

The Role of Fuzzy Logic Control in Evolutionary Robotics

Frank Hoffmann

Electrical Engineering & Computer Science Department
University of California, Berkeley
Berkeley, CA 94720
Email: fhoffman@cs.berkeley.edu

Abstract. This paper presents an evolutionary learning algorithm to facilitate the design of fuzzy controllers for mobile robots. It discusses the concepts, feasibility, benefits and limitations of current evolutionary techniques for fuzzy rule discovery and tuning. We propose an evolution strategy that optimizes the gain factors in the conclusion part of Takagi-Sugeno-Kang type fuzzy rules.

We describe two applications of genetic-fuzzy systems in detail, adapting a wall-following behavior of a mobile robot and designing an autopilot for a small-size model helicopter.

1 Introduction

Traditional AI approaches decompose robotic behaviors into a sense-model-plan-act type of hierarchy. The sensors provide perceptual information, which is used to build a model of the current environment. The planner generates a plan that enables the robot to accomplish the given task. A controller executes the actions commanded by the planner without taking novel sensor information into account. The utility of this model-based reasoning approach for the design of intelligent robots is limited due to uncertainties inherent to unstructured environments, unreliable and incomplete perceptual information and imprecise actuators.

Fuzzy systems employ a mode of approximate reasoning which enables them to make robust and meaningful decisions under uncertainty and partial knowledge. Therefore, the difficulties arising from the lack of precise and complete information on the environment make fuzzy control a suitable method to implement the behavior of a mobile robot. The robotic behavior is constituted by a set of fuzzy rules, which can be designed without requiring the complexity and precision of mathematical or logical models. The fuzzy rules describe the relation between the external and internal states of the robot and the set of possible actions. Fuzzy logic systems represent knowledge in linguistic form, which permits the designer to define a highly abstract behavior in an intuitive fashion.

Usually, a robotic behavior is designed with regard to a typical environment in which the robot is supposed to operate. The performance might deteriorate if the robot is transferred to an environment that differs from the prototype. In order to achieve true autonomy a robot depends on the ability to adapt its behavior

to changes in the environment. Training the robot with noisy and incomplete sensor information enhances the robustness of the behavior. A robot that learns from past experience can exploit particular regularities of its environment and perceptual apparatus and thereby improve its competence to achieve its goals.

For complex behaviors, the manual design of a control algorithm becomes a tedious task, since it is often difficult to obtain an exact model of the manifold interactions between the sensory input, the mechanics of the robot, the control system and the environment. A hard-wired behavior bears the risk of failure in novel situations overlooked during the design phase. As an alternative to traditional engineering design evolutionary computation provides a means for automatic behavior generation and tuning.

Soft computing is concerned with the design of adaptive, intelligent and robust systems, which imitate the human mind in its ability to deal with imprecise, incomplete knowledge and partial truth. In order to achieve this objective, soft computing employs a variety of methodologies, fuzzy logic, neural networks, probabilistic reasoning and genetic algorithms. Soft computing proposes to merge the advantageous features of complementary methods into hybrid systems. Recently, numerous publications propose to integrate the learning capabilities of evolutionary algorithms with the knowledge representation of fuzzy systems [8][7][11][30][32] [33]. These methods are described by the general term genetic-fuzzy systems. An evolutionary algorithm automates the knowledge acquisition step in fuzzy control design. In this context learning a fuzzy knowledge base becomes an optimization problem in which the search space is constituted by the fuzzy membership functions and fuzzy if-then rules.

In our case, the mobile robot perceives its environment by means of five with ultrasonic sensors. The local navigation behavior is based on the concept of general perception, which replaces a precise geometric model of the environment. An evolution strategy learns a wall-following behavior implemented by fuzzy control rules. Experiments with the physical mobile robot demonstrate, that an evolutionary algorithm can in principle design a robust fuzzy control system that is able to cope with the uncertainty and imprecision inherent to real-world situations.

In contrast to the significant amount of research concerned with fuzzy control of mobile ground robots, only a few publications describe the use of fuzzy techniques for unmanned aerial vehicles. SUGENO ET AL. were the first who designed and implemented a fuzzy control system for an automated helicopter flight [29]. PHILLIPS ET AL. [25] proposed a genetic algorithm to determine a fuzzy rule set for effective control of UH-1 helicopters through various manoeuvres.

This paper describes the evolutionary tuning of fuzzy controllers in an autopilot for a small-size model helicopter. The autopilot constitutes the continuous regulation layer of a hybrid flight vehicle management system. It is composed of four modules which control the helicopter altitude, heading, lateral and longitudinal motion. Each module is optimized in a simulation of typical helicopter manoeuvres, which command the controller to track a sequence of set points.

2 Problem Statement

Recently, numerous approaches propose the integration of human design and machine learning techniques for behavior engineering of autonomous robots. Automated learning of the robotic behavior exempts the human designer from a tedious testing and tuning process. It is often impossible to derive an accurate and complete model of the interaction among the robot, its sensors and the environment. In time-invariant classical control problems the boundary between the system composed of the controller and plant and the external world is well defined. Mobile robots are situated in a more or less unstructured environment, which is intrinsically intertwined with the robotic dynamics, perceptions and actions. For instance, control actions do not only affect the next state of the robot but also influence future perceptions of the environment.

Usually, the human designer has some qualitative a priori knowledge about the task, sensor characteristics and type of environment, which serves as a guideline for behavior design. Behavior analysis provides a valuable tool to identify the sensory perceptions, preprocessing of raw sensor data and selection of actions that are relevant for the robot to accomplish its mission.

For instance, in case of an obstacle avoidance behavior common sense suggests that the magnitude of the evading manoeuvre should increase with the proximity of the obstacle and that objects along the current direction of motion require more attention than obstacles located at the periphery. This kind of qualitative knowledge can be easily expressed in form of linguistic fuzzy if-then rules. The entire fuzzy rule base determines the relationship among sensor information and the control action to be applied in the current context. However, such a manually designed fuzzy controller might be far from optimal, especially for behaviors that involve multiple conflicting objectives.

Machine learning techniques, such as reinforcement learning or evolutionary algorithms provide a valuable tool to refine the hand-designed behavior in regard to performance specifications, generalization ability and robustness towards uncertainty and incomplete information. The general framework for learning mechanisms that incrementally adapt a robotic behavior is depicted in Fig. 1.

An autonomous agent is a behavior-based system that is situated in a dynamic environment, which it perceives by means of sensors. Usually, the sensors provide only vague and partial information about the environment. Internal states allow the agent to build an internal representation of the external world. The agent processes data from consecutive perceptions to refine its estimate of the current environmental context. Based upon the sensory perception and the internal states the agent generates an appropriate control action that affects its state in the environment. This mapping from perceptive input and internal states to a control action realizes a specific behavior of the agent.

From time to time the environment provides a reinforcement signal that indicates the performance of the agent in 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 with

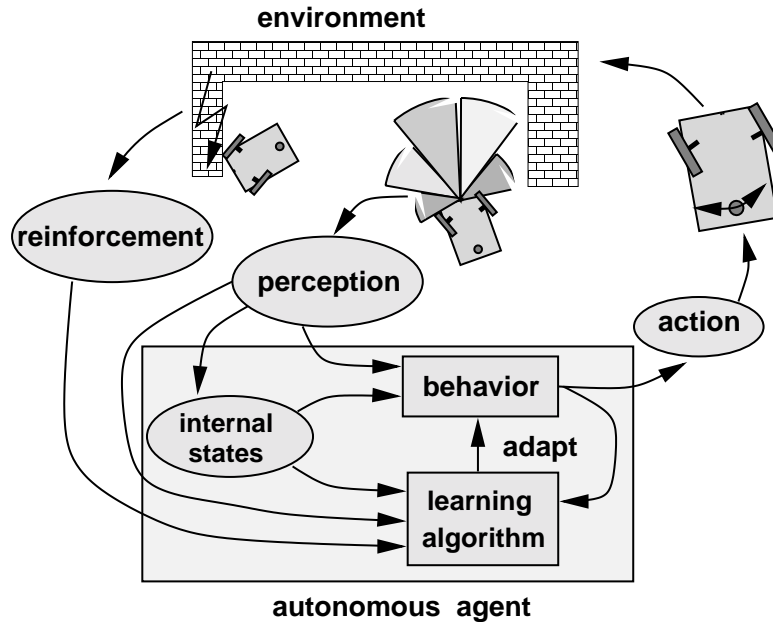


Fig. 1. Interaction of the autonomous agent with its environment

high payoff and to learn a control policy that drives the system to these favorable states.

