

Evolutionary Design of a Helicopter Autopilot

Frank Hoffmann, Tak John Koo, Omid Shakernia
Electrical Engineering & Computer Science Department
University of California, Berkeley
Berkeley, CA 94720
email: fhoffman@cs.berkeley.edu

June 22, 1998

Abstract

This paper presents an evolutionary design method for fuzzy logic controllers, which is based on a self-organizing process that learns the appropriate relationship between control input and output. Our approach employs an evolution strategy that operates on vectors of real numbers which correspond to the gain factors in the conclusion part of fuzzy rules. An incremental learning scheme gradually expands the genome and thereby refines the fuzzy knowledge base that acquires additional fuzzy rules.

The paper describes the hybrid architecture of a flight vehicle management system, which governs the operation of an autonomous model helicopter. The autopilot is composed of four modules that control the longitudinal and lateral motion, altitude and heading. The autopilot that constitutes the continuous regulation layer of the hybrid system is implemented by a set of fuzzy controllers. The evolutionary algorithm optimizes the fuzzy rule bases off-line. We compare two design approaches, learning the rule base starting without previous knowledge and tuning an predefined set of fuzzy rules.

1 Introduction

Evolutionary algorithms are search and optimization methods, which are guided by the process of natural evolution. Although they employ different genetic representations, they share the same generic features, a population of competing candidate solutions, random combination and alteration of potentially useful structures to generate new variants and selection of fitter solutions as a means to direct the exploration of the search space.

Fuzzy systems describe knowledge by means of linguistic concepts which require less complexity and precision than mathematical or logical models. Their

approximate reasoning scheme exploits the tolerance to imprecision and uncertainty inherent to many real world problems. Fuzzy control provides a flexible means to model the nonlinear relationship between input information and control output. Its robustness with respect to noise and variation of system parameters make fuzzy control a suitable method for intelligent control of unmanned helicopter [18].

The soft computing paradigm suggests to combine the beneficial properties of complementary methods into a hybrid system. Recently, numerous publications propose to integrate the learning capabilities of evolutionary algorithms with the knowledge representation of fuzzy systems [3, 5, 13, 20]. These methods are described by the general term genetic-fuzzy systems. The central idea is to automate the knowledge acquisition step in fuzzy control design by an evolutionary algorithm. 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. PHILLIPS ET AL. [14] proposed a genetic algorithm to learn fuzzy logic controllers for helicopter flight.

Section 2 introduces the functional elements and fundamental concepts of evolution strategies. Section 3 is concerned with the genetic fuzzy system and incremental learning scheme that automate the knowledge acquisition step in fuzzy control design. Section 4 describes the flight vehicle management system and the helicopter autopilot architecture in detail. Section 5 presents the approach used to design the autopilot by means of an evolutionary adaptation process. Section 6 describes the visualization tool SmartAerobots that is used to generate the helicopter animations for the online presentation of this paper.

2 Evolution Strategy

An evolutionary algorithm propagates a population of genetic variants from one generation to the next. A coding scheme transcribes the genetic blue-print of an individual into a potential solution of the optimization problem. A scalar objective function evaluates the quality of solutions in regard to the optimization task. According to Darwin's principle, highly fit individuals obtain a higher chance of reproduction. Selection focuses future search to those regions in which good solutions were already found. Genetic operators such as recombination and mutation generate new offspring which inherit their genetic material from the selected parents. This evolutionary cycle of selection, recombination and mutation gradually improves the quality of solutions that emerge within the population.

In the 1960s RECHENBERG and SCHWEFEL [2] developed evolution strategies, independently from the work of HOLLAND [8] on genetic algorithms at the same time. Both optimization methods are based on similar generic principles, a population of genetic structures, a selection scheme that increases the proportion of better solutions and genetic operators that produce new variants.

The major difference between evolution strategies and genetic algorithms is the way in which the genotype represents a candidate solution. An evolution strategy operates on a vector of real numbers, whereas genetic algorithms manipulate a string of discrete, usually binary, symbols.

In evolution strategies mutation is the major source of variation, whereas recombination merely has the function to mix the genetic material. The earliest evolution strategies did not even employ recombination at all. In genetic algorithms mutation plays a minor role by preventing premature convergence of the population. According to the building block hypothesis [8], evolutionary progress in genetic algorithms is mainly governed by recombination.

Evolution strategies are distinguished by self-adaptation of additional strategy parameters, which enables them to optimize the evolutionary process according to the structure of the fitness landscape. The selection operator in evolution strategies is completely deterministic whereas in genetic algorithms selection is a stochastic process. Due to these characteristics evolution 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.

