Soft Computing Techniques for the Design of Mobile Robot Behaviours

FRANK HOFFMANN
Berkeley Initiative in Soft Computing (BISC)
Computer Science Division
University of California, Berkeley CA 94720
email: fhoffman@cs.berkeley.edu

ABSTRACT

This paper describes a design method for mobile robot behaviours that employs a variety of soft computing techniques. Evolutionary algorithms are used to learn a wall following behaviour implemented by means of fuzzy control rules. A messy coding scheme for fuzzy rules is able to bound the size and complexity of the rule base, with the result, that the task of designing the fuzzy controller remains tractable for the genetic algorithm. An evolutionary strategy tunes the scaling factors for the controller's input and output and optimizes the adjustment of the sensors on the robot. A neural network analyzes the sensor data in order to provide the robot with additional information on its environmental context.

Keywords: genetic algorithms, fuzzy control, mobile robot, soft computing

1. INTRODUCTION

Designing controllers for mobile robots by hand becomes a difficult task as soon as the behaviour becomes more complex. In many applications the robot's environment changes with time in a way that is not predictable by the designer in advance. In addition, the information available about the environment is subject to imprecision, incompleteness and imperfection due to the limited perceptual quality of the sensors. These problems limit the utility of traditional model-based reasoning approaches for the design of intelligent robots.

1.1. Evolutionary Computation

Evolutionary computation provides an alternative design method that adapts the robot's behaviour without requiring a precisely specified model of the world. Its adaptive power enables the robot to deal with changes

655 Avenue of the Americas, New York, NY 10010 0020-0255/94/$6.00

in the environment and to acquire a robust behaviour tolerating noisy and unreliable sensor information.

Evolutionary algorithms constitute a class of search and optimization methods guided by the principles of natural evolution [15]. The major distinctions between the methods belonging to this class originate from the genetic representation of a candidate solution to the optimization problem, the genetic operators employed to generate new offspring and the selection scheme used to determine the parents for reproduction. Genetic algorithms normally work with binary strings and their progress is due to the harmonized co-operation of selection and crossover in the evolutionary progress [10]. In evolutionary strategies candidate solutions are directly represented by real-valued vectors [2]. Individuals contain an additional vector of strategy parameters that is subject to optimization as well. This self-adaptation provides the evolutionary strategy with additional flexibility by tuning the evolutionary process itself. Mutation is the most important search operator in evolutionary strategies, whereas recombination plays only a minor role. In genetic programming a population is comprised of computer programs structured in a hierarchical composition of functions and terminals [17].

Evolutionary computation plays an important role in modeling and designing artificial-life systems [19]. Essentially, artificial life and machine learning understand evolutionary algorithms as an abstraction of natural evolution in that they adapt a system to its environment. "Evolutionary robotics" is concerned with the design of systems with lifelike properties by optimizing the robot's controller with respect to some objectives. The essential distinction of the proposed methods is the way in which the evolutionary algorithm represents the controller in the genotype. A variety of different methods, such as neural nets [20], tree-structured programs [21], classifier systems [8], stimulus-response rules [11] and fuzzy control rules [4][5][13][18] [25], are used for the implementation of the robotic behaviour.

1.2. Soft Computing

Fuzzy systems employ a mode of approximate reasoning, which allows them to make decisions based on imprecise and incomplete information in a way similar to human beings. A fuzzy system offers the advantage of knowledge description by means of linguistic concepts without requiring the complexity and precision of mathematical or logical models. Fuzzy control provides a flexible tool to model the relationship between input information and control output and is distinguished by its robustness with respect to noise and variation of system parameters. Among many other applications, fuzzy logic controllers (FLC) are applied to the control of

mobile robots [3][22][23]. Sensor inputs are mapped to control actions by means of fuzzy rules constituting the robot's behaviour.

Soft computing is concerned with the design of intelligent and robust systems, which exploit the tolerance for imprecision inherent in many real world problems[26]. In order to achieve this objective, soft computing suggests a combination of fuzzy logic, neural networks, probabilistic reasoning and genetic algorithms. Soft computing advocates that the integration of these complementary methodologies, each of them adequate for its specific domain of problems, results in more powerful hybrid methods than using a single method exclusively.

Recently, numerous researchers explored the integration of evolutionary algorithms with fuzzy systems in so-called genetic fuzzy systems[1][7][12][16][24]. The majority of publications are concerned with the automatic design or optimization of fuzzy logic controllers either by adapting the fuzzy membership functions or by learning the fuzzy if-then rules. The first method results in a self-tuning controller and is primarily useful in order to optimize the performance of an already existing controller. The second approach is a self-organizing process that learns the appropriate relation between control input and output starting without any previous knowledge.

1.3. Evolutionary Design of Mobile Robot Behaviours

Promising results have been achieved by employing evolutionary methods to develop rule based behaviours for mobile robots. A significant part of the engineering task of designing an intelligent robot is delegated to the evolutionary algorithm, which explores alternative behaviours and optimizes the controller's parameters. Bonarini et al. proposed an automatic design method called S-ELF (Symbolic Evolutionary Learning of Fuzzy Rules) to learn basic robotic behaviours and to coordinate their activation [4]. Dorigo et al. employed a learning classifier system to adapt behavioural patterns of a mobile robot such as light following, searching food and avoiding predators [8]. Tunstel et al. used genetic programming to learn fuzzy control rules for mobile robot path tracking[25]. Braunstingl et al. optimized a fuzzy controller for a wall following behaviour of a mobile robot by means of a genetic algorithm [5]. Grefenstette et al. applied their genetic learning system SAMUEL to a collision avoidance and navigation task for mobile robots [11]. Leitch proposed a genetic algorithm based on a context dependent coding scheme in order to learn a fuzzy controller for a mobile robot given the tasks of wall following, turning in a corridor and parallel parking [18].

