Evolutionary Algorithms for Fuzzy Control System Design

FRANK HOFFMANN

Invited Paper

This paper provides an overview on evolutionary learning methods for the automated design and optimization of fuzzy logic controllers. In a genetic tuning process, an evolutionary algorithm adjusts the membership functions or scaling factors of a predefined fuzzy controller based on a performance index that specifies the desired control behavior. Genetic learning processes are concerned with the automated design of the fuzzy rule base. Their objective is to generate a set of fuzzy if-then rules that establishes the appropriate mapping from input states to control actions. We describe two applications of genetic-fuzzy systems in detail: an evolution strategy that tunes the scaling and membership functions of a fuzzy cart-pole balancing controller, and a genetic algorithm that learns the fuzzy control rules for an obstacle-avoidance behavior of a mobile robot.

Keywords**—***Evolutionary algorithm, fuzzy control, genetic fuzzy system, mobile robot.*

I. Introduction

Fuzzy control systems employ a mode of approximate reasoning that resembles the decision-making process of humans. The behavior of a fuzzy controller is easily understood by a human expert as knowledge is expressed by means of intuitive, linguistic rules.

A fuzzy system is usually designed by interviewing an expert and formulating her implicit knowledge of the underlying process into a set of linguistic variables and fuzzy rules. In particular for complex control tasks, obtaining the fuzzy knowledge base from an expert is often based on a tedious and unreliable trial and error approach.

Unlike neural networks, generic fuzzy systems are not able to learn from data. However, several techniques have been proposed to extract fuzzy rules from training data gathered from observations of the operator control strategy [1]–[3].

Manuscript received January 13, 2001; revised April 2, 2001. This work was supported by the NUTEK project “Architectures for Autonomous Systems” NUTEK-1k1p-99-06200 and by the EU TMR Network VIRGO.

The author is with the Royal Institute of Technology, Center for Autonomous Systems, Stockholm, Sweden (e-mail: hoffmann@nada.kth.se).

Publisher Item Identifier S 0018-9219(01)07609-5.

Neuro-fuzzy systems combine the learning capability of neural networks with the knowledge representation of fuzzy logic. Typically, the fuzzy model is transferred into a neural network-like architecture, which then is trained by some learning method, such as gradient descent or nonlinear regression techniques [4], [5]. Neuro-fuzzy approaches are suitable for supervised learning tasks, where the objective is to minimize the error between the output of the fuzzy system and the target value.

Evolutionary algorithms provide a universal optimization technique that mimics the type of genetic adaptation that occurs in natural evolution [6], [7]. Unlike specialized methods designed for particular types of optimization tasks, they require no particular knowledge about the problem structure other than the objective function itself. A population of candidate solutions evolves over time by means of genetic operators such as mutation, recombination, and selection. The different approaches, genetic algorithms (GAs), evolution strategies (ESs), evolutionary programming, (EP) and genetic programming (GP), are distinguished by the genetic structures that undergo adaptation and the genetic operators by which they generate new variants.

Recently, numerous publications have proposed evolutionary algorithms to automate the knowledge acquisition step in fuzzy system design [8]–[11]. These methods are described by the general term *genetic fuzzy systems*. Genetic fuzzy systems are applicable to control design problems in which the objective is to maximize some performance index of the closed-loop process itself, as the evolutionary optimization is solely based on a scalar objective function.

This paper provides a general overview on genetic fuzzy systems and describes a framework for the evolutionary tuning and optimization of fuzzy control systems. The performance of a fuzzy controller is improved by tuning parameterized membership functions and input–output (I/O) scaling factors with respect to the desired control behavior. A population of competing chromosomes that encode the tuning parameters evolves by means of selection, recombination, and mutation. The fitness of an individual is evaluated by observing the performance of the fuzzy controller while regulating the process using the membership functions and scaling factors encoded in the chromosome. Over the course of evolution, the evolutionary algorithm identifies the set of parameters, for which the fuzzy controller performs optimal with respect to the given performance index.

The second method presented in this paper is concerned with the automated design of the fuzzy rule base. Learning a fuzzy rule base can be considered as an optimization problem in which the search space is constituted by the set of fuzzy if-then rules. In our approach, a genetic algorithm learns the label of entries in the fuzzy decision table that associate the rule antecedents with a particular output fuzzy set. The proposed method is applied to design an obstacle-avoidance behavior for a mobile robot. The fuzzy controller maps the input perceived by a set of sonar senors to a control action, namely, the turn rate of the robot.

The majority of applications in the domain of genetic fuzzy systems is concerned with the optimization of fuzzy logic controllers [12], [13], [9], [14], [11]. Several authors proposed evolutionary algorithms for learning robotic behaviors implemented by fuzzy control rules [15]–[18].

The Evolutionary Learning of Fuzzy rules (ELF) system employs a credit assignment mechanism similar to reinforcement learning techniques [15]. The quality of an individual fuzzy rule is evaluated in the context of its cooperation with other rules in the population. Competition is restricted to subsets of rules that trigger for similar inputs. ELF has been successfully applied to learn individual robotic behaviors, the coordination of primitive behaviors within a single autonomous agent and across multiple agents.

The approach by [18] employs a hierarchy of fuzzy behaviors for the control of mobile robots similar to the subsumption style architecture in [19]. Each behavior contributes to the overall control decision according to its level of activation, which, in turn, depends on its applicability in the current context. This fuzzy approach to robot behavior coordination was originally proposed in [20]. Tunstel *et al.* employ genetic programming to evolve supervisory fuzzy rules for the coordination of low-level fuzzy behaviors. The evolved goal-seeking behavior demonstrated modest generalization capability to previously unseen situations.

In an earlier paper, the author proposed a messy genetic algorithm for the automated design of a reactive fuzzy rule-based behavior that guides the mobile robot to a known goal point, while avoiding obstacles on the way [17]. Experiments demonstrated that a fuzzy behavior evolved in a low-fidelity simulation achieves a comparable performance on the real robot.

In [16], the authors present a fuzzy classifier system that utilizes a genetic algorithm evolutionary for online-learning of a goal seeking and a wall following behavior. The fuzzy classifier system contains to a cache to store suitable fuzzy rules that are applied in future situations and are used to seed the initial population. This technique substantially speeds up the learning process, which makes the entire approach feasible for online learning of robotic behaviors (the robot learned the goal-seeking and wall-following behavior after 96 s).

The paper is organized as follows. Section II introduces the general structure of evolutionary algorithms and presents evolution strategies in more detail. Section III describes the components and principal operation of genetic fuzzy systems. Section IV presents genetic tuning processes of fuzzy controllers and shows their application to the cart-pole balancing problem. Section V on genetic learning processes compares several approaches concerned with the automated design of the fuzzy rule base. Section VI presents the application of a genetic algorithm to evolve a set of fuzzy rules that constitute a reactive obstacle-avoidance behavior of a mobile robot. Finally, in Section VII, some conclusions are drawn.

II. Evolutionary Algorithms

An evolutionary algorithm processes a population of competing candidate solutions. The chromosome decodes a set of parameters mapped into a potential solution to the optimization problem. A scalar objective function evaluates the quality of a solution. According to Darwin's principle, individuals superior to their competitors obtain a better chance to promote their genes to the next generation. Genetic operators such as recombination and mutation are applied to the parents in order to generate new variants. The interplay of exploiting good solutions via selection and exploring the search space in form of recombination and mutation constitutes the fundamental theme in evolutionary optimization. The quality of solutions improves gradually as a result of this basic cycle of selection, reproduction, recombination and mutation.

Fig. 1 depicts the principal structure of a generic evolutionary algorithm. The quality of individuals that constitute the current population $P(t)$ is evaluated according to a scalar fitness function. Individuals that achieve a higher fitness are more likely to be selected as parents and generate offspring by means of recombination and mutation. Recombination promotes the exchange of genetic information among individual members of the population. The offspring are subject to mutations, which randomly modify a gene in order to create new variants. The current population is replaced by the newly generated group of offspring, which forms the new population $P(t+1)$ in the next generation. The evolutionary algorithm terminates either if a maximum number of generations elapses or a desired level of fitness is reached.

Evolution strategies [6] and genetic algorithms [7] share the same generic features, a population of chromosomes, a selection scheme, a replacement policy and genetic operators for recombination and mutation. The major difference between evolution strategies and genetic algorithms lies in the genetic representation of candidate solutions. An evolution strategy manipulates a vector of real numbers, whereas genetic algorithms usually process a string of discrete, often binary symbols.

Evolution strategies are distinguished by self-adaptation of additional strategy parameters, which enables them to adapt the evolutionary process to the structure of the fitness

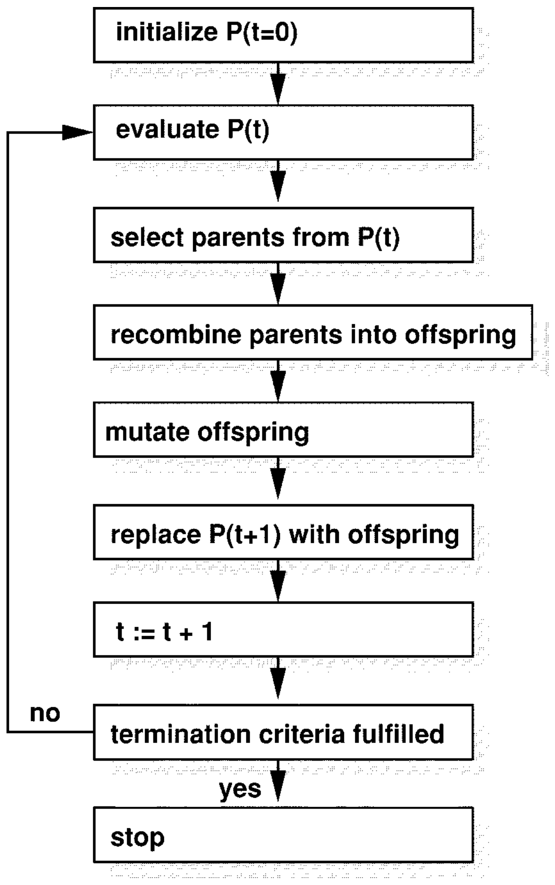

initialize P(t=0)
evaluate P(t)
select parents from P(t)
recombine parents into offspring
mutate offspring
replace P(t+1) with offspring
t := t + 1
no
termination criteria fulfilled
yes
stop

Fig. 1. Structure of an evolutionary algorithm.

landscape. The strategy parameters are either tuned by an exogenous, deterministic step-size adaptation scheme or by the same evolutionary mechanism that optimizes the original object parameters [6].

