilities. Optimization possibilities for the FIVE FRI were originally introduced in [88] and [91].

4.1 Analysis of the implementation of the FIVE method

The implemented functions can be divided into three logical groups: *preprocessing*, *conclusion gathering*, and *visualization*. The preprocessing group is responsible for calculating the vague environments and scaling functions. The conclusion gathering group has the task of calculating the conclusions of the rule bases based on the supplied observations. The visualization group supply methods which can be used for easy fuzzy set construction, application debugging and also for example presentations.

The preprocessing group has two functions: FIVEGScFunc is for calculating the scaling function of a defined fuzzy partition, with the following calling syntax in MATLAB:

$$SCF = FIVEGScFunc(U,PSC), \tag{21}$$

where *U* array contains the points of the universe of discourse, *PSC* is an array containing fuzzy set definitions, *SCF* is the generated function in the *U* universe.

Scaling points can be defined in a simple format. The fuzzy sets used with FIVE can be only triangle shaped sets. To define a fuzzy set, the position of the point in the universe where the triangle has full membership, and the gradient of the left and right side of the triangle should be defined (e.g.: [1, 2, 2] for the A_1 fuzzy set in Fig. 8.). If only one gradient value is supplied then that will be used for both sides of the triangle (e.g.: [3, 1/3] for the A_2 fuzzy set in Fig. 8.). Note that the gradient can be zero, meaning continuous full membership, i.e. a trapezoidal membership function can be composed from two triangle shaped membership function, where the first one has a zero gradient on the right side and the second has a zero gradient on the left side (e.g. [0.3, 4, 0] and [0.6, 0, 4] see Fig. 7.) Many fuzzy sets can be defined to form a fuzzy partition in a MATLAB matrix, but only one fuzzy set is also sufficient. For an example fuzzy partition and its generated scaling function see μ and *s* in Fig. 8.

The second function is like the previous one, FIVEGVagEnv generates the vague environment of a previously generated scaling function. Usage syntax:

$$VE = FIVEGVagEnv(U,SCF), \tag{22}$$

where *U* array contains the points of the universe, S*CF* is the generated function in the *U* universe and *VE* is the vague environment of *SCF*.

To gather a conclusion based on the current observations the FIVEVagConcl function has to be used. As parameters this functions accepts the defined universes, the calculated vague environments, rule bases, and the actual observations. Called with the following syntax:

$$C = FIVEVagConcl(U,VE,R,X), \tag{23}$$

where *U* array contains the points of the universe of discourse, *VE* array contains the points of the vague environment based on the defined scaling functions, *R* array contains the rules of the rule base, *X* is the actual observation, and *C* is the calculated consequent.

Rules in *R* are composed from the predefined linguistic terms of antecedents. The last element of the rule vector is the predefined conclusion linguistic term or a constant consequent value. All the observations have to be supplied, as many as the number of antecedents, hence the *X* vector must have the same number of elements as the number of antecedents in the *R* rule base.

The FIVEVagConcl function makes use other FIVE functions: FIVEVagDist, FIVERuleDist, FIVEValVag. These three functions are rarely used directly or outside the FIVEVagConcl function.

FIVEVagDist calculates the distance in the vague environment between the supplied observations and a specified rule. Called with the following syntax:

$$D = FIVEVagDist(U,VE,P1,P2), \tag{24}$$

where *U* array contains the points of the universe of discourse, *VE* array contains the points of the vague environment based on the defined scaling functions, *P1* and *P2* are arbitrary points (practically a rule and an observation), and *D* is the calculated distance between the latter two. The FIVEVagDist function works as follows: calculates the distances between all the universe points (elements of *U*) and the given points (*P1* and *P2*). The next step is to find the smallest distance among these previously calculated distances. Then calculate the distance in the vague environment based on this latter nearest distance. These steps are repeated for all the dimensions.

The FIVERuleDist function calculates all the distances between the supplied observations and all the rules in the rule base. This is achieved simply by calling the previous FIVEVagDist function in a loop for all the rules found in the rule base. The FIVERuleDist function has the following syntax:

$$RD = FIVERuleDist(U,VE,R,X), \tag{25}$$

where *U* array contains the points of the universe of discourse, *VE* array contains the points of the vague environment based on the defined scaling functions, *R* array contains the rules of the rule base, *X* is the actual observation, and *D* is the resulting calculated distance.

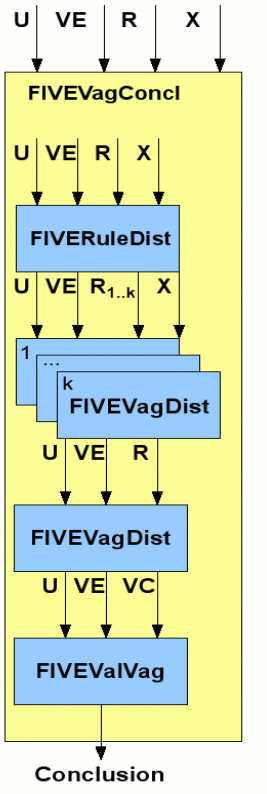

Fig. 19. Block diagram of the FIVEVagConcl function. U is the unvierse, VE is the vague environment, R is the rule base, X is the observation, $R_{1..k}$ are the rules in the rule base 1 through k, VC is the conclusion in the vague environment.

The FIVEValVag function can be used to gather the value of a given vague point in the vague environment:

$$P = FIVEValVag(U,VE,VP), \tag{26}$$

where *U* array contains the points of the universe, *VE* array contains the points of the calculated vague environment, *VP* is the scaled distance from the first element of the vague environment, and *P* is the point which is *VP* distance away from the first element in *VE* .

The membership function of a point in a vague environment can be calculated with using the FIVEVagMemb function. This latter achieves its task by calculating the membership points for the all points in the universe. The MATLAB interface syntax is the following:

$$MF = FIVEVagMemb(U,VE,P), \tag{27}$$

where *U* array contains the points of the universe, *VE* array contains the points of the calculated vague environment, *P* is the point describing the membership fuction and *MF* is the generated membership function.

In view of the functions of the conclusion gathering group the block diagram of FIVEVagConcl is presented in Fig. 19.

The first step of the FRI reasoning is to calculate the distances between the observation (*X*) and all the rules in the rule base (*R*). This is done by calling the FIVERuleDist function,

which calls FIVEVagDist for all the rules. Based on the results, FIVEVagConcl decides whether there was a direct rule hit by the observation, or if not, the Shepard interpolation should be applied. Then if the consequent has a vague environment, the FIVEVagDist is used to gather the conclusion in the vague environment (*VC*) as vague distances from the first element of the consequent unverse, otherwise (if the consequent does not have a vague environment) it is the singleton conclusion directly. Finally the conclusion gained as vague distance is transformed back to the conclusion fuzzy set applying the vague environment of the conclusion universe.

Other supplemental functions in the FIVE package are the functions intended to help the construction of fuzzy sets and partitions, and also to help debugging by visualizing the various fuzzy partitions, antecedents, conclusions. These functions are the following:

$$FIVEDrawMergedPart(U,PSCa,PSCo), \tag{28}$$

where *U* array contains the points of the universe, *PSCa* contains the points of the antecedent scaling function, *PSCo* contains the points of the observation scaling function.

Table 6. Measured run times of FIVE functions

Name of function	**Number of calls**	**Execution time**	**% of time**
FIVEVagConcl	1200000	79.970 s	6%
FIVERuleDist	1200000	561.985 s	42%
FIVEVagDist	9315028	658.764 s	50%
FIVEValVag	1200000	25.727 s	2%

This function visualizes the fuzzy partitions based on the antecedent and observation scaling functions, also the scaling functions and the merged fuzzy partition reconstructed based on the merged scaling functions.

A special case of the previous function is FIVEDrawPart, which visualizes the whole fuzzy partition for an antecedent or consequent, also its scaling function and the fuzzy partition reconstructed based on the calculated scaling function, the syntax is similar than for the previous FIVE function:

$$FIVEDrawPart(U,PSC), \tag{29}$$

where *U* array contains the points of the universe, PSC contains the points of a scaling function. See Fig. 20. for an example result of a FIVEDrawPart call. The fuzzy set points used for generating the scaling function in Fig. 20. are: [0, 3; 0.5, 1; 1, 3].

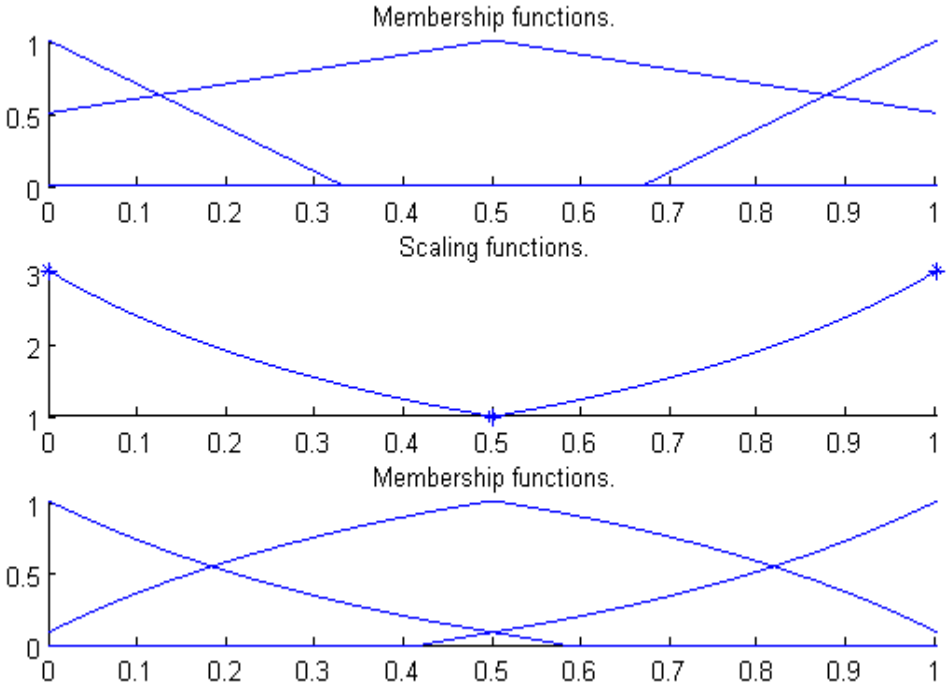

Fig. 20. FIVEDrawPart example output
From top to bottom: fuzzy partition, scaling function,
the reconstructed fuzzy partition based on the scaling function.

For calculating and visualizing all the antecedent or consequent and observation partitions on one figure the FIVEDrawAnt or FIVEDrawConc function can be used. These functions also use the above presented FIVEDrawMergedPart function in the background. These have the syntax:

$$FIVEDrawAnt(output,params), \tag{30}$$

$$FIVEDrawConc(output,params), \tag{31}$$

where *output* is a set of FIVE outputs and *params* define the size of the used universe and the power parameter for Shepard interpolation.

In ordinary applications mostly only the two generator functions ((21) and (22)) and the conclusion gathering function (23) are being used directly. These three functions are dramatically simlifying the usage of the FIVE FRI for the potential application programmer.

The resource needs of the FIVEVagConcl function only depend on the used vague environments and rule bases, which are both static while running the application.

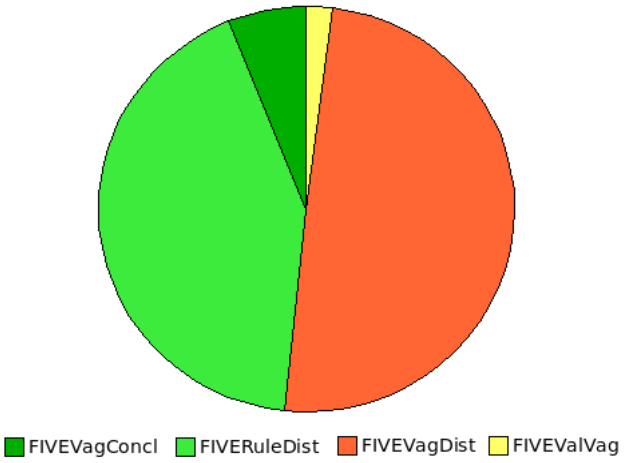

Fig. 21. The execution times of the FIVE modules when using FIVEValConcl.

Profiling the FIVE implementation with a modified sample application found originally in [83] and [92] shows that the FIVEVagDist function is the most frequently used function while gathering conclusions with FIVE, and also the most resource hungry function (see Table 6. and Fig. 21.). Another function with high computing power usage is the FIVERuleDist. For optimization purposes these two functions should be considered in the first place. Table 6. and Fig. 21. shows the run times and the ratios of run times of the functions themselves excluding the time spent in other nested FIVE functions. It can be clearly seen that FIVEVagDist is used nearly eight times more than the other FIVE functions. Further analysing this function for possible optimizations can yield in successful results.

4.2 Performance optimization of the FIVE FRI method

The analysis of the implementation of the FIVE method shows that the most frequently used function is the FIVEVagDist function. Profiling the code of a simple application example which uses FIVE also proves that the most frequently called and the most resource hungry function is FIVEVagDist (see e.g. benchmark results on Table 6. The FIVEVagDist function is responsible for calculating the scaled distance between two points in the vague environment. The first step in the function is to align the current point supplied as input to the points of a pre-defined universe.

The original FIVE implementation is capable of handling arbitrary universes (see Fig. 22./A). Experience shows that in most cases FIVE based applications (e.g. in [38], [83] and [92]) use a fixed resolution universe description (see Fig. 22./B) with points stored in an array in ascending order. Therefore it seems to be possible to simplify the function by restricting the universe description from the arbitrary to the fixed resolution without having a high impact on its practical applicability.

In the original implementation [36] for aligning (see Fig. 23.) the input points (one point for a rule in the rule base and one point for the current observation) to the pre-calculated positions of the look-up table, the method first calculates all the possible distances. Then it searches through for the nearest pre-defined universe positions (i) for each input points: .

$$i_k = minindex\left(\left|U_{k_{1..n}} - P_k\right|\right), \tag{32}$$

where P_k is the input point, $U_{k1..n}$ is the n element vector of the pre-calculated positions in the kth dimension, *minindex* is the function calculating the index of the minimal element in an array and i_k is the index of the alignment position of P_k.

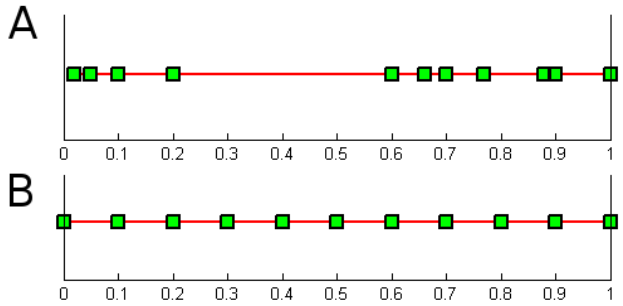

Fig. 22. Universe points of a dimension.
A: arbitrary universe,
B: universe with a fixed resolution

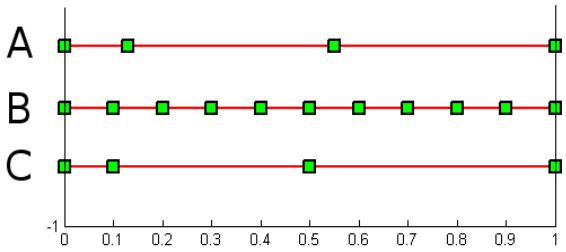

Fig. 23. Universe points of a dimension.
A: input points,
B: universe with a fixed resolution,
C: input points aligned to the points in the universe

(32) is unfolded and executed as follows:

variables:

i: float, initally 0

x: float array, initally filled with 0 values

U: float array, contains the points of the universe

minindex: integer, initially 1

minelement: integer

for every i, i ∈ U:

x(i) := abs(Uk - Pk)

i := i+1

```
minelement := x(1)
for every i,  1 ≤ i ≤ sizeof(U):
  if x(i) < minelement then
       minelement := x(i)
       minindex := i
```

output: minindex

As seen above the substraction must be executed for all the points in the universe, and after that the smallest element should be found to gather the index of the nearest pre-calculated position in the array U.

The number of calculations could be reduced by fixed resolution universe description (see Fig. 22. 3/B), where the pre-defined universe points stored in an array are in ascending order. This case it is possible to directly calculate the alignment position in one step:

$$i_k = round\left(n_k \cdot \left(P_k - U_k\right) / \left(U_{k_n} - U_{k_1}\right)\right), \tag{33}$$

where n_k is the number of the pre-calculated positions, U_{k1} is the first, U_{kn} is the last pre-calculated position in the kth dimension and *round()* is a function which round towards the nearest integer. In case P_k is exactly between two neighbouring points the original FIVE method uses the smaller value, so standard rounding will not give the required results. To preserve the original properties of the FIVEVagDist function, the calculated index is corrected when needed by checking the distance of the next neighbouring point. Hence two substractions and comparisons are made instead of the number of the elements in the U array.

The suggested modification makes the actual input point alignment process time complexity independent from the size of the look-up table representing the universe of discourse. Therefore the performance gain depends on the size of the applied universe, the larger the universe description table, the larger the performance gain becomes.

