A JAVA system supporting fuzzy controller design in robotics

G. ATTOLICO, G. MAGGIO, A. ITTA
Istituto di Elaborazione e Segnali ed Immagini
Consiglio Nazionale delle Ricerche
Via Amendola 166/5, 70126 Bari, ITALY

Abstract: - The paper addresses the problem of designing fuzzy reactive controllers for robotics applications. Describing the straight association between input and output spaces required by this kind of controllers involves several critical choices (especially fuzzification of input and output variables and rule base drawing) heavily affected by the subjective skills of the designer. Automatic tools working on suitable training samples can greatly simplify and making more objective this process.

The paper describes a system developed for supporting all the design phases of fuzzy controllers. The platform-independent JAVA language has been choosen in order to obtain the benefits of object-oriented environments and develop an easily portable system, including its graphical user interface.

The system offers a graphic user interface allowing four different modalities for interacting with a real vehicle in order to execute the desired task: the strategy used by the operator is recorded (in terms of examples of the association between sensory data and control commands). A few tools supports the operator during this phase by providing graphic and/or audio feedback about the quality of its driving with respect to previously defineable criteria. The collected examples are then analysed by a module using a machine-learning based algorithm for identifying the inputs really relevant for each situation and for drawing rules suited for replicating the control strategy.

Our experiments show that the operator does not need to be concerned with fuzzy logic and control: his only care is to execute the task at its best. As a further advantage, the use of fuzzy rules for describing the automatically derived control strategy allows its eventual validation and/or refinement by human engineers.

Five operators, with different characteristics with respect to knowledge of fuzzy logic and driving skills, have been used for obtaining different training sets related to the task of driving a real vehicle along the right-hand wall in an indoor environment. The results obtained by the corresponding rule bases, generated by the system, are shown and discussed in order to evaluate the effectiveness of the approach. CSCC'99 Proceedings, Pages:7141-7147

Key-Words: - Reactive control, fuzzy controllers, rule-base drawing, supervised learning, JAVA language

1 Introduction

Building control systems by combining basic reactive behaviors (straight associations between input and output spaces) is becoming popular in robotics for achieving generality and robustness. Non-linearity and complexity of such functions, often defined in a multidimensional domain, can make difficult their design, making desirable automatic tools supporting this activity.

Fuzzy rules, besides providing a smooth control (a very desired property), can be derived by appropriate learning algorithms and remain easily understandable by humans, for direct examination and, eventually, modification. Identifying input and output variables, defining their fuzzy partitions (terms and related membership functions) and drawing a suitable rule base expressing the control strategy involve increasing levels of subjectivity. While input and output variables naturally arise from the problem, their partition in fuzzy terms is less immediate. Finally, identifying and expressing a control strategy into working fuzzy rules can be difficult for a task less than trivial and can heavily depend on the designer skill. Supervised learning may be useful for extracting rules from appropriate and meaningful training examples [1, 7, 2, 5].

The paper describes a system supporting the development of fuzzy reactive controllers from examples recorded by observing a human operator accomplishing the task at hand. Its modular architecture (realized in Java) allows an operator to drive a vehicle, using different modalities, on the base of both external (sonar, CCD camera, ...) and internal sensory data that are stored with the corresponding control commands. We argue that this wide support increases the quality of training examples and of the automatically derived rules. Moreover the modified generation algorithms, fully exploiting the fuzzy representation of the problem, select data really relevant for each control situation and draw the rules

replicating the driving strategy of the operator. The system supports driving style evaluation (both on-line and off-line) and training set extension when critical spatial configurations are encountered.

The system has been tested on the wall-following basic behavior (moving the vehicle at a constant distance from the wall on the right hand, avoiding collisions and with minimum linear and rotational accelerations). Linear and steering velocities are set on the base of data provided by a sonar ring. The obtained results confirm that the operator just needs to exhibit a sufficient skill in driving the vehicle. He can ignore the structure of the system as long as the theories it is based upon. Even the designer, besides defining the translation of input and output spaces in fuzzy terms, just has to tune a few parameters of the algorithms generating the rule base.