A pair of real-valued vectors $(\vec{x}, \vec{\sigma})$ forms the chromosome of an individual. The first vector of object variables $(x_1, \ldots, x_n)$ constitutes a potential solution in the optimization space. The additional vector of strategy parameters $(\sigma_1, \ldots, \sigma_n)$ defines the standard deviations of mutations applied to the object variables. A mutation replaces the parent $\vec{x}^t$

$$\vec{x}^{t+1} = \vec{x}^t + \vec{N}(0, \vec{\sigma}) \tag{1}$$

with the offspring $x(t \vec{+} 1)$, where $N(\vec{0}, \sigma)$ is a normally distributed random vector with zero mean and standard deviation $\vec{\sigma}$.

Rather than using a deterministic step-size adaptation, the evolution strategy itself controls its own mutability. In this case, the additional strategy vector $\vec{\sigma}$ undergoes mutation as well

$$\vec{\sigma}^{t+1} = \vec{\sigma}^t e^{\tau' N(0,1) + \tau \vec{N}(0,1)}. \tag{2}$$

The normally distributed random variable $N(0, 1)$ is sampled once for the entire chromosome, whereas the mutations $\vec{N}(0, 1)$ are uncorrelated for different genes. The global factor $e^{\tau' N(0,1)}$ increases or decreases the overall rate of mutation, whereas $e^{\tau N_i(0,1)}$ adapts the individual step-size σ_i. The logarithmic normal distribution guarantees that the mutation standard deviations σ_i remain positive. The mutation is unbiased in the sense that an increase or decrease of σ_i by the same factor occurs with identical probability.

In the long run, selection favors those strategy parameters which are more likely to generate successful mutations of object variables in the future. Due to the mechanism of self-adaptation, an exogenous control of step-sizes, utilized by standard mathematical optimization methods, becomes obsolete.

Although mutation is the major genetic operator in evolution strategies, recombination can be helpful to improve the search. In intermediate recombination, an offspring inherits the statistical parameters of its parents $(\vec{x}^1, \vec{\sigma}^1)$, $(\vec{x}^2, \vec{\sigma}^2)$

$$(\vec{x}, \vec{\sigma}) = ((\vec{x}^1 + \vec{x}^2)/2, (\vec{\sigma}^1 + \vec{\sigma}^2)/2). \tag{3}$$

The discrete recombination is identical to uniform crossover in genetic algorithms, where each component x_i is randomly picked from either one of the parents. Often, intermediate recombination is applied to the strategy parameters, whereas discrete recombination is preferred for the object variables. Evolution strategies employ a deterministic μ, *λ)-selection* scheme in which only the μ best individuals survive as parents from which λ offspring are generated.

III. Genetic Fuzzy Systems

The objective of a genetic fuzzy system is to automate the knowledge acquisition step in fuzzy system design, a task that is usually accomplished through an interview or observation of a human expert controlling the system. An evolutionary algorithm adapts either part or all of the components of the fuzzy knowledge base. At this point, it is important to notice that a fuzzy knowledge base is not a monolithic structure but is composed of the database and the rule base, which each play a specific role in the fuzzy reasoning process. According to the distinction between database and rule base, genetic fuzzy systems are discriminated along two major approaches, genetic tuning processes and genetic learning processes. The first method is targeted at optimizing the performance of an already existing fuzzy system. The tuning process involves the adaptation of the fuzzy database, namely parameters of membership functions and I/O scaling factors. The second method is concerned with the automatic derivation of fuzzy rules in the rule base. A genetic learning process faces a much more difficult task as it has to establish the proper relationship between input and output states from scratch, rather than optimizing the performance of a fuzzy system that already operates at least approximately correct.

Designing a fuzzy rule-based system is equivalent to finding the optimal configuration of fuzzy sets and/or rules, and in that sense can be regarded as an optimization problem. The optimization criterion is the problem to be solved at hand and the search space is the set of parameters that code the membership functions, scaling functions and fuzzy rules. The genetic learning process emerges from the hybridization of an evolutionary algorithm, which by means of selection and genetic operators optimizes parameters of

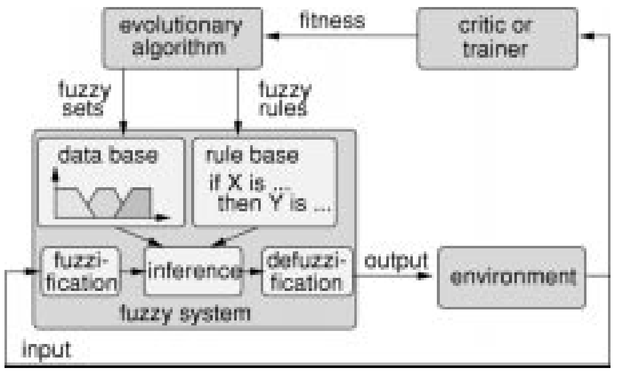


Fig. 2. Genetic fuzzy system.

the knowledge base, with the fuzzy system supposed to demonstrate a desired behavior.

Genetic fuzzy systems have been successfully applied to control system design, modeling, decision making, optimization, classification, and information retrieval [21]. The majority of publications is concerned with fuzzy rule-based control system design [15], [12], [22], [9], [10], [14], [11].

For two reasons, a fuzzy representation is particularly useful for evolutionary optimization compared to other possible parameterizations of a controller. For many real-world problems a mathematical precise and complete solution is not only unnecessary, but often also unfeasible. Fuzzy systems exploit this tolerance for imprecision by aggregation of similar states into coarse granules defined by fuzzy sets. The remaining task for the evolutionary algorithm becomes to learn the appropriate local relationships between input and output states established by fuzzy if–then rules. The locality and granularity of fuzzy rules not only decreases the complexity of the search space, but at the same time lessens the interdependency of the very genes that encode these rules. In return, limited interaction among genes expedites the evolutionary optimization process according to the *building block hypothesis* [7].

Fig. 2 shows the major components of a genetic fuzzy system. The fuzzy system lies at the core of the hybrid structure, it fuzzifies the input state, performs the inference based on the fuzzy rules, and aggregates the result of the inference process into a crisp output. Depending on the context, the environment can be a plant to be controlled, a system to be modeled, or a set of data to be classified. An external critic or trainer evaluates the performance of the fuzzy system with regard to the control task, the model accuracy, or the classification error. The performance is aggregated into a scalar fitness value on which basis the evolutionary algorithm selects better adapted chromosomes. A chromosome either codes parameters of membership functions, scaling factors, fuzzy rules, or a combination thereof. By means of crossover and mutation, the evolutionary algorithm generates new parameters for the database and/or rule base which usefulness is tested in the fuzzy system.

There are two major approaches to evolutionary learning of fuzzy controllers. In the so-called "Michigan" approach, the population is composed of a set of individual rules that compete with each other to suggest the optimal control action. The method is suitable for on-line learning tasks as the evolutionary algorithm incrementally improves the performance of the fuzzy controller constituted by the population of rules [15]. A temporal credit assignment mechanism, such as the bucket brigade algorithm, distributes the external reward among fuzzy rules activated in subsequent control steps.

The methods presented in this paper belong to the so-called "Pitt" learning approaches that operate on a population of rule bases. As the evolutionary algorithm evaluates the performance of the fuzzy controller as a whole, rather than that of isolated fuzzy rules, the credit assignment mechanism becomes obsolete.

The genetic representation of the fuzzy knowledge base is another important issue in the conception of a genetic fuzzy system. The main distinction is made between *descriptive* and *approximative* knowledge bases. In the former one, database and rule base are two clearly separate entities. All rules share the same membership functions defined in a common database. Fuzzy sets are associated with linguistic terms such as *small*, *medium*, *large*. A fuzzy if–then rule combines multiple linguistic labels in a way that facilitates an intuitive interpretation of the relationship between input and output states defined by this rule. In an approximative fuzzy system, each rule operates with its own fuzzy sets. Database and rule base merge as the rule itself contains the parameters of the underlying membership functions. Such a representation possesses more degrees of freedom which permits a more accurate approximation of the desired I/O relationship, hence, the term approximative. The price of increased accuracy is that the resulting fuzzy system becomes more difficult to analyze as an individual fuzzy set no longer coincides with a linguistic concept.

IV. Genetic Tuning Processes

Over the last years several researchers conceptualized methods for tuning the performance of fuzzy systems by means of evolutionary algorithms [9], [14]. Genetic tuning processes focus on the adaptation of the fuzzy database. The chromosome codes parameters of the membership functions and/or scaling functions for input and output variables. They are based on the assumption that the rule base has been already designed, either by a human expert or by a prior learning process.

A. Tuning Scaling Functions

A scaling function maps the universe of discourse of a linguistic variable onto a normalized range over which the fuzzy membership functions are defined. From a control design point of view, a scaling function corresponds to the gain applied to the input and output of a system. From the knowledge representation perspective, scaling functions provide context information as the define the operating range of variables in a particular application. Consider for example a linguistic variable for speed, which range is entirely different for a customary automobile and a Formula One race car.

Most applications of scaling functions utilize a linear form of scaling

$$x' = \frac{x-a}{b-a} \tag{4}$$

in which a and b determine the lower and upper bound of the operating range $[a, b]$. The scaled variable x' is defined over the normalized interval $[0, 1]$. In case of a symmetric variable, only one parameter a is required to map the interval $[-a, a]$ to $[-1, 1]$.

Once the variables are transformed to the normalized interval $[0, 1]$ or respectively $[-1, 1]$ it becomes possible to apply nonlinear scaling functions

$$x' = \operatorname{sign}(x)|x|^{\alpha} \tag{5}$$

in which the region around origin is contracted ($\alpha < 1$) or dilated ($\alpha > 1$). Whereas linear scaling has the same uniform effect on the entire operating range, nonlinear scaling according to (5) increases the sensitivity to the input in one region of the universe of discourse and decreases it in another. For example, in case of $\alpha < 1$, the control becomes more sensitive for small inputs $x \approx 0$ and relatively less sensitive for larger input values $x \approx 1$.

The effects of the scaling factors a and b on the system behavior are manifold. First of all, modifications to the parameters a and b lead to an expansion or contraction of the operating range, which in turn decreases or increases the sensitivity of the system for that variable. If the parameters a and b change by the same amount, the operating range is shifted to the left or the right, which is equivalent to an offset for that variable. Finally, the nonlinear scaling affects the relative sensitivity of areas within the operating range.

Scaling functions have a drastic impact on the overall system behavior. Therefore, one expects the genetic tuning process to adapt these global parameters early on before it locally fine-tunes the membership functions.

For each input and output variable, the evolution strategy codes the real-valued scaling parameters a, b, and α in the object vector $\{x_1, \ldots, x_n\}$.