The uniform time complexity of the original *FIVEVagDist* function can be approximated as:

$$m \cdot \left(9 \cdot n + 16\right), \tag{34}$$

where n is the average size of the look-up tables representing the m dimensional antecedent universes.

The uniform time complexity of the optimized *FIVEVagDist* function is independent from the number of antecedent dimensions, and in usual cases (n>>1) much better than the original version:

$$m \cdot 57. \tag{35}$$

The space complexity is the same for both versions of the function, as it is dominated by the size of the pre-calculated look-up table:

$$m \cdot n. \tag{36}$$

In Landau notation the time complexity of the original function is $O(mn)$, while in case of the modified function it is only $O(m)$. The space complexity is $O(mn)$ in both cases.

In typically cases, where the universes consist of about a thousand reference points (see e.g. [38], [83] and [92]), the performance gain in time complexity is significant. (See the application example section for measurement results in case of a benchmark problem.)

4.2.1 Further possible optimizations

Analysis of the implemented FIVERuleDist function yields to further enhance of the overall performance. By default, FIVE stores antecedents and consequents in the same array (the rule base itself), which is convenient from the viewpoint of the programmer. Convenient for the programmer, but it means consuming extra resources when splitting the array in the case when only the antecedents are required. This is the case with the FIVERuleDist function. In the original implementation FIVERuleDist truncates the corresponding arrays of the universes, the vague environments and the rule bases to contain only the data of the antecedents. These truncations are performed for every rule in the rule base. The rule bases, universes and vague environments do not change while the application is running, so it is possible to perform these truncations in advance and cache the result (antecedent caching). Then the truncated antecedent arrays have to be passed as parameters to the corresponding FIVE functions. This latter means a change in the application programming interface, so existing applications have to be modified slightly to follow this new syntax. Creating the mentioned truncated array (antecedent cache) will consume more memory of course, but this extra memory usage is insignificant compared to the speed gain.

The original FIVERuleDist function has the time complexity of

$$m \cdot (4 \cdot n + 8) \tag{37}$$

and space complexity of

$$m + 4 \cdot n - 3, \tag{38}$$

while that optimized version has time complexity

$$m \cdot (4 \cdot n + 2) \tag{39}$$

and space complexity of

$$m + 2 \cdot n + 2 \tag{40}$$

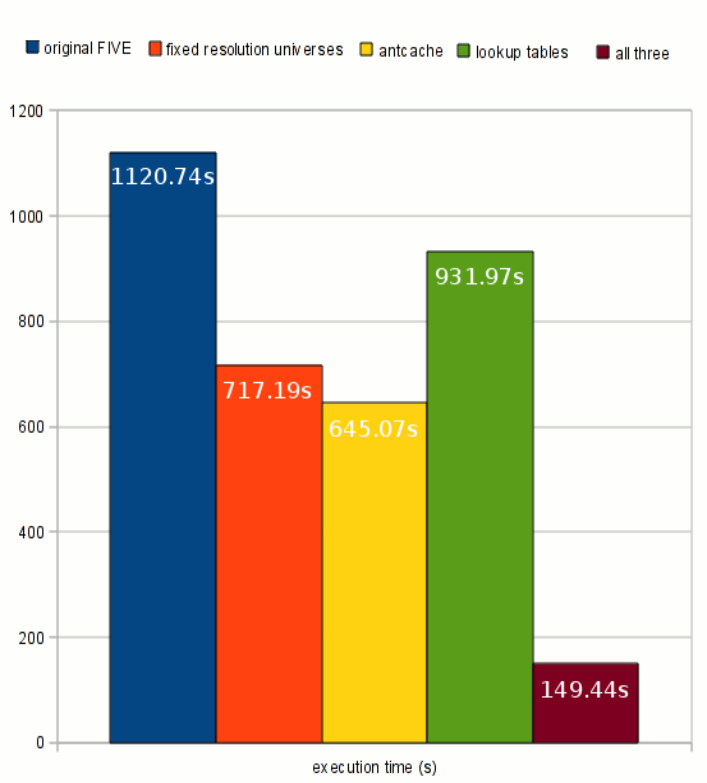

Fig. 24. Execution times of the vehicle navigation application benchmark, from left to right: unmodified FIVE, FIVE with fixed resolution universes, FIVE with antecedent caching, FIVE with conclusion lookup tables, and FIVE with all the modifications

In Landau notation, both versions have the time complexity of *O(mn)*. The high difference in the benchmark execution time (FIVE with antecedent truncation in Table 7. and Fig. 24.) is mainly inherited from the high expense of the matrix truncation function. In the applied uniform time complexity calculation, it is hidden as a single calculation step.

Table 7. Measured run times [sec] of the main FIVE functions

Name of function	**Execution time**	**Speed-up**
Unmodified FIVE	1120.74 s	1 x
FIVE with fixed resolution universes	717.19 s	≈1.6x
FIVE with antecedent caching	645.07 s	≈1.7x
FIVE with conclusion lookup tables	931.97 s	≈1.2x
FIVE with all modifications	149.44 s	≈7.5x

Another efficient method of achieving performance improvement in small embedded applications is the pre-calculation of the consequents for all the possible lookup table positions of the antecedent universe of discourse. If the lookup table resolutions of the universes are small, the rule bases with more than one antecedent can also use this approach. The space complexity should be investigated carefully because the number of required pre-calculated elements grows exponentially with the number of antecedent dimensions: $O(n^m)$. For example, with a typical resolution of a 1000-element lookup table and roughly 100,000 repeated evaluations, this simple method can yield a considerable

speed gain (the costly FIVE calculations only have to be performed 1000 times instead of roughly 100,000 times during the evaluations).

For swift application startup, caching the generated universes with their calculated vague environments and rule bases on storage is also a plausible solution.

In the age of multi-core CPU systems, it is straightforward to study parallelization possibilities in FIVE. There are two locations in the implementation where the advantages of parallel execution can be exploited. One is in the FIVERuleDist function (see Fig. 19.). The iterations of the main *for* loop of the function are data independent, because in each iteration the weight of a different rule (independent data sets) is calculated even if the same code is executed.

The next candidate for the parallelization is the FIVEVagDist function (see Fig. 19.). In this case, the observation and the rule antecedent vague distances can be calculated independently for each antecedent dimension.

For embedded real-time applications, the additional computational cost of parallelization, data distribution, and data collection also have to be taken into consideration, as the level of available parallelization (the number of the fuzzy rules in case of the FIVERuleDist and the number of antecedent dimensions in case of the FIVEVagDist) are not so high. E.g. MATLAB provides a simple method for the parallel execution of for loops (defining a CPU pool then using the *parfor* keyword instead of the regular *for* keyword). Unfortunately this method is not suitable to be used in FIVE, because the actual code in the loops are very short, and the overhead of *parfor* is much higher than the execution time of the useful code, so practically using *parfor* in this case greatly slows down the execution.

4.2.2 Application example for the evaluation of the optimization

For the first benchmark of the performance, the optimization of a simple vehicle navigation demo application is selected (see details in [83]). The goal of the application is to control an unmanned robot capable of room surveillance, making it cycle through given waypoints within a room with walls and moving obstacles to avoid. When the path of the robot seems to be blocked by an obstacle, the robot is capable of turning around and heading in the opposite direction. This application uses the original FIVE method with four simple rule bases. The application is extended to use the optimized FIVE functions. The universes have fixed resolutions of 1001 elements. One of the rule bases has only one antecedent; another one has only two. Hence, for them, lookup tables for the conclusions can be generated: the consequents must be calculated for every possible point in the universe of the antecedent. This way the lookup table will have the same number of elements as the universe array of the antecedent. Another rule base (expl_newdir) has two antecedent parts, but the first antecedent can only have two possible values (0,1), so the size of the lookup tables will only be the double of the size of the second antecedent universe, not the product of the sizes of the two antecedent universes.

The other two rule bases have 3 and 10 antecedent dimensions. As a benchmark, the main loop of the vehicle navigation application example is analyzed. The resulting execution times of 10^8 times the main vehicle navigation cycle has run are summarized in Table 7. and Fig. 24. Also the modified vehicle navigation application is available at [76].

4.3 Optimizing the FIVE FRI method for FRIQ-learning

A special application example of the optimized FIVE FRI is the FRI based Q-learning (FRIQ-learning) introduced in [82], [81] and [84]. The FRIQ-learning is an extension of the traditional Fuzzy Q-learning (FQ-learning) method with the capability of handling sparse fuzzy rule bases. The introduction of FIVE FRI in FQ-learning allows the omission of deducible fuzzy rules from the rule base representing the action-state values function. This reduction also adds the potential for applying FRIQ-learning in higher state dimensions than the original FQ-learning thanks to the sparse fuzzy rule base model of the action-state space.

The efficiency of the FRI method has a high impact on FRIQ-learning, as the methodology (similarly to the Q-learning) requires an extremely high number of repeatedly executed action-state values function model updates (FRI reasoning) in runtime. Because of the similar repeatedly executed FRI operations the optimization of the applied FIVE FRI suggested in this work could improve the FRIQ-learning to a great extent.

From the previously introduced FIVE optimization methods, first of all, the fixed resolution universe description can be applied to speed up FRIQ-learning.

Because of the high dimensionality of the action-state space, a conclusion lookup table is practically impossible to implement in this case.

Beyond the FIVE optimization methods introduced previously, because of the specialties of the FRIQ-learning calculations, another optimization method is also approaching. In FRIQ-learning in the action-state value model updates and action selection, the values of each possible action in a given state have to be calculated.

According to [82], the action-state values function FRI model has the following rule form (k^{th} rule):

$$\textbf{If } s_1 = S_{k,1} \textbf{ And } \ldots \textbf{ And } s_m = S_{k,m} \textbf{ And } a = A_u \textbf{ Then } \tilde{Q}(s_1,..,s_m,a) = Q_k \tag{41}$$

where $S_{k,i}$ is the label of the i^{th} membership function of the m dimensional state space, A_u is the label of the u^{th} membership function of the one dimensional action space, Q_k is the singleton conclusion, and $\tilde{Q}(s_1,..,s_m,a)$ is the approximated continuous action-state values function.

According to (41), the FRI action-state values function model has a multidimensional state and a single action dimension (see [82], [81] and [84]). Caching the calculated vague distances of the observed state and the rule antecedents related to the state dimensions of the action-state values FRI rule base can dramatically speed up the calculations. In each action-state value model update and action selection cycle, the vague distances of the actual state and the rule antecedents related to the state dimensions have to be calculated only once.

The performance gain that could be achieved by this modification depends on the number of the antecedent fuzzy term sets in the action universe. The higher the number of the action term sets, the better the performance gain becomes.

For execution time benchmarks, the application example presented for FRIQ-learning earlier is used.

Table 8. Run times [msec] of a best-action-selection iteration with the various optimizations

Version	Time	Speed-up
Original	≈115 ms	1x
States calculated only once	≈25ms	≈9.5x
States calculated only once and using fixed resolution alignment optimization	≈9ms	≈12.5x

Table 9. Run times [sec] for the first five episodes of a simulation run with the various optimizations

Version	Time	Speed-up
Original	≈76.1s	1x
Fixed resolution universes	≈24.6s	≈3.1x
States calculated only once	≈23.5s	≈3.2x
States calculated only once and using fixed resolution alignment optimization	≈7.9s	≈9.6x

This application has been extended with the proof-of-concept implementation of the suggested FIVE FRI optimized FRIQ-learning.

The execution times of the benchmark show a significant performance gain over the original version. Table 8. summarizes the runtimes of the first best-action-selection iteration step at the start of the application.

Table 9. summarizes the running time for the first five episodes of a simulation run from the whole application with the various optimizations disabled or enabled.

The efficiency of this optimization depends on the number of the state dimensions. In this application example, the action-state values function FRI model (41) has a four-dimensional state and a single-action universe with a resolution of 21 distinguishable actions. Applications with different number of state dimensions possibly yield different results, lower performance gain for less state dimensions and higher performance gain for applications with more state dimensions thatn the application example examined here.

In conclusion of the benchmarks, it can be stated, that by the optimized FIVE FRI based FRIQ-learning can achieve a tenfold increase in speed over the original FIVE FRIQ-learning implementation.

4.4 Summary

The presented optimizations in the FIVE FRI method yield considerable speed improvements. This performance gain could be a relevant improvement for embedded FRI real-time applications, for which rapid reasoning is the essential question of the FRI applicability. Practically there is no drawback in applying these modifications, though it reduces functionality this only affects rare special cases, which are not so common. The original FIVE source code can be simply replaced with the modified code in existing FIVE based applications, rendering the application faster without significant modifications in the application itself. In case only using the fixed resolution universe based modification, the modified FIVE source code presented here can be used as a drop in replacement to the original FIVE code to speed up an already existing application.

Data structural simplification and run-time caching methods can achieve a relevant speed-up in computational time in FRI applications. The data structural simplification speeds up lookup table data fetching while caching techniques can speed up repetitive calculations, such as rule evaluations in a single reasoning cycle (see speed-up results in Table 7.). The gain of caching can be more relevant if the FRI reasoning core is embedded into repetitive calculations, as in FRIQ-learning (see speed-up results in Table 8. and Table 9.).

Another important conclusion is that the relatively small impact of conclusion lookup tables on the FRI reasoning speed performance (Table 7. and Fig. 24.) The speed performance gain that can be achieved is negligible against the exponential space complexity loss of the conclusion lookup tables. (Antecedent universes lookup tables has a space complexity of $O(mn)$, while that of conclusion lookup tables have $O(n^m)$.) The main reason for the low reasoning speed gain of the conclusion lookup tables is inherited from the effective FRI sparse-fuzzy-rule-based knowledge representation format. This is a special feature of the FRI knowledge representation, i.e., the size of the rule base is relatively small, containing the relevant rules only. For classical fuzzy reasoning methods in which the exponentially-large, complete rule base is required, conclusion lookup tables can achieve a better speed-up factor.

The results of the performance measurements of the suggested optimization methods in a real application example show that the gain is significant for an ordinary application, in this very case 7-8 times faster than the original unmodified algorithm. In more complex FIVE based applications or applications using large universes can benefit much more from this optimization.

As a conclusion, the FIVE FRI method can be specifically optimized for the proposed FRIQ-learning method, which, along with other applicable general optimizations, significantly improves the the performance of the FRIQ-learning method (see [91], [88], [83]).

The results introducted in this chapter are supporting the statement of Thesis I.

V Fuzzy automaton for describing ethological functions

This chapter presents an ethologically inspired model realized with a fuzzy rule interpolation fuzzy automaton (introduced in [77], [78], [79] and [80]). First, the ethological inspiraton for human-robot interaction is presented, after that a brief overview on behaviour-based control, then some words on fuzzy automata can be found. After introducing the necessary prerequisites, the ethological test procedure, the suggested structure of the model and the framework for operating the model is described in details. Some samples from the rule bases of the fuzzy automaton along with example simulation executions are described. Finally the various software and hardware interfaces developed for the proposed model are presented [87].

5.1 Ethologically inspired Human-Robot Interaction

In recent years there has been an increased interest in the development of Human-Robot Interaction (HRI). Researchers have assumed that HRI could be enhanced if these intelligent systems were able to express some pattern of sociocognitive and socioemotional behaviour (e.g. [9]). Such approach needed an interaction among various scientific disciplines including psychology, cognitive science, social sciences, artificial intelligence, computer science and robotics. The main goal has been to find ways in which humans can interact with these systems in a 'natural' way. Recently HRI has become very user oriented, that is, the performance of the robot is evaluated from the user's perspective. This view also reinforces arguments that robots do not only need to display certain emotional and cognitive skills but also showing features of individuality. Generally however, most socially interactive robots are not able to support long-term interaction with humans, and the interest shown toward them wears out rapidly.

The design of socially interactive robots has faced many challenges. Despite major advances there are still many obstacles to be solved in order to achieve a natural-like interaction between robots and humans. The 'uncanny valley' effect: Mori [47] assumed that the increasing similarity of robots to humans will actually increase the chances that humans refuse interaction (will be frightened from) very human-like agents. Although many take this effect for granted only little actual research was devoted to this issue. Many argue that once an agent passes certain level of similarity, as it is the case in the most recent visual characters in computer graphics, people will treat them just as people [51]. However, in the case of 3D robots, the answer is presently less clear, as up do date technology is very crude in reproducing natural-like behaviour, emotions and verbal interaction. Thus for robotics the uncanny valley effect will present a continuing challenge in the near future.

In spite of the huge advances in robotics current socially interactive systems fail both with regard to motor and cognitive capacities, and in most cases can interact only in a very limited way with the human partner. This is a major discrepancy that is not easy to solve because there is a big gap between presently available technologies (hardware and software) and the desire for achieving human-like cognitive and motor capacities. As a consequence

recent socially interactive robots have only a restricted appeal to humans, and after losing the effect of novelty the interactions break down fast.

The planning and construction of biologically or psychologically inspired robots depends crucially of the current understanding of human motor and mental processes. However, these are one of the most complex phenomena of life. Therfore it is certainly possible that human mental models of abilities like ‘intention’, ‘human memory’ etc., which serve at present as the underlying concepts for control socially interactive robots, will be proved to be faulty.