2 Fuzzy rule base generation

The controller for the task at hand (wall-following) is composed by two tightly coupled fuzzy controllers, each in charge of a different velocity (linear and steer). Even if formally divided, they must be logically considered as one entity: a strong relation between velocities is mandatory for a safe and efficient navigation. Both the controllers have n inputs $x_1,...,x_n$ and a single output y, associated respectively with the continuous universe of discourse U_I and U_O. $A=(A_1, A_2, ..., A_n)$ denotes the multivariate fuzzy input variable (each $A_k \in U_I$, k=1,..,n, has N_k linguistic terms $A_{k1},A_{k2},...,A_{kN_k}$}). The fuzzy output variable, $B \in U_O$, has M linguistic terms $B_1,B_2,...,B_M$. Further details on the fuzzy system used in our experiments can be found in [3].

The first algorithm, derived from Sudkamp and Hammel [6, 3], set the consequent of a rule has weighted-mean of the output values associated to the training data verifying its antecedent (weights depend on the activations the training samples produce on the rule) [3]. We define:

$$T_i = \{(\bar{x}, y) \mid \mu_{A^i}(\bar{x}) > \varepsilon\}$$

(A_i is the antecedent of the rule and ε a pre-defined threshold). The crisp consequent w_i of the rule r_i is:

$$w_i = \frac{\sum_{(\bar{x},y)\in T_i} \mu_{A^i}(\bar{x})\, y}{\sum_{(\bar{x},y)\in T_i} \mu_{A^i}(\bar{x})}$$

and the corresponding fuzzy term B_i is B_{j*}, so that:

$$\mu_{Bj*}(w_i) = \max_{j=1,...,M} \mu_{Bj}(w_i)$$

The method has been modified for better coping with crisp values activating adjacent fuzzy terms with similar degree. The new system generates several rules with the same antecedent and the consequents having membership values close to the maximum. Formally, the system generates a set of rule of the form
$A_i \rightarrow B_j$ if

$$\mu_{Bj}(w_i) / \mu_{Bj*}(w_i) > \gamma$$

with $\gamma \in [0,1]$ being a threshold on similarity. Each of them receives a weight defined as:

$$weight_{ij} = \mu_{Bj}(w_i) \;/\; \Sigma_k \mu_{B_k}(w_i)$$

with k representing the terms satisfying the condition (1).

The first version of the algorithm could generate at most $N = \Pi^n_{i=1} N_i$. In the new version this number depends also on granularity of the output, that is: $N' = N \cdot M$ The previous completeness condition (N rules) becomes $\Sigma^k_{i=1} weight_i = N$ with $k \le N'$ (usually $k << N'$). The not complete coverage of the input space, depending on the distribution of sample data, can be solved by using *the principle of similarity* (distance between antecedents) giving similar consequents to rules with similar antecedents.

The second algorithm reduces the number of rules (without affecting the completeness) by selecting really relevant inputs for each control situation. A machine learning approach (ID3), proposed by Quinlan for building decision trees, had been used for reaching this goal [4,3]. The method recursively choose the most relevant input variable (with greater information content and better separation of training data) as root of the potential sub-tree. The expansion is conditioned to a significant increase in control accuracy. Rules are generated by following each path from the root of the tree to a leaf (chaining conditions on input variables in the antecedent and picking the output value from the final leaf). Each path shorter than the maximum depth covers several rules generated by the previous algorithm.

Also this algorithm has been modified for exploiting the fuzzy information available. We define:

- μ_t as the global activation available from training samples at a point of the tree (the sum of training samples memberships with respect to conditions encountered along the path to the point)
- μ_{Bj} as μ_t modulated by the membership of

output values y with respect to the fuzzy term B_j

$\mu_{A_{ki}}$ as μ_t modulated by the membership of x_k with respect to the fuzzy term A_{ki}

$\mu_{A_{ki}B_j}$ as μ_t modulated by both the membership of y with respect to B_j and of x_k with respect to A_{ki}