In the following a genetic tuning process optimizes the input scaling factors of a fuzzy cart-pole controller. A pole is pivoted to a cart moving on a horizontal track, as shown in Fig. 3. The controller can exert a force F on the cart in order to stabilize the pole angle θ, angular velocity ω, cart position x, and velocity $\dot{x}$.

The dynamics of the cart pole system are given by

$$\begin{aligned}
\dot{\theta} &= \omega \\
\dot{\omega} &= \frac{g \sin\theta - \dfrac{\cos\theta(F + m_p l \omega^2 \sin\theta)}{m_c + m_p}}{l\left(4/3 - \dfrac{m_p \cos^2\theta}{m_c + m_p}\right)} \\
\dot{x} &= v \\
\dot{v} &= \frac{F + m_p l(\omega^2 \sin\theta - \dot{\omega}\cos\theta)}{m_c + m_p}
\end{aligned} \tag{6}$$

where g is the acceleration due to gravitation, l is the half length of the pole, and m_p and m_c are the mass of the pole and the cart, respectively.

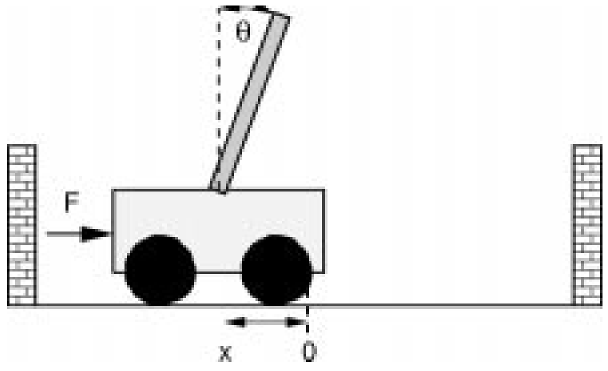


Fig. 3. Cart pole balancing problem: The objective is to balance the pole into an upright position while the cart is kept within the boundaries of the track.

The task is to balance the pole in a vertical position, while returning the cart to the origin. According to these objectives, the objective of the optimization is to minimize the cost functional

$$J = \int_{t=0}^{\inf} x'Px + u'Qu\, dt \tag{7}$$

over the state vector $x = (\theta, \omega, x, v$ and control vector $u = F$. The cost matrices P and Q reflect the desired characteristics of the controller performance. In our case, we choose P as the diagonal matrix

$$P = \begin{pmatrix} 20.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.5 \end{pmatrix} \tag{8}$$

and $Q = 0.1$ ($u = F$ and, hence, Q are scalars in case of the cart-pole system) as the penalty factor for the magnitude of the control action F. The cart-pole system is started from two different initial states $x^1 = (0.3, 0.0, 0.0, 0.0)$ and $x^2 = (0.0, 0.0, -1.5, 0.0)$ and regulated by the fuzzy controller subject to evaluation for a period of 5 s. The performance of a fuzzy controller is computed as the sum of the cost function $J = J^1 + J^2$ in both trials.

An evolution strategy optimizes the object vector $\{x_1, x_2, x_3, x_4\}$ which encodes the linear scaling factors a_θ, a_ω, a_x, a_v for each of the input variables θ, ω, x, v. As all input variables are symmetric the linear scaling function $x'_i = a_i x_i$ is fully determined by a single factor a_i. The fuzzy controller utilizes the original rule base given in Table 1, which itself is not subject to adaptation by the tuning process.

In the $(4, 12)$-evolution strategy, the four out of 12 offspring that achieve the lowest cost J constitute the parents of the next generation. The individuals of the first generation are initialized with scaling factors close to unity. The best individual that emerges within 30 generations corresponds to a set of scaling factors $a_\theta = 0.66$, $a_\omega = 0.53$, $a_x = 0.49$, $a_v = 0.56$. Fig. 4 compares the original membership functions (left) with the scaled ones obtained through the

Table 1.
Fuzzy Rule Base for Cart and Pole Stabilization

F	x is neg		θ	
	v is neg	neg	zero	pos
	neg	nb	nm	ns
ω	zero	nm	ns	ze
	pos	ns	ze	ps
F	x is neg		θ	
	v is pos	neg	zero	pos
	neg	nm	ns	ze
ω	zero	ns	ze	ps
	pos	ze	ps	pm
F	x is pos		θ	
	v is neg	neg	zero	pos
	neg	nm	ns	ze
ω	zero	ns	ze	ps
	pos	ze	ps	pm
F	x is pos		θ	
	v is pos	neg	zero	pos
	neg	ns	ze	ps
ω	zero	ze	ps	pm
	pos	ps	pm	pb

evolutionary tuning process (right). For all variables, the region around the origin becomes broader with the effect that the input gain decreases.

The evolutionary tuning of the input scaling factors a_θ, a_ω, a_x, a_v reduces the cost J in (7) added over both test cases from $J_{\text{org}} = 480$ to $J_{\text{opt}} = 209$, which reflects itself in the time evolution of the system states depicted in Fig. 5. The pole stabilizes within the first 2 s after which the cart is smoothly returned to the origin without the pole angle to overshoot. The control action F evolves more continuously and the overall amount of force exerted on the cart is reduced. Although the scaling factors are only tuned for two particular test cases, the resulting fuzzy controller demonstrates an improved performance over the entire spectrum of initial states.

B. *Tuning Membership Functions*

Fuzzy sets are usually defined by parameterized membership functions. The number of parameters depends on their shape, triangular fuzzy sets are defined by three characteristic points, trapezoidal fuzzy sets by four points and Gaussian fuzzy sets by center and width. The number of parameters can be reduced if certain constraints are imposed on the fuzzy partition. Most fuzzy systems employ normalized fuzzy sets, that require that the membership values $\mu_{A_i}(x)$ of all fuzzy sets A_i sum up to unity

$$\forall_x \sum_i^L \mu_{A_i}(x) = 1. \tag{9}$$

It is therefore sufficient only to define the center points $c_1, \ldots, c_L$ of normalized, triangular membership functions in order to specify the entire fuzzy partition of a variable.

The order of fuzzy sets A_i is not supposed to change, namely the center point c_i of a fuzzy set A_i with lower index $i < j$ must stay to the left $c_i < c_j$ of the fuzzy set A_j. This property is maintained by a representation that codes the distances Δ_i between adjacent fuzzy sets rather than their absolute locations

$$c_i = c_{i-1} + \Delta_i = c_0 + \sum_{k=1}^{i} \Delta_i \tag{10}$$

as shown in Fig. 6.

The point c_0 defines the lower bound of the variable range. Due to the normalization constraint in (9), the left l_i and right r_i point of each fuzzy set A_i coincide with the center points of the left c_{i-1} and right neighbor c_{i+1}. This type of representation requires one parameter Δ_i per fuzzy set A_i.

In the following, we present an evolution strategy that tunes the center points of the triangular output membership functions $\{NB, NM, NS, ZE, PS, PM, PB\}$ of the fuzzy cart pole controller.

For a fixed set of rules, the control surface merely depends on the center point of the output fuzzy sets rather than the overlap between adjacent fuzzy sets. Therefore, we restrict the shape of the output membership functions to symmetric triangles of fixed width and only adjust the center location of the fuzzy sets. The structure of the control problem suggests that the fuzzy set pairs (NB, PB), (NM, PM) and (NS, PS) are equidistant to the origin and that the center point of the fuzzy set ze lies at the origin itself. The object vector $\{x_1, x_2, x_3\}$ encodes the three remaining free parameters $\Delta_{NS,PS}$, $\Delta_{NM,PM}$, $\Delta_{NB,PB}$, as shown in Fig. 7. As each $\Delta_i > 0$ determines the positive offset between adjacent fuzzy sets, their order does not change (e.g., the fuzzy set PS cannot shift to the right of the fuzzy set PM).

Tuning decreases the cost functional J in (7) summed over both test cases from $J_{\text{org}} = 480$ for the original output fuzzy sets (Fig. 8, top) to $J_{\text{opt}} = 250$ for the optimized output fuzzy sets (Fig. 8, bottom). The time history of the state variables shown in Fig. 9 illustrates that the optimized fuzzy controller stabilizes cart and pole more rapidly and reduces the overall amount of force applied to the cart.

V. Genetic Learning Processes

Whereas genetic tuning processes require a previously defined rule base, genetic learning of fuzzy rules constitutes a self-organizing process that establishes the proper relationship between the observed input state and the desired control action, starting without any previous knowledge. Fixed-length chromosomes are applicable to encode a static fuzzy rule base, represented by a relational matrix or a decision table. A fuzzy relational matrix defines a fuzzy subset over

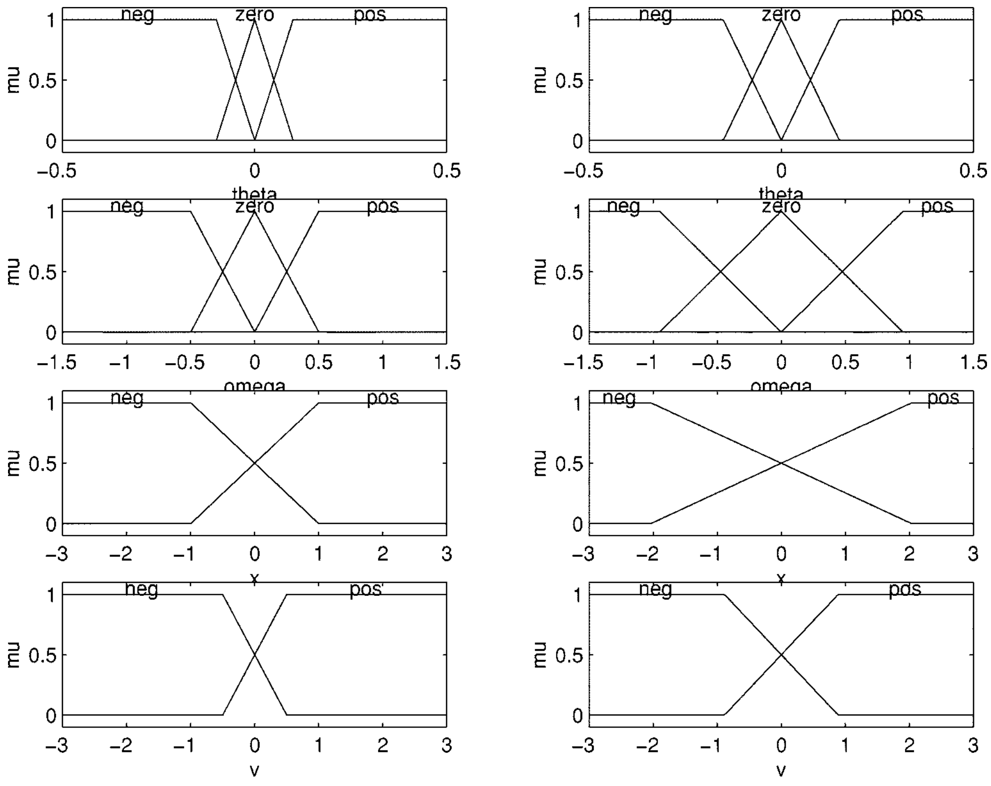


Fig. 4. Fuzzy membership functions of the original controller on the left compared with the tuned membership functions on the right for scaling factors $a_\theta = 0.66$, $a_\omega = 0.53$, $a_x = 0.49$, $a_v = 0.56$.