In an evolution strategy the genome of a single individual consists of two sets of real-valued variables. The vector (x_1, \dots, x_n) contains the object variables, which are tuned by recombination and mutation in order to optimize the objective function. The additional strategy parameters are stored in the vector $(\sigma_1, \dots, \sigma_n)$ that defines the step-size of mutations applied to the object variables. The mutation operator modifies the corresponding object variable x_i by adding normally distributed noise $N_i(0, \sigma_i^2)$ with variance σ_i^2 and zero mean.

$$x'_i = x_i + N_i(0, \sigma_i^2) \quad (1)$$

The strategy parameters σ_i do not remain constant. They undergo mutation

$$\sigma'_i = \sigma_i e^{\tau' N(0,1) + \tau N_i(0,1)} \quad (2)$$

where τ' is a global mutation factor and τ is a local mutation factor. The random number $N(0, 1)$ is only sampled once for the complete genotype, whereas the mutations $N_i(0, 1)$ are uncorrelated for different genes. The learning rates depend on the size n of the genotype, usually one chooses constant factors $\tau' \sim 1/\sqrt{2n}$ and $\tau \sim 1/\sqrt{2\sqrt{n}}$. Notice, that in case of a variable-size genetic representation these factors are updated accordingly, whenever the genotype acquires additional genes.

The mutation of strategy parameters enables the evolution strategy to adapt itself to the structure of the fitness landscape. At long sight, selection favors those strategy parameters, which are more likely to generate mutations to object variables that substantially improve their fitness. 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. The discrete recombination resembles uniform crossover in genetic algorithms, where each single variable is randomly picked from either one of the parents. Normally, intermediate recombination is chosen for the strategy parameters, while discrete recombination is preferred for the object variables.

3 Evolutionary Tuning of TSK Fuzzy Controllers

Among many other applications, fuzzy logic control has been applied to control an intelligent unmanned helicopter [18]. A fuzzy logic controller is a knowledge based system characterized by a set of linguistic variables and fuzzy if-then rules. 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 rule consequence is defined by fuzzy sets [11]. A typical Mamdani type fuzzy rule R_n has the structure:

$$R_n \quad : \quad \text{if } X_1 = A_{1n} \text{ and } \cdots X_m = A_{mn} \text{ then } Y = B_n$$

The fuzzy output of a rule depends on the degree of activation of its antecedent. The Mamdani inference scheme aggregates these outputs into a single fuzzy set for the variable Y . Finally, defuzzification transforms this output fuzzy set into a crisp control value, usually by calculating its centroid.

Takagi, Sugeno and Kang (TSK) proposed an inference scheme in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of the crisp inputs rather than a fuzzy set [19].

$$\text{if } X_1 = A_1 \text{ and } \cdots X_m = A_m \text{ then } y = c_0 + c_1 x_1 + \cdots + c_m x_m$$

A TSK fuzzy inference system smoothly interpolates among multiple linear controllers defined for specific operating points in the input space. TSK fuzzy systems can be tuned by a learning algorithm, that adapts the continuous gain factors c_i of the linear control rules. For reasons of simplicity we neglect the constant offset term c_0 in the following and only consider the gain factors.

3.1 Incremental Learning Scheme

The number of rules in a fuzzy controller grows rapidly with an increasing number of variables and fuzzy sets. Learning all fuzzy rules at once becomes less and less feasible due to the large number of parameters to be adapted simultaneously. To overcome this problem of increasing complexity, we propose an incremental learning scheme, which partitions the design task into smaller steps.

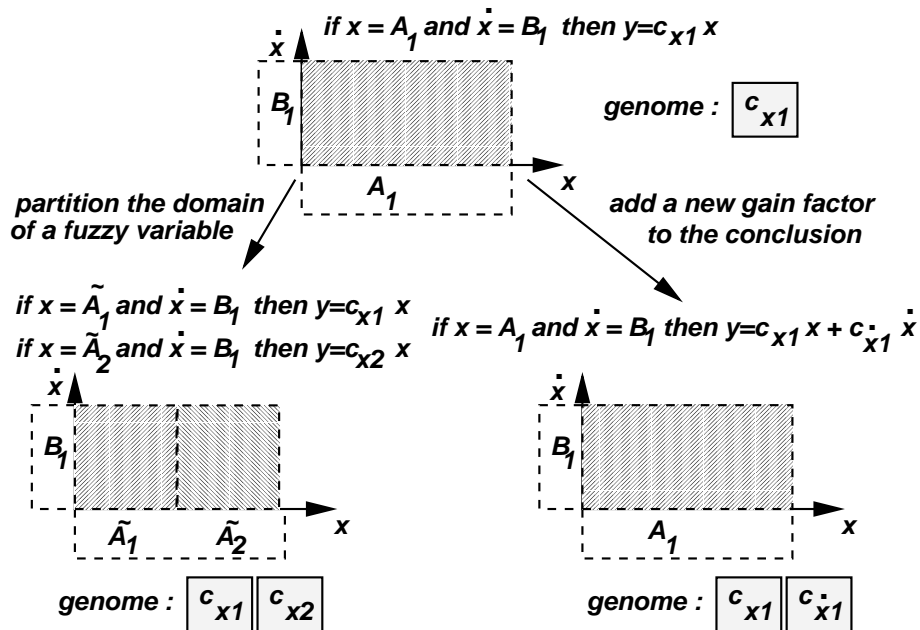


Figure 1: Incremental Refinement of the Fuzzy Rule Base

The evolution strategy first optimizes a fuzzy knowledge base with a small number of rules each of them covering a large region of input space. In later generations the rule base is refined it through superposition of additional rules with finer granulation. Fig. 1 shows an example of the refinement for a fuzzy system with two input variables x and \dot{x} . The rule base tuning procedure starts with a single TSK fuzzy rule

$$R_1 : \quad \text{if } x = A_1 \text{ and } \dot{x} = B_1 \text{ then } y = c_{x1} x$$

that covers the entire input space. Its linear control law only depends on the input variable x . The genome is constituted by a single gene that corresponds to the gain factor c_{x1} and its mutation step-size σ_{x1} .¹ This primitive rule R_1 constitutes the point of departure for subsequent refinements of the rule base. The evolution strategy starts to optimize the single gain factor c_{x1} . Within a few generations the population converges to a preliminary solution.