Assume the robotic platform, environment and sensory apparatus are given. The fundamental questions faced by the designer during the conception of the autonomous agent architecture are

- How to process the raw sensor data and which internal states adequately represent the relevant features of the external world?
- Which structure is used to implement the behavior and which parameters of this structure are subject to adaptation?
- How to adapt the behavior? This includes the issue whether learning takes place on the real robot or in a simulation of the robot and its environment.
- Often, the agent only receives sporadic reinforcement after executing a sequence of multiple actions. How to assign credit to individual actions based on the delayed reinforcement?

A general discussion of these problems and an overview on proposed approaches is beyond the scope of this paper; the interested reader is referred to [21]. The remaining part of this paper describes how evolutionary robotics and in particular genetic fuzzy systems try to address these issues.

2.1 Evolutionary Robotics

Evolutionary robotics is concerned with the design of intelligent systems with

life-like properties by means of simulated evolution. Evolutionary computation automatically synthesizes controllers that enable the robot to perform useful tasks in complex environments. An evolutionary algorithm searches through the large space of behaviors in order to find a controller that maps sensory perceptions to suitable control actions. A population of candidate controllers is processed from one generation to the next by means of selection, recombination and mutation. A scalar fitness function describes the performance of the controller in regard to the desired mission of the robot.

Numerous examples of real-world robots demonstrate that evolutionary algorithms are in principle able to facilitate the design of robot control systems [5][10][13][23] [18][32]. On the other hand, most of the evolutionary techniques presented in the literature so far generated rather simple behaviors. Often the development of the evolutionary algorithm itself surmounts the expenditure of work required to design the control system by hand. The litmus test for evolutionary robotics becomes the automatic generation of more complex behaviors that demonstrate more robustness and tolerance to uncertainty than manually designed control systems.

An essential distinction among the approaches in evolutionary robotics is in the genetic structure that undergoes adaptation. CLIFF ET AL. use a neural network controller which architecture is evolved by a genetic algorithm [5]. COLOMBETTI ET AL. propose classifier systems as another means for the design and adaptation of robotics behaviors [6]. GREFENSTETTE ET AL. designed the SAMUEL learning system, which is based on sequential stimulus-response rules to control a simulated robot [10]. Recently, NORDIN ET AL. propose genetic programming as a learning technique for robotic behavior adaptation [23]. The behavior is implemented by a tree-structured program, which maps perceptions to control actions.

Several authors suggested the combination of evolutionary algorithms with fuzzy control for automated robot behavior design [3][4][13] [33]. These so called genetic fuzzy systems bridge the gap between engineering and evolution in that they allow the designer to incorporate domain knowledge into the learning process. The essential part in fuzzy controller design is the acquisition of the knowledge base. The knowledge base of a FLC is usually derived by observing or interviewing a human expert controlling the system. A learning process aiming to achieve a desired system behavior can automate this knowledge acquisition step. When learning is applied to a rule-based controller, the objective becomes to meet user specified performance criteria of the system. In this context, learning a fuzzy knowledge base can be regarded as an optimization problem in which the set of possible fuzzy rules and fuzzy sets constitutes the search space.

Evolutionary optimization techniques are usually highly general and robust and therefore applicable to a broad spectrum of optimization tasks. On the other hand they are often inferior to specialized optimization techniques, such as gradient descent, designed for a narrow spectrum of problems. In this context, fuzzy logic offers a framework to incorporate problem specific domain knowledge in order to enhance the evolutionary search process. Fuzzy systems and evolutionary

algorithms can be hybridized in many different ways, for example optimization of fuzzy rule-based systems or tuning of parameterized fuzzy sets from training data.

The human expert formulates her domain knowledge either in terms of linguistic variables or fuzzy if-then rules. In the first case, the user determines the context of optimization by defining the fuzzy database, including the number of variables, number of fuzzy sets and membership functions. For many real-world problems, in particular robot behavior design, a mathematical precise and complete solution is not only evitable, but also often infeasible. Fuzzy systems exploit this tolerance for imprecision by aggregating similar states into coarse granules. The remaining design task is to find the correct set of fuzzy if-then rules, rather than learning a general analytic function from sensory perceptions to control actions. Furthermore, a granular representation avoids the tendency of evolutionary algorithms to generate brittle solutions that are over-adapted to peculiar features of the problem.

In addition to the fuzzy database, the human expert can use his domain knowledge about the robot's mission and environment to find an initial set of fuzzy rules. Usually, this initial rule-set does not constitute a complete solution but merely a small collection of primitive features supposedly beneficial to achieve the desired behavior. The rules are seeded into the initial population in order to facilitate the search during the first few generations. The user-defined rules provide the building blocks from which the evolutionary algorithm generates the entire behavior through refinement and superposition of additional rules.

2.2 Mobile Robot



Fig. 2. Mobile robot

The mobile robot depicted in Fig. 2 is supposed to demonstrate a wall-following behavior in an indoor environment. The environment is formed by segments

with different geometries and dimension, such as right-angled turns, dead-ends, corridors and obstacles.

The robotic motion is governed by the non-holonomic dynamics of a unicycle

$$\begin{aligned} \dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \omega = \frac{\Delta\phi}{\Delta t} \end{aligned} \tag{1}$$

with the two inputs velocity v and yaw rate ω . Stepping motors drive the two front wheels with independent velocities u_1, u_2 . The robot performs a right turn by reducing the speed u_1 of the right wheel. Instead of using u_1, u_2 directly as control actions, the controller computes the change in heading $\Delta\phi = \omega * \Delta t$ during the next control step of duration Δt . The spectrum of manoeuvres ranges from a strict forward motion to a sharp turn on the spot in case of counter-rotating front wheels. Although the robot is in principle able to move backwards, such a manoeuvre is not recommended since there are no sensors pointing backward. The location of the axis of rotation located at the front of the robot bears the inherent drawback that the rear of the robot describes a wide swing during turns. This unfavorable characteristic considerably complicates manoeuvres in confined space.

The robot perceives its environment by means of five ultrasonic sensors mounted on the robot's front end. An individual sonar sensor covers an angular range of approximately $\pm 25^\circ$. The chance of omitting an echo signal increases for objects located at the surroundings of the perceptive area. The sensor axes can be adjusted in order to optimize the perception of obstacles. The optimal configuration is a compromise between a sufficient overlap of the individual sensors and a wide total field of vision. The genotype embodies an additional gene for the sensor orientation [16].

The limited angular resolution of ultrasonic sensors and the occurrence of multiple reflections makes it difficult to extract an accurate geometric model of the environment from the sensor data. In this book FABRIZI ET AL. present an online algorithm that generates a fuzzy map of the environment from a series of ultrasonic distance measurements [9]. The environment is represented by a fuzzy set defined over an occupancy grid of points. Fuzzy logic provides an adequate means to cope with the uncertainty and incompleteness inherent to the ultrasonic sensing process. A path-planning algorithm utilizes the fuzzy map to compute a collision-free robot trajectory.

In this paper we follow a different approach to robotic behavior design based on a stimulus-response relationship among perceptions and control actions rather than a deliberative control strategy. The concept of *general perception* introduced by BRAUNSTINGL ET AL. [4] attempts to model the environment indirectly through a representation of the perception itself rather than reconstructing its geometry in a straightforward way. Bypassing the explicit modeling of the environment facilitates the behavior design and avoids the necessity of planning a feasible trajectory. On the other hand, an internal representation that utilizes a

restricted world model sometimes prevents the generation of sophisticated behaviors necessary for more complex missions and environments.

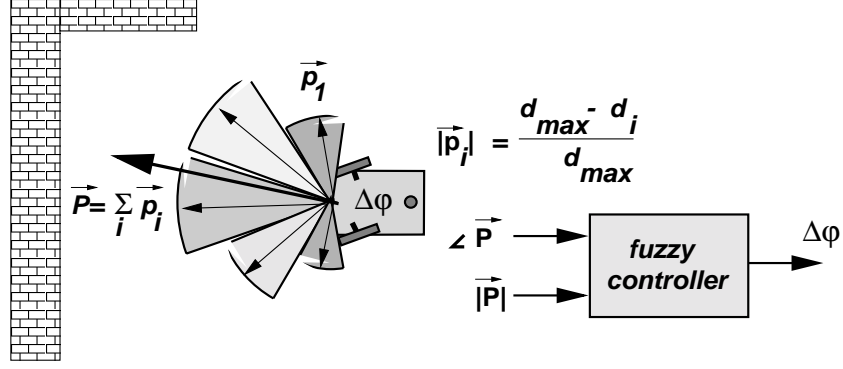


Fig. 3. Mapping from sonar distance measurements to perception vector

In our scenario, the agent maps the distance information provided by the five sensors into a two-component perception vector defined in the body coordinate system as depicted in Fig.3. The bearing of the perception vector vaguely indicates the direction of nearby walls and obstacles. Its magnitude reflects the proximity of the nearest objects.