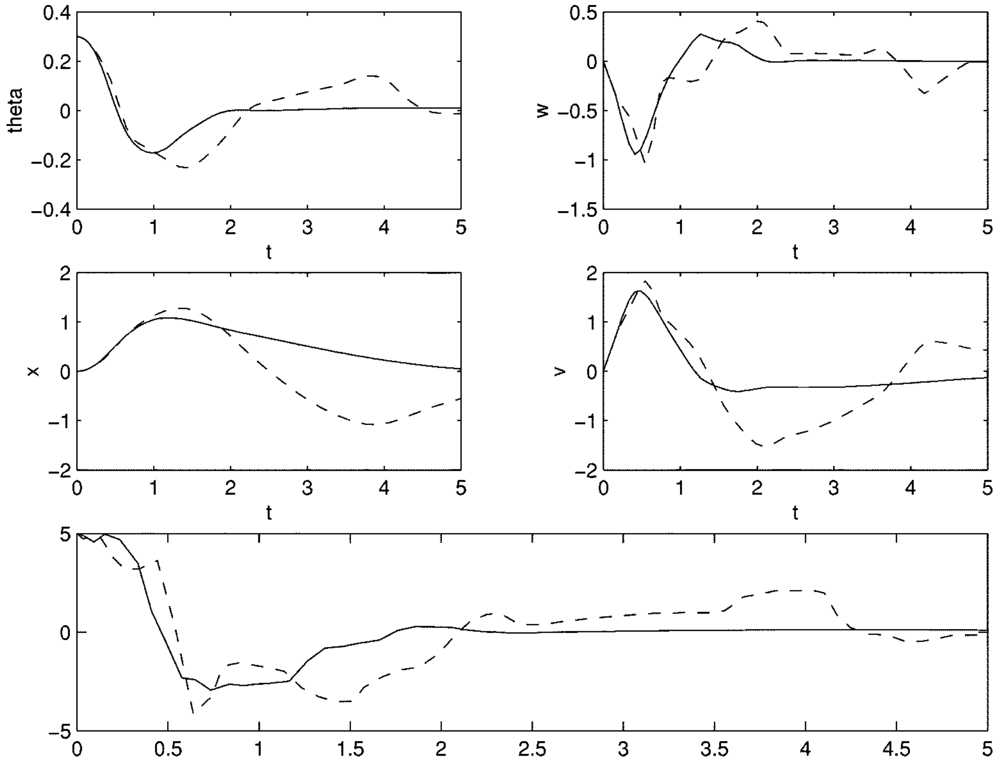


Fig. 5. Time evolution of the state variables θ, ω, x, v and control action F for the original (dashed) and tuned (solid) fuzzy controller.

the Cartesian product of input and output variables. A fuzzy decision table defines a crisp relation which connects the sets of fuzzy input and output variables.

In our case the code is positional, that is, the functionality of a gene is determined by its position, and its value defines the corresponding attribute. A specific location in a

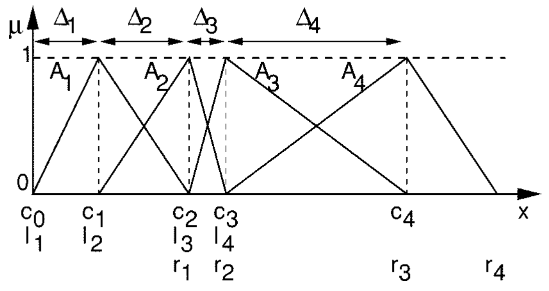


Fig. 6. Genetic representation of the fuzzy partition: The chromosome codes the distances Δ_i between the center points c_i of adjacent fuzzy sets A_i and A_{i+1}.

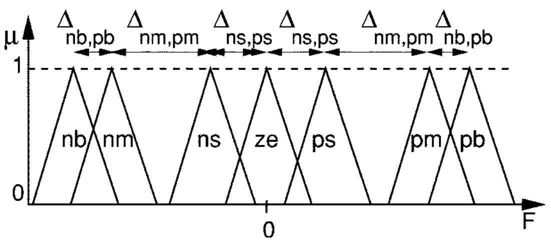


Fig. 7. The object vector $\{x_1, x_2, x_3\}$ in the evolution strategy encodes the offsets $\Delta_{NS,PS}$, $\Delta_{NM,PM}$, $\Delta_{NB,PB}$ between the center points of adjacent output fuzzy sets $\{NB, NM, NS, ZE, PS, PM, PB\}$.

positional chromosome refers to a particular entry in the decision table and the gene value defines the output fuzzy set associated to that entry. The number of genes in the chromosome is identical to the number of elements in the decision table or the relational matrix. Our approach assumes that the linguistic variables, the number of fuzzy sets, and the corresponding membership functions in the fuzzy database are defined in advance.

The size of the decision table or relational matrix grows rapidly with the number of input variables and linguistic terms. Therefore, other genetic learning approaches operate with a variable rather than a static number of rules in order to keep the learning task feasible for large input dimensions [13], [14].

A second distinction is made for genetic learning processes that employ approximate rather than linguistic or descriptive knowledge bases [14], [11]. In an approximate knowledge base each rule defines its own unique fuzzy sets, whereas rules in a descriptive fuzzy controller refer to a linguistic label that points to a fuzzy set externally defined in the database commonly shared by all rules. Approximate fuzzy systems possess more degrees of freedom and, therefore, achieve a better accuracy than descriptive ones. However, the comprehensiveness of the stored knowledge deteriorates as the fuzzy rules no longer share a unique linguistic interpretation.

The genetic learning approach presented in this paper is concerned with a positional, fixed-length genetic representation of a descriptive knowledge base. We assume that the membership functions of the linguistic variables are defined in advance and are not subject to adaptation. However, it is possible to augment the rule learning phase with a posterior genetic tuning process to refine the knowledge base.

The fuzzy decision table is a common way to represent the rule base of a fuzzy controller. Each entry in the decision table associates a combination of fuzzy input sets in the rule antecedent with a fuzzy output set in the rule consequent. Table 2 depicts a fuzzy decision table for a two input (X_1, X_2), single output (Y) fuzzy controller. The input variables are partitioned into three fuzzy sets (A_{11}, A_{12}, A_{13}; A_{21}, A_{22}, A_{23}) each, the output variable contains four fuzzy sets (B_1, B_2, B_3, B_4).

The decision table is scanned row-wise and the chromosome codes the output fuzzy set of the current entry as an integer. Therefore, the sequence of output fuzzy sets (B_1, B_1, B_2, B_1, B_2, B_3, B_1, B_3, B_4) in the decision table shown in Table 2 is represented by the integer vector (1, 1, 2, 1, 2, 3, 1, 3, 4). Even though the code is integer based rather than binary, the standard two-point crossover operator commonly used in genetic algorithms remains applicable. The mutation operator either increases or decreases the integer by one, and thereby substitutes the current output fuzzy set with one of its neighbors (e.g., an entry with B_2 might be replaced by B_1 or B_3). Mutation of the left- and rightmost output fuzzy sets B_1, B_4 is restricted such that the integer values remain bounded between 1 and 4.

In a Takagi–Sugeno–Kang (TSK) fuzzy controller, the rule consequent is formed by a linear combination of the crisp inputs

$$\begin{aligned}R_i\text{: if } X_1 \text{ is } A_{1i} \text{ and } \cdots \text{ and } X_n \text{ is } A_{ni}\\ \text{then } y = c_{0i} + c_{1i}x_1 + \cdots + c_{ni}x_n\end{aligned}$$

rather than by fuzzy sets as in a conventional Mamdani controller. The crisp TSK rule consequent constitutes a local, linear approximation to the desired I/O relationship, that is valid in the region specified by the fuzzy rule antecedent. The additional degrees of freedom enable TSK fuzzy controllers to achieve the same accuracy with a fewer number of rules. However, the control behavior becomes less intuitive as the linguistic interpretability gets lost.

Since the antecedent structure of Mamdani and TSK fuzzy systems is identical, it is possible to represent a TSK rule base in a similar structure as a decision table. The same positional code maps from rule antecedents to entries in the decision table. The only difference between both representations lies in the information stored in the entries of the table, the label B_i to a linguistic term of the output variable (Mamdani rules) or the parameter vector $c_{0i}, \ldots, c_{ni}$ in the linear equation (TSK rules).

The chromosome is a vector of real numbers formed by the set of individual rule parameters $(c_{0i}, \ldots, c_{ni})$ concatenated in the order of entries in the decision table. The real-valued object vector employed in evolution strategies provides a natural representation for the TSK rule base parameter vector $((c_{00}, c_{10}, \ldots, c_{n0}), \ldots, (c_{0m}, c_{1m}, \ldots, c_{nm}))$. This approach has been successfully applied to design a TSK fuzzy controller for an obstacle-avoidance behavior of a mobile robot [23]. In [10], a genetic algorithm optimizes the TSK fuzzy rule base, hence, the parameter vector is encoded in a binary string using eight bits per coefficient c_{ij}.