In order to enable a more distinguished control behavior the rule base acquires additional degrees of freedom. The already adapted rule R_1 can be extended in two different ways.

¹In section 2 x_i denoted the object variables employed by the evolution strategy. In the following x always relates to the crisp value of input variable x , whereas c_i is used to describe the gain factors encoded by the object variable i .

1. Adding a term for the input variable \dot{x} to the consequence part of the rule R_1

$$R'_1 : \quad \text{if } x = A_1 \text{ and } \dots \dot{x} = B_1 \text{ then } y = c_{x1}x + c_{\dot{x}1}\dot{x}$$

with the result that the control law becomes a linear function of both inputs. The genome is extended by a second object variable for the additional gain factor $c_{\dot{x}1}$ plus its corresponding strategy parameter $\sigma_{\dot{x}1}$.

Additional genes are incorporated into the genome in a way that minimizes deleterious side-effect off the already adapted genes. Therefore, the new gene $c_{\dot{x}1}$ is initialized with a zero value to guarantee the congruence among the new rule R'_1 with its genetic ancestor R_1 . The evolution strategy exploits the extra degree of freedom by adjusting the control surface through subsequent mutations applied to the new gain factor.

2. Partition the domain of an input variable, for example x into two partially overlapping fuzzy sets \tilde{A}_1 and \tilde{A}_2 , with the result that the knowledge base now contains two rules R'_1, R'_2 instead of one.

$$\begin{aligned} R'_1 : & \quad \text{if } x = \tilde{A}_1 \text{ and } \dots \dot{x} = B_1 \text{ then } y = c_{x1}x \\ R'_2 : & \quad \text{if } x = \tilde{A}_2 \text{ and } \dots \dot{x} = B_1 \text{ then } y = c_{x2}x \end{aligned}$$

The genome is expanded by duplicating the gain factor c_{x1} of the original rule R_1 which preserves the previously adapted knowledge expressed by R_1 . Afterwards, the genome contains two gain factors c_{x1} and c_{x2} associated to fuzzy rules that cover different regions of input space. In the region where the fuzzy sets \tilde{A}_1 and \tilde{A}_2 overlap the control output is interpolated among the two fuzzy rules.

The evolution strategy adapts the additional parameters for a few generations before the next rule base expansion takes place. Each expansion step gradually increases the complexity of the fuzzy knowledge base, by either adding a new input variable to the consequence part of all fuzzy rules or by partitioning a fuzzy set of an input variable, resulting in an increased number of rules with finer granulation.

The proper initialization of the newly incorporated genes preserves the input-output mapping already adapted in previous generations. The modularity of fuzzy rules allows it to gradually evolve complex genetic representations starting with a simple genome that only comprises a few parameters.

Nevertheless, some disturbance is caused by the first mutations applied to the new genes, especially since in the beginning their strategy parameters are not yet adapted to the structure of the fitness landscape. The cycle of genome expansion and adaptation of the additional genes continues until the rule base accumulates enough parameters to solve the control task.

In each expansion step the fuzzy knowledge base can be extended in multiple ways. Either one can add another gain factor to the rule consequences or partition an arbitrary fuzzy set. A simple solution is to hand on the decision to the designer, who generates an expansion schedule based on her knowledge about the structure of the control problem.

Instead of a predefined schedule of expansion, a constructional selection mechanism can be used to regulate the augmentation of the fuzzy knowledge base. It evaluates the functional effect of multiple possible genome expansions selects the one that results in optimal evolvability, or in other words most likely promises the production of fitter offsprings. The idea is to reject gene additions that have a neutral or even negative effect on the performance and to stably embody genes that explore novel degrees of freedom in a way that increases the fitness. The constructional selection mechanism is described in [7] in more detail.

4 Autonomous Helicopter

Unmanned autonomous aerial vehicles have been found indispensable for various applications where human intervention is considered difficult or dangerous. Despite its poor cursing performance, helicopter can be operated in different flight modes, such as vertical take-off/landing, hovering, longitudinal/lateral flight, coordinated turn. Due to its versatile in maneuverability, a helicopter is capable to manoeuvre in and out of restricted areas and to hover efficiently for long periods of time. These features make helicopter a valuable tool for power line inspection, terrain surveying, and investigation and clean-up of hazardous waste sites.