The relevance of each input variable (estimated by the Quinlan's *Gain ratio* becomes:

$$GR(x_k) = \frac{INF(y) - M_k}{INF(x_k)}$$

where:

$$INF(y) = -\sum_j^M \frac{\mu_{B_j}}{\mu_t} \log_2 \frac{\mu_{B_j}}{\mu_t}$$

is the information content of the potential sub-tree with respect to the output variable,

$$M_k = \sum_i^{N_k} \frac{\mu_{A_{k_i}}}{\mu_t} \left(\sum -\frac{\mu_{A_{k_i}B_j}}{\mu_{A_{k_i}}} \log_2 \frac{\mu_{A_{k_i}B_j}}{\mu_{A_{k_i}}}\right)$$

is the expected information content of the sub-tree having as root the input variable x_k with respect to output values,

$$INF(x_k) = -\sum \frac{\mu_{A_{k_i}}}{\mu_t} \log_2 \frac{\mu_{A_{k_i}}}{\mu_t}$$

is the information content associated to the input variable x_k as root of the potential sub-tree.

3 System architecture

The system is composed by three applications: **Interactor** supporting the collection of training data; **Generator** for the automatic derivation of suitable fuzzy rules reproducing the driving strategy of the operator; **Driver** executing controllers and supporting their verification in driving the real vehicle. They have all been implemented in Java, obtaining an almost complete hardware independency. A basic layer (two packages of classes implementing the interface with the robot and the basics of fuzzy systems) does support the application layer.

A meaningful training set is crucial for any supervised learning strategy: its completeness (coverage of the whole input space) and consistency (constancy in associating control actions to sensory data) heavily affect the performance of the resulting controller. The system designer is charged with the identification of training situations covering most of the possible sensory configurations (usually a trial-and-error process). Consistency of data is a major concern. Humans can provide different commands in similar conditions (due to lack of attention, to wrong perception of the world, to rough understanding of the task, ...). The system support greatly improves the final results by providing: a simple and natural user interface, related to the task, without fuzzy logic or learning algorithms concerns; a sensory representation giving to the operator the same information available to the robotic systems; a feed-back information about the driving consistency with the requirements of the task at hand.

The **Interactor** provides four different modalities for acquiring examples (fig. 2). The **Direct Operator Modality** allows driving by the hardware joystick available on the vehicle while the module records sensory data and associated commands. This natural mode simply requires a sufficient skill in moving the joystick. The system can eventually provides on-line (through a voice synthetizer) the measures acquired by the sensory systems. In this modality the operator can use sensory information that are not available to the control system, resulting in a strategy that is not detectable into the recorded training data. In the **Remote Controller Modality** a working controller drives the vehicle while the system records the association between sensory data and commands. Due to the absolute consistency of this driving, these data are particularly suitable for early testing of new generation algorithms. In the **Remote Operator Modality** the sensory data acquired by the robotic system are presented on the screen and the operator drives using a software joystick. He must rely only on the data available to the vehicle: therefore if he is able to accomplish the task, also the controller is expected to do the same. This usefully checks the balance between task complexity and amount/kind of information provided by the sensory systems. The **Mixed Operator/Controller Modality** is useful when failures arise in specific situations (due to poor or insufficient training data). The controller drives the vehicle and the operator (using the hardware or the software joystick) takes control in those situations supplying further correct examples.

The **Generator** stand-alone application implements the two previously described derivation algorithms. The system designer must set few parameters of them (in our tests they have been set once and left

unchanged for all the experiments) using a very straightforward interface for direct editing structured text files. During the generation, the designer can check the state of progress and evaluate the produced rules. The rule base, stored in structured text files, can be edited, evaluated and modified. While direct modifications are discouraged, a direct examination often provides enough information for recognising poor controller, preventing the loss of time for a test on the real vehicle.

	Alg. WM	Alg. FDT
Exec. Time Pentium 133 / Win98	59’ 30’’	13’ 30’’
Exec. Time Alpha 500 / Unix	12’ 30’’	3’
Number of rules	116	32
Covered input space	< 33 %	100%

The **Driver** module read the structured text files produced by **Generator**: fuzzy variables and related terms, a few parameters influencing the specific behavior (desired distance from the wall and minimum distance allowed from a frontal obstacle in the case of wall-following) and the separated sets of rules (controlling linear and rotational speeds of the vehicle). Then it runs the associated controllers.