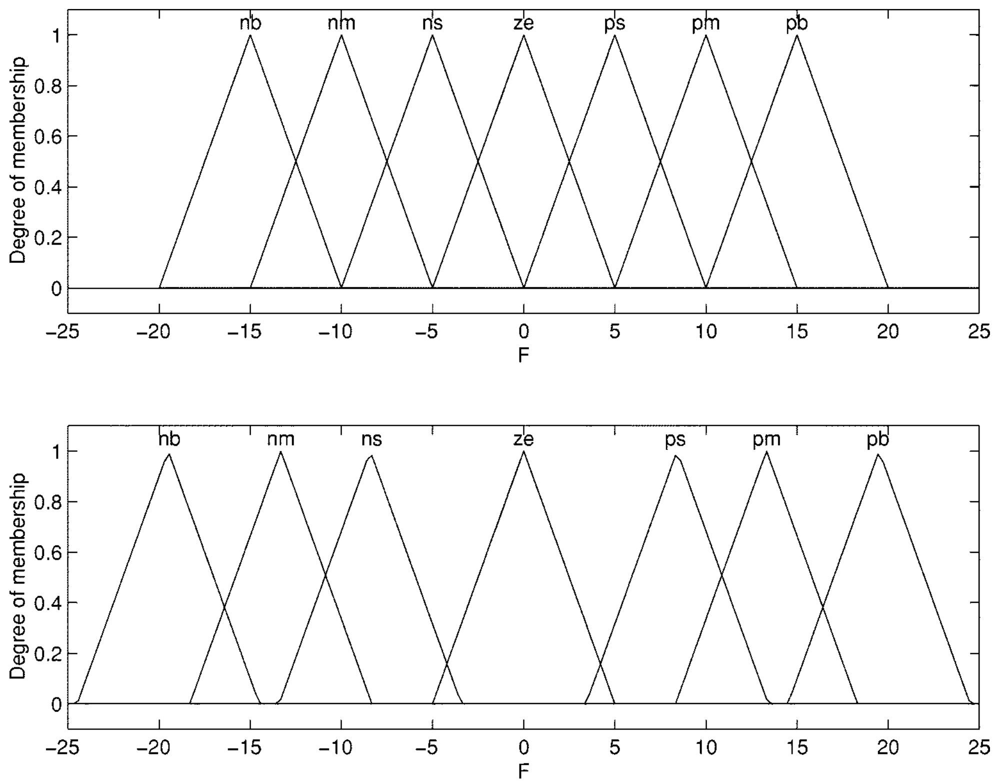


Fig. 8. Equally distributed fuzzy membership functions of the original controller at the top compared with the tuned membership functions $\{NB, NM, NS, ZE, PS, PM, PB\}$ at the bottom.

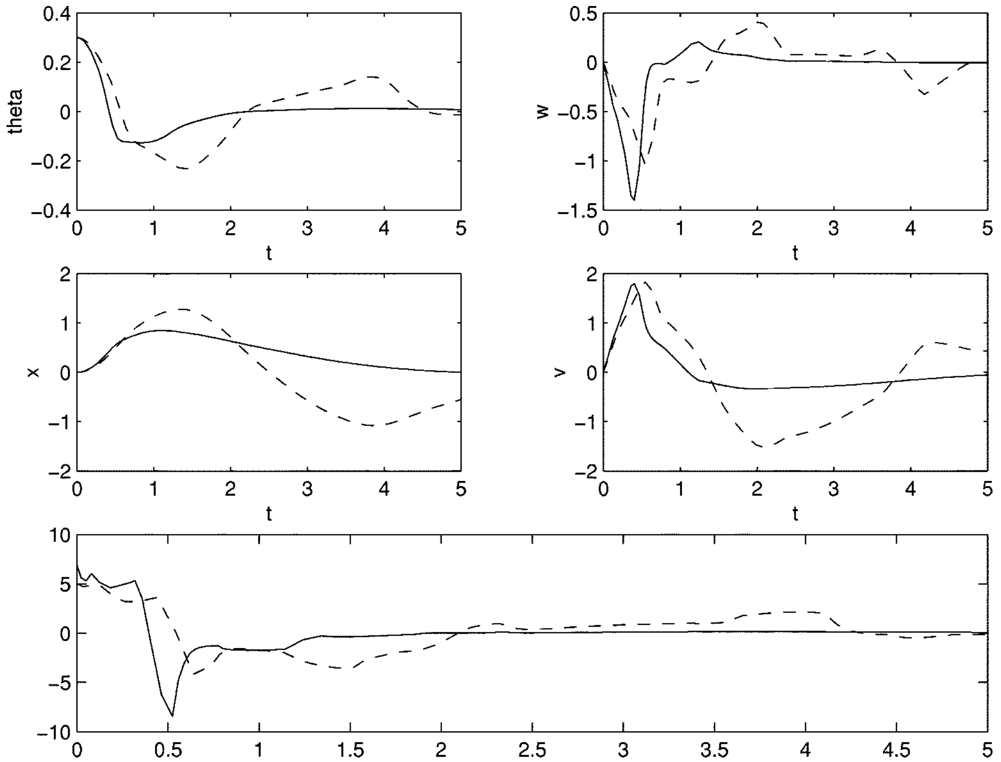


Fig. 9. Time evolution of the state variables θ, ω, x, v and control action F for the original (dashed) and tuned (solid) fuzzy controller.

VI. Evolutionary Design of an Obstacle Avoidance Behavior

An autonomous agent is a behavior-based system that operates in a dynamic environment, as shown in Fig. 10. As the sensors are limited in their perception capacity, the knowledge of the agent about its own state and the state of the environment is usually imprecise and incomplete. Internal states allow the agent to build an internal representation of the ex-

Table 2.
Fuzzy Decision Table

	A_{21}	A_{22}	A_{23}
A_{11}	B_1	B_1	B_2
A_{12}	B_1	B_2	B_3
A_{13}	B_1	B_3	B_4

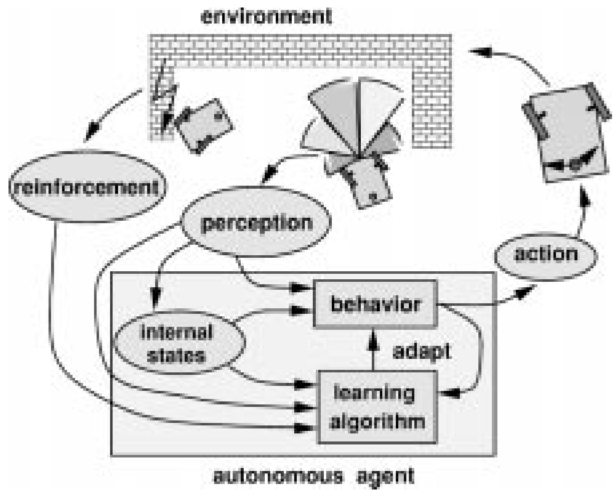


Fig. 10. Perception, action, reward cycle of an autonomous agent.

ternal world. The agent responds to a new perception with an action that affects its future state in the environment. This mapping from perceptive input and internal states to a control action implements a specific behavior of the agent.

From time to time, a teacher provides an external or internal reinforcement signal to the agent that indicates its performance with regard to the given task. The learning algorithm adapts the behavior in order to maximize future reward payoffs. The machine learning problem involves two steps: the identification of states that yield a large payoff, and to learn a control policy that leads the agent to these favorable states.

A. Behavior-Based Robotics

The traditional artificial intelligence approach to robotics assumes a central, usually symbolic representation of the world as the basis for planning, reasoning, and action. The common sense-plan-act scheme builds a world model from its sensory input and generates a plan based on the model, which is then executed. This approach suffers from the drawback that perceptual information is subject to imprecision and uncertainty, which in all but the most predictable worlds makes it impossible to obtain an accurate, complete model of the environment. Prior knowledge about the context in which the robot operates, for example, in form of a map of the environment, provides valuable information for planning, navigation, and localization. However, robot behaviors that entirely rely on prior information about these environments in

Fig. 11. Nomad scout.

order to decide upon the necessary actions are prone to fail as metric maps are approximate and incomplete and the location of objects and landmarks might change over time. In addition, the external environmental conditions, such as lighting, occlusion, and sensor noise, might lead to incorrect interpretation of the sensor data.

In a behavior-based robotic system, perceptions and actions are tightly coupled in a way that strongly depends on an accurate model of the environment [19]. Behavioral responses result from the interaction between perception and action rather than through deliberation based on apparent goals and plans. The two key aspects of the approach are *situatedness*, which means to avoid abstract, symbolic representations, and *embodiment*, which refers to the fact that robots experience the world as physical entities.

Primitive behaviors are the building blocks from which more complex, deliberate behaviors are assembled. The modular structure facilitates the integration of additional behaviors in a way that requires a redesign of the already implemented components. In contrast to many approaches of classical control theory, fuzzy control system design does not depend on a precise, mathematical model of the underlying process, which in particular for mobile robot applications is difficult or impossible to obtain. The robotic behavior is constituted by a set of fuzzy rules that describe the relation among perceptions of the environment and the set of possible actions. The linguistic nature of fuzzy variables and rules allows the expert to formulate his description of the desired robotic behavior in an intuitive fashion.

For complex behaviors, the manual design of a control algorithm becomes a tedious task, as it is often difficult to keep track of the manifold interactions among the incoming sensor data, the physical make-up of the robot, and the environment. A static behavior faces the risk of failure in unforeseen situations that were not considered in the design phase. Evolutionary algorithms provide an alternative for automated learning and tuning of robotic behaviors [24].

In the past, fuzzy control systems have been successfully used to implemented robotic wall following, obstacle avoidance, and navigation behaviors [1], [25], [22], [20]. In addition, roboticists used fuzzy logic for behavior coordination in

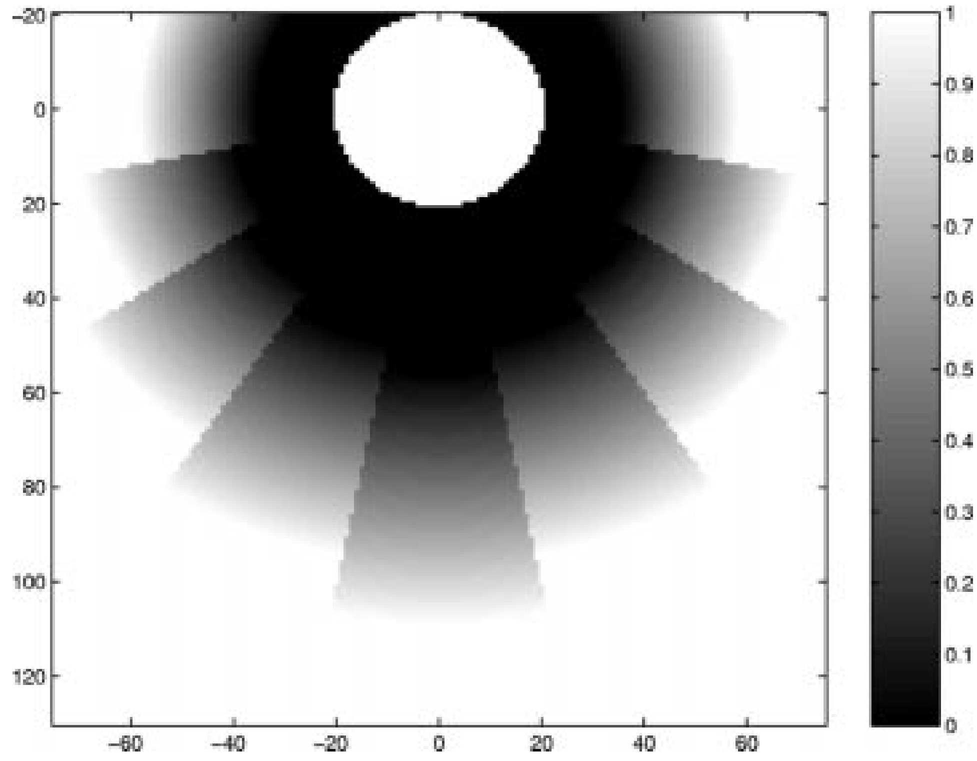


Fig. 12. Robot safety zone.

a way that allows a smooth transition among different behaviors as an alternative to strictly prioritized arbitration scheme [20], [18]. The arbitration policy is formulated by fuzzy context rules that define the degree to which a behavior is activated in a particular situation.