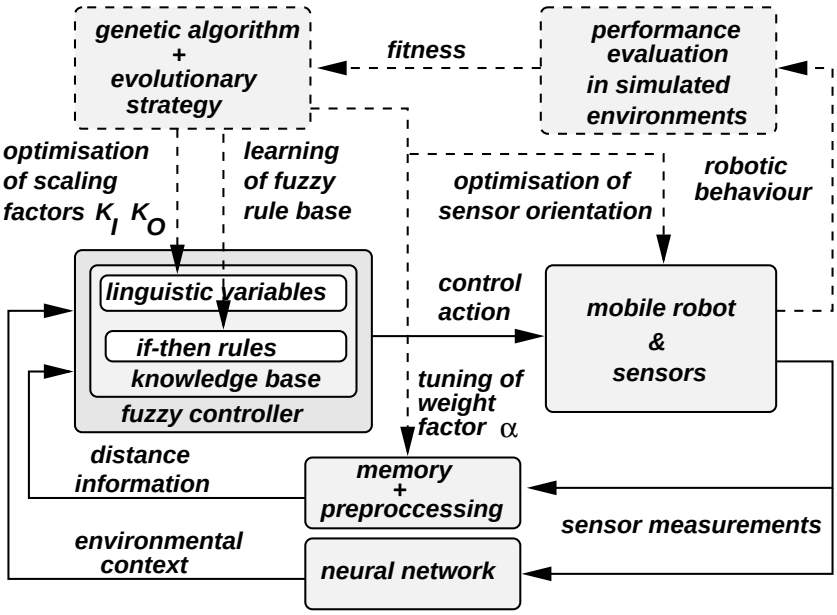

Fig. 1. System architecture

2. System Architecture

Following the guideline of soft computing we integrated different methodologies into a hybrid system in order to design the robotic behaviour. Fig.1 shows the role of each of the soft computing partners, fuzzy control, genetic algorithm, evolutionary strategy and neural network. The evolutionary learning process marked by the dashed lines takes place in a simulation of the robot and its environment. The solid lines represent the components of the system that are transferred to the real robot after the learning phase is concluded.

The behaviour of the mobile robot is implemented by means of fuzzy control rules. Fuzzy controller employ a method of approximate reasoning which allows them to find a suitable control action even if the information about the environment is imprecise and incomplete. The relationship between perceived control input and desired control action is stored in the knowledge base constituted by the linguistic variables and the fuzzy if-then rules.

Based on the performance criteria specified by the human designer the quality of a controller is evaluated by simulating the robotic behaviour in

Fig. 2. Mobile robot

a set of training environments. The purpose of the genetic algorithm is to learn the fuzzy if-then rules that are suitable to perform the task of wall-following. In addition, the evolutionary strategy tunes the scaling factors of control input and output and optimizes the adjustment of the sensors in order to achieve an optimal perception of the environment. The neural network classifies the environmental context based on the measurements provided by the ultrasonic sensors.

2.1. Mobile Robot

The mobile robot depicted in Fig. 2 is driven by two independent stepping motors which enable the controller to keep track of the robot's position. It perceives its environment by means of five ultrasonic sensors, each of them covers an angular range of approximately $\pm 20°$. The probability of omitting an object increases at the borders of the perceptual field. The orientation of the sensors on the robot is adjustable. An optimal perception of obstacles is achieved by a compromise between a sufficient overlap of the individual sensors and a wide total field of vision. The robot is used as a testbed for the evolutionary design method for fuzzy logic controllers proposed in the following. The given task is to learn a wall following behaviour which includes to avoid collisions of the robot with the walls and to perform an adequate turning manœuvre in dead-end situations.

2.2. Neural Network

The purpose of the feed-forward neural network shown in Fig. 3 is to classify different environmental situations such as dead-ends, corners and walls.

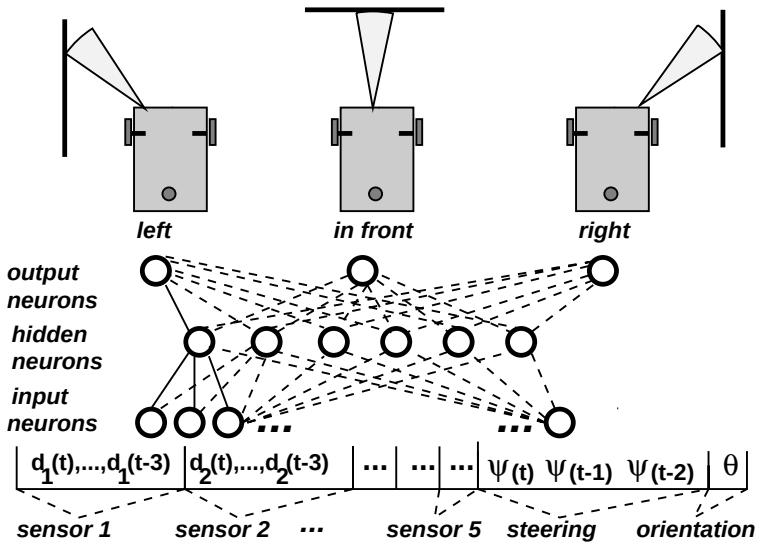

Fig. 3. Neural network for the classification of the environmental context

The neural network incorporates previous sensor data and steering angles in its analysis, since the environmental context of the robot can not be inferred solely from the current set of sonar measurements. The neural network obtains the last four distance measurements $d_i(t), \ldots, d_i(t-3), \{i = 1, \ldots, 5\}$ of the five sensors $s_1, \ldots, s_5$ as input. If an object closer than $1.5m$ is detected by the sensor s_i at time $t-n$, the input $d_i(t-n)$ becomes 1, otherwise it is 0. Additional input neurons contain the four previous steering angles of the robot $\psi(t), \ldots \psi(t-3)$, and the orientation of the sensors θ as depicted in Fig. 8. The input is mapped to three output neurons, which become active, if an obstacle appears to the left, in front of or to the right of the robot. The network is able to distinguish among eight different basic situations, starting with no obstacle detected at all and ending with a dead-end in which case all three output neurons become active at the same time. The network was trained with the back-propagation algorithm using a training set of 4000 input-output pairs generated by means of a simplified sensor model.