Each ultrasonic sensor computes its own local perception vector $\mathbf{p}_i(t)$, which direction coincides with the sensor axis while its length increases for decreasing distance measurements $d_i(t) \in [0, d_{max}]$:

$$|\mathbf{p}_i(t)| = \frac{d_{max} - d_i(t)}{d_{max}} \quad (2)$$

where $d_{max} = 2m$ specifies the upper range limit of the sonar sensors. The overall perception vector $\mathbf{p}(t)$ equals the geometric resultant of the individual perceptions $\mathbf{p}_i(t)$

$$\mathbf{p}(t) = \sum_{i=1}^5 \mathbf{p}_i(t) \quad (3)$$

Sometimes, a sensor fails to detect an echo signal, especially if it is reflected into the boundaries of its perceptive region. The sensors tend to underestimate the actual proximity of a corner for echo signals that undergo multiple reflections. Therefore, it seems inadvisable to establish the control decision on a single set of sensor measurements. A time-weighted average of previous sensor measurements improves the reliability of the perceptual information.

The perception vector $\mathbf{P}(t)$, ultimately used for control, is computed as a weighted means of the current perception $\mathbf{p}(t)$ and the previous perception vector $\mathbf{P}(t - \Delta t)$.

$$\mathbf{P}(t) = \frac{\mathbf{p}(t) + \alpha \Omega(-\Delta\varphi(t)) \mathbf{P}(t - \Delta t)}{1 - \alpha} \quad (4)$$

Notice, that during a control step of time Δt the robot turns by the amount $\Delta\varphi$. Under the assumption of a stationary environment, the previous perception is therefore mapped into the current body system by means of the rotation matrix $\Omega(\Delta\varphi(t)) \in SO_2$. The decay factor α determines the weight attributed to the previous perception compared to the current sensor information. In addition to the fuzzy control rules, the evolution strategy also optimizes the decay factor α and the orientation of the sensor axes [16].

The robot reacts to the perceived input by a change in heading $\Delta\varphi$ during the next control interval. A fuzzy controller maps the perception vector \mathbf{P} to a desired change in heading $\Delta\varphi$, which is converted into the angular velocities of the driven wheels.

The conception and design of a fuzzy controller for the wall-following behavior does not pose an extremely difficult engineering problem. Nevertheless, the particular realization of the robot causes additional challenges

- The perception vector provides only limited information on the environment, for instance it does not discriminate sufficiently between right-angle turns and dead-ends or does not indicate if a major obstacle in a corridor prevents the robot’s passage.
- The corridors vary in their dimensions and geometry, so that a controller needs to generalize over different environments. Adaptation to a particular prototypical environment bears the risk of brittleness and failure in novel situations.
- Turns in confined space require precise and smooth manœuvring because of the wide swing described by the robot’s tail.
- Noisy and incomplete sensor data can mislead the robot about its actual situation.

2.3 Helicopter Autopilot

Unmanned autonomous aerial vehicles (UAV) have been found indispensable for various applications where human intervention is considered difficult or dangerous. A helicopter operates in different flight modes, such as vertical take-off/landing, hovering, longitudinal/lateral flight, pirouette, and bank-to-turn. Due to its versatile in maneuverability, a helicopter is capable to manœuvre in and out of restricted areas and to hover efficiently for long periods of time. These characteristics make helicopter invaluable for terrain surveying, surveillance and clean up of hazardous waste sites.

BERkeley AeRobot team, *BEAR*, has built an autonomous helicopter and is currently developing a planning and control system based on hybrid system theory [20]. A Flight Vehicle Management System (FVMS) resolves conflicts among air vehicles, plans the flight path, generates a proper sequence of flight modes, calculates a feasible trajectory and regulates the helicopter along the nominal trajectory. The design and implementation of a stable and reliable autopilot constitutes a mandatory step in the conception of the hybrid FVMS.

The design of an autopilot for an aerial robot poses different challenges than controller design for a mobile ground-based robot. A helicopter is inherently unstable and the degrees of motion are coupled. A helicopter requires constant corrective control actions to maintain its steady state, whereas a mobile robot is able to stop in case it needs additional time to reason about the next action.

The external influence of the environment becomes less significant, at least as long as the air space of helicopter operation contains no other objects than the ground. On the other hand, the dynamics of the aerial robot become far more complex. The autopilot design is based on a nonlinear helicopter model valid in the hover and low velocity regime [?]. The dynamic equations of motion are obtained by equating the linear and angular momentum of a 6-DOF rigid body motion with the forces and moments generated by the main and tail rotor. The helicopter is controlled by four inputs: main rotor collective δ_{coll} , longitudinal δ_{long} and lateral δ_{lat} cyclic pitch, and tail rotor collective pitch δ_{tr} . Servo actuators are linked to these control surfaces and are modeled by first-order transfer functions.

The main rotor pitch regulates the vertical thrust, which lifts the helicopter. An independent engine governor controls the engine throttle in order to maintain a constant rotor speed. The longitudinal and lateral cyclic pitch of the main rotor govern the pitch and roll angle, which on their part determine the forward and side-ward motion of the helicopter. Finally, the tail rotor pitch controls the yaw motion.

Choosing the position and the yaw angle as the four control outputs decouples the dynamics and reduces the complexity of the control design. Consequently, the autopilot is composed of four separate modules depicted in Figs. 4 & 5, each associated with a specific control output. The control outputs correspond to the three translational degrees of motion and the yaw motion.

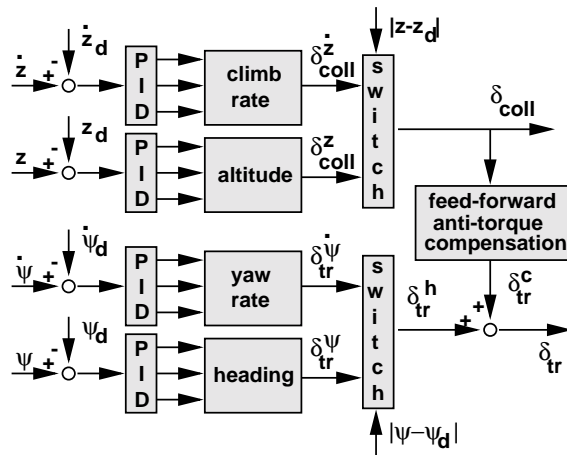


Fig. 4. Fuzzy controller architecture for collective and tail rotor pitch

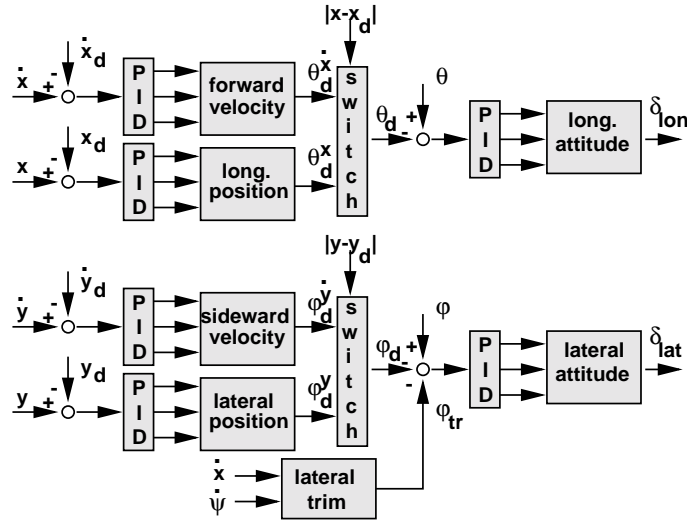


Fig. 5. Fuzzy controller architecture lateral and longitudinal cyclic and tail rotor pitch

The collective pitch control block attempts to follow a commanded altitude in vertical climb and descent. The heading control block governs the yaw motion during turns and compensates the anti-torque generated by the main rotor in order to maintain a desired heading. The longitudinal and lateral block regulate the horizontal motion and at the same time control the attitude to maintain helicopter stability.

The longitudinal and lateral modules accommodate multiple fuzzy PID control blocks organized in a hierarchical manner. Modules employ a fuzzy switch that shifts between position and velocity control of the corresponding output variable. The switching is either explicitly governed by the hybrid FVMS or automatically by means of fuzzy interpolation of controls depending on the current state. Position control is only activated when the output value is close to the nominal reference position. In an intermediate regime the fuzzy switch smoothly interpolates among the control actions proposed by the velocity and position fuzzy PID control block. The functionality and structure of the flight mode modules are described in more detail in [15].