4.1 Flight Vehicle Management System

The Flight Vehicle Management System (FVMS) is responsible for planning and controlling the helicopter's operation. An image processing system detects objects on the ground and identifies landmarks used for navigation. The design is based on a hybrid control approach [1] in which a discrete event systems governs the behavior of a continuous-state plant. The FVMS is responsible for resolving conflicts between air vehicles, planning of the flight path, generating a feasible trajectory and a proper sequence of flight modes, and regulating the helicopter along the trajectory. The system architecture proposed in [10] including the image processing system is shown in Fig. 2.

The Strategic planner resides on top of the FVMS and is concerned with the planning and execution of the central helicopter mission. The strategic planner commands the tactical planner to execute a sequence of proper behaviors which enable the helicopter to accomplish the overall mission. The strategic layer is able to communicate with other FVMS in order to coordinate missions

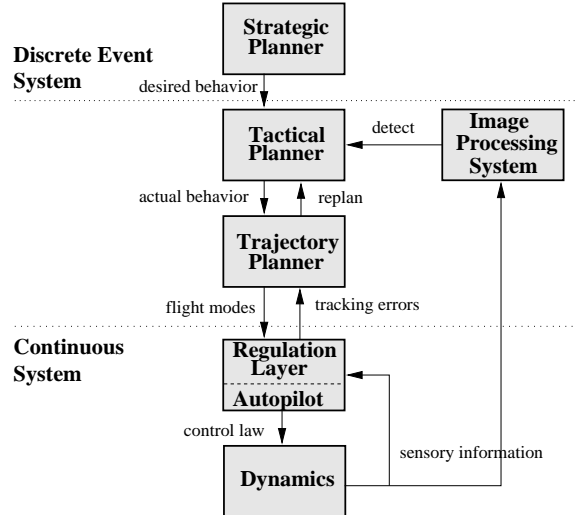


Figure 2: FVMS Architecture for autonomous helicopter

that require the cooperation of multiple helicopters. The tactical planner is responsible for the coordination and execution of behaviors, such as landing, searching an area, steering towards a way-point, collision avoidance or inspection of objects on the ground. The tactical planner is able to overrule the behavior proposed by the strategic planner, in case of safety critical situations such as collision avoidance, failure of sensors or strong wind gusts. The tactical planner returns to the original behavior as soon as the conflict among the prioritized safety manoeuvre and accomplishing the mission is resolved.

The trajectory planner decomposes the behavior commanded by the tactical planner into a sequence of primitive manoeuvres, such as forward-, side ward-flight, hovering, vertical climb and turns. The trajectory planner can employ a variety of basic flight mode controllers which are designed to control different variables in the helicopter dynamics. Each controller emphasizes different performance attributes, such as tracking accuracy, robustness and execution speed and therefore offers advantages for a specific type of manoeuvres. The trajectory planner activates those basic flight controllers that are best suited to achieve the proposed behavior considering the context of operation and the current flight mode.

The regulation layer commands the autopilot to track a dynamic trajectory by using the flight mode controllers assigned by the trajectory planner. In the presence of large external disturbances (such as wind shear or malfunctions), however, tracking can severely deteriorate. The regulation layer has access to sensory information about the actual state of the helicopter dynamics, and can calculate tracking errors. These errors are passed back to the trajectory

planner, to facilitate replanning of the trajectory or even switching the controller if necessary. The autopilot is implemented by means of multiple fuzzy controllers associated to different flight modes and control actuators. SHIM ET AL. [17] provide a comparative study on helicopter autopilot design using fuzzy control, robust control and feedback linearization.

4.2 Helicopter Dynamics

Flying a helicopter is a complex control problem including multiple inputs and outputs which in addition are partially coupled. A helicopter is a nonlinear system which is inherently unstable and very sensitive to external disturbances like wind and ground effects. Three rotational and three translational degrees of freedom have to be controlled by four inputs.

The main rotor pitch regulates the vertical thrust which lifts the helicopter. The engine throttle is controlled by an independent engine governor and is not used for modifying the thrust. It maintains a constant main rotor RPM that provides the optimal trim condition for the collective pitch.

The longitudinal and lateral cyclic pitch of the main rotor affect 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 rate and in addition compensates the anti torque generated by the main rotor.

4.3 Controller Architecture

The helicopter autopilot is composed of four separate modules which correspond to the control actuators collective pitch, tail rotor pitch, longitudinal and lateral cyclic pitch. The complexity of the control design can be reduced by carefully considering the helicopter dynamics when subdividing the system into separate modules.

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 the helicopter stable. Individual modules contain multiple fuzzy controllers organized in a hierarchical manner that reflects the effect and coupling of different controls. Each block employs a switch which selects the controller that is specialized for the current flight mode. During a transition between position and velocity control the switch smoothly interpolates among the proposed control actions.