As an alternative to the tedious manual design of robotic systems, evolutionary robotics is concerned with the automatic design of robotic behaviors by imitating the processes that occur in natural evolution [24]. The chromosome encodes parameters of the behavioral controller, for example, the gains associated to the input stimuli. The designer implicitly specifies the desired behavioral properties by means of a scalar objective function that describes how well the robot accomplishes the task at hand. In case of a collision avoidance behavior, for example, the objective function punishes the robot for bumping into an obstacle. In all but a few cases, evolutionary behavior design becomes a trial and error process in which the human designer refines the original objective function upon observing the robotic behavior evolved by means of the evolutionary algorithm.

As evolutionary algorithms typically require a large number of fitness evaluations, most of the adaptation schemes for robotic behaviors utilize a simulation rather than learning in real-time on the robot itself. However, evolving in simulation bears the risk of overadapting the robotic behavior to peculiar features of the simulation scheme. This kind of brittleness is avoided if noise is added to the simulated sensors and actuators, and by grounding the behavior directly on the perception itself rather than some abstract representation of the world.

In the context of evolutionary robotics, fuzzy control systems are a valuable tool to bridge the gap between evolutionary and human design. They allow the engineer to integrate her domain knowledge in the design process and to analyze and interpret the resulting robotic behavior.

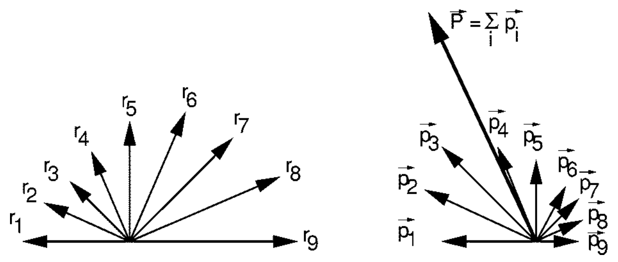


Fig. 13. Perception vector.

B. Mobile Robot

The mobile robot used in the experiments is a Nomad scout, shown in Fig. 11. It is equipped with ultrasonic, tactile, and odometry sensors. A low-level controller governs the sensor data acquisition, motion commands, and communications. The high-level behavioral control either runs on an on-board computer or on a remote workstation that communicates with the robot via radio Ethernet. The robot possesses a two-wheel differential drive located at the geometric center, which allows omni-directional steering at zero turn radius. The platform has a diameter of 41 cm and moves at a speed of up to 1 m/s. Six bumpers for collision detection are arranged around the outside perimeter.

A ring of 16 uniformly distributed sonar sensors measures distances to objects in a range from 15 to 650 cm with a typical error margin of about ±1%. Objects that are larger than the sonar wavelength of ≈7 mm reflect the incoming sound

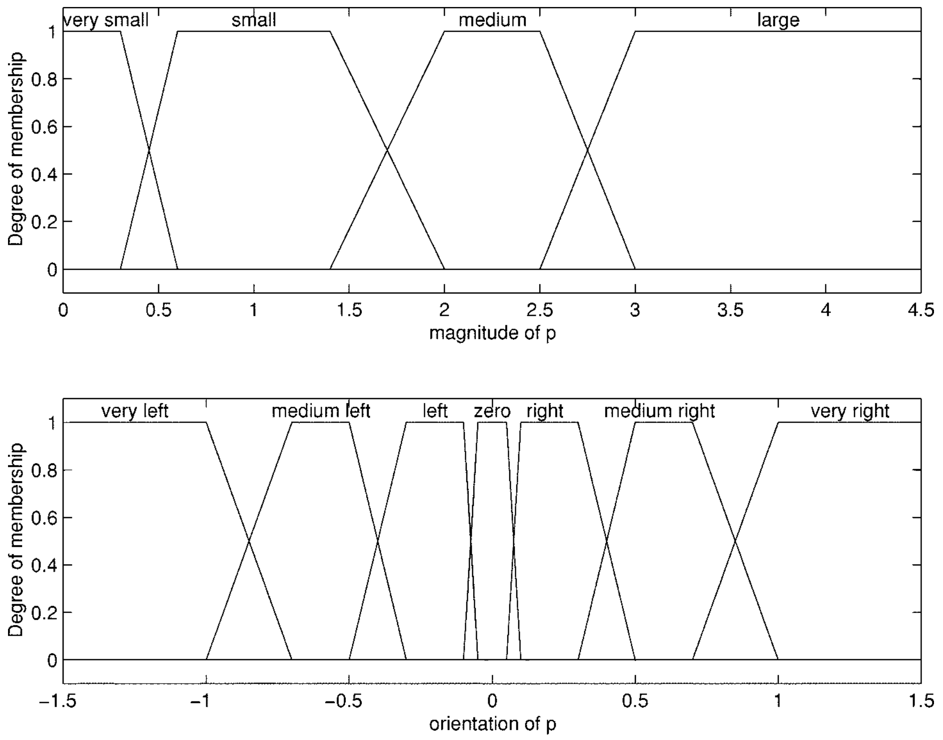


Fig. 14. Membership function for input variables magnitude and orientation of the perception vector $\vec{P}$.

wave, whereas objects smaller than the wavelength act as a diffractor. Objects with smooth surfaces such as walls, doors, or plates act as mirrors with respect to sound waves. Due to this mirror-like property, a sonar echo might undergo multiple reflections before reaching the receiver, which results in a potential overestimation of the distance to the object. Convex sharp edges such as door frames and table legs produce a specular echo. Leonard *et al.* provide a detailed discussion of sonar sensing for mobile robot navigation [26].

The basic motion command for the 2-DOF differential drive specifies the left- and right-wheel velocities for the 2-DOF differential drive. However, for navigation purposes, the desired robot motion is more intuitively described through the commanded translational $v \in [-50 \text{ cm/s}, 50 \text{ cm/s}]$ and turn rate $\omega in \in [-45°/\text{s}, 45°/\text{s}]$ from which the wheel velocities are computed. The robot state is given by its position x, y and orientation ϕ and evolves according to

$$\begin{aligned}\dot{x} &= v\cos\phi\\ \dot{y} &= v\sin\phi\\ \dot{\phi} &= \omega.\end{aligned} \tag{11}$$

The obstacle-avoidance behavior only takes distance measurements of the nine front sonar sensors into account. The translational velocity v and turn rate ω are controlled independent of each other. The translational velocity control is based on the concept of an inner and outer safety zone around the robot, shown in Fig. 12. The white circle at the origin depicts the robot base. The coordinate system is in the robot frame with the current direction of motion pointing downward along the y axis. The gray level indicates the danger of objects perceived under a certain angle and distance. Dark shades indicate a substantial danger of collision, whereas a light shade corresponds to an momentarily negligible or at least not immediately dangerous object.

Notice that the safety zone of the front sonar extends farther than of sonars which major axis points sideways. The overall safety index s is computed as the minimum over one complete cycle of sonar distance measurements $r_1, \ldots, r_9$ as

$$s = \min_i s_i(r_i) \tag{12}$$

where $s_i(r_i)$ is the safety index at a radius r_i along the axis of sonar i. The commanded velocity v assumes a value in the interval $[v_{\min}, v_{\max}] = [1 \text{ cm/s}, 25 \text{ cm/s}]$ in a way that the robot moves slower as the safety index decreases

$$v = v_{\min} + s(v_{\max} - v_{\min}). \tag{13}$$

The low angular resolution of sonar distance measurements makes it difficult to obtain exact, unambiguous geometric information based on a single sonar scan. In order to generate a map of the environment, the robot has to actively explore its environment and collect sonar data from multiple perspectives. However, reactive robotic behaviors, such as obstacle avoidance, do not require a complex representation but rather operate in a stimulus-response mode which establishes the control action directly on the perception rather than a geometric model of the environment.

The relevant perception in an obstacle-avoidance behavior has to somehow capture the proximity and location of the nearest object. Therefore, a complete sonar scan $\{r_1, \ldots, r_9\}$ is aggregated into a planar perception vector,

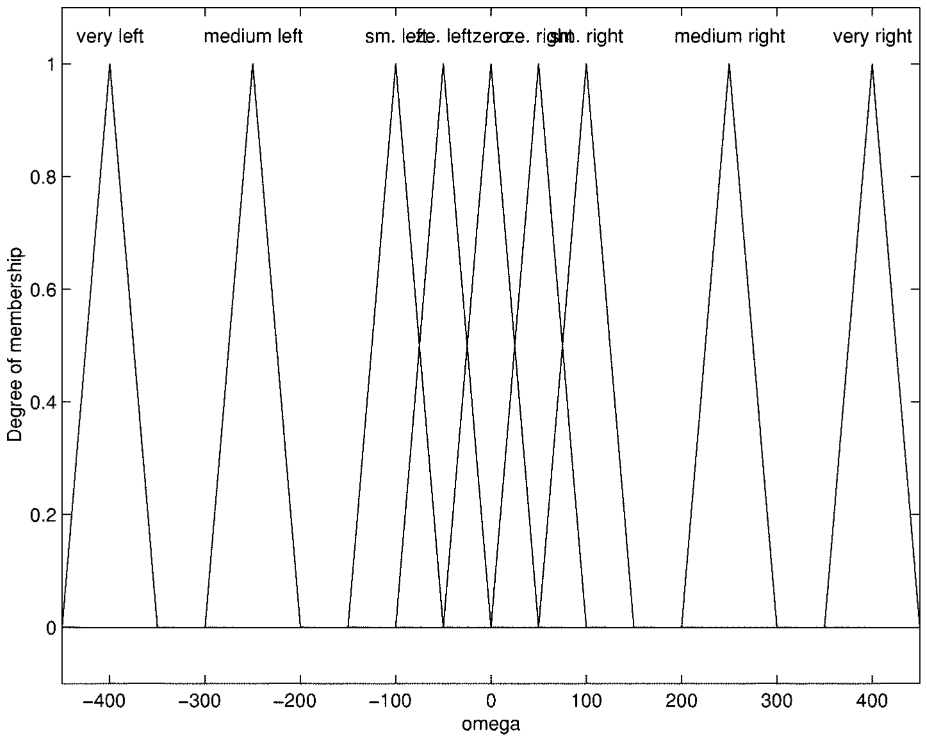


Fig. 15. Membership function for output variable ω.