2.3. Fuzzy Controller

Compared to optical sensors, sonar distance information is much more afflicted with incompleteness and imprecision because of the physical char-

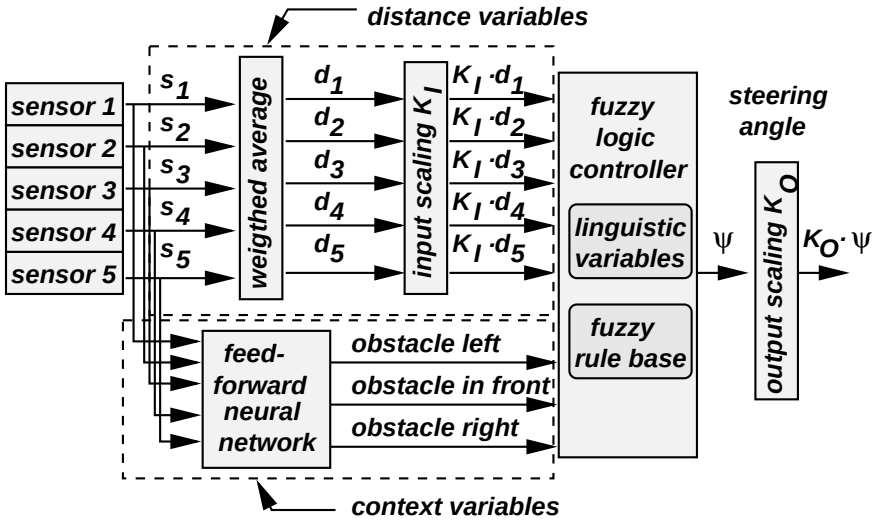

Fig. 4. Control scheme of the mobile robot

acteristics of ultrasonic signals and varying reflection properties for different kinds of objects. A sensor sometimes fails to detect an object, especially if it is located at the borders of its perceptual field. Even if the sensor information is correct, a single set of sonar measurements does not reveal enough information to determine the entire state of the environment. As a result, former controllers turned out to be unable to detect dead-ends in a reliable way [13]. In order to cope with real world situations the agent has to exhibit more than a purely reactive behaviour based on the current sensor stimuli. The agent requires some kind of memory to take previous sensor information into account for its decision making on the next control action.

Fig. 4 shows the control scheme of the mobile robot of which behaviour is implemented by means of fuzzy control rules. The fuzzy controller employs five *distance variables* $d_1, \ldots, d_5$ containing preprocessed distance information which takes previous sensor measurements into account. Ultrasonic data is afflicted with a large amount of uncertainty. For example echo signals are not detected by the sonar receiver if they are not reflected perpendicular to the object. As a result, a single sonar measurement may fail to reveal the presence of an object that was detected by the same sensor one time step ago. Due to multiple reflections sonar sensors sometimes overestimate the true distance to an object. In our case the obstacles are static and the robot moves at a relatively slow speed. Therefore, the crisp

value of each distance variable d_i is computed as a weighted average over previous sensor values $s_i(t)$ neglecting those measurements that received no echo signal.

$$d_i = \sum_{n=0}^{N} \alpha^n e_i(t-n) s_i(t-n) / \sum_{n=0}^{N} \alpha^n e_i(t-n) \tag{1}$$

where $e_i(t) \in \{0, 1\}$ is 1 if the sensor s_i received an echo, at time t and 0 if no signal was detected. The weight factor $\alpha \in [0, 1]$ determines the significance allocated to previous sensor measurements. The optimal value of α is a compromise between compensating the impact of incorrect sensor data and exploiting the most up-to-date information.

Each distance variable is subdivided into four trapezoidal fuzzy sets and an additional fuzzy term in case no echo signal is detected by the sensor. As depicted in Fig. ??, the distance variables are scaled by a common gain factor K_I before they are provided as input to the controller. The steering angle ψ serves as the control output. It is inverse proportional to the turning radius and defines the way in which the robot moves in the next timestep. The crisp value of ψ is also multiplied by an output gain factor K_O. An evolutionary strategy is employed in order to find the optimal values for the real valued parameters α, K_I, K_O.

In addition to the *distance variables* the controller obtains supplementary information on the environmental situation provided by the three output neurons of the neural network depicted in Fig. 3. The fuzzy controller uses these *context variables* in order to define situation dependent constraints on the rule antecedents. The idea is, that each behaviour has its own domain of applicability and that the corresponding rule is only activated in the appropriate context [22]. For example, the rules to execute a turning manœuvre are only applied in a dead-end situation. Each of the three context variables contains two linguistic terms, corresponding to a firing and non-firing output neuron.

3. Evolutionary Algorithm

Evolutionary algorithms process a population of competing candidate solutions from one generation to the next [2][10][15]. Each individual is represented by a set of parameters called a genotype. The algorithm evaluates the quality of an individual in regard to the optimization task by means of a scalar fitness function. According to Darwin's principle, highly fit individuals obtain a better chance to reproduce offspring to the next generation. Genetic operators such as crossover and mutation are applied

to the selected parents in order to generate new candidate solutions. As a result of this evolutionary cycle of selection, crossover and mutation more and more suitable solutions to the optimization problem emerge within the population.

3.1. Messy Coding Scheme

The choice of an appropriate genetic representation of candidate solutions plays a crucial role in the design of an evolutionary algorithm. In a conventional coding scheme for fuzzy rule bases the chromosome contains one output term for every possible combination of statements in the antecedent. The number of rules and thereby the size of the chromosome increase rapidly with the number of input variables, with the result that the evolutionary optimization becomes less and less feasible.