Because of the goal of mimicking a human, socially interactive robots do not utilize more general human abilities that have evolved as general skills for social interaction. Further, the lack of evolutionary approach in conceptualizing the design of such robots hinders further development, and reinforces that the only goal in robotics should be the produce ‘as human-like as possible’ agents.

In order to overcome some of the challenges presented above ethologically inspired HRI models can be applied. The concept of ethologically inspired HRI models allows the study of individual interactions between animals and animals and humans. This way of handling Human-Robot Interaction is based on the concept that the robot acts like an animal companion to human. According to this paradigm the robot should not be molded to mimic the human being, and form human-to-human like communication, but to follow the existing biological examples and form inter-species interaction. The 20.000 year old human-dog relationship [46] is a good example for this paradigm of the HRI, as interaction of different species. One good reason of this approach in HRI is the lack of the ‘uncanny valley’ effect [47], i.e. increasing similarity of robots to humans will actually increase the chances that humans refuse interaction (will be frightened).

Such robots do not have to rely on the exact copy of human social behaviour (including language, etc.) but should be able to produce social behaviours that provide a minimal set of actions on which human-robot cooperation can be achieved. Such basic models of robots could be improved with time making the HRI interaction more complex.

In ethological modeling, mass of expert knowledge exists in the form of expert’s rules. Most of them are descriptive verbal ethological models. The knowledge representation of an expert’s verbal rules can be very simply translated to the structure of fuzzy rules, transforming the initially verbal ethological models to a fuzzy model. On the other hand, in case of the descriptive verbal ethological models, the ‘completeness’ of the rule-base is not required (thanks to the descriptive manner of the model), which makes implementation difficulties in classical fuzzy rule based systems, as stated earlier.

For ethologically inspired HRI model implementation a fuzzy model structure built upon the previously mentioned framework of low computational demand Fuzzy Rule Interpolation (FRI) method FIVE [36] and fuzzy automaton is suggested. The application of FRI methods fits well the conceptually sparse rule-based (‘incomplete’) structure of the existing descriptive verbal ethological models.

The main benefit of the FRI method adaptation in ethological model implementation is the fact, that it has a simple rule-based knowledge representation format. Because of this, even after numerical optimization of the model, the rules remain ‘human readable’, and

helps the formal validation of the model by the ethological experts. On the other side due to the FRI base, the model has still low computational demand and fits directly the requirements of the embedded implementations.

For implementing ethologically inspired HRI models the classical behaviour-based control structure is suggested, described in the followings.

5.2 Behaviour-based control

The main building blocks of Behaviour-based Control (BBC, a comprehensive overview can be found in [50]) are the behaviour components themselves. The behaviour components can be copies of typical human or animal behaviors, or can be artificially created behaviours. The actual behaviour response of the system can be formed as one of the existing behaviour component, which gives the best match to the actual situation, or a fusion of the behaviour components based on their suitability for the actual situation. Encoding the behaviour components can be realized with e.g. simple reflexive agents, which assign an output response to each input situation.

In case when more than one behaviour components are simultaneously competing for the same ‘actuator’ the aggregation or selection of the behaviour components is necessary. Handling multiple behaviour components in a BBC system can be done in two ways. The first is the competitive way, when the behaviour components are assigned with priorities, and the behaviour component with the highest priority takes precedence, while the behaviours with lower priorities are simply ignored. The second is the cooperative way when the outputs are fusioned based on various criteria.

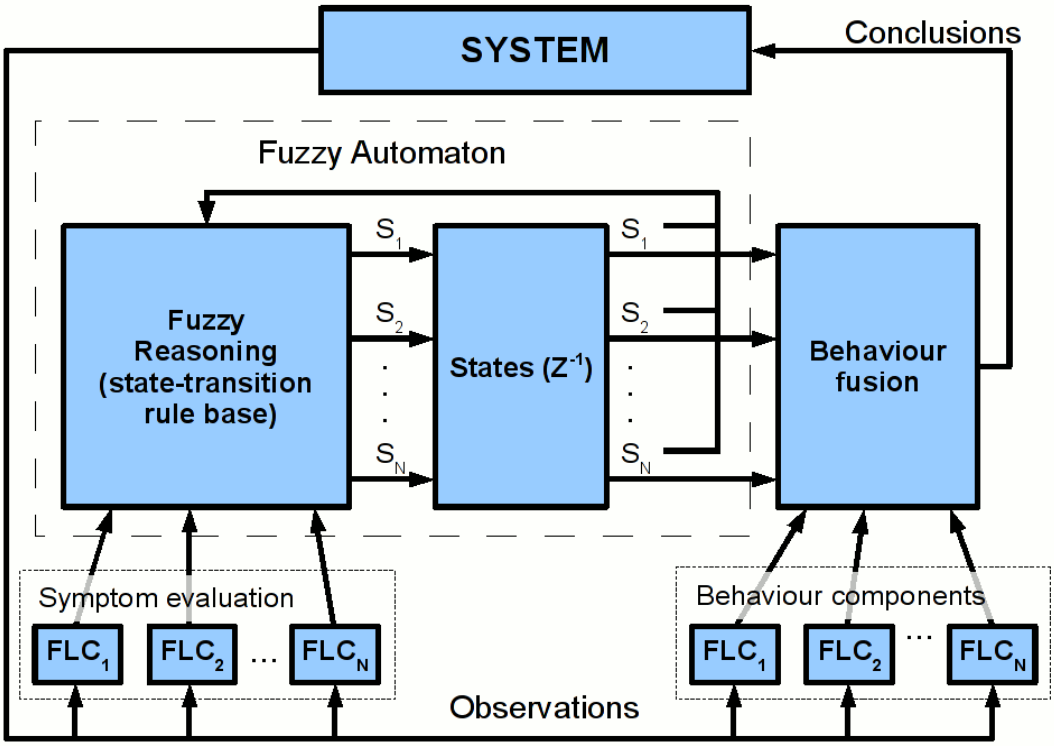

Fig. 25. Diagram of the fuzzy automaton

This structure has two main tasks. The first is a decision of which behaviour is needed in an actual situation, and the levels of their necessities in case of behaviour fusion. The second is the way of the behaviour fusion. The first task can be viewed as an actual system state approximation, where the actual system state is the set of the necessities of the known behaviours needed for handling the actual situation. The second is the fusion of the known behaviours based on these necessities. In case of the suggested fuzzy behaviour based control structures both tasks are solved by FRI systems. If the behaviours are also implemented on FRI models, the behaviours together with the behaviour fusion modules form a hierarchical FRI system.

The application of FRI methods in direct fuzzy logic control systems gives a simplified way for constructing the fuzzy rule base. The rule base of a fuzzy interpolation-based model, is not necessarily complete, it could contain the most significant fuzzy rules only without risking the chance of having no conclusion for some of the observations. The model always gives a usable conclusion even if there are no rules defined for the actual observations. In other words, during the construction of the fuzzy model, it is enough to concentrate on the main actions (rules could be deduced from the others could be intentionally left out from the model, radically simplifying the rule base creation, saving time consuming work.).

For demonstrating the main benefits of the FRI model in behaviour-based control, here the focus is only on the many cases most heuristic part of the structure, on the behaviour coordination. The task of behaviour coordination is to determine the necessities of the known behaviours needed for handling the actual situation.

In the suggested behaviour-based control structure for achieving the decision related to the relevance of the behaviour components this work adapts the concept of the finite state fuzzy automaton [37] (see Fig. 25. and the next chapter for more details), where the state of the finite state fuzzy automaton is the set of the suitabilities of the component behaviours. This solution is based on the heuristic, that the necessities of the known behaviours for handling a given situation can be approximated by their suitability. And the suitability of a given behaviour in an actual situation can be approximated by the similarity of the situation and the prerequisites of the behaviour. (Where the prerequisites of the behaviour is the description of the situations where the behaviour is applicable). This case instead of determining the necessities of the known behaviours, the similarities of the actual situation to the prerequisites of all the known behaviours can be approximated.

The system consists (see Fig. 25.) of not only the fuzzy automaton but the behaviour fusion component and various component behaviours implemented on fuzzy logic controllers (FLC). The state variables characterize the relevance of the component behaviours. The state-transition rule base of the automaton applies fuzzy rule interpolation for state-transition evaluation. The previous states are fed back to the automaton and the conclusion given by the automaton is used as weights in the behaviour fusion component for determining the final conclusion of the BBC. The conclusion of the fuzzy automaton will be the new system state for the next step of the behaviour fusion. The behaviour fusion component can be also implemented by fuzzy reasoning (e.g. using fuzzy rule interpolation), or simply as a weighted sum. The symptom evaluation (from the terminology

of fault classification) components provide a kind of preprocessing for the automaton based on the gathered observations. These components also can employ FRI techniques.

Therefore the first step of the system state approximation is determining the similarities of the actual situation to the prerequisites of all the known behaviours. The task of symptom evaluation is basically a series of similarity checking between an actual symptom (observations of the actual situation) and a series of known symptoms (the prerequisites – symptom patterns – of the behaviour components). These symptom patterns are characterizing the systems states where the corresponding behaviours are valid. Based on these patterns, the evaluation of the actual symptom is done by calculating the similarity values of the actual symptom (representing the actual situation) to all the known symptoms patterns (the prerequisites of the known behaviours). There are many existing methods for fuzzy logic symptom evaluation. For example fuzzy classification methods e.g. the Fuzzy c-Means fuzzy clustering algorithm [7] can be adopted, where the known symptoms patterns are the cluster centers, and the similarities of the actual symptom to them can be fetched from the fuzzy partition matrix. On the other hand, having a simple situation, the fuzzy logic symptom evaluation could be an FRI model too.

One of the main difficulties of the system state approximation is the fact, that most cases the symptoms of the prerequisites of the known behaviours are strongly dependent on the actual behaviour of the system. Each of the behaviours has its own symptom structure. In other words, for the proper system state approximation the approximated system state is needed itself. A very simple way of solving this difficulty is the adaptation of fuzzy automaton. This case the state vector of the automaton is the approximated system state, and the state-transitions are driven by fuzzy reasoning ('Fuzzy Reasoning (state-transition rule base)' on Fig. 25.), as a decision based on the previous actual state (the previous iteration step of the approximation) and the results of the symptom evaluation.

5.3 Fuzzy automaton (fuzzy state machine)

As mentioned previously, the structure of the proposed model follows the behavior-based control concept [50], i.e. the actual behavior of the system is formed as a fusion of the known component behaviors appeared to be the most appropriate in the actual situation. For behavior fusion in the applied model the concept of the 'Fuzzy Automaton' is adapted. Numerous versions and understanding of the fuzzy automaton can be found in the literature (a good overview can be found in [10]). The most common definition of Fuzzy Finite-state Automaton (FFA, summarized in [10]) is defined by a tuple (according to [5], [10] and [49]):

$$\widetilde{F} = \left(Q, \Sigma, \delta, R, Z, \omega\right), \tag{42}$$

where Q is a finite set of states, $Q=\{q_1,q_2,...,q_n\}$, Σ is a finite set of input symbols, $\Sigma=\{a_1,a_2,...,a_m\}$, $R \in Q$ is the (possibly fuzzy) start state of $\widetilde{F}$, Z is a finite set of output symbols, $Z=\{b_1,b_2,...,b_n\}$, $\delta : Q \times \Sigma \times Q \to [0,1]$ is the fuzzy transition function which is used to map a state (current state) into another state (next state) upon an input symbol, attributing a

value in the fuzzy interval [0,1] to the next state, and is $\omega: Q \to Z$ the output function which is used to map a (fuzzy) state to the output.

Extending the concept of FFA from finite set of input symbols to finite dimensional input values turns to the following:

$$\tilde{F} = (S, X, \delta, P, Y, \omega), \tag{43}$$

where S is a finite set of fuzzy states, $S = \{\mu_{s1}, \mu_{s2}, ..., \mu_{sn}\}$, X is a finite dimensional input vector, $X = \{x_1, x_2, ..., x_m\}$, $P \in S$ is the fuzzy start state of $\tilde{F}$, Y is a finite dimensional output vector, $Y = \{y_1, y_2, ..., y_l\}$, $\delta: S \times X \to S$ is the fuzzy state-transition function which is used to map the current fuzzy state into the next fuzzy state upon an input value, and $\omega: S \times X \to Y$ is the output function which is used to map the fuzzy state and input to the output value. See e.g. on Fig. 26.

In case of fuzzy rule based representation of the state-transition function $\delta: S \times X \to S$, the rules have $n+m$ dimensional antecedent space, and n dimensional consequent space. Applying classical fuzzy reasoning methods, the complete state-transition rule base size can be approximated by the following formula:

$$|R| = n \cdot i^n \cdot j^m, \tag{44}$$

where n is the length of the fuzzy state vector S, m is the input dimension, i is the number of the term sets in each dimensions of the state vector, and j is the number of the term sets in each dimensions of the input vector.

According to (44) the state-transition rule-base size is exponential with the length of the fuzzy state vector and the number of the input dimensions. Applying FRI methods for the state-transition function fuzzy model can dramatically reduce the rule base size. See e.g. [32], where the originally exponential sized state-transition rule base of a simple heuristical model turned to be polynomial thanks to the FRI.

In case of direct application of the suggested FRI based Fuzzy Automaton for behavior-based control structures, the output function $\omega: S \times X \to Y$ can be decomposed to parallel component behaviors and an independent behavior fusion. In this case the structure of the above introduced fuzzy automaton can turn to a very similar form as it is expected in behavior-based control (see Fig. 27 and Fig. 25.). Some more details of the model implementation can be found in [77], [78] and [80].

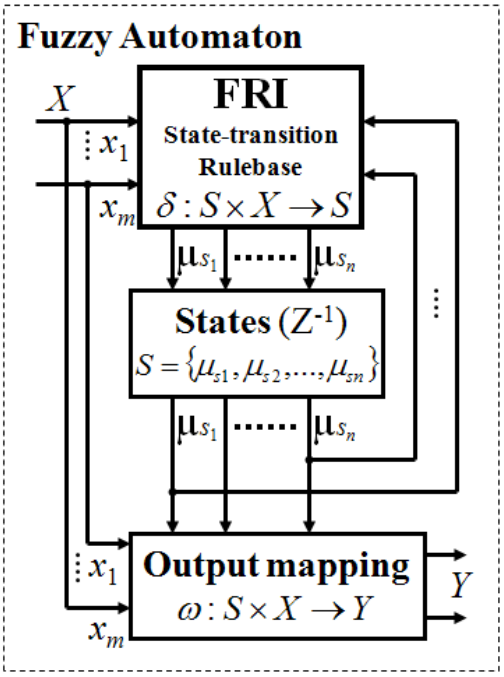

Fig. 26. FRI based Fuzzy Automaton.

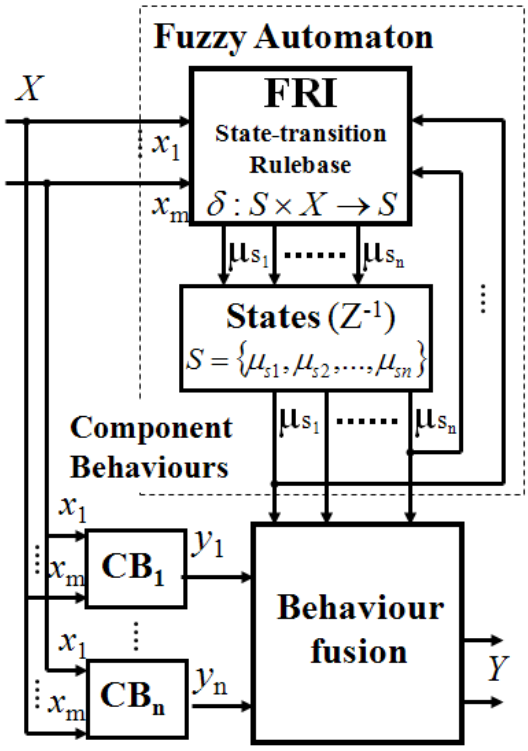

Fig. 27. The suggested FRI behaviour-based structure

5.4 Simulation of the ‘Strange Situation Test’ (SST)

The ethological test procedure presented here has been developed for studying the affiliative relationship between a dog and its owner. The procedure is made up from seven episodes, each lasting 2 minutes, where the dog is in different situations: first with the owner, then with a stranger, or alone (according to a pre-defined protocol). Through the test, the dog’s behavior is evaluated mainly focusing on dogs’ responses related to their proximity seeking with the owner [63].

In the first episode of the test, the dog and the owner are in the test room. First the owner is passive, and then he/she stimulates playing. The second episode is where the stranger comes into the room and starts to stimulate play with the dog. Then the owner leaves the room, so in the third episode the dog is separated from the owner; the stranger tries to play with the dog then sits for a while and offers petting. The owner comes back in the beginning of the fourth episode, this is the first reunion. Meanwhile the stranger leaves the room and the dog is with the owner for two minutes. Then the owner also leaves, so in the fifth episode the dog is alone in the room. In the next episode the stranger returns and tries to stimulate playing or comfort the dog by petting it. The returning of the owner marks the beginning of the last episode. The stranger leaves the room, the owner interacts with the dog for two minutes and the test ends.