4 Experimental results

The experiments have been done using a Nomad vehicle running in a real environment. A sonar ring, composed by 16 sonar sensors, has been used as sensory system for all the trials.

Up to three training runs have been done on each of five representative situations for collecting a variety of examples. Five different operators have driven the vehicle using the hardware joystick. They had different knowledge about fuzzy logic and its application to control, different understanding of the effects of their driving style on the quality of examples (and therefore on the rule-base automatically derived from them) and different skills in using the joystick. The Interactor module has been used for collecting training samples, providing, at the same time, the required support to the operator.

The Generator module has been used for deriving a rule-base from each collection of examples, confirming superior performance of the decision tree based algorithm in deriving complete and compact rule bases. All the results shown in fig. 3 have been obtained using the Driver module for running the controllers generated by this method: the number of available training samples, the number of rules derived for controlling linear and steering speeds and a score evaluated using suitable penalty functions are provided.

In two cases (Op \#3 and Op \#5) the rule-bases derived by the automatic algorithms hit the wall: a simple manual intervention on a couple of rules produced a satisfactory (even if not very good) behavior. This is an advantage of the clarity of representation typical of fuzzy rules.

Moreover we had experimental evidence that ability in executing the task (driving the vehicle along the wall) was definitely more important than knowledge about fuzzy logic, control and algorithms. The operator \#4 outperformed the operator \#2, due to its superior skill in driving the vehicle, in spite of its reduced knowledge about the principles and techniques used by the whole system.

Op #1	Deep knowledge about fuzzy logic applied to robotics. Good understanding of the desired behavior, of driving style effects on examples and on derived fuzzy rules. High driving skill. Supported by vocal messages.
Op #2	Deep knowledge about fuzzy logic applied to robotics. Good understanding of the desired behavior, of driving style effects on examples and on derived fuzzy rules. Supported by vocal messages and landmarks on floor.
Op #3	Limited knowledge about fuzzy logic applied to robotics. Lower understanding of the desired behavior, of driving style effects on examples and on derived fuzzy rules. Low driving skill. Supported by vocal messages.
Op #4	No knowledge about fuzzy logic applied to robotics. Lower understanding of the desired behaviors, none of driving style effects on examples and on derived fuzzy rules. Good driving skill. Supported by vocal messages and landmarks on floor.
Op #5	No knowledge about fuzzy logic applied to robotics. Limited understanding of the desired behavior, none of driving style effects on examples and on derived fuzzy rules. Low driving skill. Not supported.

Figure 1. The five operators, each with different skills, used for testing the approach.

5. Conclusions

A system has been realized for supporting the design of fuzzy controllers in robotics. Special care has been paid to the collection of training samples, a critical point for every supervised learning approach. A friendly user interface and four different modalities

(each especially suitable for particular tasks and/or training needs) have been designed for acquiring examples. Two algorithms for the automatic generation of rule baseshave been realized. The representation of derived fuzzy rules is easily understandable and, eventually, modifiable by human designers. Finally, the system provides facilities for both testing and running the controllers.

Extensive tests, accomplished by deriving fuzzy controller for a basic reactive behavior (wall-following), have been done on a real vehicle moving in an indoor environment. The system allows an operator, with no knowledge about fuzzy logic and control, to build a fuzzy controller for a reactive behavior by simply providing a few executions of the task.

The system is being extended to the design of fuzzy partitions for input and output variables (applying unsupervised learning techniques to the training examples) in order to reduce the subjectivity of the whole process.

References:

[1] S. Abe and M. Lan. Fuzzy rules extraction directly from numerical data for function approximation. IEEE Transactions on Systems, Man and Cybernetics, 25(1):119-129, Jan. 1995.

[2] C. Baroglio, A. Giordana, M. Kaiser, M. Nuttin, and R. Piola. Learning controllers for industrial robots. Machine Learning, 1996. (in press).