CHOWDURY ET AL. proposed a messy genetic algorithm which optimizes the structure of a neuro-fuzzy controller [6]. LEITCH developed a *context dependent coding scheme* for fuzzy logic controllers in which the meaning of a gene is not determined by its absolute position but depends on the surrounding genes [18]. This paper presents a new messy coding scheme for fuzzy rules and rule base. The coding is able to bound the size and complexity of the rule base, especially for a larger number of input variables. Therefore, the genetic algorithm is able to maintain the efficiency, robustness and comprehensiveness of the fuzzy controller. A fuzzy rule base is constituted by a smaller number of fuzzy rules, which correspondingly encompass larger regions of input space. Due to the more compact genetic representation, the task of designing the fuzzy controller remains tractable for the genetic algorithm.

The proposed coding scheme imitates the genetic representation used in messy genetic algorithms proposed by GOLDBERG ET AL.[9]. The meaning of a gene is part of its coding instead of being defined by its position within the chromosome. This property enables the formation of chromosomes with variable length in which genes can be arranged in any order. A messy genetic algorithm has more freedom to generate efficient building blocks, since the coding scheme considers both the allele value and its position. Adapting the gene ordering enables the genetic algorithm to find the optimal linkage format, by minimizing the defining length of highly fit schemata.

First of all, the linguistic variables and terms are numbered by integers without distinguishing between input and output variables. In the example depicted in Fig. 5, the input variables *direction*, *distance* and the output variable *steering* are associated to the integers $1, 2, 3$ marked by the boxes

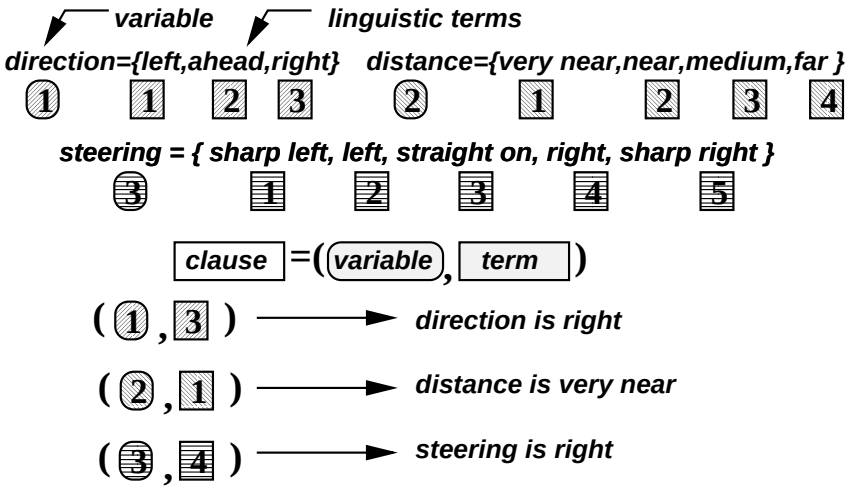

Fig. 5. Enumeration of input and output variables and their linguistic terms, genetic representation of fuzzy statements

with rounded corners. Each variable is composed of several linguistic terms which are numbered consecutively as well, marked by the rectangular boxes in Fig. 5.

Fuzzy statements of the form *variable is term* constitute the basic elements of our coding scheme. In the genotype, a fuzzy statement is represented by a single gene build of two integer values. The first integer value defines the fuzzy variable, the second value specifies the associated linguistic term. For reasons of clarity the boxes in Fig. 5,6,7 are filled with different patterns, whereby each pattern represents a specific variable or rule. These patterns are not used in the actual coding, in which the meaning of a gene is determined by the first integer value and the surrounding genes. In the first example depicted in Fig. 5 the gene $(1,3)$ encodes the fuzzy statement *direction is right*, in which the integer 1 defines *direction* as the variable, and the value 3 represents the associated linguistic term *right*.

Chromosomes encoding fuzzy rules, are formed by any arbitrary sequence of statement genes as shown in Fig. 6. The first example with the three integer pairs $(2,1),(3,1),(1,3)$ contains exactly one gene for every variable and represents the fuzzy rule *if direction is right and distance is very near then steering is sharp left*. This rule might also be encoded by the sequence $(3,1)(1,3)(2,1)$ build from the same basic genes arranged in a different order.