During evaluation, ethologists record pre-defined behavioral variables for describing the dogs’ responses related to both the owner and the stranger and they analyze data to reveal significant differences in behaviors showed towards the two persons.

This complex ethological model has been implemented based on the presented structure. The agent controlled by the model is the representation of the dog, other participants in the test are behaving according to a programmed scenario or are controlled by a human operator. This agent can execute a set of behaviours (moving towards specified objects, pick up or drop the toy object, wag its tail, etc.), some of them can be active and executed in parallel in the same time, but most of them are blocking each other, this latter situation is handled by the behaviour fusion component.

The model is basically designed for simulating the dog’s behaviour in the test described in [63], but the model is flexible enough to react in an ethologically correct way for arbitrary situations.

A framework for simulating the environment hence the inputs (observations from the environment) and for visualization and behaviour evaluation (updating the environment and system states) has been developed. A simplified block diagram of the structure of the implemented simulation is shown in Fig. 28. The framework also simulates the participants in the test as objects with all the required data (position information, state, etc.). The participants can be controlled by an operator via the user interface or by a pre-defined scenario definition playback file, which is useful when testing requires exactly same inputs in every test iteration.

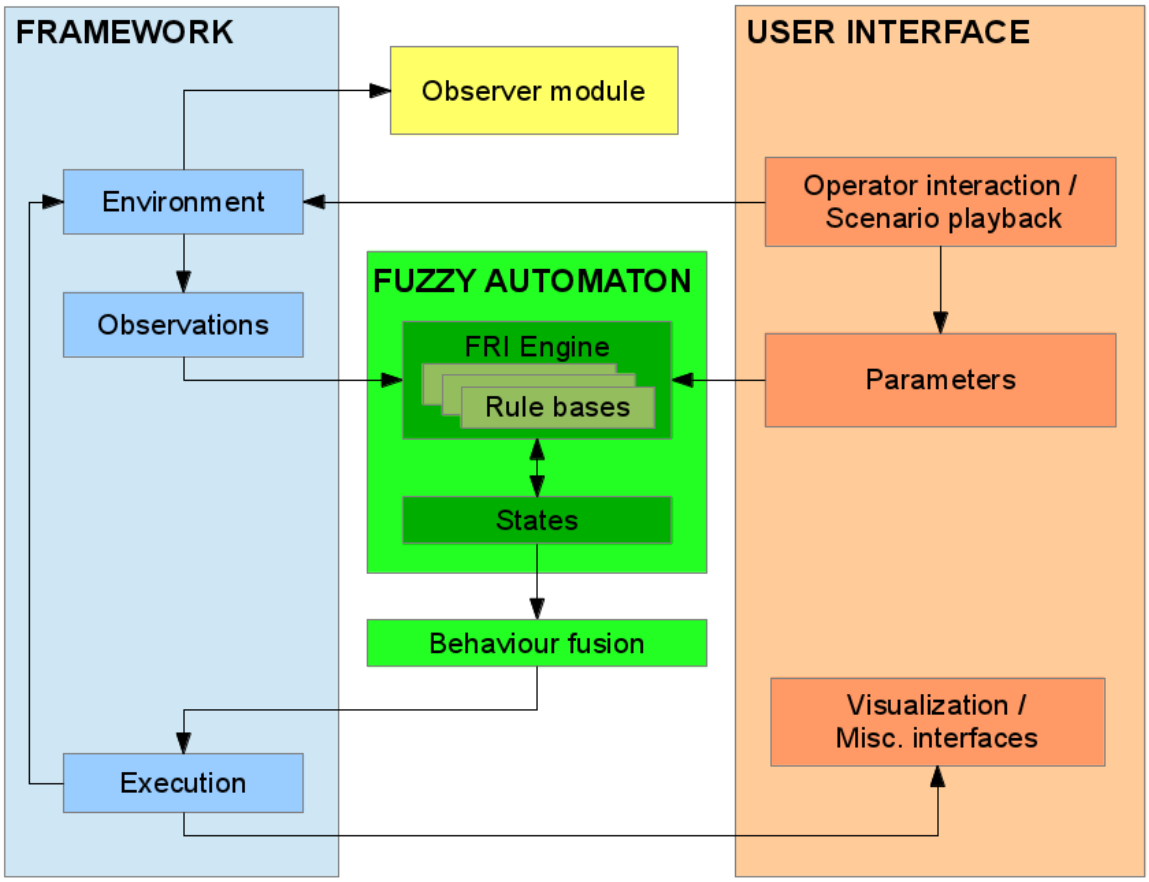

Fig. 28. Structure of the simulation implementation

A screenshot of the simulation framework is shown in Fig. 29., where a room can be seen viewed from above, where the interactions take place. The humans are symbolized with squares, the blue square represents the owner of the dog, the magenta square represents a human unfamiliar (stranger) to the dog. The dog is represented by two circles, the large circle is the body of the dog, the smaller circle is the head of the dog, which is useful for visualizing the direction of the dog A thin green line on the dog's body represents the tail of dog, which can change its length and also its angle (tail wagging). The color of the dog is dynamically changing showing the actual anxiety level of the dog. The room has a door where the human participants can leave or enter the room, the dog cannot go outside. Also a toy object is present (small blue circle, with a hole inside), which can be picked up and dropped by all the participants.

The other labels and vertical sliders are for displaying the actual states of the fuzzy automaton and also the observer module. The buttons, horizontal sliders are for the operator of the simulation for setting parameters, starting/stopping the simulation, etc.

In the followings some example behaviours from the system and their definitions are presented.

The first example behaviour set is built upon two separate component behaviours, namely 'DogExploresTheRoom' and 'DogGoesToDoor'. The 'DogExploresTheRoom' is an exploration dog activity, in which the dog 'looks around' in an unknown environment (see the track marked with white in Fig. 30.). The 'DogGoesToDoor' is a simple dog activity, in

which the dog goes to the door, and than stands (sits) in front of it (waiting for its owner to show up again). The definition of the related state-transition FRI rules of the fuzzy automaton acts as behaviour coordination in this example.

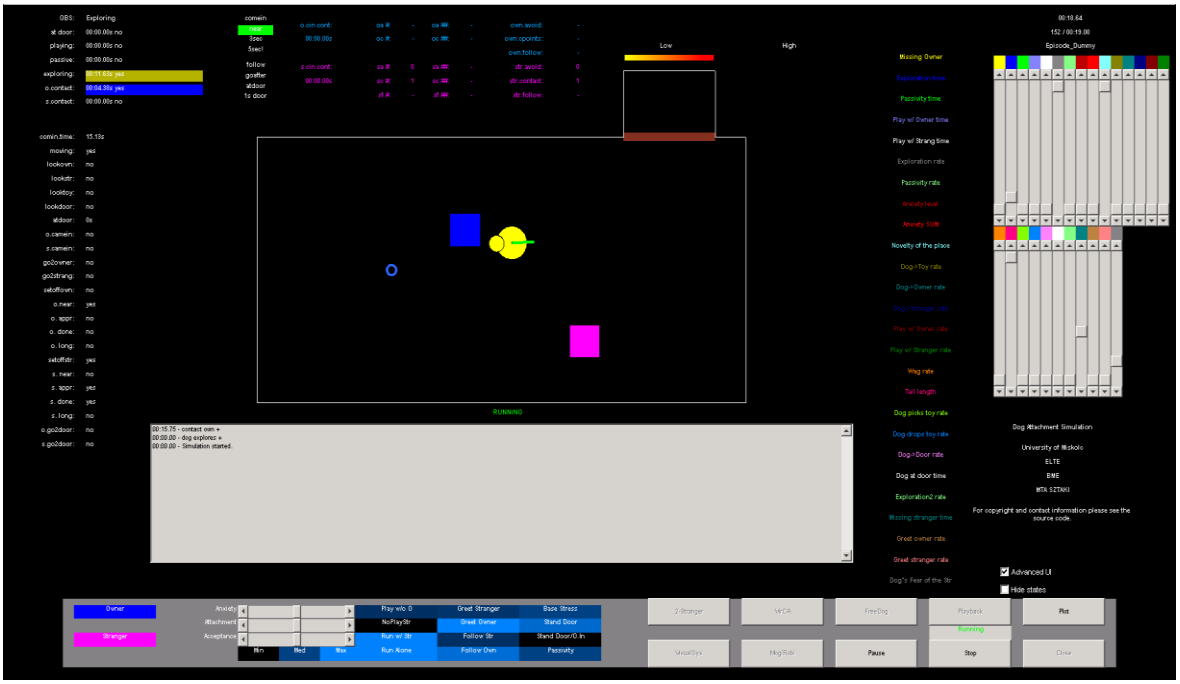

Fig. 29. Screenshot of the simulation application

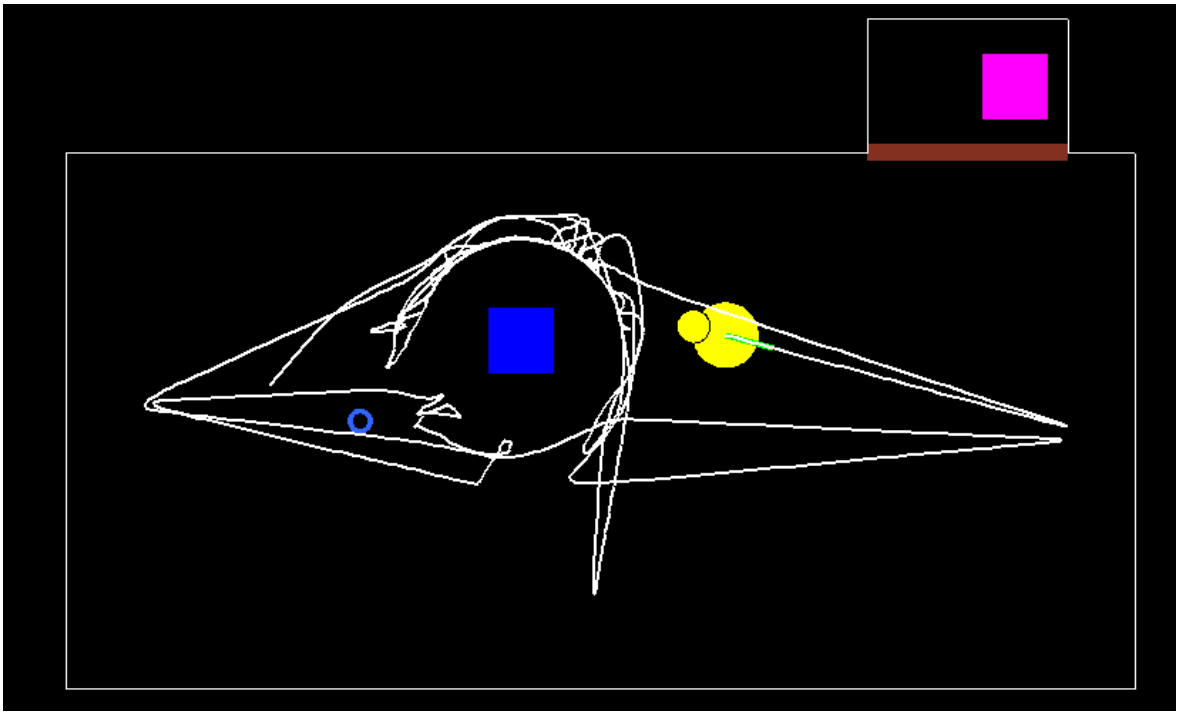

Fig. 30. A sample track induced by the exploration behaviour component

The states concerned are the following:

- 'Hidden' states, which have no direct task in controlling any of the above mentioned behaviours, but has an importance in the state-transition rule base: 'Missing the owner mood of the Dog' (DogMissTheOwner) and 'Anxiety level of the Dog' (DogAnxietyLevel).
- 'Normal' states, which have a direct task in controlling the corresponding 'DogExploresTheRoom' and 'DogGoesToDoor' behaviours: 'Going to the door mood of the Dog' (DogGoesToDoor) and 'Room exploration mood of the Dog' (DogExploresTheRoom).

As a possible rule base structure for the state-transitions of the fuzzy automaton, the following is defined (considering only the states and behaviours for this example):

- State-transition rules related to the missing the owner mood (state) of the Dog:
  - **If** OwnerInTheRoom=*False* **Then** DogMissTheOwner=*Increasing*
  - **If** OwnerInTheRoom=*True* **Then** DogMissTheOwner=*Decreasing*
- State-transition rules related to the anxiety level (state) of the Dog:
  - **If** OwnerToDogDistance=*Small* **And** Human2ToDogDistance=*High* **Then** DogAnxietyLevel=*Decreasing*
  - **If** OwnerToDogDistance=*High* **And** Human2ToDogDistance=*Low* **Then** DogAnxietyLevel=*Increasing*
- State-transition rules related to the going to the door mood (state) of the Dog:
  - **If** OwnerInTheRoom=*False* **And** DogMissTheOwner=*High* **Then** DogGoesToDoor=*High*
  - **If** OwnerInTheRoom=*True* **Then** DogGoesToDoor=*Low*
- State-transition rules related to the room exploration mood (state) of the Dog:
  - **If** DogAnxietyLevel=*Low* **And** OwnerStartsGame=*False* **And** ThePlaceIsUnknown=*High* **Then** DogExploresTheRoom=*High*
  - **If** ThePlaceIsUnknown=*Low* **Then** DogExploresTheRoom=*Low*
  - **If** DogAnxietyLevel=*High* **Then** DogExploresTheRoom=*Low*

The text in italic represent the linguistic terms (fuzzy sets) of the FRI rule base.

Also it should be noted that the rule base is sparse, containing the main state-transition FRI rules only.

A sample run of the example is introduced in Fig. 30. and in Fig. 31. At the beginning of the scene, the owner is in the room and the stranger is outside. The place is unknown for the dog ('ThePlaceIsUnknown=High' in the rule base). According to the above rule base, the dog starts to explore the room (see Fig. 30.). After a little while the owner of the dog leaves the room, and then the stranger enters and stays inside (dashed white tracks in Fig. 31.). As an effect of the changes (according to the above state-transition rule base), the anxiety level of the dog and the 'Missing the owner' state is increasing and as a result, the dog goes to

and stays at the door (white track in Fig. 31.), where its owner has left the room. See the state changes in Fig. 32.

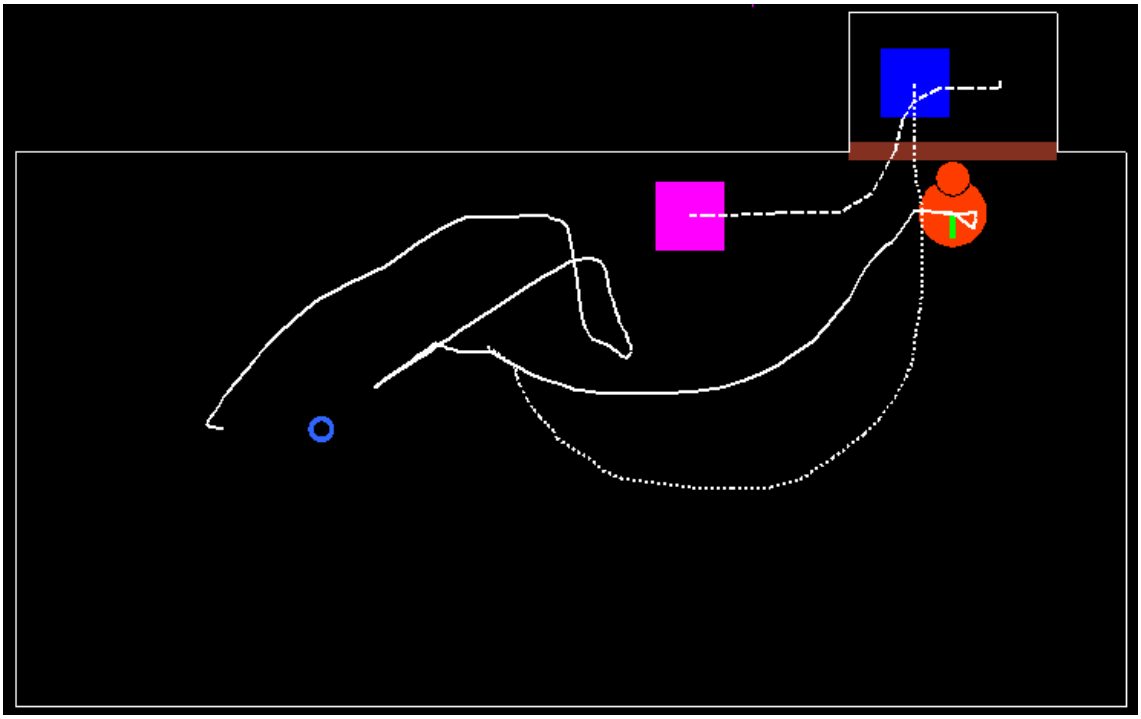

Fig. 31. A sample track induced by the 'DogGoesToDoor' behaviour component

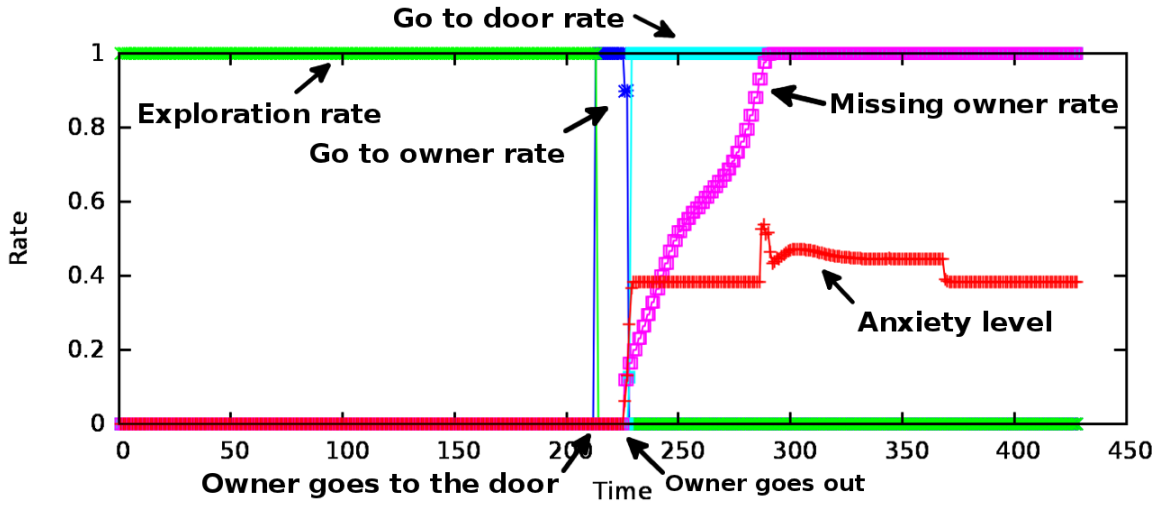

Fig. 32. Some of the state changes during the sample run introduced in Fig. 31.