The collective control block is composed of two parallel PID fuzzy controllers, which respectively control the helicopter altitude z and the climb rate \dot{z} . If the altitude deviates significantly from the commanded height the climb rate control action passes through the switch. For an intermediate range 1-2m away from

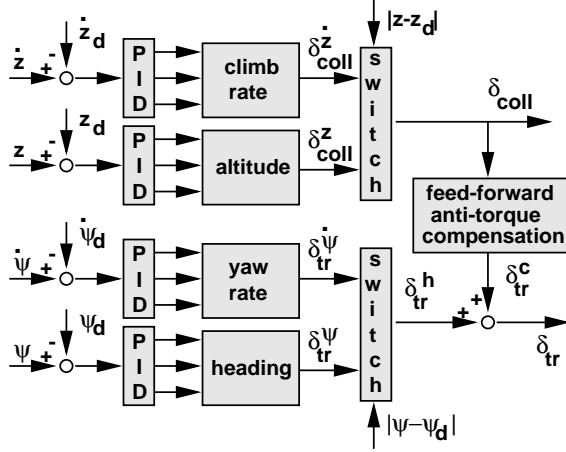


Figure 3: Fuzzy controller architecture for collective and tail rotor pitch

the set-point a linear combination of the control outputs is used. When the error in height becomes less than 1m the altitude controller determines the collective pitch δ_{coll} .

The control block for the tail rotor pitch δ_{tr} operates in a similar manner. Two fuzzy PID controllers control the heading ψ and the yaw rate $\dot{\psi}$ for hovering and coordinated turns. The main rotor exerts an anti-torque on the helicopter body which has to be compensated by the tail rotor. Feed-forward control trims the tail rotor pitch δ_{tr} in response to a change in collective pitch δ_{coll} .

The control inputs x, x_d, y, y_d are mapped from spatial to body coordinates before they are passed to the longitudinal and lateral position controllers. This transformation is not shown in Fig. 4. The control inputs $\dot{x}, \dot{x}_d, \dot{y}, \dot{y}_d$ for the velocity controllers are already specified in body coordinates.

The longitudinal control block employs three PID fuzzy controllers to compute the longitudinal cyclic pitch δ_{lon} as a function of the pitch angle θ , longitudinal position error e_x and forward velocity \dot{x} . The pitch attitude determines the thrust component in forward direction and therefore the longitudinal motion of the helicopter. The longitudinal position and velocity controllers command a desired pitch angle θ_s to the longitudinal attitude controller. A switch mediates between position and velocity control based on the actual region in state space. The cascaded architecture of the block significantly reduces the number of fuzzy rules compared to a single fuzzy controller that aggregates all three inputs in one step. Connecting the fuzzy controllers in sequence implicitly guarantees a stable pitch attitude while controlling the longitudinal motion.

The lateral control block is based on a similar hierarchical architecture. The lateral position and velocity controllers command the lateral attitude controller

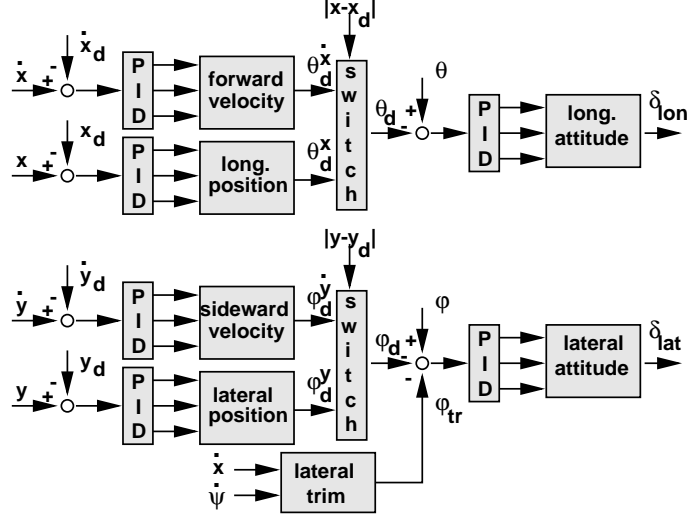


Figure 4: Fuzzy controller architecture for longitudinal and lateral cyclic pitch

to hold a desired roll angle ϕ_s . In forward flight and coordinated turns the lateral velocity \dot{y} should be zero to prevent side-slip. In a coordinated turn roll angle ϕ , yaw rate $\dot{\psi}$ and forward velocity \dot{x} are constrained through the equation

$$\phi = \arctan(\dot{\psi} \dot{x} / g) \quad (3)$$

assuming zero lateral velocity. Therefore, a trim offset ϕ_{tr} is added to the desired roll angle ϕ_s according to Eq.3.

We manually derived fuzzy control rules for each block by a process based on trial and error and some knowledge obtained from a linearized model. At this preliminary design stage each fuzzy controller employed a single PID-type control rule. The control system was able to stabilize the hovering helicopter for small deviations from the nominal operating point.

The performance of the helicopter controllers generated by the evolution strategy is evaluated in a simulation based on a nonlinear model of the helicopter. The model was derived from literature on helicopter dynamics [15, 12]. The aerodynamic equations that describe the forces and moments generated by the main and tail rotor can be simplified assuming that the helicopter operates in hover or low velocity conditions.

5 Evolutionary Tuning of the Autopilot

The genetic-fuzzy system approach allows the designer to specify his performance goals directly by means of a scalar objective function and releases him

from a tedious manual tuning and optimization procedure. The method can be used to generate a collection of basic controllers that emphasize on different performance aspects such as robustness, accuracy and speed of execution. This repertoire of basic flight mode controllers constitutes the continuous regulation layer or autopilot of the hybrid flight vehicle management system described in Section 4.1.