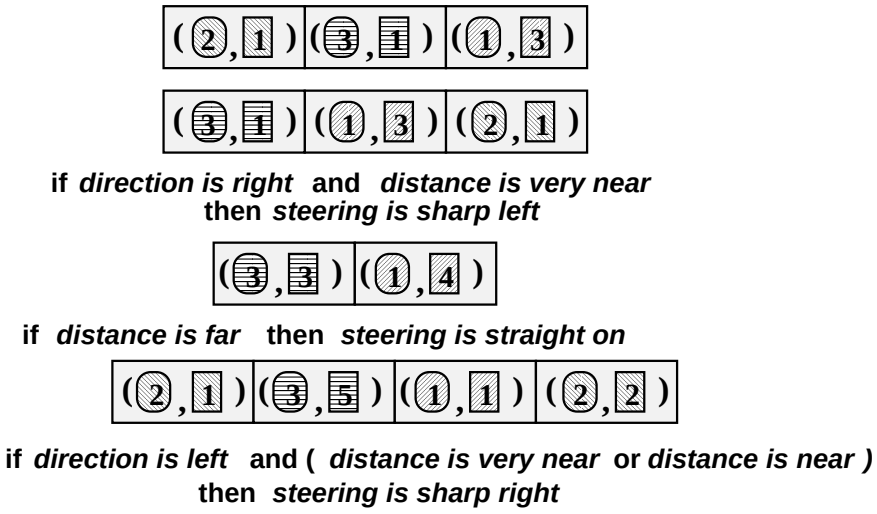

Fig. 6. Genetic representation of fuzzy rules

In a messy coding scheme, the size of a chromosome is variable, with the result that a chromosome might contain multiple duplicates of the same gene having different allelic values. In case the chromosome is lacking a specific gene, the messy genetic algorithm has to deal with incomplete genetic information when generating the phenotype. The chromosome $(3,3)(1,4)$ is an example of such an underspecification, since it contains no gene for the variable *direction* with an integer 2 in the first position. In this situation, the coding scheme considers *direction* as a do-not-care variable of which the corresponding fuzzy statement is omitted in the antecedent. More general rules with shorter antecedents like *if distance is far then steering is straight on* encompass larger regions of input space. Using these type of compact, general rules enables the genetic algorithm to reduce the size and complexity of the rule base. Fuzzy rules which are lacking a conclusion to the output variable become meaningless. Therefore, the operators for the initialization and the recombination of rule genes are slightly modified in order to guarantee, that each rule chromosome contains at least one statement gene for the output variable.

In a messy coding scheme the dual problem to underspecification is overspecification, which arises whenever multiple genes for with conflicting linguistic terms for the same variable occur in the chromosome. In the last example in Fig. 6, the chromosome $(2,1)(3,5)(1,1)(2,2)$ possesses two conflicting genes $(2,1)$, $(2,2)$ for the input variable *distance*. Both of their

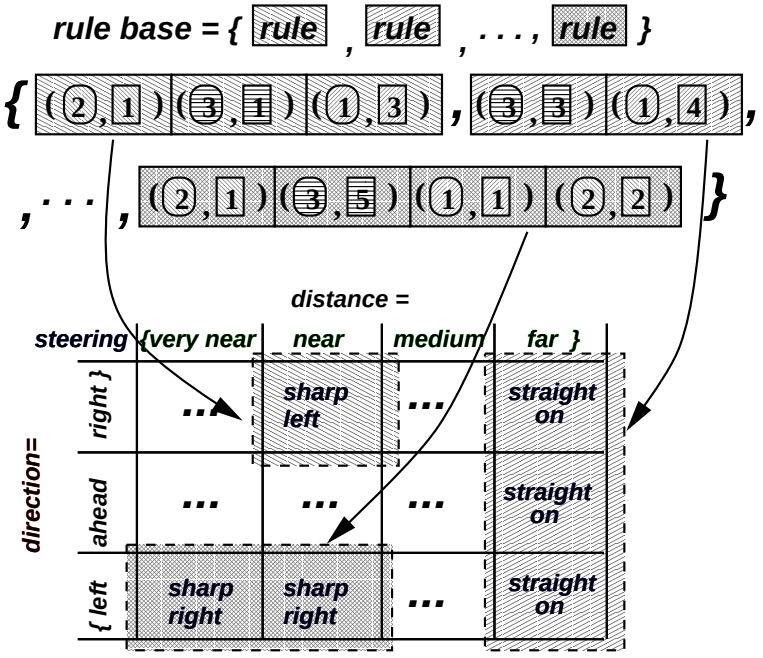

Fig. 7. Genetic representation of the fuzzy rule base

corresponding statements *distance is very near* and *distance is near* are comprised in the conclusion of the rule by combining them with a fuzzy OR operation.

Finally, the chromosome representing the entire rule base is formed by a set of rule genes as depicted in Fig. 7. Due to the flexible genetic representation of antecedents, some of the rules encompass more than one entry of the rule matrix or even correspond to entire rows or columns, which is helpful to reduce the overall number of rules. It can be easily understood, that such a compact representation of the rule matrix becomes even more efficient in case of more than only two input variables. For example, the fuzzy controller for the mobile robot depicted in Fig. 4 employs eight input variables resulting in a total of number 25.000 entries in the rule matrix. Normally, the rule bases adapted by means of the messy coding scheme manage the control task with less than 200 rules.

The coding scheme for the entire rule base requires no additional enumeration of genes since the order of rules does not matter. Nevertheless, the twin problems of over- and underspecification emerge for the rule base coding as well. The chromosome is overspecified if several rules with dif-

ferent conclusions apply to the same input situation. A fuzzy controller that employs the usual inference scheme computes the control output from the interpolation of the control actions suggested by the firing rules. The decision making of the fuzzy controllers presented in this paper is modified in a way that rules with more specialized antecedents receive priority to general rules with only a few input statements. The conclusion suggested by a general rule is ignored in favour to the output of a more complex rule that becomes active at the same time. The usual inference mechanism is employed in case of a conflict among rules having the same level of complexity.

Underspecification occurs, if none of the rules match the encountered control situation. In this case, a rule generation process is put into operation, which inserts a new rule gene of which antecedent matches the current input. During the initialization of the population the new rule acquires a randomly chosen control action. In later generations the rule base is completed by copying the gene from another member of the population that possesses the missing rule and achieved a high fitness on the same training situation. The gene segment of the new rule is inserted into the chromosome next to the gene of the rule that became active in the previous control step. This mechanism ensures that the genes of rules which are activated one after another in time are adjacent in the chromosome. The result is a tight linkage among genes which are assumed to have high epistasis, since the overall performance of the controller strongly depends on the co-operation of rules adjacent in the rule matrix.

In addition to a modified coding scheme, messy genetic algorithms also employ different genetic operators for recombination [9]. The usual crossover operator is replaced by cut and splice operations. Cut and splice operators are applied on the level of rules, as well as on the level of the rule base.

Cut and splice operations on the rule level modify the rule antecedent by incorporating new or deleting existent input statements inherited from the parents. A matching procedure compares the statement genes of the two parent rules, so that only rules bearing a minimal resemblance in their antecedent and conclusion part are mated. The offspring rule might become more specific than its parents if it inherits new input variables. On the other hand a rule might become more general if it inherits an additional statement corresponding to an already present input variable, or looses an input variable in its antecedent. Sometimes, cut and splice result in an offspring containing no gene for the output variable. In this case, the chromosome is "repaired" by subsequently adding the missing gene from one of the parents.