Table 3.
Decision Table of Manually Designed Fuzzy Controller

ω		$\angle\vec{P}$						
		VL	ML	SL	ZE	SR	MR	VR
	VS	ZE	ZE	ZE	NZ	ZE	ZE	ZE
$\lvert\vec{P}\rvert$	SM	ZE	NZ	NZ	NZ	PZ	PZ	ZE
	ME	ZE	NZ	NS	NM	PS	PZ	ZE
	LG	NZ	NS	NM	NB	PM	PS	PZ

Table 4.
Decision Table of Evolutionary Designed Fuzzy Controller

ω		$\angle\vec{P}$						
		VL	ML	SL	ZE	SR	MR	VR
	VS	ZE	PS	NM	NB	NS	NB	PM
$\lvert\vec{P}\rvert$	SM	NZ	NZ	NB	PZ	NB	NM	NZ
	ME	NZ	PS	NS	PZ	NM	ZE	NB
	LG	PS	PZ	NZ	NZ	ZE	NB	PZ

which direction vaguely indicates the location of obstacles and which magnitude reflects the proximity of the nearest obstacle. A local perception vector $\vec{p}_i$ is calculated for each sonar sensor, as shown in Fig. 13. Its direction coincides with the sensor axis while its length is inversely proportional to the sonar reading $r_i \in [0, d_{\max}]$

$$|\vec{p}_i(t)| = \begin{cases} 0: & r_i > d_{\max} \\ \dfrac{d_{\max} - r_i}{d_{\max}}: & r_i \leq d_{\max} \end{cases} \tag{14}$$

where $d_{\max} = 2\,\mathrm{m}$ specifies an upper range of reaction above which obstacles are ignored. The global perception vector $\vec{P}$ is computed as the vector sum

$$\vec{P} = \sum_{i=1}^{9} \vec{p}_i \tag{15}$$

of the individual perceptions $\vec{p}_i$.

C. Obstacle-Avoidance Fuzzy Controller

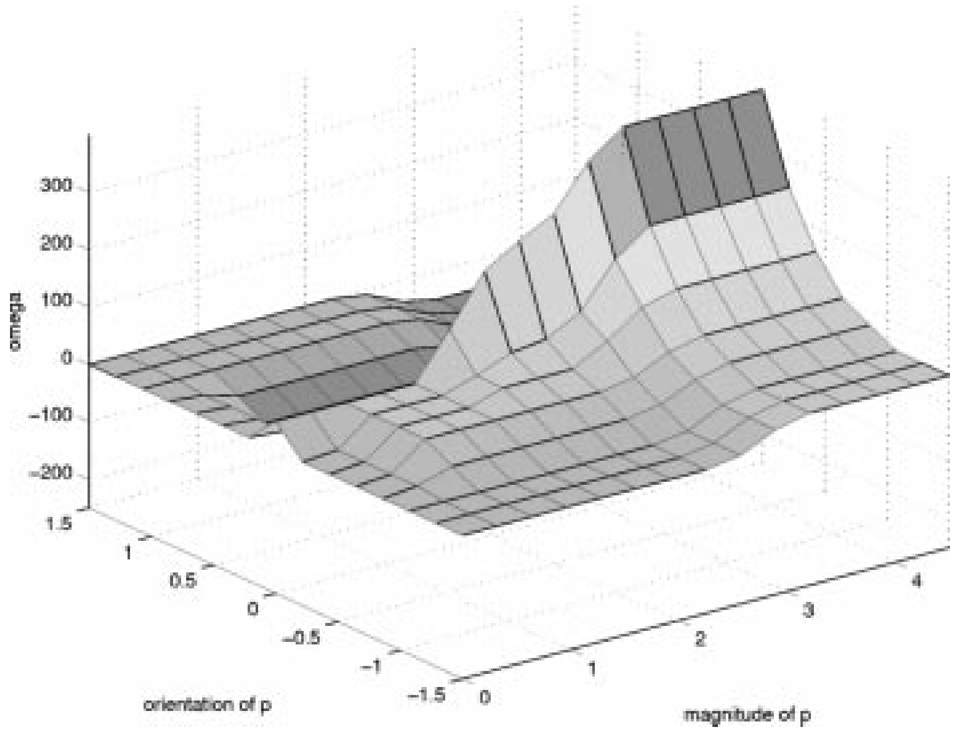
The magnitude $|\vec{P}|$ and orientation $\angle\vec{P}$ of the perception vector $\vec{P}$ constitute the two input variables to the obstacle-avoidance fuzzy controller that regulates the turn rate ω. The input space is partitioned into four trapezoidal fuzzy sets labeled $\{VS, SM, ME, LG\}$ for the magnitude and seven trapezoidal fuzzy sets labeled $\{VL, ML, SL, ZE, SR, MR, VR\}$ for the orientation of the perception vector, as shown in Fig. 14.

The domain of the turn rate ω is partitioned into nine triangular output fuzzy sets zero, negative zero, negative small, negative medium, negative big denoted by $\{PB, PM, PS, PZ, ZE, NZ, NS, NM, NB\}$ in Fig. 15. PB corresponds to a sharp left turn at near maximum rate of 40°/s. The terms $\{PS, PZ, ZE, NZ, NS\}$ correspond to relatively smooth turns with $|\omega| \leq 10°/\mathrm{s}$.

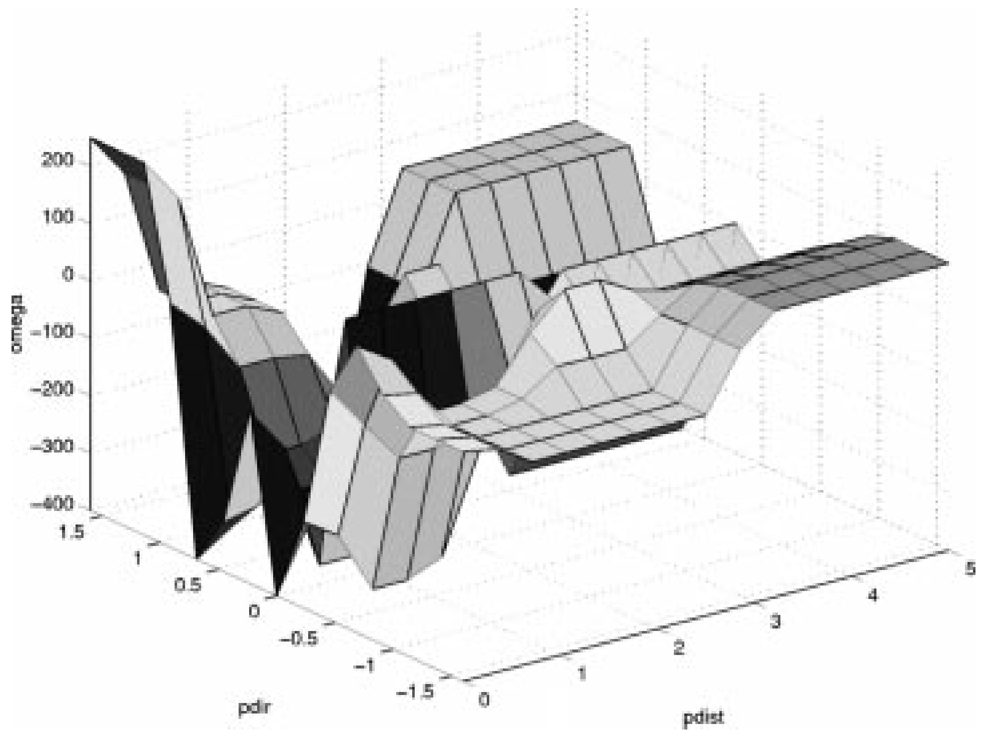
The obstacle-avoidance behavior is implemented by a standard Mamdani fuzzy controller whose rule base contains 28 rules. The decision table of a manually designed controller is depicted in Table 3.

D. Genetic Fuzzy System

The genetic fuzzy system that is utilized for designing the obstacle-avoidance behavior is based on the "Pitt" approach in which the chromosome encodes an entire rule base. Chromosomes $\{s_1, \ldots, s_{28}\}$ are of fixed length and represent



(a)



(b)

Fig. 16. Control surfaces of the (a) manual and (b) evolutionary designed fuzzy obstacle-avoidance controller.

the 28 entries of the decision table coded as integers. Each gene $s_i \in [1, 2, \ldots, 9]$ encodes a possible output fuzzy label $\{PB, PM, PS, PZ, ZE, NZ, NS, NM, NB\}$ for the corresponding entry of the fuzzy decision table. The mutation operator shifts the integer either up or down, such that the offspring output fuzzy label for a particular rule is always a neighbor of the parent label. For example, mutation changes the label PM either to PB or PS. Despite its positional bias one-point crossover is used for recombination, as the influence on the behavior of fuzzy rules that are neighbors in the decision table is likely to be correlated.

Each controller is evaluated on the Nomadic simulator starting from different initial positions. The training environment contains cluttered obstacles of various dimensions. A single run either stops if the robot collides with an obstacle as indicated by the bumper state or until a maximum number $t_{\max}$ of control steps elapses. In order to demonstrate a meaningful obstacle-avoidance behavior, the robot is only supposed to turn when an obstacle is present and otherwise travel as straight as possible. In order to prevent the evolution of a behavior that simply gyrates the robot on the spot, the fitness increases as the average turn rate ω decreases. The fitness attributed to the behavior for a single control step t_i is

$$f(t) = \frac{v(t)}{v_{\max}} * \frac{\omega_{\max} - |\omega(t)|}{\omega_{\max}}. \tag{16}$$

The fitness $f(t) \in [0, 1]$ increases with decreasing absolute turn rate $|\omega(t)|$ and increasing velocity $v(t)$. Notice that the fuzzy controller only governs ω but has no direct control over $v(t)$, which is regulated externally according to the safety index. However, by timely steering the robot away from obstacles, the fuzzy controller is able to affect the future safety index and indirectly the translational velocity as well. The objective of the adaptation process is to find a suitable compromise that minimizes the steering action while at the same time maintains a safe distance to obstacles.