3 Design

Fig.6 shows the general system architecture composed of the mobile robot, the fuzzy control system, the evolution strategy that adapts the fuzzy knowledge base and a simulation environment to evaluate the quality of a behavior. In the following we will describe the structure of each components and its role for the design process.

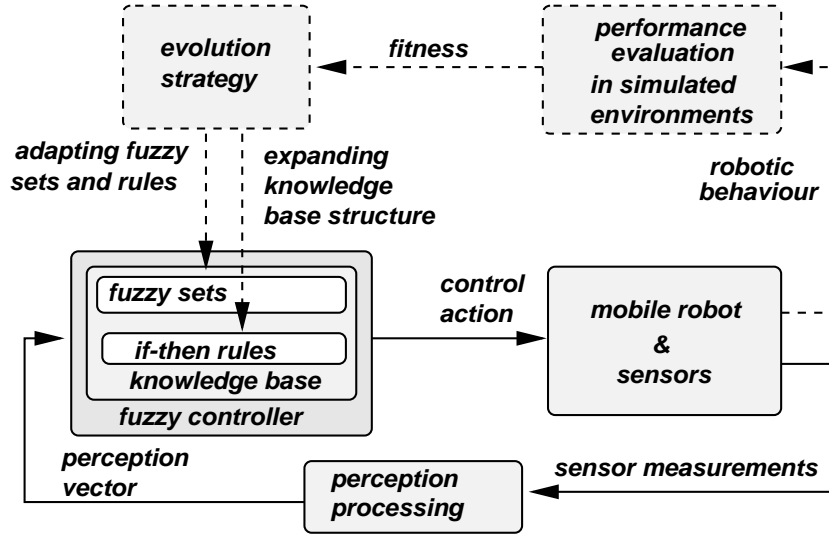


Fig. 6. System architecture

3.1 Fuzzy Controller

A fuzzy logic controller is a knowledge-based system characterized by a set of rules, which model the relationship among control input and output. The reasoning process is defined by means of the employed aggregation operators, the fuzzy connectives and the inference method. The fuzzy knowledge base contains the definition of fuzzy sets stored in the fuzzy data base and a collection of fuzzy rules, which constitute the fuzzy rule base.

Fuzzy rules are defined by their antecedents and consequents, which relate an observed input state to a desired control action. Most fuzzy systems employ the inference method proposed by Mamdani in which the consequence parts are defined by fuzzy sets [22]. The controller for the wall-following behavior employs the linguistic input variables *distance* and *bearing* which correspond to the crisp magnitude $|\mathbf{P}|$ and orientation $\angle \mathbf{P}$ of the perception vector \mathbf{P} . The input space is partitioned by five trapezoidal fuzzy sets along each dimension. Due to the symmetry of the robot and sensor configuration we only consider positive values of $\angle \mathbf{P}$. The fuzzy output variable *change-heading* corresponding to the steering action $\Delta\phi$ contains seven fuzzy terms implemented as singletons. A typical Mamdani type fuzzy rule has the form:

if distance=small and bearing=left-ahead then change-heading =sharp-right

The outputs of the active rules R_n are aggregated into a single fuzzy set for the output variable *change-heading*. The crisp control action $\Delta\phi$ is obtained through defuzzification, which in case of singleton fuzzy sets becomes equivalent

to a weighted average of the individual rule outputs. We manually derived a Mamdani fuzzy knowledge base using our intuition on the control task. First of all, the control output $\Delta\phi$ is supposed to increase with the proximity of the nearest object indicated by the magnitude of the perception vector $|\mathbf{p}|$. In order to avoid a collision with obstacles in front of the robot, the controller repulses the perception vector from the current direction of motion. Therefore, the change of heading $\Delta\phi$ increases with smaller absolute values of $\angle p$.

Besides the more common fuzzy inference method proposed by Mamdani, Takagi, Sugeno and Kang introduced a modified inference scheme [31]. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator are exactly the same. A Takagi-Sugeno-Kang (TSK) type controller employs different implication and aggregation methods than the standard Mamdani controller. Instead of using fuzzy sets in the conclusion part of a rule, the output becomes a linear combination of the crisp inputs.

$$\text{if distance=small and bearing=left-ahead then } \Delta\phi = c_0 + c_1|\mathbf{p}| + c_2\angle p$$

Since the consequence of a rule is crisp, the defuzzification step is obsolete in the TSK inference scheme. Instead, the control output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive than calculating the center of gravity. Since the output of each rule is a linear function of the inputs rather than a constant, the TSK knowledge base uses a sparse granulation with only three fuzzy sets per input variable.

A TSK fuzzy inference system is extremely well suited to the task of smoothly interpolating gains across the input space. This method of interpolating multiple linear controllers for different operating points is known as *gain scheduling* in classical control system design. In addition, learning techniques are applicable to TSK fuzzy controllers, by adapting the continuous gain factors c_i of the linear control rules. In our case, an evolution strategy optimizes the consequence parts of 9 rules with a total of 27 gain factors in order to model the relation among control input and output. Sugeno's method has the drawback that rules become less intuitive than in the Mamdani case, which makes it more difficult for a human expert to interpret the control behavior.

3.2 Evolution Strategies

An evolutionary algorithm processes a population of competing candidate solutions. The genome 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 the quality of solutions improves gradually.

In the 1960s RECHENBERG and SCHWEFEL developed evolution strategies [1], independent from the work of HOLLAND on genetic algorithms at the same time [17]. Both methods share the same generic features of evolutionary algorithms, a population of genomes, 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 process a string of discrete, often binary symbols. In evolution strategies mutation is the main spring of progress, whereas in genetic algorithms mutation only plays a minor role. The opposite holds for recombination. The original evolution strategies did not even employ recombination at all. In genetic algorithms, recombination in form of genetic crossover plays a central role for the generation of new candidate solutions. Evolution strategies employ a deterministic selection mechanism whereas in genetic algorithms the selection operator is usually stochastic. Due to these characteristics evolutionary strategies are preferable for problems in the domain of continuous optimization whereas genetic algorithms are an approved method for problems of discrete or combinatorial nature.

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 landscape. The strategy parameters are either tuned by an exogenous, deterministic step-size adaptation scheme (like Rechenberg's 1/5 success rule) or by the same evolutionary mechanism that optimizes the original object parameters [1].

A pair of real-valued vectors $(\mathbf{x}, \boldsymbol{\sigma})$ forms the genome of an individual. The first vector of object variables (x_1, \dots, x_n) corresponds to a point in search space. The additional vector of strategy parameters $(\sigma_1, \dots, \sigma_n)$ defines the standard deviations of mutations applied to the object variables. A mutation replaces the parent \mathbf{x}^t

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{N}(0, \boldsymbol{\sigma}) \quad (5)$$

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

The 1/5 success-rule after Rechenberg proposes to update the standard deviation $\boldsymbol{\sigma}$ based on the measured ratio p among successful and unfavorable object parameter mutations.

$$\begin{aligned} \boldsymbol{\sigma}^{t+1} &= \boldsymbol{\sigma}^t / c^{n/2} & \text{if } p > 1/5 \\ \boldsymbol{\sigma}^{t+1} &= \boldsymbol{\sigma}^t * c^{n/2} & \text{if } p < 1/5 \\ \boldsymbol{\sigma}^{t+1} &= \boldsymbol{\sigma}^t & \text{if } p = 1/5 \end{aligned} \quad (6)$$

Schwefel showed that a value of $c = 0.817$ is optimal for the sphere model [27].

As an alternative to a deterministic step-size adaptation, the evolution strategy itself may control its own mutability. In this case, an additional strategy vector $\boldsymbol{\sigma}$ becomes part of the individual and undergoes mutation as well.

$$\boldsymbol{\sigma}^{t+1} = \boldsymbol{\sigma}^t e^{\tau' N(0,1) + \tau \mathbf{N}(0,1)} \quad (7)$$

The normally distributed random variable $N(0, 1)$ is sampled once for the entire genome, whereas the mutations $\mathbf{N}(0, 1)$ are uncorrelated for different genes. At long sight, 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.

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

$$(\mathbf{x}, \boldsymbol{\sigma}) = ((\mathbf{x}^1 + \mathbf{x}^2)/2, (\boldsymbol{\sigma}^1 + \boldsymbol{\sigma}^2)/2) \quad (8)$$

. The discrete recombination resembles uniform crossover in genetic algorithms, where each component x_i is randomly picked from either one of the parents $q_i \in 1, 2$

$$(\mathbf{x}, \boldsymbol{\sigma}) = (x_1^{q_1}, \dots, x_n^{q_n}), (\sigma_1^{q_1}, \dots, \sigma_n^{q_n}) \quad (9)$$

Often, intermediate recombination is applied to the strategy parameters, whereas discrete recombination is preferred for the object variables.