Cut and splice operations are also applied on the rule base level, on which they recombine entire sets of fuzzy rules. The idea is, that two par-

ents which have already adapted suitable control rules for different regions of input space transmit their complementary knowledge to the common offspring. After the recombination, the offspring is subject to a rule base completion procedure which prevents the loss of important parent rules.

As a result of a cut and splice operation, chromosomes might grow or shrink in size which enables the genetic algorithm to adapt the number of rules and the number of statements in the antecedent to the complexity of the control problem. Instead of optimizing all entries of the entire rule matrix at once, the genetic algorithm starts with a few general rules which in the course of evolution are gradually refined and supplemented by more specific rules. The cut and splice operators as well as the mutation operator are described in more detail in [12].

3.2. Evolutionary Strategy

Evolutionary strategies are a class of evolutionary algorithms that optimize a vector of real-valued parameters, which makes them especially useful for continuous optimization problems [2]. The real-valued representation of candidate solutions avoids additional multimodality of the fitness landscape sometimes introduced by the conventional binary representation used by genetic algorithms. The mutation operator in evolutionary strategies adds a normally distributed random vector with zero mean to the parameter vector of the candidate solution. Each individual contains an additional vector of strategy parameters specifying the variance of the mutation applied to the usual variables. These strategy parameters are subject to mutation and recombination just as the usual parameters. This mechanism of self-adaptation enables the evolutionary strategy to find those meta parameters that have an advantageous impact on the evolutionary progress of the object parameters. Due to these benefits evolutionary strategies are preferable to genetic algorithms for problems in the domain of continuous optimization.

The role of the evolutionary strategy in the overall system depicted in Fig. 1 is to optimize four additional parameters of the controller and the sensors. Each candidate solution embodies scaling factors for the input and output, the orientation of the sensors and the weight factor determining the impact of preceding sensor measurements on the distance variable.

This paragraph describes the idea of tuning the fuzzy sets in order to optimize the performance of the controller, without relaxing the linguistic interpretation of the fuzzy knowledge base too much. The membership functions characterize the underlying fuzzy sets of linguistic terms. Normally, they are defined in advance in a way that reflects the designer's

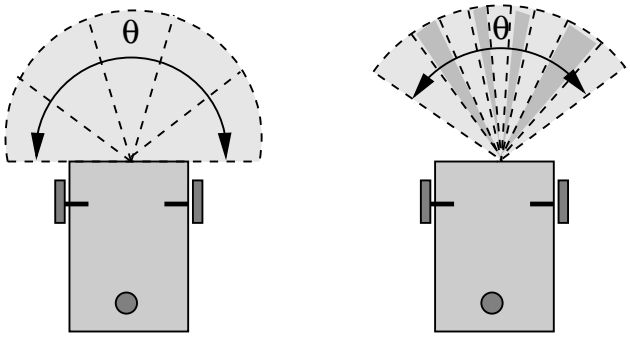

Fig. 8. Field of sonar perception for large (left) and small (right) value of θ

perception with regard to the dimensions of variables, e. g. *near* distance or *sharp left* steering. Therefore, the designer is able to comprehend and to interpret the rule base as automatically generated by the genetic algorithm. On the other hand, an inappropriate definition of membership functions may result in a suboptimal control behaviour. A reasonable compromise between the clarity of fuzzy rules on one hand and an optimal approximation to the control task on the other hand can be achieved, by introducing two linear scaling factors K_I, K_O for input and output. The first one scales the sensor input of all distance variables in common, while the second one plays the role of an output gain applied to the steering angle as shown in Fig. 1. The meaning of the linguistic terms is preserved, while at the same time the evolutionary strategy is able to adjust the sensitivity of the controller in regard to perceived objects and to tune the magnitude of the control action.

In addition to the scaling factors, the evolutionary strategy optimizes the orientation of the five sensors on the robot. Each ultrasonic sensor covers an angle of vision of approximately $\pm 20°$. The overall angle θ depicted in Fig. 8 determines the way in which the five sensors perceive objects in the frontal half plane of the robot. A large value of θ (Fig. 8 left) enables a wide total field of perception, but allows only a minimal overlap among the individual sensory fields. A small value of θ (Fig. 8 right) guarantees the reliable detection of objects even if they appear under a direction that lies at the boundaries of two adjacent sensory fields, but offers a poor total range of perception. The evolutionary strategy tunes the parameter θ in order to achieve an optimal compromise between these two conflicting objectives.

Finally, the fourth parameter α defines the weight given to previous sensor measurements when comprised within the distance variable according

to Eq. 1. In case of a large value of α the controller becomes more robust in regard to incorrect and imprecise sensor data, since the distance value is based on an average over several measurements. A small value of α improves the topicality of the distance information, so that the controller responds faster to perceived changes of the environment. The task of the evolutionary strategy is to find a value of α that suffices both objectives.

3.3. Fitness Evaluation

The evaluation of the control behaviour by means of a scalar fitness function depends on the objectives and constraints of the robotic application. There are only a few examples [4][21], in which the real robot is used to evaluate the controller. Often, online performance evaluation is too time consuming or even becomes impossible in case a poor controller may cause severe damage to the robot. Therefore, most of the approaches employ a quantitative simulation model of the robot and its environment.

For practical reasons, the simulation of the robot is based on some idealistic assumptions about its dynamics, environment and sensors, but still bears sufficient resemblance to the real world, e. g. it models incomplete and noisy distance measurements. The genetic algorithm benefits from the robustness of fuzzy control in respect to noise and imperfection, which allows it to utilize a simplified model for the fitness evaluation of the controllers. Previous experiments with the robot demonstrated that the controllers adapted in the simulation achieve an identical performance in the physical reality[13].