The manually designed autopilot controllers for the altitude, heading, lateral and longitudinal control are tuned in a simulation of typical training situations. The performance of each block is tested in a helicopter manoeuvre that commands a sequence of set-points. The fitness function evaluates the time weighted mean square error to the reference signal.

5.1 Lateral and Longitudinal Controller

In the following, the longitudinal and lateral position controller will serve as an example to illustrate the tuning process. The design objective is a fast and accurate system response to a simultaneous longitudinal and lateral position step-input. The simulation lasts 30 seconds and is divided into three sections of 10 seconds each. At time $t = 10$ the helicopter is commanded to move simultaneously 1m forward and side-ward, at time $t = 20$ to return to the origin (see Fig.5).

The objective function is formed by a weighted sum of cumulated state errors. Depending on the way in which the errors are aggregated over time the designer can emphasize different performance aspects such as short reaching time, minimal overshoot or minimal oscillation. The positional error is defined by a two component vector $\vec{e} = [x - x_d, y - y_d]$. The integral of the square error (ISE) which is defined as

$$E = \int_t \vec{e}^T \circ \vec{e} dt \quad (4)$$

is often used as a performance index. It achieves a good compromise between fast execution of a manoeuvre and sufficient damping.

In our case the helicopter is given the mission to locate and recognize objects on the ground by means of a visual sensor. In order to obtain a stable camera image accurate positioning and fast damping becomes more important than high manoeuvrability. Therefore we use a performance index based on the integral of time multiplied by the square error (ITSE)

$$E = \sum_{i=1}^3 \int_t^{T_i} t * \vec{e}^T \circ \vec{e} dt \quad (5)$$

which reduces the effect of large initial errors and penalizes poorly damped oscillations occurring later in the response. Notice, that the index i is used to distinguish the three sections of the simulation.

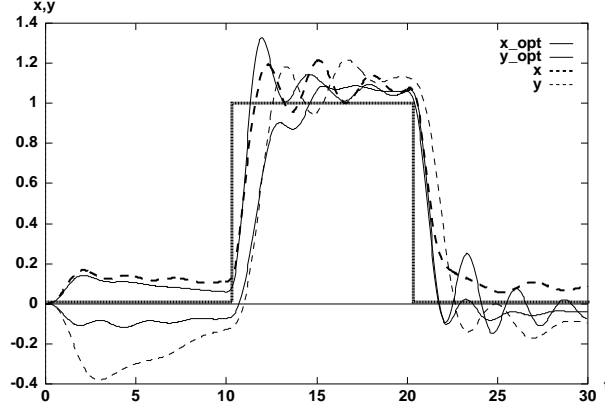


Figure 5: Lateral and longitudinal motion in response to a step-input for original (- - -) and optimized (—) fuzzy controllers.

Fig.5 compares the performance of the original hand designed controllers with the fuzzy rules tuned by the evolution strategy. The ITSE performance criteria was improved by 35% for the longitudinal and 62% for the lateral motion.

5.2 Altitude Controller

The altitude module is optimized in a similar way as the longitudinal and lateral block. At time $t = 10$ the helicopter is commanded to climb to an altitude of $2m$, at time $t = 20$ to descend to its original altitude. The evaluation of the manoeuvre is based on the fitness function in Eq.5.

The rule base of the altitude controller is composed of five PID type fuzzy rules of the form

$$R'_i : \quad \text{if } e_z = A_i \text{ then } \delta_{coll} = c_i^{e_z} e_z + c_i^{e_z^D} e_z^D + c_i^{e_z^I} e_z^I + c_i$$

in which E_z , E_z^D and E_z^I denote the error in altitude, the change in error, the integral of error and δ_{coll} defines the collective pitch of the main rotor blade. The domain of the altitude error is partitioned into the fuzzy sets A_i . The rule consequences are defined by three gain factors and a constant offset c_i , resulting in an overall number of 20 parameters.

In the following we compare the incremental learning scheme with an optimization process that employs a static structure of the complete parameter set. Both methods were tested in two different scenarios, one in which the parameters of the initial population are randomly drawn and the one in which they are initialized with the gain factors of a manually designed PID controller.