A simple vector of strategy parameters $\boldsymbol{\sigma}$ restricts the mutation probability distribution to hyper-ellipsoids oriented along the coordinate axes. In order to generate arbitrary normally distributed mutations the general covariance matrix σ_{ij} of the distribution is parameterized by the variances $\boldsymbol{\sigma} \in R_+^n$ and the rotation angles $\boldsymbol{\alpha} \in [-\pi, \pi]^{(n * (n - 1)/2)}$ [27]. Correlated mutations enable the evolution strategy to align the mutation hyper-ellipsoid along any directions in search space. OSTERMEIER ET AL. proposed an adaptation scheme for arbitrarily normally distributed mutations which uses the distance vector among the best offspring and its parents to generate the optimal mutation distribution [24].

3.3 Genetic Fuzzy Systems

Recently numerous researchers explored the integration of evolutionary algorithms with fuzzy logic in genetic fuzzy systems [7][8][11][30]. 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.

There are two alternative approaches in applying evolutionary algorithms to rule based learning. In the Michigan approach a genetic fuzzy systems operates on a population of rules. In these so-called classifier systems, the genotype represents a single fuzzy rule and the entire population constitutes a solution. Evaluation, selection and recombination occur at the level of individual classifier rules. A classifier rule triggers whenever its condition part matches the current perceptions, in which case the proposed action is send to the environment. A credit apportionment system assigns strength to each rule according to the rewards following its activation. The evolutionary algorithm generates new classifiers rules based on the rule strengths acquired during a training episode.

The performance of a fuzzy system depends on the cooperation of multiple fuzzy rules. A fuzzy behavior is not constituted by a single rule but rather by an activation sequence of mutually collaborating fuzzy rules. On the other hand, an evolutionary algorithm based on the Michigan approach processes a population of competing fuzzy rules. In other words, a genetic fuzzy system integrates the antagonistic roles of competition and cooperation, to adapt a collection of fuzzy rules. Bonarini [2] proposed a credit assignment mechanism that evaluates the quality of an individual fuzzy rule in the context of its cooperation with other rules in the population. Competition is restricted to subsets of rules that trigger for similar inputs.

The Pittsburgh approach circumnavigates the credit assignment problem by evaluating the fitness for the entire knowledge base. The genetic fuzzy system evolves a population of knowledge bases rather than individual fuzzy rules. Evaluation of entire knowledge bases comes for the price of an increased computational burden and a substantial increase in the number of parameters and complexity of the search space. These drawbacks limit the usefulness of genetic fuzzy systems based on the Pittsburgh approach for optimization tasks in which fitness evaluations are computational expensive or time consuming such as online learning.

Genetic fuzzy systems are also distinguished by the structure that undergoes adaptation. The learning process can either target the fuzzy sets defined in the fuzzy database or the fuzzy rules that constitute the knowledge base. The first method results in a self-tuning controller in which an evolutionary algorithm adapts the fuzzy membership functions. The genome encodes parameters such as center and width of trapezoidal, triangle or Gaussian membership functions. This approach requires a previously defined rule base and is primarily useful in order to optimize the performance of an already existing controller.

The second approach, which is pursued in this paper, is based on a self-organizing process that learns the appropriate relationship between control input and output starting without any previous knowledge. In our case, the evolutionary algorithm processes a population of individuals corresponding to fuzzy knowledge bases. Each individual employs the same predefined fuzzy variables and fuzzy sets stored in the fuzzy data base. The genotype encodes the conclusion parts of TSK fuzzy rules. The real-valued vector of object variables represents the gain factors in the linear control laws as depicted in Fig. 7. The size of the genotype depends on the number of input variables and fuzzy sets. A TSK fuzzy controller usually needs a smaller number of rules, because their output is already a linear function of the inputs rather than a constant fuzzy set.

In case of the mobile robot controller, the input *distance* is partitioned into three fuzzy sets, the variable *bearing* into five fuzzy sets, resulting in fifteen fuzzy rules. This number is reduced to nine rules by taking advantage of the symmetry of the sensor configuration along the longitudinal axis. The control action $\Delta\phi$ acquires the opposite sign, when the perception vector falls into the right half-plane instead to the left. The genome contains 27 object variables, that correspond to 18 gain factors $c_i \angle \mathbf{p}$ and $c_{i|\mathbf{p}|}$ 9 constant offsets c_{i0} .

For the autopilot design, the genetic fuzzy system adapts each module separately using a training manoeuvre that is typical for the corresponding flight mode. The genome concatenates the gains and offsets of the individual PID fuzzy controllers. Again, symmetry and design considerations are helpful to reduce the number of independent parameters. The three PID fuzzy blocks in the lateral module depicted in Fig. 5 employ three TSK fuzzy rules each. Their gain factors and offsets are encoded by 20 object variables similar to the coding scheme shown in Fig. 7. The object variables are not initialized at random, but with the gains from a manually designed linear PID controller. The linear PID controller is stable for a narrow flight envelope around the hovering operating point. The initial parameters serve as point of departure to achieve an improved performance in a broader flight envelope. In this case, the evolutionary algorithm tunes an existing control system, rather than discovering novel rules as in the mobile robot case.

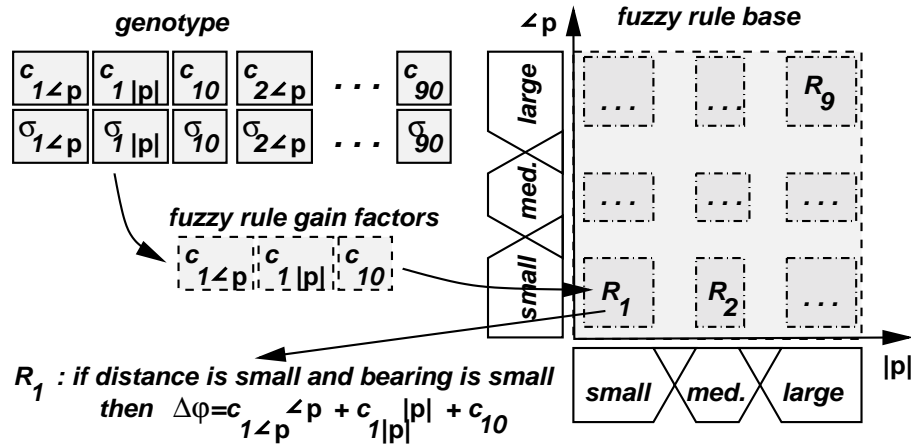


Fig. 7. Genetic representation of the TSK fuzzy knowledge base

3.4 Fitness Function Design

Designing the fitness function for behavior evaluation requires human insight into the problem domain and is often based on a sequence of trials and errors. Proper fitness function design becomes even more difficult for behaviors that have to consider multiple conflicting subgoals.

Controllers evolved in simulation tend to exploit peculiarities of the training environments or the synthesized sensor data. Therefore, we introduced sensor and actuator noise into the simulation in order to prevent over-adaptation and to improve the robustness of control.

Evolving in simulated domains requires some physical model of the robot, the sensors the environment and their mutual interaction. This model requires

a sufficient degree of accuracy to be able to transfer evolved controllers to the real robot. If on the other hand such a precise physical model is readily available it can be utilized to design a robot controller using a traditional model-based approach. Recently, authors presented impressive experiments in which the evolutionary learning takes place on the real robot [23]. The question is whether this approach can be applied to the evolution of more complex behaviors that require a larger number of fitness evaluations. The advantages of both approaches can be combined, by pre-adapting a provisional behavior in simulation, which is refined by online evolution on the physical robot. In this context, the modular structure of the fuzzy rule base enables smooth, local adaptations of individual fuzzy rules without the risk of fatal deterioration of the overall behavior.

Generality of adapted behaviors constitutes another important issue in evolutionary robotics. Fuzzy controllers demonstrate robustness towards external disturbances or changes in the system parameters. Fuzzy robot behaviors tend to be less brittle due to their tolerance to imprecision and uncertainty. This feature is beneficial for evolutionary robot controller design by closing the gap between simulation and reality. In addition, we evaluate the wall-following behavior in a representative collection of different environments, to enhance the generalization quality.

3.5 Mobile Robot

The desired robotic wall-following behavior is specified by a scalar fitness function describing the performance of the controller. Each controller is tested for a training set of environments, which differ in the geometry and dimension of walls and obstacles. The robot is started in every environment at various initial positions and headings. A single run either ends when the robot collides with a wall or if a maximal number of control steps N_{max} is exceeded.