[3] G. Castellano, G. Attolico, and A. Distante. Automatic generation of fuzzy rules for reactive robot controllers. Robotics and Autonomous Systems, 22:133-149, 1997.

[4] S. Hsu, J. Hsu, and I. Chiang. Automatic generation of fuzzy control rules by machine learning methods. In Proceedings of the IEEE International Conference on Robotics and Automations, pages 287-292, Nagoya, Japan, 1995.

[5] P. Reignier. Supervised incremental learning of fuzzy rules. Robotics And Autonomous Systems – Elsevier, 16:57-71, 1995.

[6] T. Sudkamp and R. Hammell. Interpolation, completion, and learning fuzzy rules. IEEE Transactions on Systems, Man and Cybernetics, 24(2):332-342, Feb. 1994.

[7] L. Wang. Stable adaptive fuzzy control of nonlinear systems. IEEE Transactions on Fuzzy Systems, 1(2):146-155, May 1993.

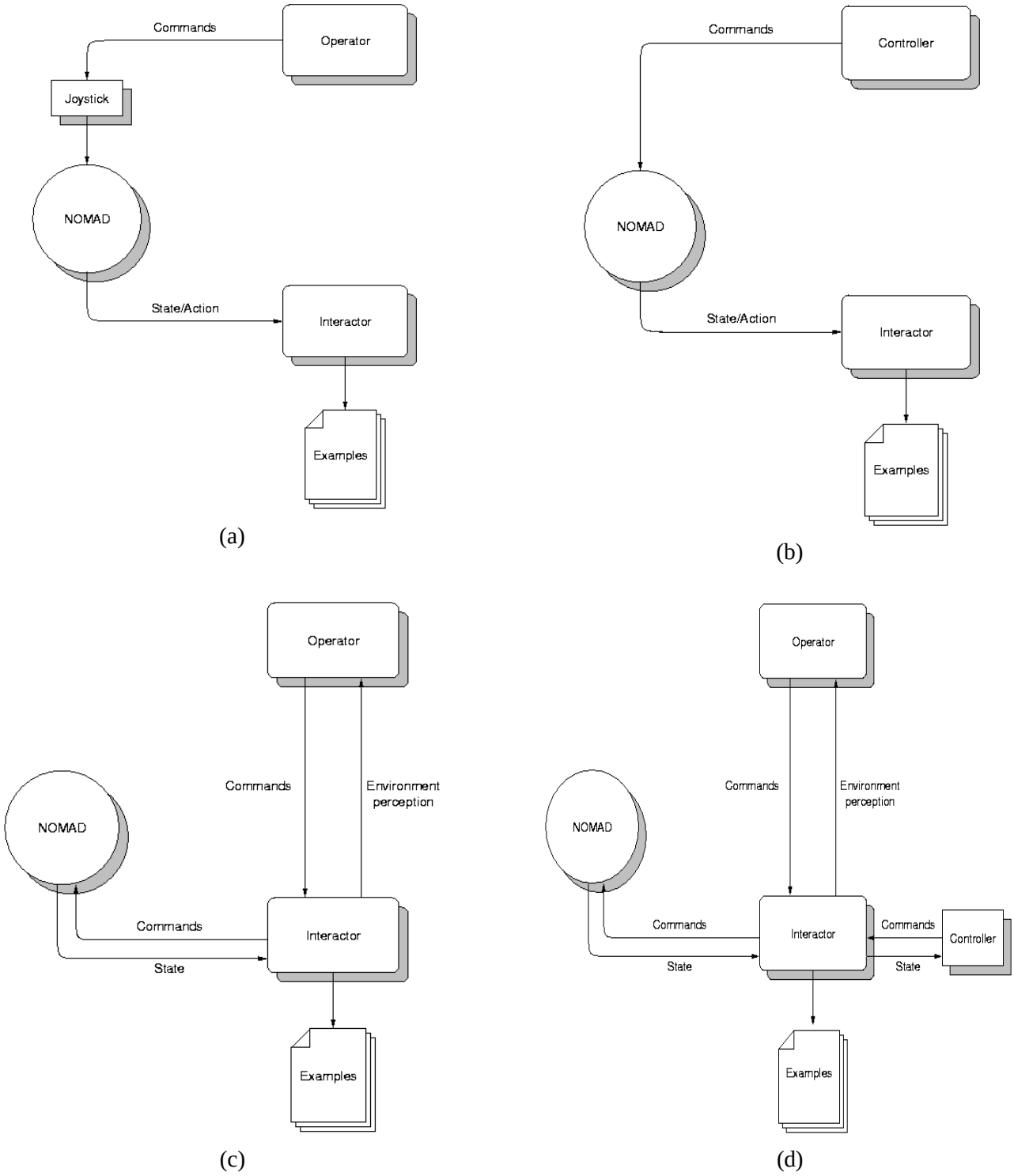
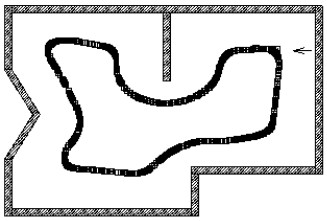
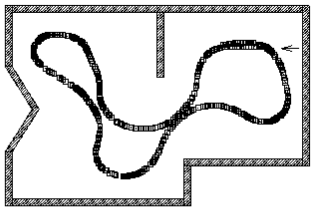