During the fitness evaluation, the fuzzy controllers are tested on the simulated robot, which is placed in several different kinds of corridors. Leitch suggested to evaluate the performance of the controllers on a variable set of test cases , instead of using a static set of training situations [18]. A second, simultaneous genetic algorithm adapts the set of training environments to the performance of the current population of controllers, so that the learning is based on the competition of two evolutionary processes. The robustness of the adapted robotic behaviour is improved, because the controllers are tested preferably on those environments that turned out to be difficult for previous generations [14].

The training environments are formed by a combination of dead-ends, straight segments and branchings at a right angle finally leading to an exit of the corridor. In order to find its way out of the corridor, the robot has to distinguish among these different types of segments. In each environment the robot is started two times heading towards opposite initial directions. Fig. 9 shows a training situation in which the robot collides with the lower

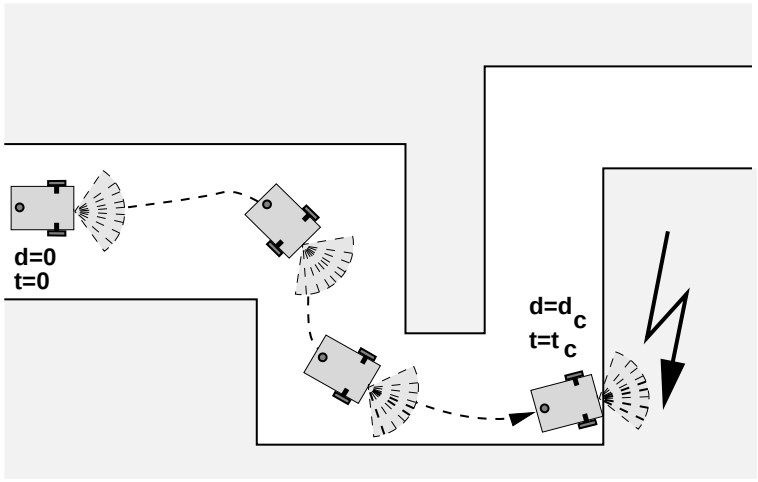

Fig. 9. Fitness evaluation of controllers, t_c : time until collision, d_c : absolute distance from start point until collision

right corner because the controller failed to initiate a left turn in time. A simulated run of the robot either stops if it collides with a wall or if a maximum evaluation time is exceeded. The fitness obtained by the controller for a single run is proportional to the time t_c the robot travels without having a collision. The most simple obstacle avoidance behaviour is to turn the robot on the spot. In order to elude the evolution of this trivial, but inadequate control strategy, the fitness is multiplied by the absolute distance d_c the robot covered from the starting point. The total fitness

$$F = \sum_{E_i} t_c^i * d_c^i \tag{2}$$

assigned to a controller is computed from the sum of fitness values achieved in separate environments E_i.

Fig. 10 shows a simulation of the robot behaviour adapted by means of the proposed method. The robot is initially heading towards the dead-end situation. Due to the additional context information provided by the neural network the controller is able to perceive the dead-end. The controller initiates a turning manœuvre which lasts until the robot is heading away from the dead-end. Afterwards, the normal wall following behaviour guides the robot to the exit of the corridor by keeping a safe distance to the left wall.

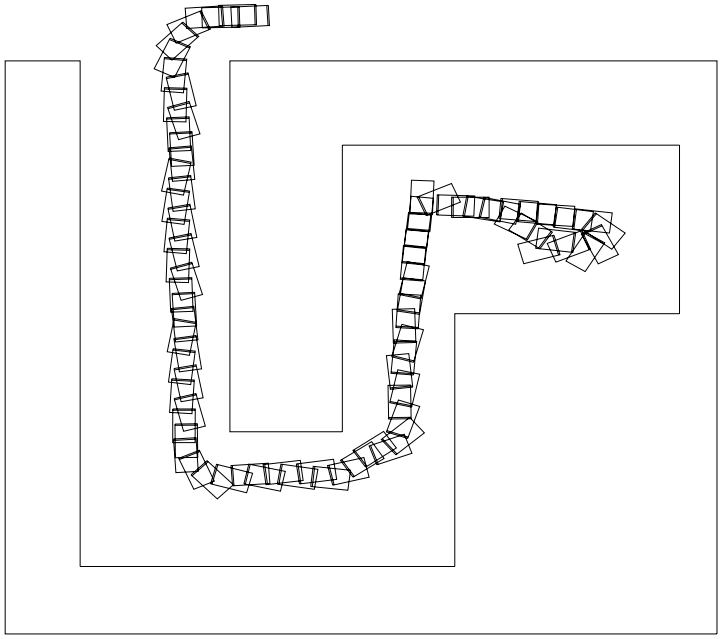

Fig. 10. Simulated run of the robot controlled by the evolutionary designed fuzzy rule base

4. Conclusions

This paper presented a soft computing approach for the design of mobile robot behaviours which combines evolutionary algorithms, fuzzy control and neural networks into a hybrid system. A wall following behaviour is implemented and adapted by means of a genetic fuzzy system. The proposed messy coding scheme enables the formation of fuzzy rules with antecedents of variable size and structure. As a result, the chromosomal representation of the rule base becomes more flexible and compact, which significantly reduces the complexity of the design task for the genetic algorithm. An evolutionary strategy optimizes additional continuous parame-

ters of the controller and the robot. By tuning the scaling factors of the input and output the evolutionary strategy optimizes the performance of the controller, without relaxing the linguistic meaning of the fuzzy sets. The evolutionary strategy adapts the adjustment of the sensors, in order to achieve an optimal perception of the environment. A neural network analyzes the current as well as past sensor data in order to classify the environmental situation. This additional information provided by the neural network is used by the controller in order to impose context dependent constraints on the activation of fuzzy rules for specific manœuvres. A short term memory stores previous sensor measurements which the controller takes into account for the computation of distance information. These two extensions to the former purely reactive control behaviour [13] enable the controller to detect dead-end situations and to execute a suitable turning manœuvre of the robot.