In a gross abuse of ethological terminology, the robot experiences *pain* when it collides with an object and receives *pleasure* for going straight. A collision terminates a run and hinders the robot from experiencing further *pleasure*. Or in other words, *pain* manifests itself through the denial of future *pleasure*. A robot that gyrates on the spot without moving forward avoids *pain*. In order to elude the evolution of this trivial, but inadequate control strategy, the fitness function includes a *pleasure* component that at each control step rewards the robot for going straight. The fitness value f_i for the i -th training situation becomes

$$f_i = \sum_{n=1}^{\max N_{coll.}, N_{max.}} \left(1.0 - \frac{|\Delta\phi(n)|}{|\Delta\phi_{max}|}\right) \quad (10)$$

where $N_{coll.}$ denotes the number of control steps until a collision occurs, $\Delta\phi(n)$ is the commanded change in heading at step n and $\Delta\phi_{max}$ is the maximum change in heading. Notice, that the fitness is maximized by increasing the length of the sum $N_{coll.}$ as well as maximizing each of its elements $(1.0 - \frac{|\Delta\phi(n)|}{|\Delta\phi_{max}|})$. The overall

fitness of a controller is computed as the mean of worst and average fitness in all M training situations, in order to promote the evolution of a robust behavior.

$$F = \frac{1/M \sum_i^M f_i + \min_i^M f_i}{2} \quad (11)$$

3.6 Helicopter Autopilot

This subsection describes the evolutionary tuning process for the lateral module (Fig. 5) that governs the side-ward motion of the helicopter. The genetic fuzzy system utilizes the dynamical non-linear model to evaluate the performance of the controller. Each controller is tested in a variety of side-ward manoeuvres, which differ in amplitude, system parameters and external disturbances. Without this diversity of training situations the evolutionary algorithm tends to over-adapt the fuzzy system to a peculiar manoeuvre rather than the desired robust and general performance. It is important that the training situations cover all possible states of the prospective flight envelope. The lateral position controller is supposed to handle horizontal displacements in the interval $[-2.0m, 2.0m]$. Outside this hovering region, the FVMS uses the lateral velocity control block to bring the helicopter into the vicinity of the commanded position.

The engineer specifies the desired transient response in the time-domain choosing among a variety of performance indexes. The most common criteria are integral of squared error

$$S = \int_{t=0}^T \epsilon^2(t) dt \quad (12)$$

where ϵ is the output error, integral of energy consumption

$$S = \int_{t=0}^T u^2(t) dt \quad (13)$$

where u is the control action or the general optimal control formulation

$$S = \int_{t=0}^T x^T(t)Qx(t) + u^T(t)Ru(t) dt \quad (14)$$

where $x(t)$ is the state vector and Q and R are positive-definite real symmetric matrices. Additional performance indexes such as time of first transit through set-point, amplitude of overshoot or time to settle oscillations to an ϵ -band around the set-point provide a more intuitive description. In any case, the designer specifies the desired performance p_i by means of a soft constraint, described by a lower p_i^{min} and an upper bound p_i^{max} . The fitness f_i for the i -th performance index

$$f_i = \begin{cases} 1 & \text{if } p_i < p_i^{min} \\ 1 - \frac{p_i - p_i^{min}}{p_i^{max} - p_i^{min}} & \text{if } p_i^{min} < p_i < p_i^{max} \\ 0 & \text{if } p_i > p_i^{max} \end{cases} \quad (15)$$

is therefore normalized to the interval $[0, 1]$. The total fitness is computed as the product of the f_i

$$F = \prod_i f_i \quad (16)$$

in a way that encourages a balanced performance in regard to all indices. A controller obtains a zero fitness, if any of the indices exceeds its upper limit p_i^{max} . The performance of the hand-designed PID controller provides a suitable guideline to determine the upper bounds.

In case of the lateral control module five performance criteria contribute to the fitness. Three of them, overshoot, reaching time and settling time relate to the transient response in the lateral position y . An additional criteria constraints the roll angle φ to a small band $\pm 10^\circ$ around the hovering conditions. Notice, that the cascaded control architecture implicitly guarantees the stability of the roll motion. Finally, we penalize large values of high frequency components in the Fourier spectrum of the control signal δ_{lat} . This criterion guarantees a smooth control signal and avoids saturation of the actuator dynamics. Each controller is tested in multiple side-ward manoeuvres of different magnitude. In addition the simulated helicopter is subject to wind gusts which exert an additional drag force on the body.

The other modules, longitudinal, altitude and heading control are tuned in a similar fashion, using the type of training manoeuvre that characterizes the corresponding flight mode. For the altitude module tuning, we vary the helicopter mass by $\pm 10\%$ over multiple climbs and descents, to account for changes in payload and fuel weight.

4 Implementation

The implementation of the controller on the real robot does not pose a substantial challenge in regard to speed or computational complexity. The fuzzy controller is based on the same C++-code used in the simulation. The duration of a control cycle is merely determined by the echo delay of sonar distance measurements. In order to reduce the costs, the five sensors use the same echo detection electronics in a sequential manner. A complete sensor measurement takes about $500ms$. An on-board 486 PC handles the sensor data acquisition, the perception preprocessing, the fuzzy inference and the motor control. The program logs the sensor and motion data to a file for later debugging and analysis of the behavior.

It took a number of manual testing, evaluation and refinement cycles before the robot achieved the desired performance. Refining and validating the sensor model constituted the major engineering challenge in the design process. Once we obtained a sonar sensor model that matched the true echo characteristics fairly well, the controllers demonstrated the same performance in reality as in the two-dimensional virtual world in the simulation. The substantial amount of human design necessary to derive the sensor model establishes a serious handicap for off-line behavior adaptation schemes. On the other hand, a simulation model

only has to capture those aspects of robot environment interaction relevant for the desired behavior.

The real-time implementation of the helicopter autopilot becomes substantially more difficult and critical. In addition to the fuzzy inference, the on-board computer handles the sensor data acquisition, the ground station communication, the engine speed regulation, the servo actuation and the INS-GPS integration using a Kalman filter. The real-time operating system QNX has to schedule all these tasks within a control cycle of $20ms$ (INS update even every $10ms$). Therefore, we replace the fuzzy inference by a simple linear interpolation scheme among the linear control rules. The control output can be computed as a piecewise quadratic function over $\prod_i^n (2m_i)$ intervals, if n is the number of fuzzy variables and m_i the number of fuzzy membership functions of the i -th variable. The PID fuzzy controllers are only partitioned along the P -dimension, which further reduces the memory requirements and computational complexity. We plan to test the fuzzy controller on our helicopter prototype by the end of this year.

Safety is a major concern in the operation of an autonomous aerial robot. The inherently unstable system dynamics impose narrow permissible limits to system deterioration and failure. The accuracy of the observed state highly depends on the availability of precise GPS data. In LT-2 mode the differential GPS provides a relative position estimate of $2cm$ accuracy. In case of satellite occlusion by objects or atmospheric disturbances the accuracy can suddenly drop to $20cm$. In case of such an event, the hybrid control system initiates a controlled emergency landing. The INS can maintain a sufficient attitude estimate for a couple of seconds. The lateral and longitudinal modules switch to mere attitude control and discard the $x - y$ -position in order to land the helicopter in a horizontal orientation.

5 Experimental Results

Fig.8 shows two runs of the best controller implemented on the physical robot. In the left figure, the robot starts facing north, traverses the corridor and turns around when perceiving the dead-end and finally follows the corridor to the exit. For reasons of clarity, the final part of the robot's path to the exit is not plotted anymore. The right figure shows an environment with isolated obstacles avoided by the robot. Notice that both situations are different from the training environments in which the reactive navigation behavior was adapted.

Our experiments as well as simulations show that the robot can handle most of the environments for which a feasible trajectory exists. Only in situations with confined space for manoeuvring the robot sometimes gets stuck in a configuration where it no longer can avoid a collision. The purely reactive wall-following behavior alone does not allow the robot to pursue a meaningful mission in an indoor environment. It has to be augmented by deliberative strategies, a navigation system and a mission planner to master more complex tasks [26][34].