Fig. 32. shows the values of four state variables on a time scale (iteration number). Two immediate changes can be seen which are both caused by events related to the movement of the owner. First when the dog realizes that the owner is heading for the door (around the 200th iteration) the exploration rate drops rapidly and the rate of the 'DogGoesToOwner' behaviour component is high, till the owner finally leaves the room (around the 250th iteration). When the owner had left the room, the 'DogGoesToDoor' behaviour component

will be active, causing the value of the ‘Missing the owner’ state to increase. At the same time the anxiety level of the dog also increases with the leaving of the owner.

In the second example the ‘DogGoesToOwner’ behaviour will be presented in details from another point of view than the previous example. As its name suggests, this behaviour component is responsible for determining the need to go to the owner for the dog. The description of this behaviour component is more complex than the rule base in the previous example. The complete rule base for the ‘DogGoesToOwner’ behaviour is shown in Table 10., where the last column means the consequent part, all the other columns are antecedent dimensions (the ‘*x’* value in the rule description means that the value of that antecedent is omitted while reasoning – does not have effect in the rule). The explanation of the terms used in the rule base is the following (‘observation’ in this case means that the value is gathered (measured) directly from the environment, ‘previous state’ means that the value was calculated in the previous iteration as a conclusion of a rule base):

- **dgtd** – previous state – dog goes to door rate (zero-large) – see Fig. 36. for the corresponding fuzzy partition
- **dgto** – previous state – dog goes to owner rate (zero-large) – see Fig. 36. for the corresponding fuzzy partition
- **oir** – observation – owner is inside (true-false) – see Fig. 33. for the corresponding fuzzy partition
- **ddo** – observation – distance between dog and owner (zero-large) – see Fig. 34. for the corresponding fuzzy partition
- **dgtt** – previous state – dog goes to toy rate (zero-large) – see Fig. 33. for the corresponding fuzzy partition
- **dgro** – previous state – dog greets owner rate (zero-large) – see Fig. 33. for the corresponding fuzzy partition
- **dpmo** – previous state – dog’s playing mood rate with the owner (zero-large) – see Fig. 33. for the corresponding fuzzy partition
- **dpms** – previous state – dog’s playing mood rate with the stranger (zero-large) – see Fig. 33. for the corresponding fuzzy partition
- **danl** – previous state – dog’s anxiety level (zero-large) – see Fig. 35. for the corresponding fuzzy partition
- **ogo** – observation – owner is going outside (true-false) – see Fig. 33. for the corresponding fuzzy partition
- ***dgto (consequent)*** – next state – dog goes to owner rate (zero-large) (output)

Table 10. Rule base for the 'DogGoesToOwner' behaviour component

#	dgtd	dgto	oir	ddo	dgtt	dgro	dpmo	dpms	danl	ogo	dgto
1	dgtdl	x	oirt	x	x	x	x	x	x	x	dgtoz
2	dgtdz	x	oirt	x	dgttz	dgrol	x	dpmsz	x	ogof	dgtol
3	dgtdz	x	oirt	x	dgttz	x	x	dpmsz	danll	ogof	dgtol
4	dgtdz	x	oirt	x	dgttz	x	dpmol	x	x	ogof	dgtol
5	dgtdz	x	oirt	x	x	x	x	x	x	ogot	dgtol
6	dgtdz	x	oirt	x	x	dgroz	dpmoz	x	danlz	ogof	dgtoz
7	x	x	oirf	x	x	dgroz	x	x	x	ogof	dgtoz
8	dgtdz	x	x	x	dgtth	dgroz	x	dpmsz	x	ogof	dgtoz
9	dgtdz	x	x	x	x	dgroz	x	dpmsl	x	ogof	dgtoz
10	dgtdz	dgtol	oirt	ddoz	dgttz	dgroz	dpmoz	dpmsz	x	ogof	dgtoz
11	dgtdz	dgtol	oirt	ddol	dgttz	x	x	dpmsz	x	x	dgtol

The rules can be interpreted as the following textual description, in the corresponding order:

1. dog goes to door → do not go to owner
2. dog should greet the owner → go to owner
3. dog is nervous → go to owner
4. dog's playing mood with owner is high → go to owner
5. owner is going out of the room → go to owner (follow the owner)
6. dog is calm and dog is not playing and not greeting → do not go to owner
7. owner outside → do not go to owner (can't go to the owner)
8. dog is going to the toy → do not go to owner
9. dog's playing mood with stranger is high → do not go to owner
10. already going to owner and owner is near → do not go to owner (arrived at the owner)
11. already going to owner and owner is far → going to owner (keep on going)

The presented rule base is suitable to define a complex behaviour component with only eleven rules in spite of the relatively high antecedent dimension count (ten variables) thanks to the usage of FRI. In Fig. 33. through Fig. 36. the fuzzy partitions used for the various antecedents are presented. From top to bottom the curves in the mentioned figures show the membership functions, the scaling functions (see FIVE FRI in Chapter II.) and the reconstructed membership functions (based on the scaling function). As it can be seen most of the fuzzy partitions are simple (six of the nine antecedents use the same simple fuzzy partition), consisting of only two triangular fuzzy sets with their cores at 0 and 1, and steepness of 1 on the right and the left side respectively. This is true also for the other rule

bases in the model, most of the antecedent variables are defined by fuzzy partitions exactly like the one shown in Fig. 33.

Most of the behaviour components defined in the system are responsible for coordinating the movement (heading direction) of the dog, which must be aggregated by some kind of behaviour fusion. The simulation uses a simple weighted sum of the currently required behaviour components for determining the movement vector of the agent. Other behaviour components can be executed in parallel with movement related components. Also there are rule bases which are responsible for calculating the rate of behaviour components which are not directly executed, but instead their values are used by other rule bases as input values (in a hierarchical manner). E.g. the 'DogGreetsOwner' component is not executed directly, as shown in Table 10. the consequent of 'DogGreetsOwner' is used as an input value in the rule base of the 'DogGoesToOwner' behaviour component, hence the actual output behaviour will be triggered by the activation of the 'DogGoesToOwner' behaviour component.

The model can be customized for different types of dogs. The system provides three factors for customization ('Parameters' in Fig. 28.), which can be set in the interval [0, 1]: 'Anxiety', 'Acceptance', and 'Attachment'. The 'Anxiety' parameter defines the level of anxiety of the dog on a scale between calm and nervous, 'Acceptance' defines the level of the dog's acceptance towards the stranger (between friendly and unfriendly), 'Attachment' defines the level of attachment of the dog to its owner (between weak and strong attachment). In fact these parameters are not directly used in the behaviour description rule bases. Secondary variables are calculated first based on the three parameters. Then these secondary parameters are used in the corresponding fuzzy rule bases. The three external parameters ('Anxiety', 'Attachment', 'Acceptance') mentioned earlier are used as antecedents in various rule bases, which have the task of determining the values of secondary parameters. These secondary parameters are then directly used as antecedents in the rule bases of the fuzzy automaton driving the simulation. These calculated secondary parameters are the following:

- Play without the owner (when the owner is outside)
- Do not play with the stranger (even when it could be possible)
- Run in circles (caused by anxiety) when stranger is inside
- Run in circles (caused by anxiety) when the dog is alone
- Base rate of greeting the stranger
- Base rate of greeting the owner
- Base rate of following the stranger, when he/she is leaving the room
- Base rate of greeting the owner, when he/she is leaving the room
- Base rate of stress (anxiety)
- Base rate of standing at the door
- Base rate of standing at the door, even when the owner is inside
- Base rate of passivity

This hierarchical strategy is useful for minimizing the input dimensions of the rule bases (one input variable is used instead of three input values in this case), hence fewer rules are required. The values of the secondary parameters are also calculated using fuzzy rule interpolation, each secondary parameter having its own rule base for determining the exact value based on the three input parameters. These are calculated at the time of initialization before starting the simulation, but can be also adjusted while the simulation is running, allowing the operator to switch between different types of dogs in real-time.

A distinct, but important part of the framework is the so called observer module (see Fig. 28.), which has the task of evaluating the model (the correctness of the constructed rule bases) itself. The observer module only gets information from the environment (just what an outsider would see), no data about the inner states of the fuzzy automaton or behaviour weights are supplied to the observer. Based on these observations the observer module is able to determine what kind of activity is going on in the simulation. It can distinguish among ‘playing’, ‘exploring’, ‘standing at door’ and ‘passive’ states. If the dog is in contact (close the human and looking at the human) with the owner or the stranger, that is also measured and logged. Furthermore a special scoring system developed by ethologists used for evaluating the separation and reunion episodes (human leaves dog / human comes back) is also implemented in the observer module. Also the overall time spent in the various states along the different episodes is measured. A summary of the measurements and the scores is calculated at the end of the simulation, which can be directly used by ethologist experts to evaluate the simulated dog in the same manner as they would with real dogs.

As the implementation of the simulation fits the requirements of the usability of the optimizations presented for the FIVE FRI in Chapter IV., hence those are successfully applied (except the optimization especially for FRIQ-learning) in the implementation of the simulation model, making it significantly faster for the end user.

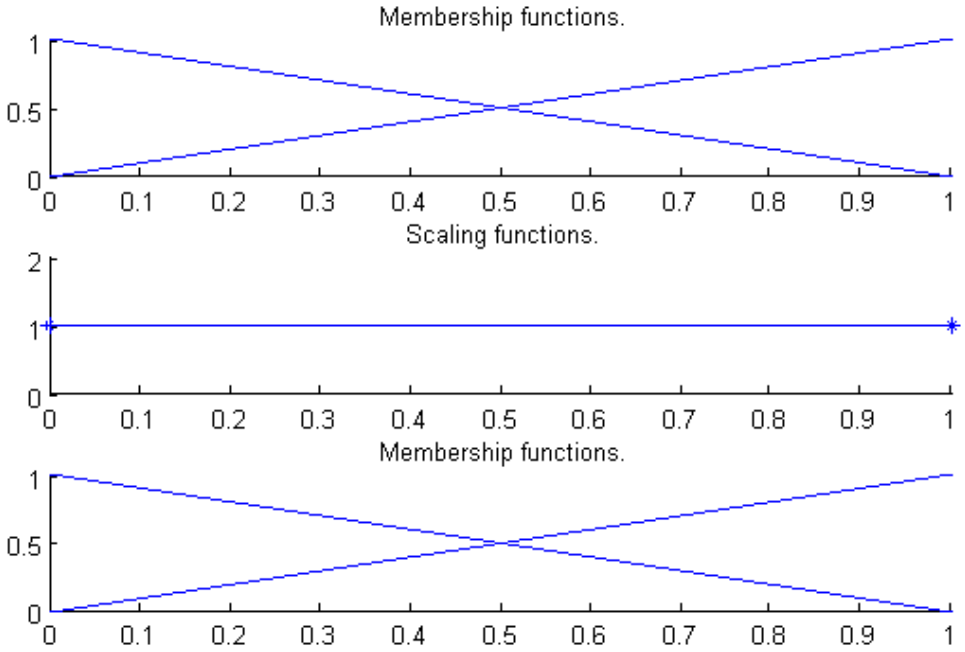

Fig. 33. Fuzzy partition of the following terms:
dgro - dog greets owner,
dpmo - dog's playing mood with the owner,
dpms - dog's playing mood with the stranger,
dgtt - dog goes to toy,
dgtd - dog goes to door,
oir - owner is inside,
ogo - owner is going outside

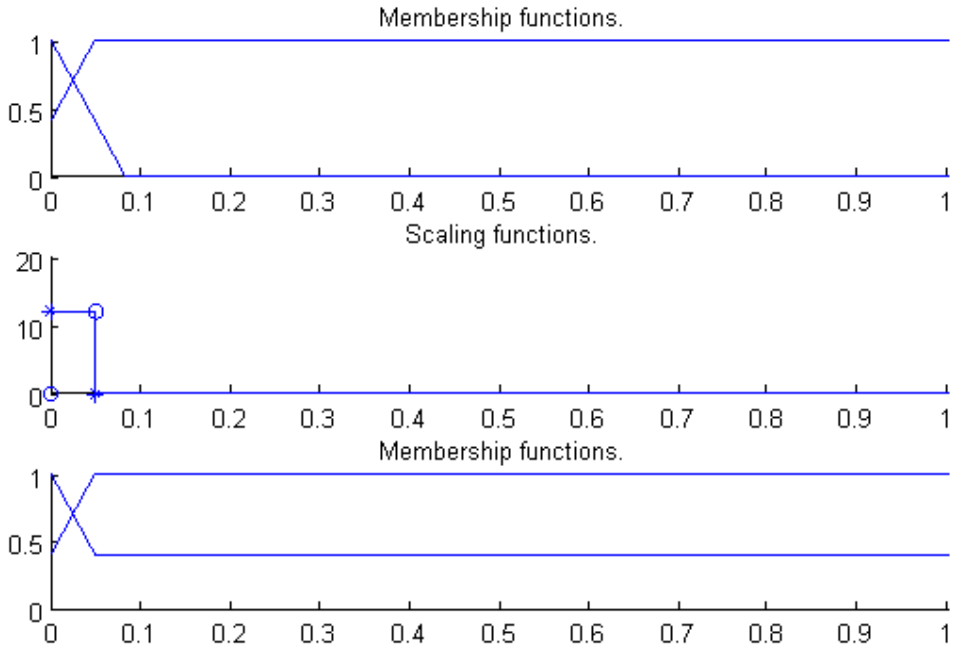

Fig. 34. Fuzzy partition of the term *ddo (*distance between dog and owner*)*

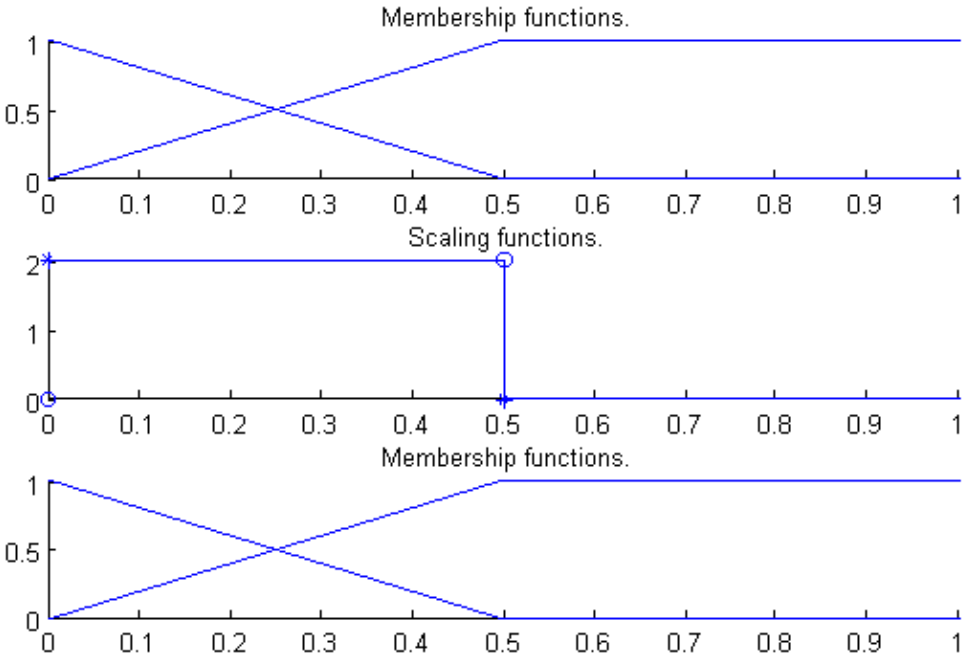

Fig. 35. Fuzzy partition of the term *danl* (dog's anxiety level)