In case of the random initialization of fuzzy rules none of the 20 initial controllers with a complete static rule base was able to stabilize the helicopter. Due

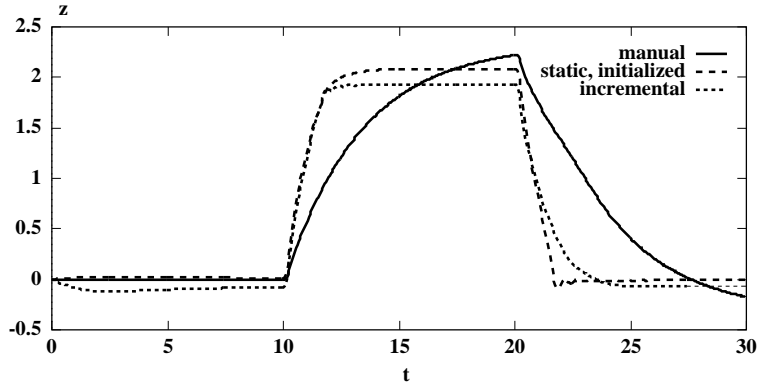


Figure 6: Upward and downward motion in response to a step-input for the manually designed PID (—) controller, the controller designed by a self-tuning process (- - -) and the fuzzy controller designed by the self-organizing process (- . -).

to the large number of parameters, feasible solutions are sparsely distributed over the vast search space. Therefore, even the chances of finding a second-rate controller through random search are minimal. Aiming for an easier objective by weakening the fitness function presents one way to overcome this problem. Instead of using the ITSE as a criteria the controllers are evaluated according to the time elapsed before the helicopter becomes unstable. The evolutionary algorithm starts using the original performance criteria as soon as the controllers are able to handle the basic stabilization task.

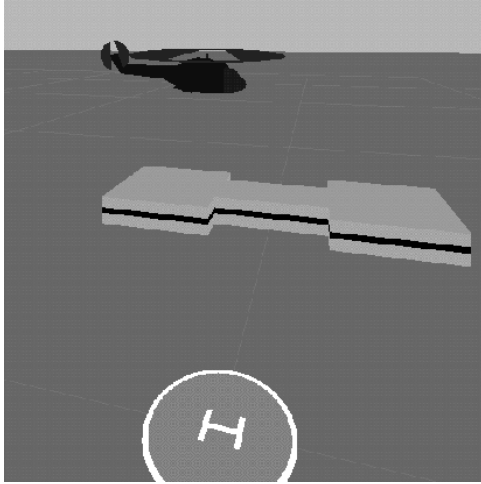
The initialization problem can be also be solved by designing an approximative PID controller manually and use its parameters to initialize the gain factors of the first generation. For the altitude controller we derived an ad hoc PID controller through trial and error. This candidate solution is far from optimal in the sense that a profound controller design based on classical linear control theory will provide a much better performance. This approach constitutes a self-tuning process in which the evolutionary algorithm optimizes the performance of an already existing solution. Fig. 6 compares the performance of the manually derived ad hoc PID controller with the TSK fuzzy controller that arises from the evolutionary tuning. The ITSE criteria from Eq.5 is improved by 89%, although most of the progress is due to the poor performance of the initial PID controller.

In the incremental learning case, the optimization becomes a self-organizing process that learns the fuzzy rules starting without any previous knowledge. The genome of the initial population is composed of fewer parameters, which significantly reduces the dimension of search space. Therefore, the likelihood that random initialization produces a stabilizing controller increases. In case of

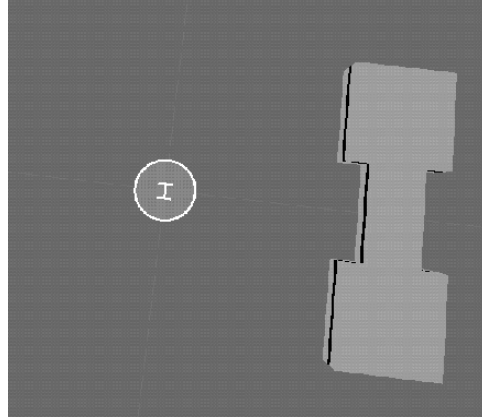
the altitude controller 20% of the initial population manage to keep the helicopter stable during the 30 seconds of the trial. The initial genome is composed of four genes that represent a single fuzzy rule. The number of rules is increased to three after five generations and finally to five after ten generations. The ITSE of the best solution is 34% larger than the fuzzy controller designed by the tuning process (see Fig.5). This result demonstrates that the incremental learning scheme is able to design suitable controllers starting from scratch, which performance is comparable to controllers obtained from a tuning of predesigned fuzzy rules.

6 SmartAerobots - Simulation of Autonomous Aerial Robots

In addition to the conference paper, our online contribution will include animations of the helicopter flying in a 3D virtual environment. The flight animations are generated by the visualization tool SmartAerobots and provide a global as well as an on-board camera view of the helicopter and the environment. The user is able to choose among a variety of helicopter manoeuvres and compare the performance of fuzzy controllers of different complexity. SmartAerobots is a 3D virtual environment simulation that is a visualization tool for the control schemes and vision algorithms designed for DV8. SmartAerobots is built on top of a simulation based on mathematical models of the helicopter dynamics. As a visualization tool, SmartAerobots presents the simulation results in a form that is easier to analyze for the designer. In its current state of development, SmartAerobots takes the trajectory and heading information of the helicopter produced by the simulation and animates either the helicopter flying in a 3D virtual environment, or the view of the virtual environment from the helicopter's camera (see Figure 7). Currently, we are extending the SmartAerobots simulation to become a platform for the design and verification of the vision system of the DV8 helicopter. The simulation will include implementations of vision algorithms for tasks such as motion estimation based on the image sequences produced by the helicopter camera's view. The performance of motion estimation algorithms can be evaluated by comparing the estimated motion with the helicopter's true motion. The vision system will also perform tasks such as obstacle detection and target recognition. Based on the position of the recognized obstacle or target in the camera's image, the vision system will send new reference inputs to the path planner to regenerate the flight trajectory. In this way, DV8 will perform higher level tasks such as obstacle avoidance and target tracking based on visual servoing. In this respect, SmartAerobots will be an invaluable tool in the design of the coupling between the control and vision algorithms, and as a platform for the verification of these algorithms.