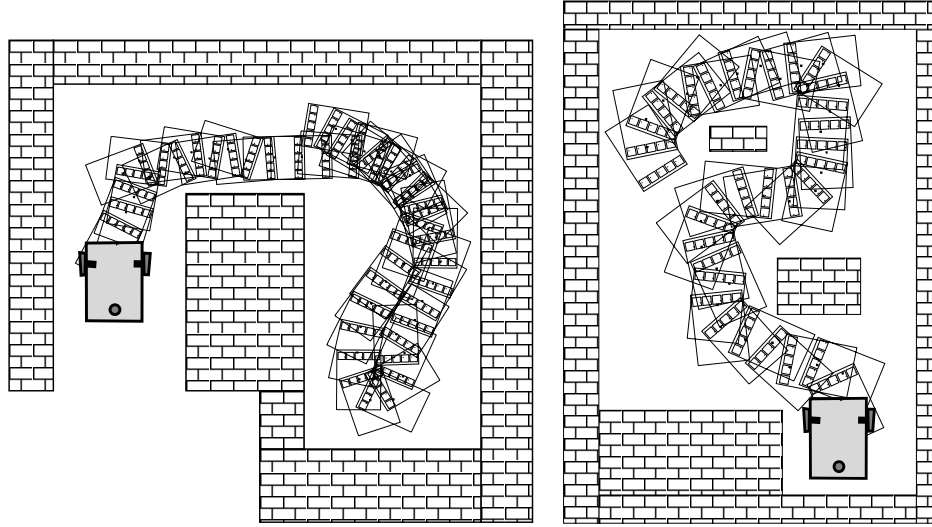


Fig. 8. Paths of the mobile robot in a real world environment

Table 1 compares the performance of the manually, the evolutionary designed fuzzy controller and a reinforcement learning scheme. Each controller is evaluated in simulation in thirteen different types of geometries, such as dead-ends, straight corridors, L-shaped segments or intersections of two corridors. For each scenario we generated 40 environments that differ in the width and length of the corridor segments. For example the corridor width varies in a range from 130 – 270cm. We consider a trial a success if the robot manages to move at least 40% of the maximal distance away from start. Notice, that sometimes obstacles block a corridor in a way that makes it impossible for the robot to pass. None of the learning methods explicitly rewarded a behavior that moves the robot away from the start, but it rather emerges indirectly as a result of the reward for traveling in a straight manner.

The task possesses an inherent trade-off between the more riskier strategy to pass obstacles to reach the open space and to the more conservative policy to avoid narrow passages with the inherent risk of a collision. The reinforcement learning scheme defines this trade-off by the amount of negative reward associated to a terminal state that ends in a collision. Therefore, the average fitness values stated for the reinforcement learner, does not reflect the actual reward function that is subject to policy optimization. The small number of "successes" does not indicate failure of the reinforcement learner since wandering away from the start provided no direct reward. Training the robot in changing environments introduces non-stationarity, which violates the Markov-process assumption of reinforcement learning schemes.

The number of training episodes for evolutionary algorithm and the reinforcement learner were approximately identical. All three controllers demonstrate a

similar average fitness over the test environments. The evolutionary designed fuzzy controller is most successful in avoiding collisions. This robustness stems from the explicit contribution of the worst fitness value in each environment according to Eq.11.

Env.	manually designed FLC			evolutionary designed FLC			reinforcement learner		
	avg. fit.	# coll.	# success	avg. fit.	# coll.	# success	avg. fit.	# coll.	# success
1	0.83	0	40	0.85	0	40	0.87	0	40
2	0.76	1	38	0.77	0	39	0.85	0	40
3	0.72	2	29	0.72	0	26	0.68	4	7
4	0.73	3	19	0.76	0	24	0.70	0	3
5	0.74	1	24	0.72	1	23	0.61	0	2
6	0.70	3	12	0.69	0	16	0.55	1	6
7	0.76	1	29	0.74	1	31	0.63	5	7
8	0.73	2	39	0.74	3	39	0.66	4	10
9	0.82	0	39	0.82	0	37	0.63	5	23
10	0.83	0	40	0.83	0	40	0.69	4	21
11	0.84	0	40	0.86	0	40	0.68	0	0
12	0.70	2	27	0.66	2	32	0.59	1	0
13	0.71	2	32	0.69	2	29	0.60	1	0
Σ	0.76	17/520	398/520	0.76	9/520	416/520	0.73	30/520	159/520

Table 1. Performance of manually and evolutionary designed fuzzy controllers and reinforcement learner in 13 simulated environments of different structural complexity. The numbers for collision and successes refer to 40 trials per environment. Each trial differed in the dimensions of the corridor, location of obstacles and initial configuration of the robot

We have not tested the autopilot on the real helicopter yet, since the state information provided by the integrated GPS-INS sensors is currently not reliable enough for automated flight. Fig. 9 shows the transient response of the simulation model to a commanded climb of $2m$. The graphs refer to the original hand-designed PID controller, the evolutionary tuned TSK fuzzy controller with nominal payload of $10kg$ and with an extra payload of $5kg$. Notice, that in the training scenario the maximum extra payload did not exceed $1kg$. The optimized controller reaches the commanded altitude much faster, even in case of the increased payload, although a small steady-state error remains in the later case.

6 Discussion

As stated earlier, fuzzy logic offers a number of benefits for mobile robot behavior design. Fuzzy logic provides a form of linguistic representation that allows a human designer to integrate his domain knowledge into the design process in

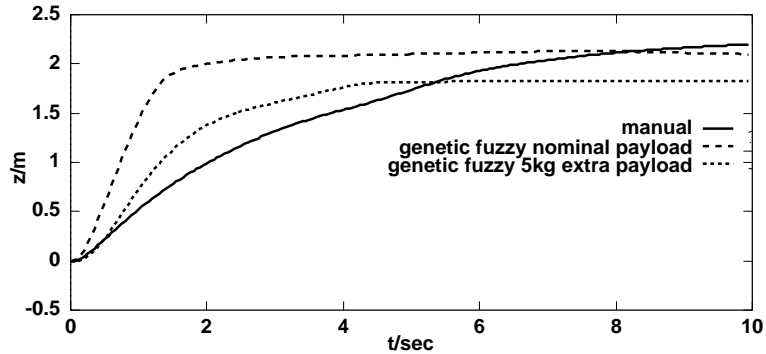


Fig. 9. Transient response of the manual (—) and evolutionary designed (- - -) controllers to a commanded climb manoeuvre

an intuitive manner. If a fuzzy system is enhanced by a learning method such as evolutionary algorithms, the adapted knowledge remains comprehensible for a human designer due to the local context of fuzzy rules. In our experience, robot behavior design is usually not a straight-forward, unerring process but a repeated cycle of conception, implementation, testing and refinement. The human engineer plays an indispensable role in the design because robotic tasks, environments and platforms are too diverse for fully automated design. In our context, fuzzy logic bridges the gap among human engineering and evolutionary design. Fuzzy logic enables the designer to use his domain knowledge to constrain the search space or bias the evolutionary algorithm.

In the helicopter scenario, the evolutionary search was biased with the hand-designed linear PID rules of the initial population. In the mobile robot case, the space of possible behaviors is constraint by comprising the sensor information into a perception vector and by the level of granulation defined by the fuzzy membership functions. This complexity reduction not only makes learning easier, but also avoids over-adaptation and brittleness of solutions. Constraining the search carries the drawback that too much engineering might prevent the evolution of creative, novel design solutions that the human designer did not conceive in advance.

Our research provides no evidence that a fuzzy system approach, compared to other design methods, boosts the quality of control to unprecedented levels of performance. In the helicopter case we achieved comparable results with linear robust multi-variable control methods [28]. More systematic design methods, such as nonlinear feedback linearization are superior in case an accurate analytical model of the system is available. For many practical applications although, it is very expensive or impossible to model all dynamic aspects and to determine its parameters precisely. Uncertainties and incomplete knowledge often make it impossible to derive a dynamic model of the robot, its sensors and the interaction with the environment even for simple task. A behavior based approach takes these inherent limitations into account. Within the framework of behavior

based robotics fuzzy techniques facilitate the conception and design process.

Evolutionary robot behavior design in simulated environments is based on the assumption that the real world bears enough resemblance with the simulation domain. One can argue, that the effort to conceive and implement a simulation model outweighs a direct human-based behavior design. On the other hand the simulation model can be kept fairly simple, as long as it includes those features relevant to perform the desired behavior. JAKOBI developed a theoretical and methodological framework called "minimal simulation" for the easy construction of fast-running robotic simulators [18]. By only modeling the relevant aspects of a robotic setup, evolved controllers are less prone to brittleness and more likely to bridge the gap among simulation and reality.

Although our results and those from other experimenters show that it is possible to evolve controllers in simulation, it is not clear whether this technique will scale up with the complexity of behaviors. Currently, evolutionary robotics is not powerful enough to entirely replace the human engineer in the design process. To our understanding it is important to clearly separate the behavior adaptation process from the issue of behavior coordination. The intuitive interpretation of fuzzy rule based systems helps the engineer to integrate a repertoire of primitive behaviors into a meaningful, more sophisticated robotic system [12][32].