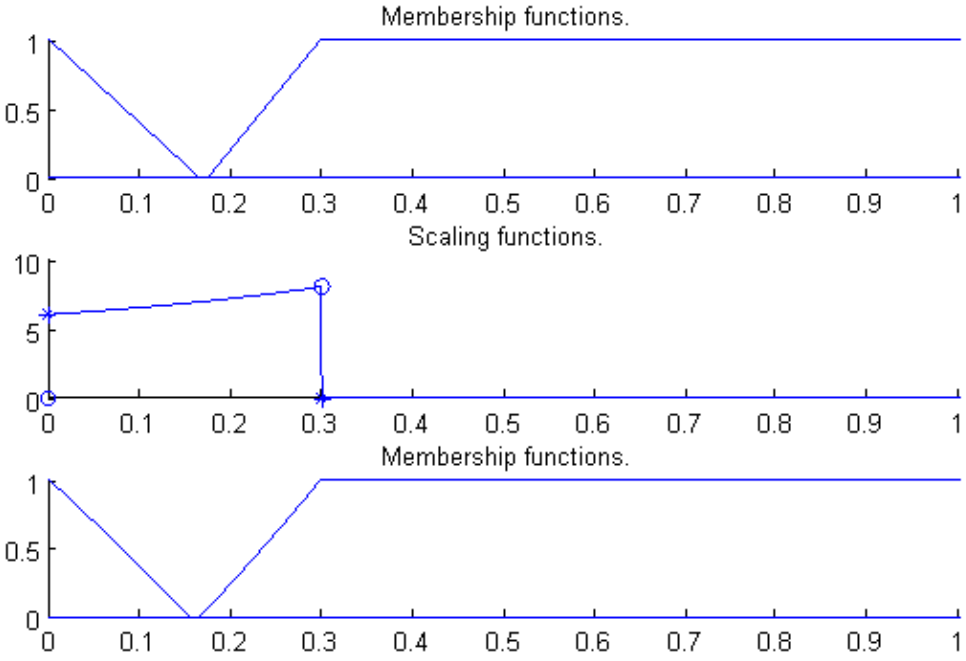

Fig. 36. Fuzzy partition of the term *dgto* (dog is going to owner)

5.5 Interfaces for the simulation

The following subchapters present the human-robot interfaces developed for the previously introduced simulation model. These interfaces have been originally introduced in [87].

5.5.1 The original 2D interface

As presented earlier, the original interface for the test procedure is a simple two dimensional plan view displaying the test room and the participants (a screenshot of the application screen is shown in Fig. 29.). It was intended only for testing and development purposes for engineers.

The main area is the test room with a door leading outside. The two human participants are symbolized by the two filled rectangles, the blue one is the owner and the magenta is the stranger. The circular shaped object is the dog, which also has a head and tail sub-object. The head is used for determining the orientation of the dog. Also a toy object is defined, which can be used for interaction between the participants. The objects can be controlled using conventional personal computer input devices (e.g. mouse and keyboard). And of course according to the ethological test procedure the humans can be controlled automatically based on a text script file defining the scenario to be played. This way the dog will act according to the model, in real-time.

The user interface of the simulation application also incorporates ways to give information on the inner states of the model, which is useful for engineers. The various states of the fuzzy automaton are shown in real-time on slider widgets, also a text-box can be found in the center which logs the occurred events.

To help in easier testing for ethologists, an object control and movement recording feature was incorporated. An ethologist at a remote location can record the exact coordinates and movement of the objects, this way, ethologists can exactly show where and what adjustments are required to improve the model and also in case of errors, those will be reproducible.

Also the application has a sophisticated observer module (see previous subchapter), which evaluates the behaviour of the dog based only on data which can be observed from the outside, without knowing any information of the states of the model. This observer module is a key part in the evaluation of the whole model's correctness.

5.5.2 The 3D Virtual Collaboration Arena interface

The Virtual Collaboration Arena (VirCA) [64] is a modular, easy to use 3D framework supporting the development of augmented reality applications. The augmented reality, the mixture of real and virtual environments, gives the unbeatable chance for experiencing the most realistic direct personal interaction with a virtual entity available only in a virtual environment.

The main idea of VirCA is to place physical devices in a computer generated virtual space, where objects can interact. These objects are called CyberDevices and can be either

representations of physically existing objects or completely virtual objects. VirCA is a mixture of different technologies. For visualization it uses a 3D engine called OGRE 3D (Object-Oriented Graphics Rendering Engine, see [22] for details), which is a scene-oriented, flexible 3D engine designed to make it easier and more intuitive for developers to produce applications utilizing hardware-accelerated 3D graphics. Also a component called Bullet is integrated, which is an open source physics engine featuring 3D collision detection, soft body dynamics, and rigid body dynamics. VirCA can connect research groups and distant laboratories (devices) over the Internet using standard protocols. For communication between the components the Robot Technology Middleware (RTM) [1] is applied. RTM communication is based on the well-known CORBA middleware, but can also make use of the flexible communication channels of the Internet Communication Engine (ICE) [40]. According to the RTM concept, the VirCA is also an RT-Component, which brings the components together and provides a simple to use visualization and user interface by utilizing the aforementioned technologies.

This way VirCA can provide a solution for the users to collaborate with each other to control physically existing devices remotely in an easy to use mixed real and virtual 3D environment. For example when industrial robots are working in dangerous environments the presence of human operators is an unnecessary risk. In this case the devices can be controlled with methods close to real world methods, avoiding direct human presence.

To apply the services provided by VirCA, special VirCA interfaces (RTM) are needed in the actual programming environment. The original simulation application was implemented in MATLAB, which has no native RTM interface (which could be used for direct VirCA communication). Hence an intermediate adapter was developed, which acts as a proxy between the two systems. For communication between the proxy and the simulation application, the standardized UDP (User Datagram Protocol) over IP (Internet Protocol) was chosen to be used.

As described earlier, the three participants of the test procedure are the dog, the owner and the stranger. On the original 2D interface these are represented with very simple objects. The two human objects can be freely controlled by the operator of the application, but it can be also controlled by the simulation application based on the pre-defined commands stored in a simple script text file. The movement and behavior of the dog object is strictly controlled by the application, as this is the main purpose of the implemented model.

These objects we are referring to are called CyberDevices in the VirCA terminology. Each CyberDevice has its own function and ports. For example, the dog component is responsible for receiving and processing messages from the MATLAB simulation program, and then it has to send the rescaled and recalculated coordinates to the VirCA component via RTM protocol.

In the original standalone MATLAB model the dog is composed from two circular objects, the body and the head (see Fig. 29.). However in VirCA, the dog is one solid detailed dog shaped object with head, torso, legs and tail. The position of the dog's head and the dog's body from the 2D space is used for calculating the orientation of the dog. This additional value is required for displaying the 3D dog object properly.

The owner and stranger components are like the same from the viewpoint of CyberDevice communication, but the reverse directions also have to be handled. This latter means that users can interact with these objects in the VirCA augmented space and the new positions should be transferred back to the MATLAB model. The dog component communicates only one-way, from MATLAB to VirCA. The new coordinates of the dog object are calculated by the model based on the observations and the inner system states. The toy component is somewhat different, in addition to handling the coordinate conversions between the two environments. The toy object can be picked up, dropped and thrown by the other participants.

The virtual room in the augmented VirCA environment was constructed resembling to the real laboratory room used for conducting the real test. Fig. 37. shows a screenshot of the model running in the VirCA environment. For more details on the interface between MATLAB and VirCA see [85], [86] and [89].

Fig. 37. The 3D VirCA interface of the simulation application

5.6 Intelligent Space interfaces

Intelligent Space [43] (iSpace) reverses the traditional concept of robot development. Traditionally the control logic, sensors, etc. are built into the robot hardware itself. Intelligent Space instead embeds sensors into the environment, centralizes the control logic, and the robot hardware itself becomes only an actuator. This makes robot hardware simpler, more complex processing and more resources can be allocated for the control logic, cooperation can be also simpler, because it can be coordinated centrally. The drawback could be that robots lose their independence this way, and also Intelligent Spaces are not easy to mobilize, and could not be used in every environment (e.g. open air spaces).

An interface incorporating the Intelligent Space concept has been developed for the model of the test, this will be described in the following subchapter.

5.6.1 ‘MOGI robi’

The 'MOGI robi' is a robot hardware [30] especially built as an interface for the presented ethological test procedure simulation. The robot hardware was designed at the Budapest University of Technology, in the Department of Mechatronics, Optics, and Mechanical Engineering Informatics (in Hungarian the abbreviation of the name of the department is MOGI, also the word ‘robi’ is a nickname for robot, hence the name ‘MOGI robi’).

Connecting the ‘MOGI robi’ and the simulation application together, takes the incarnation of the model to a higher level. Ethologists can conduct real-life tests in a real test room, with real human participants, exactly as the same way they would do it with a real dog.

The ‘Intelligent Space’ setup consists of three separate parts in this case: the robot hardware, a vision system, and the simulation application itself. See Fig. 38. for a schematic diagram.

The robot hardware, ‘MOGI robi’ is based on holonomic wheel drive (three separate wheels), hence it can move in 3 degrees of freedom (3DoF). The robot has a separate head part, which can be controlled also in 3DoF. Another important part of the robot is a gripper, which is used for grabbing and releasing the toy. It is also equipped with a sophisticated mechanical tail for mood expression. Fig. 39. shows the ‘MOGI robi’ [30].

The vision system uses ordinary video cameras to gather samples of the environment. The participants have to be tagged with different coloured independent light sources, so they can be identified and positioned in the given space by the vision system. The simulation application has been extended to be able to communicate with the vision system and to control the robot hardware.

As for the vision system, the communication is only one way. The vision system periodically sends the actually sensed positions of the various objects participating in the simulation. This is achieved by sending data encapsulated in UDP over IP, similarly to the VirCA interface described earlier. The simulation application checks for the received UDP packets periodically (at a lower rate than the packets are sent). As only the newest position

information is required, in case of multiple packets are received, all of them are dropped except the latest one.

As for the robot hardware the communication is bi-directional, which is physically realized by communicating via a Bluetooth adapter. The simulation application sends the rates of the required behaviour to a bridge application using UDP packets. After the bridge application converts the received behaviour rates to a movement vector or other action, it sends the desired position to the robot via the Bluetooth adapter, which is accessed as a simple serial port (RS-232) device. The desired position cannot be always reached in time because of several possible reasons, e.g. small unseen obstacles, rough terrain, not enough acceleration, etc. In this case 'in time' means the time till the execution of the next iteration cycle of the simulation application. In every iteration, data from the vision system are received, this way the simulation application can check whether the robot reached the desired position or not (or maybe overrun on it). Based on the new data the application can recalculate and send the new desired direction to the robot hardware.

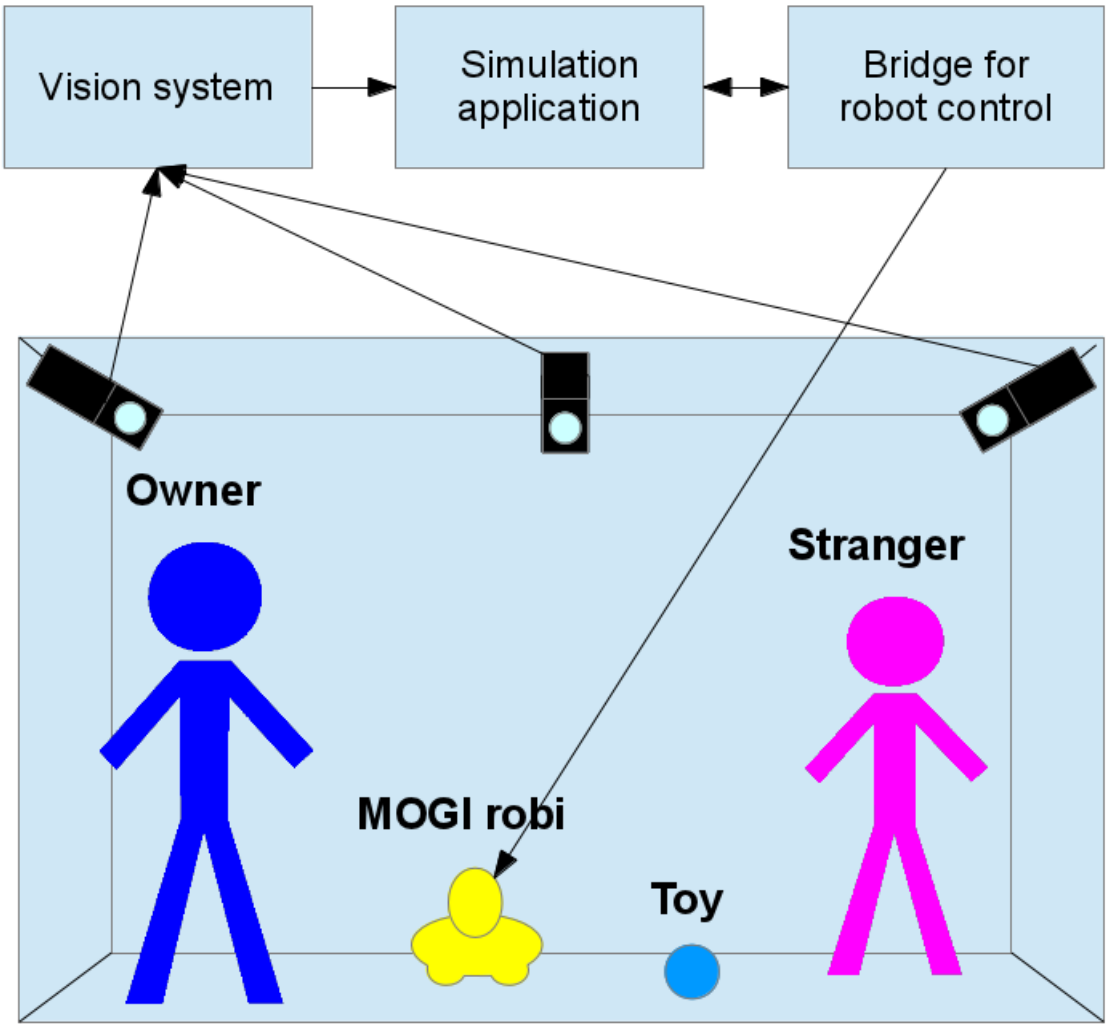

Fig. 38. Schematic diagram of the 'MOGI robi' in Intelligent Space

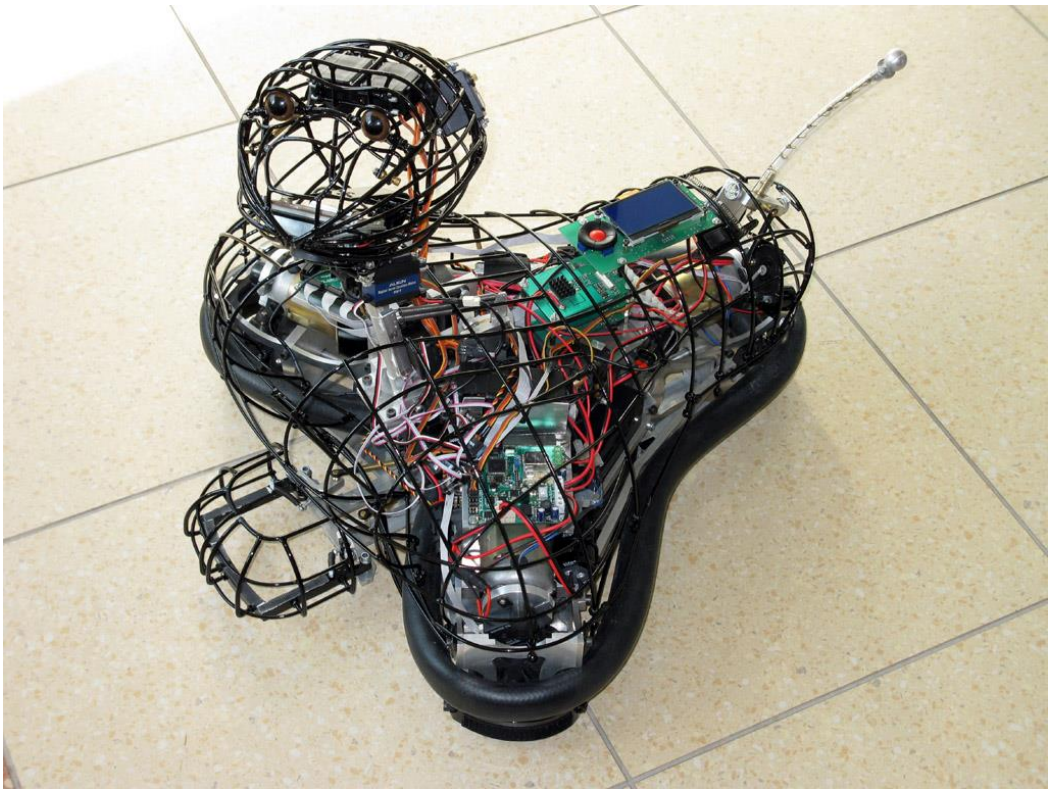

Fig. 39. The ‘MOGI robi’ [30] without outer skin

The mechanical tail of the robot is also controlled by the simulation application, based on the output of the model, the actual wagging rate and the length of the dog’s tail is sent and interpreted by the robot.