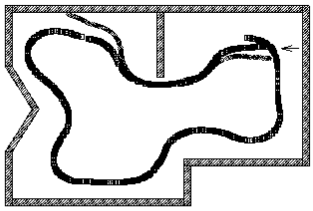
Figure 2. The four different interaction modalities available in the system. In the Direct Operator Modality (a) the operator drives the vehicle by using the aboard joystick, while the module Interactor does record the association between sensory situation and selected commands. In the Remote Controller Modality (b) a software controller can drive the vehicle: again the Interactor module records all the sensory situations and related commands. In the Remote Operator Modality (c) the operator can drive using the software joystick provided by the system and using only the sensory data available to the vehicle (displayed on the screen). Finally in the Mixed Operator/Controller Modality (d), the operator can take temporarily control of the vehicle whenever the controller does exhibit poor behavior: the new examples provided to the system can integrate the original training set in order to improve the performance of the software controller.



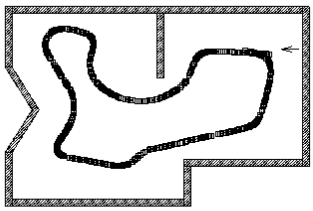
Operator #1	
Number of examples	7718
Number of rules for speed	11
Number of rules for steer	33
Score obtained	0.7250



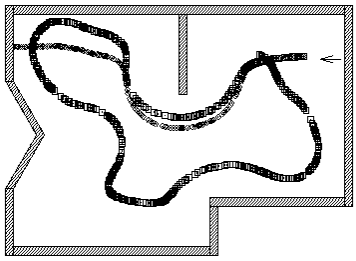
Operator #2	
Number of examples	5147
Number of rules for speed	42
Number of rules for steer	33
Score obtained	0.5817



Operator #3	
Number of examples	6498
Number of rules for speed	1
Number of rules for steer	48
Score obtained (2^{nd} trial)	0.5601



Operator #4	
Number of examples	6151
Number of rules for speed	23
Number of rules for steer	57
Score obtained	0.6573



Operator #5	
Number of examples	4878
Number of rules for speed	36
Number of rules for steer	52
Score obtained (2^{nd} trial)	0.4902

Figure 3. Results obtained by the controllers derived from the training examples provided by the different operators. The arrow shows the starting point. The best controllers (Op #1 and Op #4) come from examples provided by the operators with better knowledge about the right way for performing the task and better manual skill in using the hardware joystick. The knowledge of fuzzy logic and robotic control does not help: Op #2 performs poorer than Op #4 in spite of his superior knowledge about these topics. The scores provided for Op #3 and Op #5 have been obtained after a manual tuning of the respective rule bases (in both cases the first trial resulted in a collision).