Although our wall-following behavior generalizes fairly well over different types of environments, it is hard to imagine that a non-adaptive robot can cope with all possible real world situations. A real robot that learns during lifetime can refine and optimize its behavior to the particular environment in which it operates. On the other hand, behavior adaptation on the real robot constitutes a time consuming and tedious process and requires fast learners. Under the reasonable assumption, that the robot encounters similar situations in simulation and reality, evolutionary designed behaviors provide a valuable, preliminary basis for the challenge of learning in real world scenarios.

7 Conclusions and Future Work

This contribution presented a genetic-fuzzy system approach to robotic behavior design. The proposed design method was applied to learn a wall-following behavior of a mobile robot and to tune an autopilot for an autonomous aerial robot. In the context of evolutionary robotics, fuzzy control systems are a valuable tool to bridge the gap among evolutionary and human design. They allow the engineer to integrate her domain knowledge in the design process and to analyze and interpret the resulting behavior.

The wall-following behavior learned in a simulation was able to generalize from the training environments to previously unseen situations. Experiments with the physical mobile robot demonstrate, that an evolutionary algorithm can in principle design a robust control system that is able to cope with the uncertainty and imprecision inherent to real-world situations. The ultimate criterion for the success of evolutionary robotics is whether its techniques are able to learn more complex behaviors that can not be easily designed by a human expert.

In the future we plan to augment the evolutionary design process in simulation by a reinforcement learning phase in a real environment. The fuzzy behavior constitutes the initial policy subject to evaluation by the reinforcement learner. Based on the observed rewards and values of state-action pairs, the learner gradually improves the initial policy to maximize the rewards in the particular environment of operation.

For the helicopter autopilot, the genetic-fuzzy tuning process generated controllers, which performance and robustness match the quality of robust control design. The adapted controllers stabilize the helicopter over a broad flight envelope and are able to tolerate parameter uncertainties and external disturbances.

8 Acknowledgment

This work was supported by the Army Research Office under grant DAAH 04-96-1-0341, the ONR under grant N00014-97-1-0946 and the Deutsche Forschungsgemeinschaft under grant Ho 1790/2-1.

References

1. 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. Periaux, M. Gala, P. Cuesta, pp. 111-126, pp. 127-140, John Wiley & Sons, (1995).
2. A. Bonarini, "Evolutionary Learning of fuzzy rules : competition and cooperation", in *Fuzzy Modeling: Paradigms and Practice*, Kluwer Academic Press, pp. 265-283, (1996).
3. A. Bonarini, "Anytime learning and adaptation of hierarchical fuzzy logic behaviors" *Adaptive Behavior Journal, Special Issue on Complete Agent Learning in Complex Environments*, Ed. :M. Mataric, (1997).
4. 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)
5. D. Cliff, I. Harvey, P. Husbands, "Incremental evolution of neural network architectures for adaptive behaviour", *Proc. European Symposium on Artificial Neural Networks, ESANN '93*, Brussel, p. 39-44, (1993).
6. M. Colombetti, M. Dorigo, G. Borghi, "Behaviour Analysis and Training - A Methodology for Behavior Engineering", *IEEE Transactions on Systems, Man and Cybernetic*, Part B: Cybernetics, vol. 26, no. 3, (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. Oscar Cordon, Francisco Herrera, Manuel Lozano, "A Classified Review on the Combination Fuzzy Logic Genetic Algorithms Bibliography: 1989-1995", *Genetic Algorithms and Fuzzy-Logic Systems. Soft Computing Perspectives* E. Sanchez, T. Shibata, L. A. Zadeh (Eds.), World Scientific, (1996).

9. E. Fabrizi, G. Oriolo, G. Ulivi, "Fuzzy map building for moderately dynamic environments" in "Fuzzy Logic Techniques for Autonomous Robot Navigation", Ed. D. Driankov, A. Saffiotti, Springer-Verlag, (1999).
10. J. J. Grefenstette, A. C. Schultz, "An Evolutionary Approach to Learning in Robots", *Machine Learning Workshop on Robot Learning*, (1994).
11. 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, p. 279-305, (1996).
12. S. Goodridge, M. Kay, "Multi-layered fuzzy behavior fusion for reactive control of autonomous robots" in *Fuzzy logic techniques for autonomous vehicle navigation*, Ed. A. Saffiotti, D. Driankov, Springer, (1999).
13. F. Hoffmann, G. Pfister, "Evolutionary Design of a Fuzzy Control Rule Base for a Mobile Robot", *International Journal of Approximate Reasoning*, vol. 17, no. 4, (1997).
14. F. Hoffmann, "Incremental Tuning of Fuzzy Controllers by Means of an Evolution Strategy", in *Genetic Programming Conference*, Madison, Wisconsin, (1998).
15. F. Hoffmann, T. J. Koo, O. Shakernia, "Evolutionary Design of a Helicopter Autopilot", in *3rd On-line World Conf. on Soft Computing (WSC3)*, (1998).
16. F. Hoffmann, "Soft Computing Techniques for the Design of Mobile Robot Behaviors", accepted for publication in *Journal of Information Sciences*, (1998)
17. J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, (1992).
18. N. Jakobi, "The Minimal Simulation Approach To Evolutionary Robotics" In Proceedings of ER'98, T. Gomi (ed.), AI Systems Books (1998).
19. T. J. Koo, S. Sastry, "Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization", *37th IEEE Conference on Decision and Control*, Tampa, Florida, (1998). @
20. T. J. Koo, F. Hoffmann, B. Sinopoli, and S. S. Sastry, "Hybrid Control of Model Helicopter", *Proceedings of IFAC Workshop on Motion Control*, Grenoble, France, (1998).
21. P. Maes. "Modeling Adaptive Autonomous Agents", *Artificial Life Journal*, edited by C. Langton, Vol. 1, No. 1 & 2, pp. 135-162, MIT Press, (1994).
22. E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant", *Proceedings of the Institution of Electrical Engineers*, vol. 121, no. 12, pp. 1585-8, (1974).
23. P. Nordin, W. Banzhaf, "An On-Line Method to Evolve Behavior and to Control a Miniature Robot in Real Time with Genetic Programming" *Adaptive Behaviour*, 5 (2), pp. 107-140, (1997).
24. A. Ostermeier, A. Gawelczyk, N. Hansen, "Step-size Adaptation Based on Non-local Use of Selection Information", *Proc. Parallel Problem Solving from Nature - PPSN III*, Ed. Y. Davidor, H.-P. Schwefel, pp. 189-198, Springer, Berlin, (1994).
25. C. Phillips, C. L. Karr, and G. Walker, "Helicopter flight control with fuzzy logic and genetic algorithms", in *Engineering Applications of Artificial Intelligence*, vol. 9, no. 2, pp. 175-84, (1996).
26. A. Saffiotti, "Principled integration of planning and control using fuzzy logic" in *Fuzzy logic techniques for autonomous vehicle navigation*, Ed. A. Saffiotti, D. Driankov, Springer, (1999).
27. H.-P. Schwefel, *Evolution and Optimum Seeking*, Wiley, Chichester, (1995).

28. H. Shim, T. John Koo, F. Hoffmann, S. Sastry, "A Comprehensive Study on Control Design of Autonomous Helicopter", *Submitted to the 37th IEEE CDC*, Tampa, Florida, (1998).
29. M. Sugeno, I. Hirano, S. Nakamura, and S. Kotsu, "Development of an intelligent unmanned helicopter", in *IEEE International Conference on Fuzzy Systems*, vol. 5, pp. 33-4, (1995).
30. H. Takagi, M. Lee, "Integrating Design Stages of Fuzzy Systems using Genetic Algorithms", *Proc. of the Second IEEE Int. Conf. on Fuzzy Systems*, pp. 612-617, (1993).
31. T. Takagi, M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no.1, pp.116-32, (1985).
32. E. Tunstel, "Fuzzy-behavior synthesis, coordination, and evolution in an adaptive behavior hierarchy" in *Fuzzy logic techniques for autonomous vehicle navigation*, Ed. A. Saffiotti , D. Driankov, Springer , (1999).
33. T. Yoshikawa, T. Furuhashi, Y. Uchikawa, "Knowledge acquisition of fuzzy control rules for mobile robots using DNA coding method and pseudo-bacterial GA", *Simulated Evolution and Learning, First Asia-Pacific Conference, SEAL '96*, Taejon, South Korea, Ed. X. Yao, J. -H. Kim, T. Furuhashi, Springer Verlag, pp. 126-35, (1997).
34. J. Zhang, A. Knoll "Design and learning of fuzzy controller integrating deliberative and reactive strategies" in *Fuzzy logic techniques for autonomous vehicle navigation*, Ed. A. Saffiotti , D. Driankov, Springer , (1999).