The toy gripper has to be activated by the simulation application also. First the model commands the robot to go for the toy object and when the robot gets near the toy, the bridge application drives the robot into the correct position and orientation and activates the gripper (open/close) to grab the toy. Releasing the toy is a simpler task, because it can be done practically in any position, but considerations have to be made, because after opening the gripper, the robot has to move backwards to free the toy.

Data can be received from the robot hardware also. This is mainly used for checking that the robot is online, ready for duty, charge level of the batteries, etc. The current estimated position can be queried from the robot (based on a reference starting point and the occurred movement since the start), which can be useful in some temporary cases when the robot cannot be seen by the vision system (e.g. other participants are blocking the sight).

5.6.2 A monitoring system for home-care support

To further demonstrate ethologically inspired robotics, another robot hardware working in Intelligent Space is presented in this section. This system is not intended to be used by ethologists, but a real-life application of the concept of ethologically inspired robot design.

A monitoring system intended for home-care was developed [48] at the Human-System Laboratory, Department of Precision Mechanics, Chuo University in Tokyo, Japan, which applies an ethologically inspired approach for human-robot communication. The main task

of this monitoring system is to monitor elderly people, and in case an unusual event occurs (e.g. person lying on floor or person is at a forbidden location) alert a caregiver person.

The system is based on the concept of the model presented earlier, of course with different rule bases.

As opposed to ‘MOGI robi’, instead of using a vision system with cameras for gathering the positions of the participants in Intelligent Space, an ultrasonic positioning system was constructed. This overcomes the problem of tagging participants with constant light markers, which can be uncomfortable for them. The ultrasonic transmitters can be worn hanging in the neck without disturbing the comfort of the subject.

The robot hardware consists of a Pioneer3-DX platform and a Nabaztag, see Fig. 40. This robot is used to deliver alerts to a caregiver person. Alerts can be expressed in different ways. For more details on this monitoring system see [48].

Fig. 40. The robot hardware used in the monitoring system [48]

5.7 Summary

A working ethologically inspired model built on fuzzy rule interpolation based fuzzy automaton was successfully constructed. The rule bases of the fuzzy automaton were constructed based on knowledge extracted from an ethologist expert. The rule bases describe various behaviour components of a dog in the strange situation test, which is a standard ethological test procedure applied on dogs. The system consists of many rule bases, one rule base for each and every behaviour component, some of them are arranged in a hierarchical fashion, rule bases can use the output of another rule base as an input. For every behavior component a weight for the necessity of execution of the corresponding

behaviour is calculated. Based on these weights the final behaviour components should be selected. There are behaviour components which can be executed in parallel, but some of them are contradicting each other. Resolving these contradictions is the task of the behaviour fusion part in the model. Having chosen and merged the final behaviour components and all the states are updated, the required behaviour components can be executed by the agent (dog).

A corresponding framework was also developed which makes the operation of the model possible. The main task of the framework is to simulate the measurement from the environment (simulate inputs data) and to simulate the execution of the commands from the fuzzy automaton (simulate the processing of outputs). Via the framework an operator person can interact with model and see the results in real-time.

A separate module in the framework, the observer module was implemented with the purpose of evaluating the correctness of the model. The observer can determine the current action of the dog, also can measure the times spent with the different actions. Based on the measured times ethologists can evaluate the model and the agent itself, in the same manner as they would do with real dogs.

Various interfaces were developed for this simulation, including the original 2D interface intended for developers, also a 3D 'virtual reality' interface using the VirCA system. Furthermore, robot hardware was also constructed, which is controlled remotely by this simulation application.

The scientific results of the work presented in this chapter (see [86], [87], [77], [78], [79], [80], [85] and [89] for in-depth details) summarized as a thesis is the following:

Thesis III.: Based on descriptive verbal ethological models, which have simple rule-based knowledge representation format and where the completeness of the rule-base is not required, the Fuzzy Rule Interpolation-based fuzzy automaton (fuzzy state machine) extended with a capable behaviour fusion engine is suitable for describing ethologically inspired behaviour models. I concluded that the human-dog version of the 'Strange Situation Test' can be modeled using FRI-based fuzzy automata by transforming the initially verbal model to a fuzzy model and defining the appropriate behaviour components along with the strategy of behaviour fusion.

Contributions and future research directions

This work contributes to the field of fuzzy systems (especially fuzzy rule interpolation), reinforcement learning and also simulation.

A novel method, the FRIQ-learning (Fuzzy Rule Interpolation-based Q-learning) has been developed, along with methods to create minimal sized rule bases for operating the system. This method not only allows the original Q-learning method to be used in continuous spaces, but makes it possible to use spare fuzzy rule bases with reinforcement learning, thanks to fuzzy rule interpolation. This way only the cardinal rules have to be present in the rule base, the derivable and unnecessary rules can be omitted, which makes the overall rule base size significantly smaller. An extension of the method for creating minimal sized rule bases automatically works by incrementally constructing then decrementally reducing the rule base. Starting with a minimal, generated rule base, the method inserts new rules or updates the existing rules based on the rewards gathered in each and every iteration step. The incrementally constructed rule base can contain rule, which were only necessary during the construction, but not in the final rule base, these rules can be identified and removed by the decremental reduction strategy. See Thesis I. and Thesis II. below, and also the third chapter for detailed description of the methods.

The incorporated fuzzy rule interpolation method, FIVE, has been successfully optimized specifically for FRIQ-learning, taking the performance of the method to a higher level. See the fourth chapter for in-depth discussion.

Furthermore an ethologically inspired behaviour model using fuzzy rule interpolation based fuzzy automata for handling the state-transitions in a standard ethological test was developed along with a sophisticated framework. The model incorporates the rules for controlling an agent, which behaves like a dog in given situations in a standard ethological test, the strange situation test. The model can be operated by an ethologist expert in real time via the developed framework, or via other software and hardware interfaces specially developed for this model. Details on the structure of the model can be found in the fifth chapter, and also see Thesis III. below.

Regarding further research possibilities in the future, investigation of the possibilities of automatic knowledge extraction in ethologically inspired models seems promising.

Extracting expert knowledge in the form of verbal rules is not necessarily possible in every case, because the principles of operation of certain behaviours and function could be unknown or unavailable. Instead, expert knowledge should be extracted in the form of defining the desired goals of the various behaviour components. If these desired goals can be composed as reward functions in a given state-action space, then FRIQ-learning can be applied for automatic construction of rule bases for the behaviour components. This way the extracted expert knowledge can be transformed into a rule-based knowledge representation form. Furthermore, with the help of the decremental reduction method the rule base possibly can be made so small, that the principles of operation can be directly extracted in a human readable form.

Also it could be worth investigating the possibilities of developing additional strategies for the incremental construction and decremental reduction of FRI rule bases used in FRIQ-learning. These could include adding and removing groups of rules at once, which could be an efficient way of improving the performance of the methods. Also different rule insertion and removal policies could be investigated.

The scientific results of the research presented in this work summarized as theses can be read in the followings:

Thesis I.: **[82], [84], [90], [81], [91], [88]**

The FIVE based fuzzy rule interpolation (FRI) model is suitable for describing the Q function of the Q-learning method (FRIQ-learning). I concluded that describing the Q function with the FIVE based FRI model results in a possible continuous space extension of the action-state space, where the original Q-learning algorithm was defined in discrete action-state space. Furthermore I concluded, that describing the Q function with the FIVE based FRI model allows the omission of some (redundant) states, that is, the simplification of the model. Also, the FIVE FRI method can be specifically optimized for the proposed FRIQ-learning method.

Thesis II.: **[84], [90], [81]**

In case of FRIQ-learning the cardinal rules, also the number of rules describing the action-state space (Q function) can be determined automatically (in an incremental/decremental fashion) in run-time with the appropriate evaluation of the reward function starting from an automatically generated base rule base. I concluded that the state-transition rule-base of the operating FRI-based fuzzy automaton can be extracted from the rule-based action-state space model created this way.

Thesis III.: **[86], [87], [77], [78], [79], [80], [85], [89]**

Based on descriptive verbal ethological models, which have simple rule-based knowledge representation format and where the completeness of the rule-base is not required, the Fuzzy Rule Interpolation-based fuzzy automaton (fuzzy state machine) extended with a capable behaviour fusion engine is suitable for describing ethologically inspired behaviour models. I concluded that the human-dog version of the 'Strange Situation Test' can be modeled using FRI-based fuzzy automata by transforming the initially verbal model to a fuzzy model and defining the appropriate behaviour components along with the strategy of behaviour fusion.

References

[1] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., Woo-Keun Yoon: RT-middleware: distributed component middleware for RT (robot technology), in Proceedings of IROS 2005 - IEEE/RSJ International Conference on Intelligent Robots and Systems, 2-6 Aug. 2005, pp. 3933-3938.

[2] Appl, M.: Model-based Reinforcement Learning in Continuous Environments. Ph.D. thesis, Technical University of München, München, Germany, dissertation.de, Verlag im Internet, 2000

[3] Baranyi, P., Kóczy, L. T., Gedeon, T. D.: A Generalized Concept for Fuzzy Rule Interpolation, IEEE Trans. on Fuzzy Systems, vol. 12, No. 6, 2004, pp 820-837.

[4] Bellman, R. E.: Dynamic Programming. Princeton University Press, Princeton, NJ 1957

[5] Belohlavek, R.: Determinism and fuzzy automata, Inf. Sci. 143, pp. 205–209., 2002

[6] Berenji, H.R.: Fuzzy Q-Learning for Generalization of Reinforcement Learning. Proc. of the 5th IEEE International Conference on Fuzzy Systems, pp. 2208-2214., 1996

[7] Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function, Plenum Press, New York., 1981

[8] Bonarini, A.: Delayed Reinforcement, Fuzzy Q-Learning and Fuzzy Logic Controllers. In Herrera, F., Verdegay, J. L. (Eds.) Genetic Algorithms and Soft Computing, (Studies in Fuzziness, 8), Physica-Verlag, Berlin, D, (1996), pp. 447-466.

[9] Dautenhahn, K.: Methodology and themes of human-robot interaction: A growing research field. Int. J. of Advanced Robotic Systems, 4, pp. 103-108. (2007)

[10] Doostfatemeh, M., Kremer, S. C.: New directions in fuzzy automata, International Journal of Approximate Reasoning 38, 2005, pp. 175-214.

[11] Dubios, D., Ostasiewicz, W., Prade, H.: Fuzzy Sets: History and Basic Notions, in: Fundamentals of Fuzzy Sets, ISBN 978-0-7923-7732-0, Kluwer Academic, 2000

[12] Horiuchi, T., Fujino, A., Katai, O., Sawaragi, T.: Fuzzy Interpolation-Based Q-learning with Continuous States and Actions. Proc. of the 5th IEEE International Conference on Fuzzy Systems, Vol.1 (1996) pp. 594-600.

[13] Jenei, S.: Interpolating and extrapolating fuzzy quantities revisited – an axiomatic approach, Soft Computing, Vol. 5., 2001, 179-193.

[14] Jenei, S., Klement, E. P., Konzel, R., Interpolation and extrapolation of fuzzy quantities – The multiple-dimensional case, Soft Computing, Vol. 6., 2002, pp. 258-270.

[15] Johanyák, Zs. Cs., Tikk, D., Kovács, Sz., Wong, K. W.: Fuzzy Rule Interpolation Matlab Toolbox – FRI Toolbox. Proc. of the IEEE World Congress on Computational Intelligence (WCCI'06), 15th Int. Conf. on Fuzzy Systems (FUZZ-IEEE'06), July 16--21, Vancouver, BC, Canada, Omnipress. ISBN 0-7803-9489-5, 2006, pp. 1427-1433.

[16] Johanyák, Zs. Cs., Kovács, Sz.: Fuzzy Rule Interpolation Based on Polar Cuts, Computational Intelligence, Theory and Applications, Springer Berlin Heidelberg, 2006, pp. 499-511

[17] Johanyák, Zs. Cs., Kovács, Sz.: Fuzzy Rule Interpolation by the Least Squares Method, 7th International Symposium of Hungarian Researchers on Computational Intelligence (HUCI 2006), November 24-25, 2006 Budapest, pp. 495-506.

[18] Johanyák, Zs. Cs.: Performance improvement of the fuzzy rule interpolation method LESFRI, in Proceedings of the IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 21-22 Nov. 2011, pp. 271-276.

[19] Johanyák, Zs. Cs., Kovács, Sz.: Vague Environment-based Two-step Fuzzy Rule Interpolation Method, in Proceedings of the 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics (SAMI 2007), January 25-26, 2007 Poprad, Slovakia, pp. 189-200.

[20] Johanyák, Zs. Cs.: Vague Environment Based Set Interpolation, A GAMF Közleményei, Kecskemét, XXI. évfolyam (2006-2007), ISSN 0230-6182, pp. 33-44.

[21] José Antonio Martin H., Javier De Lope: A Distributed Reinforcement Learning Architecture for Multi-Link Robots, in the Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2007), 2007

[22] Junker, G.: Pro OGRE 3D programming, ISBN-13: 978-1-59059-710-1, Springer-Verlag New York, Inc., Apress, 2006

[23] King, P.J., Mamdani, E.H.: The application of fuzzy control systems to industrial processes, Automatica, Vol. 13, Issue 3, May 1977, pp. 235–242.

[24] Klawonn, F.: Fuzzy Sets and Vague Environments. Fuzzy Sets and Systems, 66, 1994, pp. 207-221.

[25] Kóczy, L. T., Hirota, K., Rule interpolation by α-level sets in fuzzy approximate reasoning, In J. BUSEFAL, Automne, URA-CNRS. Vol. 46. Toulouse, France, 1991, pp. 115-123.

[26] Kóczy, L. T., Hirota, K., Gedeon, T. D.: Fuzzy rule interpolation by the conservation of relative fuzziness, Technical Report TR 97/2. Hirota Lab, Dept. of Comp. Int. and Sys. Sci., Tokyo Inst. of Technology, Yokohama, 1997.

[27] Kóczy, L. T., Kovács, Sz., On the preservation of the convexity and piecewise linearity in linear fuzzy rule interpolation, Tokyo Inst. Technol., Yokohama, Japan, Tech. Rep. TR 93-94/402, LIFE Chair Fuzzy Theory, 1993.

[28] Kóczy, L. T., Kovács, Sz.: Shape of the Fuzzy Conclusion Generated by Linear Interpolation in Trapezoidal Fuzzy Rule Bases, in Proceedings of the 2nd European Congress on Intelligent Techniques and Soft Computing, Aachen, 1994, pp. 1666–1670.

[29] Kóczy, L. T., Sugeno, M.: Explicit functions of fuzzy control systems, in International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 4:515-535, 1996.

[30] Kovács, B., Száyer, G., Korondi, P., Gácsi, M., Miklósi, Á., Kovács, Sz., Vincze, D.: Ethologically Inspired Robot Design, The 2nd International Conference on Cognitive Infocommunications (CogInfoCom2011), Budapest, Hungary, July 7-9, 2011.

[31] Kovács, Sz.: New Aspects of Interpolative Reasoning. Proceedings of the 6th. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, 1996, pp. 477-482.

[32] Kovács, Sz., Kóczy, L. T.: Approximate Fuzzy Reasoning Based on Interpolation in the Vague Environment of the Fuzzy Rule base as a Practical Alternative of the Classical CRI. Proceedings of the 7th International Fuzzy Systems Association World Congress, Prague, Czech Republic, 1997, pp. 144-149.

[33] Kovács, Sz., Kóczy, L. T.: The use of the concept of vague environment in approximate fuzzy reasoning. Fuzzy Set Theory and Applications, Tatra Mountains Mathematical Publications, Mathematical Institute Slovak Academy of Sciences, Bratislava, Slovak Republic, vol.12, 1997, pp. 169-181.

[34] Kovács, Sz.: Extending the Fuzzy Rule Interpolation 'FIVE' by Fuzzy Observation, Advances in Soft Computing, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer Germany, ISBN 3-540-34780-1, 2006, pp. 485-497.

[35] Kovács, Sz.: SVD Reduction in Continuous Environment Reinforcement Learning, Lecture Notes in Computer Science, Vol. 2206, Computational Intelligence, Theory and Applications, Bernard Reusch (Ed.), Springer, Germany, 2001, ISBN 3-540-42732-5, pp. 719-738.

[36] Kovács, Sz.: Interpolative Fuzzy Reasoning in Behaviour-based Control, Advances in Soft Computing, Vol. 2, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer, Germany, 2005, ISBN 3-540-22807-1, pp.159-170.