In addition, the fitness function penalizes collisions with obstacles. The fitness values $f(t)$ of individual control steps are accumulated over time

$$f = \sum_{t=0}^{\min\{t_{\text{coll}}, t_{\max}\}} f(t). \tag{17}$$

Fitness accumulation either terminates upon collision t_{coll} or after the maximum allowed time $t_{\max}$ in case no collision occurs. A behavior that results in a collision is deprived of the possibility to collect rewards in future control steps ($t_{\text{coll}} < t < t_{\max}$). Therefore, behaviors that reliably avoid a collision achieve a better overall fitness even though they might gather less fitness per individual control step. Finally, the fitness values f_i computed for the individual trials i are aggregated into an overall fitness F. In order to emphasize the evolution of a robust behavior, the total fitness F used in the selection step becomes

$$F = \frac{\langle f_i \rangle + \min_i f_i}{2} \tag{18}$$

the mean of the average fitness $\langle f_i \rangle$ and the worst fitness $\min_i f_i$.

The population size is 30 individuals, the initial generation is seeded randomly in that the nine output label occur with equal probability. Table 4 shows the best decision table that emerged in the course of 50 generations. The resulting control surface is depicted in Fig. 16. In general, the evolved fuzzy controller tends to steer the robot to the right ($\omega < 0$) in order to avoid an obstacle. In those cases in which the robot turns left (PZ, PS, PM, PB), the magnitude of ω is usually smaller than for right turns (NZ, NS, NM, NB), as nine out of 28 entries in the decision table correspond to sharp left turns (NM, NB) but in only one case the robot turns sharp right (PM, PB). It is a sensible strategy to prefer one turning direction over the other as a symmetric behavior bears the risk of cornering the robot by alternating left–right turns.

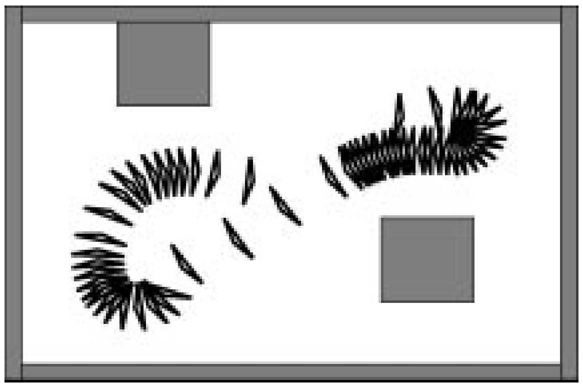


Fig. 17. Robot path for obstacle-avoidance behavior with the fuzzy decision table in Table 4.

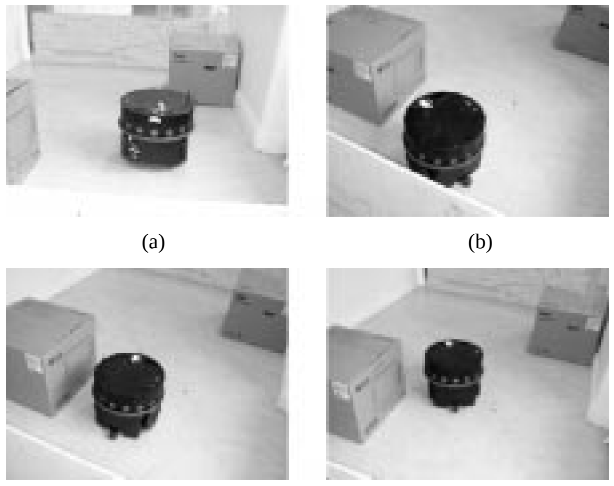

(a) (b) (c) (d)

Fig. 18. Robot during experiment.

Fig. 18 shows images from a robot experiment in a real environment. Fig. 17 depicts the path described by the robot. A snapshot of the robot location and orientation is plotted every ten control steps based on the odometry data obtained from the wheel encoders. Notice that the distance between two snapshots indicates the velocity at which the robot is moving. A wide gap corresponds to a large velocity, densely spaced snapshots indicate a slow translational motion. For reasons of clarity, the figure only depicts the first 40 snapshots of the entire trajectory.

The robot starts in the top right corner facing east [see Fig. 18(a)]. It moves straight ahead until it encounters the right wall and performs a 180° turn [see Fig. 18(b) and (c)]. As the robot finds more space after passing the rectangular obstacle it moves diagonally across the room at a large velocity [see Fig. 18(d)]. It again slows down as it reaches the lower left corner and performs another sharp right turn. In the following the robot continues along a loop (not shown) in which it performs right turns similar to the ones shown in the figure.

VII. Conclusion

This paper provides an overview on evolutionary algorithms for fuzzy control system design and optimization. Genetic fuzzy systems are a valuable tool to tune a fuzzy controller based on a cost functional that captures the user-specified performance criteria. We applied an evolution strategy to tune the scaling factors and membership function of a fuzzy controller for the cart-pole balancing problem. The genetic tuning process was able to reduce the cost functional that measures how quickly the controller stabilizes the cart-pole system by 56% (input scaling factors) and 48% (output membership functions).

Genetic learning processes adapt the fuzzy rule base with the objective to automate the knowledge acquisition step in fuzzy control system design. The evolutionary design approach was applied to learn an obstacle-avoidance behavior of a mobile robot. The obstacle-avoidance behavior adapted in simulation was successfully transferred to the real robot, where it demonstrated the same performance in previously unseen environments. Due to the robustness of the fuzzy controller the behavior needed no further adaptation on the robot, despite the limited quality of the simulation environment that did not capture the uncertainty and imprecision inherent to real-world situations. However, the purely reactive obstacle-avoidance behavior did not pose a particularly challenging task, and it remains an open question whether evolutionary techniques are able to design more complex robot behaviors that are otherwise difficult to conceive manually.

References

[1] G. Castellano, G. Attolico, and A. Distante, "Automatic generation of fuzzy rules for reactive robot controllers," *Robot. Auton. Syst.*, vol. 22, pp. 133–149, 1997.

[2] C. M. Higgins and R. M. Goodman, "Fuzzy rule-based networks for control," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 82–88, Feb. 1994.

[3] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, Feb. 1992.

[4] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, May-June 1993.

[5] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. Chichester, U.K.: Wiley, 1997.

[6] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. London, U.K.: Oxford Univ. Press, 1996.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[8] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, "Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases," in *Advances in Fuzzy Systems*. Singapore: World Scientific, 2001.

[9] C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 6, pp. 26–33, Feb. 1991.

[10] M. A. Lee, "Automatic design and adaptation of fuzzy systems and genetic algorithms using soft computing techniques," Ph.D. dissertation, Univ. California, Berkeley, 1994.

[11] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 39–47, Jan. 1994.

[12] F. Herrera, M. Lozano, and J. L. Verdegay, "A learning process for fuzzy control rules using genetic algorithms," *Fuzzy Sets Syst.*, vol. 100, pp. 143–158, 1998.

[13] F. Hoffmann and G. Pfister, "Learning of a fuzzy control rule base using messy genetic algorithms," in *Genetic Algorithms and Soft Computing*. ser. Studies in Fuzziness and Soft Computing, F. Herrera and J. L. Verdegay, Eds. Heidelberg, Germany: Physica-Verlag, 1996, vol. 8, pp. 279–305.

[14] L. Magdalena and F. Monasterio, "A fuzzy logic controller with learning through the evolution of its knowledge base," *Int. J. Approx. Reason.*, vol. 16, pp. 335–358, 1997.

[15] A. Bonarini, "Evolutionary learning of fuzzy rules: Competition and cooperation," in *Fuzzy Modeling: Paradigms and Practice*, W. Pedrycz, Ed. Norwell, MA: Kluwer, 1996, pp. 265–284.

[16] H. Hagras, V. Callaghan, and M. Colley, "Online learning of fuzzy behaviors using genetic algorithms and real-time interaction with the environment," in *Proc. IEEE Int. Fuzzy Systems Conf.*, vol. II, Seoul, Korea, 1999, pp. 668–673.

[17] F. Hoffmann and G. Pfister, "Evolutionary design of a fuzzy knowledge base for a mobile robot," *Int. J. Approx. Reason.*, vol. 17, no. 4, pp. 447–469, 1997.

[18] E. Tunstel, T. Lippincott, and M. Jamshidi, "Behavior hierarchy for autonomous mobile robots: Fuzzy-behavior modulation and evolution," *Int. J. Intell. Autom. Soft Comput.*, vol. 3, no. 1, pp. 37–50, 1997.

[19] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, pp. 14–23, Feb. 1986.

[20] A. Saffiotti, K. Konolige, and E. H. Ruspini, "A multi-valued logic approach to integrating planning and control," *Artif. Intell.*, vol. 76, no. 1–2, pp. 481–526, 1995.

[21] F. Herrera, O. Cordón, and M. Lozano, "On the combination of fuzzy logic and evolutionary computation: A short review and bibliography," in *Fuzzy Evolutionary Computation*, W. Pedrycz, Ed. Norwell, MA: Kluwer, 1997, pp. 57–77.

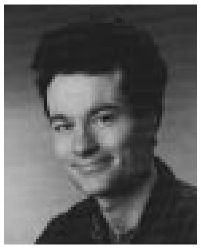
[22] F. Hoffmann, "Soft computing techniques for the design of mobile robot behaviors," *J. Inform. Sci.*, vol. 122, pp. 241–258, Feb. 2000.

[23] ——, "The role of fuzzy logic in evolutionary robotics," in *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*. ser. Studies in Fuzziness and Soft Computing, D. Driankov and A. Saffiotti, Eds. Berlin, Germany: Springer-Verlag, 2000.

[24] S. Nolfi and D. Floreano, *Evolutionary Robotics—The Biology, Intelligence, and Technology of Self-Organizing Machines*, ser. Intelligent Robotics and Autonomous Agents. Cambridge, MA: MIT Press, 2000.

[25] D. Driankov and A. Saffiotti, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, ser. Studies in Fuzziness. Berlin, Germany: Springer-Verlag, 2000.

[26] J. Leonard and H. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Norwell, MA: Kluwer, 1992.



Frank Hoffmann received the Ph.D. degree in physics from the University of Kiel, Germany, in 1996.

From 1996 until 2000, he was a visiting researcher at the Computer Science Division, University of California, Berkeley. He is currently a Research Associate and Lecturer at the Center for Autonomous Systems of the Royal Institute of Technology, Stockholm, Sweden. His current research interests are in the areas of machine learning, genetic fuzzy systems, and behavior-based robotics.