(a) View of Helicopter in Flight



(b) View from Helicopter's Camera

Figure 7: SmartAerobots Simulation of Helicopter Approaching Landing Platform

7 Summary

This paper proposed a genetic fuzzy system that automates the knowledge acquisition step in fuzzy control design. An evolution strategy adapts a vector of real numbers, which represent the gain factors of TSK-type fuzzy rules. The optimization process gradually expands the structure and number of fuzzy rules by incorporating additional genes into the genome. The rule base design is decomposed into smaller subproblems that become more feasible to evolutionary optimization.

The proposed learning method was applied to design a helicopter autopilot that constitutes the continuous regulation layer of a flight vehicle management system. A collection of controllers for the lateral, longitudinal, altitude and heading control modules are optimized according to the desired control behavior and context of helicopter operation. The visualization tool SmartAerobots is helpful to verify the control design and enables the integration of control and vision algorithms.

The incremental design method was able to learn the rule base of the altitude controller starting without any previous knowledge, whereas the optimization of the full structure controller in one step required the proper initialization of fuzzy rules by means of some previously derived plausible parameters. The results indicate that incorporating existing domain knowledge into the initial population facilitates the evolutionary optimization task. The evolution strategy that starts with a single fuzzy rule circumvents the curse of finding a feasible set of

initial parameters in a high-dimensional search space. The altitude controllers designed by the self-organizing process demonstrated comparable performance to the fuzzy controllers obtained from the tuning approach.

Acknowledgment

This work was funded by a research grant (Ho 1790/2-1) from the Deutsche Forschungsgemeinschaft and supported by the Army Research Office under grants DAAH 04-96-1-0341.

References

- [1] Antsaklis, P., Kohn, W., Nerode, A., Sastry, S., "Hybrid Systems II", Springer-Verlag, (1995).
- [2] Th. Bäck, H. P. Schwefel, "Evolution Strategies I: Variants and their computational implementation", "Evolution Strategies II: Theoretical aspects and implementation", *Genetic Algorithms in Engineering and Computer Science*, Ed. G. Winter , J. Perieaux, M. Gala, P. Cuesta, pp. 111-126, pp. 127-140, John Wiley & Sons, (1995).
- [3] 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).
- [4] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Reading Massachusetts: Addison-Wesley, (1989).
- [5] 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).
- [6] 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).
- [7] F. Hoffmann, "Incremental Tuning of Fuzzy Controllers by Means of an Evolution Strategy", *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Ed. Koza, J., Banzhaf, W. et al. , Madison, Wisconsin, Morgan Kaufmann, (1998).
- [8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, (1992).

- [9] C. L. Karr, "Design of a Cart-Pole Balancing Fuzzy Logic Controller using a Genetic Algorithm" *SPIE Conf. on Applications of Artificial Intelligence*, Bellingham, WA, (1991).
- [10] T.J. Koo, F. Hoffmann, H. Shim, B. Sinopoli and S. Sastry, "Hybrid Control of Model Helicopter", IFAC Workshop on Motion Control, Grenoble, France, (1998).
- [11] 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).
- [12] Barnes W. McCormick, "Aerodynamics Aeronautics and Flight Mechanics", John Wiley and Sons, (1995).
- [13] N.E. Nawa, T. Hashiyama, T. Furuhashi, Y. Uchikawa, "Fuzzy logic controllers generated by pseudo-bacterial genetic algorithm with adaptive operator", Proc. IEEE International Conference on Neural Networks, vol.4, p.2408-2413, Houston, Texas, (1997).
- [14] C. Phillips, C.L. Karr, G. Walker, "Helicopter flight control with fuzzy logic and genetic algorithms", *Engineering Applications of Artificial Intelligence*, vol.9, (no.2), Elsevier, p.175-84, (1996).
- [15] Raymond W. Prouty, "Helicopter Performance, Stability, and Control", Krieger Publishing Co., (1995).
- [16] N.N. Schraudolph, R.K. Belew, "Dynamic parameter encoding for genetic algorithms", *Machine Learning*, vol.9, (no.1), p.9-21, (1992).
- [17] H. Shim, T. John Koo, F. Hoffmann, S. Sastry, "A Comprehensive Study on Control Design of Autonomous Helicopter", submitted to the 37th IEEE Conference on Decision and Control CDC'98, Tampa, Florida, (1998).
- [18] M. Sugeno, I. Hirano, S. Nakamura, S. Kotsu, "Development of an intelligent unmanned helicopter", *Proceedings the Fourth IEEE International Conference on Fuzzy Systems*, p.33-4 vol.5, (1995).
- [19] 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).
- [20] 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).