[37] Kovács, Sz.: Interpolative Fuzzy Reasoning and Fuzzy Automaton in Adaptive System Applications, Proceedings of the IIZUKA2000, 6th International Conference on Soft Computing, October 1-4, Iizuka, Fukuoka, Japan, 2000, pp. 777-784.

[38] Kovács, Sz., Kóczy, L. T.: Application of the Approximate Fuzzy Reasoning Based on Interpolation in the Vague Environment of the Fuzzy Rulebase in the Fuzzy Logic Controlled Path Tracking Strategy of Differential Steered AGVs, Computational Intelligence - Theory and Applications, Lecture Notes in Computer Science, 1226, Springer, Germany, 1997, pp. 456-467.

[39] Krizsán, Z., Kovács, Sz.: Gradient based parameter optimisation of FRI 'FIVE', Proceedings of the 9th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, Budapest, Hungary, November 6-8, ISBN 978-963-7154-82-9, pp. 531-538, 2008.

[40] Krizsán, Z.: ICE Extension of RT-Middleware Framework. Proceedings of the 10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, Budapest, pp. 513-521., 2010

[41] Kwong, C.P.: Fuzzy Inference without Membership Function, in Future Directions of Fuzzy Theory and Systems, 1995, World Scientific Publishing, ISBN 981-02-1919-9

[42] Larsen, P. M.: Industrial application of fuzzy logic control. Int. J. of Man Machine Studies, (12) 4, 3-10., 1980

[43] Lee, J.H., Hashimoto, H.:. Intelligent Space - concept and contents, Advanced Robotics, Vol. 16, No. 3, 2002, pp. 265-280.

[44] Mamdani, E. H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. of Man Machine Studies, (7), 1-13., 1975

[45] Mesiar, R.: Triangular Norms - An Overview, Computational Intelligence in Theory and Practice, Advances in Soft Computing Vol. 8, 2001, pp. 35-54.

[46] Miklósi, Á.: Dog behaviour, evolution and cognition. Oxford University Press, 2007

[47] Mori, M.: The Uncanny Valley, Energy, 7 (4), 1970, pp. 33-35.

[48] Niitsuma, M., Beppu, W., Ichikawa, T., Kovács, Sz., Korondi, P., Hashimoto, H.: Implementation of Robot Behaviors Based on Ethological Approach for Monitoring Support System in Intelligent Space. Proceedings of The IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2011), Budapest, Hungary, July 3-7, 2011, pp. 536-541.

[49] Omlin, W., Giles, C.L., Thornber, K.K.: Equivalence in knowledge representation: automata, rnns, and dynamical fuzzy systems, Proc. IEEE 87 (9), 1999., pp. 1623–1640.

[50] Pirjanian, P.: Behavior Coordination Mechanisms - State-of-the-art, Tech-report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, October, 1999

[51] Potal, M.: Overcoming the uncanny valley. IEEE Computer Graphics and Applications, 4 11-17. , 2008

[52] Precup, D., Sutton, R. S., Dasgupta, S.: Off-policy temporal-difference learning with function approximation. Proceedings of the 18th International Conference on Machine Learning, 2001. pp. 417-424.

[53] Rummery, G. A., Niranjan, M.: On-line Q-learning using connectionist systems. CUED/F-INFENG/TR 166, Cambridge University, UK., 1994

[54] Shepard, D.: A two dimensional interpolation function for irregularly spaced data. Proc. 23rd ACM International Conf., 1968, pp. 517-524.

[55] Sugeno, M.: An introductory survey of fuzzy control. Information Science, (36), 1985, pp. 59-83.

[56] Sutton, R. S., Barto, A. G.: Reinforcement Learning: An Introduction, MIT Press, Cambridge, 1998

[57] Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on SMC, (15), 1985, pp. 116-132.

[58] Tesauro, G.: Temporal difference learning and TD-Gammon, in Communications of the ACM, 1995, 38.3: pp. 58-68.

[59] Tikk, D.: Notes on the approximation rate of fuzzy KH interpolator. Fuzzy Sets and Systems, 138(2), Sept., 2003, pp. 441–453.

[60] Tikk, D., Baranyi, P.: Comprehensive analysis of a new fuzzy rule interpolation method, In IEEE Trans. Fuzzy Syst., vol. 8, No. 3, June, 2000, pp. 281-296.

[61] Tikk, D., Baranyi, P., Gedeon, T. D., Muresan, L., Generalization of a rule interpolation method resulting always in acceptable conclusion, Tatra Mountains Math. Publ., 21, 2001, pp. 73-91.

[62] Tikk, D., Joó, I., Kóczy, L. T., Várlaki, P., Moser, B., Gedeon, T. D.: Stability of interpolative fuzzy KH-controllers, Fuzzy Sets and Systems, 125(1), January, 2002, pp. 105-119.

[63] Topál, J., Miklósi, Á., Csányi, V., Dóka, A.. 1998. Attachment behavior in dogs (Canis familiaris): A new application of Ainsworth's (1969) Strange Situation Test. Journal of Comparative Psychology, Vol. 112(3), pp. 219-229.

[64] Vámos, A., Reskó, B., Baranyi. P. 2010. Virtual Collaboration Arena, IEEE 8th Slovakian-International Symposium on Applied Machine Intelligence, Herl'any, Slovakia, pp. 159-164.

[65] Vass, G., Kalmár, L., Kóczy, L. T.: Extension of the fuzzy rule interpolation method, in Proc. Int. Conf. Fuzzy Sets Theory Applications (FSTA'92), Liptovsky M., Czechoslovakia, 1992, pp. 1-6.

[66] Watkins, C. J. C. H.: Learning from Delayed Rewards. Ph.D. thesis, Cambridge University, Cambridge, England (1989)

[67] Watkins, C. J. C. H., Dayan, P.: Q-learning, in Machine Learning, Vol. 8 (3/4), 1992., pp. 279-292.

[68] Wong, K. W., Gedeon, T. D., Tikk, D.: An improved multidimensional α-cut based fuzzy interpolation technique, In Proc. Int. Conf Artificial Intelligence in Science and Technology (AISAT'2000), Hobart, Australia, 2000, pp. 29-32.

[69] Wong, K. W., Tikk, D., Gedeon, T. D., Kóczy, L. T.: Fuzzy Rule Interpolation for Multidimensional Input Spaces With Applications, IEEE Transactions on Fuzzy Systems, ISSN 1063-6706, Vol. 13, No. 6, December, 2005, pp. 809-819.

[70] Yam, Y., Kóczy, L. T.: Representing membership functions as points in high dimensional spaces for fuzzy interpolation and extrapolation, Dept. Mech. Automat. Eng., Chinese Univ. Hong Kong, Tech. Rep.CUHK-MAE-97-03, 1997.

[71] Zadeh, L. A.: Fuzzy Sets, in Information and Control, Vol. 8, 1965, pp. 338-353.

[72] Zadeh, L. A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. on SMC, (3), 1973, pp. 28-44.

[73] MathWorks: MATLAB Documentation: Fuzzy Logic Toolbox: chapter: Fuzzy Logic Toolbox Examples, section: Membership Function Gallery

[74] The cart-pole example for discrete space can be found at: http://www.dia.fi.upm.es/~jamartin/download.htm

[75] The MATLAB FRI Toolbox is available at: http://fri.gamf.hu/

[76] The modified FIVE implementation can be found at: http://fri.gamf.hu/ and http://www.iit.uni-miskolc.hu/~vinczed/

Author's publications and independent citations

[77] Kovács, Sz., Vincze, D., Gácsi, M., Miklósi, Á., Korondi, P.: Interpolation based Fuzzy Automaton for Human-Robot Interaction, Preprints of the 9th International Symposium on Robot Control (SYROCO'09), The International Federation of Automatic Control (IFAC), Nagaragawa Convention Center, Gifu, Japan, 2009, pp. 451-456.

Cited by: (2 independent)

Solvang, B., Sziebig, G.: On industrial robots and cognitive info-communication, In: IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom). Kosice, Slovakia, 2012, pp. 459-464.

Niitsuma, M., Numakunai, R., Onodera, A.: Tuning of behavioral characteristics in an ethologically inspired robot behavior model based on verbal communication. IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 2013, pp. 7855-7861.

[78] Kovács, Sz., Vincze, D., Gácsi, M., Miklósi, Á., Korondi, P.: Fuzzy automaton based Human-Robot Interaction, IEEE 8th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, January 28-30, ISBN 978-1-4244-6422-7, 2010, pp. 165-169.

Cited by: (1 independent)

Kifor, T., Gottdank, T., Hajnal, Á.: Ethology and Mobile Technology in One: EtoPhone, In: CogInfoCom 2011. 2nd international conference on cognitive infocommunications. 2011, Budapest, Hungary

[79] Kovács, Sz., Gácsi, M., Vincze, D., Korondi, P., Miklósi, Á.: A novel, ethologically inspired HRI model implementation: Simulating dog-human attachment, 2nd IEEE International Conference on Cognitive Infocommunications: CogInfoCom 2011, Budapest, Hungary

Cited by: (1 independent)

Devecseri, V., Farkas, Z., Bartok, A., Halachy, N., Samu, D.: The technology behind PhotoRobot, an interactive Kinect-based installation, In: 13th IEEE International Symposium on Computational Intelligence and Informatics (CINTI). Budapest, Hungary, 2012, pp. 401-404.

[80] Kovács, Sz., Vincze, D., Gácsi, M., Miklósi, Á., Korondi, P.: Ethologically inspired robot behavior implementation, 4th International Conference on Human System Interactions (HSI 2011), Yokohama, Japan, 19-21 May 2011, (ISSN 2158-2246, ISBN 978-1-4244-9638-9, doi: 10.1109/HSI.2011.5937344), 2011, pp. 64-69.

Cited by: (1 independent)

Rodic, A., Mester, G.: Sensor-based Navigation and Integrated Control of Ambient Intelligent Wheeled Robots with Tire-Ground Interaction Uncertainties, Acta Polytechnica Hungarica Vol. 10: (3) 113-133 (2013)

[81] Vincze, D., Kovács, Sz.: Reduced Rule Base in Fuzzy Rule Interpolation-based Q-learning, 10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, CINTI 2009, November 12-14, 2009, Budapest Tech

[82] Vincze, D., Kovács, Sz.: Fuzzy Rule Interpolation-based Q-learning, SACI 2009, 5th International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, May 28-29, 2009, ISBN: 978-1-4244-4478-6, pp. 55-59.

Cited by: (4 independent)

Niu, J., Deng, Z.: Distributed self-learning scheduling approach for wireless sensor network, In Ad Hoc Networks (2010)

Niu, J.: Evolutionary self-learning scheduling approach for wireless sensor network, In: 2010 International Conference on Intelligent Computation Technology and Automation, ICICTA 2010. Changsha, China, 2010, pp. 245-249.

Niu, J.: Self-learning scheduling approach for wireless sensor network, In: Proceedings of the 2010 2nd International Conference on Future Computer and Communication, ICFCC 2010. Wuhan, China, 2010, pp. 3253-3257.

Kumar, R., Nigam, M. J., Sharma, S., Bhavsar, P.: Temporal Difference based Tuning of Fuzzy Logic Controller through Reinforcement Learning to Control an Inverted Pendulum, 2012, International Journal of Intelligent Systems and Applications (IJISA) 4: (9) pp. 15-21.

[83] Vincze, D., Kovács, Sz.: Behaviour Based Control with Fuzzy Automaton in Vehicle Navigation, Production Systems and Information Engineering, Volume 5 (2009), University of Miskolc, Hungary, HU ISSN 1785-1270., pp. 151-166.

[84] Vincze, D., Kovács, Sz.: Incremental Rule Base Creation with Fuzzy Rule Interpolation-Based Q-Learning, I. J. Rudas et al. (Eds.), Computational Intelligence in Engineering, Studies in Computational Intelligence, Volume 313/2010, Springer-Verlag, Berlin Heilderberg, 2010, pp. 191-203.

Cited by: (6 independent)

Johanyák, Zs. C., Papp, O.: Comparative analysis of two fuzzy rule base optimization methods, In: SACI 2011 - 6th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings. Timisoara, Romania, 2011, pp. 235-240.

Johanyák, Zs. Cs.: Performance improvement of the fuzzy rule interpolation method LESFRI, In: 12th IEEE International Symposium on Computational Intelligence and Informatics, CINTI 2011 - Proceedings. Budapest, Hungary, 2011, pp. 271-276.

Johanyák, Zs. Cs., Papp, O.: A Hybrid Algorithm for Parameter Tuning in Fuzzy Model Identification, Acta Polytechnica Hungarica Vol. 9: (6) 153-165 (2012)

Johanyák, Zs. Cs., Papp, O.: Benchmark Based Comparison of Two Fuzzy Rule Base Optimization Methods, In: Topics in Intelligent Engineering and Informatics. Springer Berlin Heidelberg, 2012. pp. 83-94.

Johanyák, Zs. Cs.: Clonal Selection Based Parameter Optimization for Sparse Fuzzy Systems, In: IEEE 16th International Conference on Intelligent Engineering Systems (INES 2012). Lisbon, Portugal, 2012, pp. 369-373.

Johanyák, Zs. Cs.: Fuzzy Modeling of Thermoplastic Composites' Melt Volume Rate, COMPUT INFORM 32: (4) pp. 845-857 (2013)

[85] Vincze, D., Kovács, Sz., Baranyi, P. 2011. Interconnecting the Spatial Eto-Motor and the VirCA Environment. Proceedings of the 2nd International Conference on Cognitive Infocommunications (CogInfoCom2011), Budapest, Hungary, July 7-9.

[86] Vincze, D., Kovács, Sz., Gácsi, M., Korondi, P., Miklósi, Á., Baranyi, P.: A Novel Application of the 3D VirCA Environment: Modeling a Standard Ethological Test of

Dog-Human Interactions, Acta Polytechnica Hungarica, Journal of Applied Sciences, Vol. 9. No. 1., 2012, Óbuda University, Budapest, Hungary, ISSN 1785-8860, pp. 107-120. ***(IF=0.588)***

Cited by: (4 independent)

Sziebig, G. , Øritsland, T.A.: Navigating in 3D immersive environments: A VirCa usability study, In: Proceedings of 10th IFAC Symposium on Robot Control, SYROCO 2012, Dubrovnik, Croatia, pp.380-384.

Aryania, A., Daniel, B., Thomessen, T., Sziebig, G.: New trends in industrial robot controller user interfaces, In: IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom). Kosice, Slovakia, 2012., pp. 365-369.

Devecseri, V., Dóka, A., Molnár, J., Tamás, P.: An ethological motion capture system, In: 12th IEEE International Symposium on Computational Intelligence and Informatics, CINTI 2011 - Proceedings. Budapest, Hungary, 2011., pp. 487-491.

Devecseri, V.: Motion Capture System for Ethological Observation, In: Sixth Hungarian Conference on Computer Graphics and Geometry, Budapest, Hungary, 2012., pp. 23-29.

[87] Vincze, D., Kovács, Sz., Niitsuma, M., Hashimoto, H., Korondi, P., Gácsi, M., Miklósi, Á.: Ethologically inspired human-robot interaction interfaces, HCCE '12 Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments, Hamamatsu, Japan, 2012. ISBN: 978-1-4503-1191-5, pp. 51-57.

[88] Vincze, D., Kovács, Sz.: Performance Issues of the Implemented FRI 'FIVE' Proceedings of the 11th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, CINTI 2010, November 18-20, 2010, Óbuda University, ISBN 978-1-4244-9278-7, pp. 131-136.

[89] Vincze, D., Kovacs, S., Korondi, P., Baranyi, P.: A simple interface to the Virtual Collaboration Arena for MATLAB applications, IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI 2011), Budapest, Hungary

[90] Vincze, D., Kovács, Sz.: Extending Fuzzy Q-learning with Fuzzy Rule Interpolation Method "FIVE", Scientific Bulletin of "Politehnica" University of Timisoara, Romania, Transactions on Automatic Control and Computer Science, vol. 54(68) No. 4 / 2009, pp. 173-178. ISSN 1224-600X.

[91] Vincze, D., Kovács, Sz.: Performance Optimization of the Fuzzy Rule Interpolation Method 'FIVE', Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol. 15. No. 3., Special issue on Fuzzy Rule Interpolation, 2011, Fuji Technology Press, Tokyo, Japan, ISSN 1343-0130, pp. 313-320.

[92] Vincze, D., Kovács, Sz.: Using Fuzzy Rule Interpolation-based Automata for Controlling Navigation and Collision Avoidance Behaviour of a Robot, In Proceedings of the 6th IEEE International Conference on Computational Cybernetics, Stará Lesna, Slovakia, 2008, pp. 79-84.

Cited by: (1 independent)

Johanyák, Z. Cs., Berecz, A.: Survey on Practical Applications of Fuzzy Rule Interpolation, In Proceedings of the 1st International Scientific and Expert Conference TEAM 2009, Slavonski Brod, Croatia, 2009, pp. 205-213.

This list refers only to the publications related to this dissertation. The overall publication statistics are the following: Publications: 25 Independent citations: 20, Impact factor: 0.588.