Acknowledgements

Thanks to Gerd Pfister and Oliver Malki from the Institut für Angewandte Physik of the University of Kiel for their supporting work on the mechanics, sensors, electronics and control software of the mobile robot. This work was funded by the MURI (ARO DAAH04-96-1-0341) and BISC program of UC Berkeley.

REFERENCES

1. Th. Bäck, F. Kursawe, "Evolutionary Algorithms for Fuzzy Logic: A Brief Overview", *Fifth International Conference IPMU: Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 659-664, (1996).
2. Th. Bäck, H. P. Schwefel, "Evolution Strategies I: Variants and their computational implementation", "Evolution Strategies II: Theoretical aspects and implementation", *Genetic Algorithms in Engineering and Computer Science*, Ed. G. Winter, J. Perieaux, M. Gala, P. Cuesta, pp. 111-126, pp. 127-140, John Wiley & Sons, (1995).
3. H. R. Beom, H. S. Cho, "A Sensor-Based Navigation for A Mobile Robot Using Fuzzy Logic and Reinforcement Learning", *IEEE Transactions on SMC*, vol. 25, no. 3, pp. 464-477, (1995).
4. A. Bonarini, F. Basso, "Learning Behaviours implemented as Fuzzy Logic Controllers for Autonomous Agents", *2. Online Workshop on Evolutionary Computation*, (1996).

5. R. Braunstingl, J. Mujika, J. P. Uribe, "A Wall Following Robot With a Fuzzy Logic Controller Optimized by a Genetic Algorithm", *Proc. FUZZ-IEEE'95*, vol. V, pp.77-82, Yokohama, Japan (1995).
6. M. Munir-ul M. Chowdury, Yun Li, "Messy Genetic Algorithm Based New Learning Method for Structurally Optimised Neurofuzzy Controllers", *IEEE Int. Conf. Industrial Technology*, Shanghai, China (1996).
7. O. Cordon, F. Herrera, "A General Study on Genetic Fuzzy Systems", *Genetic Algorithms in Engineering and Computer Science*, pp. 33-57, John Wiley & Sons, (1995).
8. M. Dorigo, U. Schnepf, "Genetics-Based Machine Learning and Behaviour-Based Robotics: A New Synthesis", *IEEE Trans. on SMC*, vol. 23, no. 1, pp. 141-154, (1993).
9. D. E. Goldberg, B. Korb, K. Deb, "Messy Genetic Algorithms Motivation, Analysis, and First Results", *Complex Systems*, vol. 3, pp. 493-530, (1989).
10. D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Reading Massachusetts: Addison-Wesley, (1989).
11. J. J. Grefenstette, A. C. Schultz, "An Evolutionary Approach to Learning in Robots", *Machine Learning Workshop on Robot Learning*, (1994).
12. F. Hoffmann, G. Pfister, "Learning of a Fuzzy Control Rule Base Using Messy Genetic Algorithms", *Genetic Algorithms and Soft Computing*, Ed. F. Herrera, J. L. Verdegay, Physica-Verlag, (1996).
13. F. Hoffmann, O. Malki, G. Pfister, "Evolutionary Algorithms for Learning of Mobile Robot Controllers", *Proc. EUFIT'96*, vol. II, pp. 1105-1109, (1996).
14. F. Hoffmann, G. Pfister, "Evolutionary Design of a Fuzzy Control Rule Base for a Mobile Robot", to appear in *International Journal of Approximate Reasoning*
15. J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, (1992).
16. C. L. Karr, "Design of a Cart-Pole Balancing Fuzzy Logic Controller using a Genetic Algorithm" *SPIE Conf. on Applications of Artificial Intelligence*, Bellingham, WA, (1991).
17. J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge MA, (1994).
18. Donald D. Leitch *A New Genetic Algorithm for the Evolution of Fuzzy Systems*, PhD thesis, University of Oxford, (1995).
19. M. Mitchell, S. Forrest, "Genetic Algorithms and Artificial Life", *Artificial Life 1 (3)*, pp. 267-289, (1994).
20. S. Nolfi, D. Floreano, O. Miglino, F. Mondada, "How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics", *Proc. of the 4. Int. Workshop on the Synth. and Sim. of Living Sys.* , MIT Press, (1994)
21. P. Nordin, W. Banzhaf, "Genetic Programming Controlling a Miniature Robot", *AAAI Fall Symposium on Genetic Programming*, (1995).
22. A. Saffiotti, E. H. Ruspini, K. Konolige, "Using Fuzzy Logic for Mobile

Robot Control", *Handbook of Fuzzy Sets and Possibility Theory*, Kluwer Academic, (1997).

23. H. Surmann, J. Huser, L. Peters, "A Fuzzy System for Indoor Mobile Robot Navigation" *Fourth IEEE Int. Conf. on Fuzzy Systems*, pp. 83-88, (1995)
24. H. Takagi, M. Lee, "Integrating Design Stages Stages of Fuzzy Systems using Genetic Algorithms", *Proc. of the Seond IEEE Int. Conf. on Fuzzy Systems*, pp. 612-617, (1993).
25. E. Tunstel, M. Jamshidi, "On Genetic Programming of Fuzzy Rule-based Systems for Intelligent Control", *Int. Journal of Int. Automation & Soft Comp.* , vol. 2, no. 2, (1996).
26. Lotfi A. Zadeh, "The Roles of Fuzzy Logic and Soft Computing in the Conception, Design and Deployment of Intelligent Systems", *BT Technology Journal*, vol. 14, no. 4, (1996).