First Applications of the Orbex Coprocessor: Control of Unmanned Vehicles

R. Garcia Rosa and T. De Pedro
Instituto de Automática Industrial, CSIC
La Poveda, 28500 Arganda del Rey, Madrid. Spain
{ricardo,tere}@iai.csic.es

Abstract

The ORBEX coprocessor has been designed to execute the typical fuzzy operations of a system based on fuzzy rules. The first real application has been fuzzy controllers for electric cars. The values of the input variables, the position and the orientation of the car with respect the desired trajectory of reference, are obtained from the data provided by a DGPS boarded in the vehicle. The values of the output variables provided by the controller are the angle that the steering wheel has to be turned and the increment of the velocity. Due to the power of the ORBEX coprocessor the controllers can be written with few though meaningful rules.

Key words: fuzzy coprocessor, fuzzy control and unmanned vehicles.

1 The ORBEX Coprocessor

The ORBEX coprocessor has been designed to achieve a process unit for any kind of system based on fuzzy rules. Two are the key words that characterize this co-processor *fuzzy* and *general purpose*. The first means that ORBEX is for typical fuzzy operations only, the second means that ORBEX is able of dealing with any problem modeled in terms of fuzzy sets and rules and linguistic variables, and not only for fuzzy control applications.

As other similar systems the ORBEX process unit has the elemental instructions necessary to accomplish the fuzzy reasoning procedures. But besides ORBEX can process conditions in which appear linguistic values with fuzzy modifiers. Among the permitted modifiers there are some conventional as *no, very, extremely* and *more or less.* But ORBEX can process in exclusiveness other modifiers as *more than* and *less than*. Applications have probed the power of these modifiers in two ways, first reducing the number of rules and second bringing the rules nearer to natural sentences.

This paper explains the functional model of the ORBEX co-processor, the user instructions and the internal data structure. Besides, to verify his performances, a demanding application has been implemented: the control of an unmanned vehicle.

1.1. Functional model evolution

From the first functional design [1] [2] to the present one [3] the ORBEX1 coprocessor has been adapted to several technological requirements. For instance, the variety of shapes allowed for the membership functions of the first model has been reduced to trapeziums and the calculus have been reduced to operations between integers. Now, due to the decreasing prices of the conventional processors, it is accepted that ORBEX can be implemented on a conventional processor, recovering the riches of the first model. In this way the ORBEX implementation coincides with the simulation done to verify his performances.

In the application, fuzzy controllers for electric cars, an ORBEX implementation has been done on a CIRYX processor embedded in a development board for industrial applications. The application, included in the COVAN2 project, is adequate to test the ORBEX performances, because the fuzzy controller has to be executed in real time on the computer boarded in the vehicle.

1.2. Problem context

The first step to model a problem as a system based on fuzzy rules is to represent the problem in terms of fuzzy sets and rules. In other words, to create the base of knowledge or problem context that contains the definitions of the linguistic variables involved in the problem and the fuzzy rules that model its behavior.

The variable definitions are done in function of their linguistic values, which in turn contain the shape and parameters of the associated membership functions. In the present ORBEX model the membership functions are trapezoidal.

As it is known, a rule has two parts, antecedent and consequent. The antecedent is a set of conditions, being each one a fuzzy proposition over the input values. Analogously the consequent is a set of conclusions, being each one a fuzzy proposition over the output values. In the ORBEX model, from the formal point of view, there is no reason to distinguish between conditions and conclusions, but attending to their real meaning there are two differences between conditions and conclusions: 1) The conditions can be affirmative or negative and include fuzzy modifiers while conclusions can not. 2) The conditions of a rule can be joined by and conjunctions, if they have to be satisfied at once, also the conditions can be

[1] ORBEX: CICYT, TIC 96 1393-C06-03
[2] COVAN: CAM 06T/042/9

joined by *or* conjunctions, if they can be satisfied alternatively; by the contrary all the conclusions have to be satisfied, so they are joined by *and* conjunctions only.

In other words, while it has sense that there are conditions positive or negative, the conclusions have to lead to real actions, so they can not be negative. (For instance *open valve* and *close valve* can be conclusions in a controller, *add element* or *eliminate element* in a database can be conclusions in fuzzy decision system. By the contrary, *not close valve, not add element*; … etc. are not real actions). By other hand it is nonsense that there were alternative actions with the same effect over the process (for instance, the conclusion *open valve1* or *open valve2*). Normally all the possible control actions are determined a priori.

As an example, in a scrap-smelting controller, an input variable can be the *temperature* of the blast furnace. A condition can be *temperature no very high*, being *high* a linguistic value of the variable. The membership function associated with high can be a trapezium defined by the parameters 800ºC, 1000ºC, 1500ºC y 2000ºC. The modifiers *no* and *very* are defined by a complement operator and an intensification operator respectively. Analogously an output variable can be valve and conclusion can be *valve open*, being open one of his linguistic values. The membership function associated with *open* can be an increasing ramp between 0 y 1. Finally the expression: “IF *temperature no very high* AND ... OR ... THEN *valve* open AND ... " can be a control rule.

In conclusion, ORBEX can process meaningful rules. It must be noticed that the form of the consequent is not a limitation, but the verification of what conclusions mean in practice, because the same procedures dealing with the antecedent can deal with the consequent.

1.3. Fuzzy ORBEX instructions

The ORBEX coprocessor works in two modes: load and execution. In the load mode the processor fulfills the data structures of the internal representation of the problem. In the execution mode ORBEX performs the typical fuzzy procedures.

Essentially two data structures, base of variables and base of instructions, form the problem representation in memory. In the load mode these databases are fulfilled with the variables and rules contained in the problem context [4]. There are two types of load instructions: *load variable* and *load instruction*. The instruction *load variable* stores the name of the variable, the names of its linguistic values and the associated membership functions in the elements of the base of variables.

The *load instruction* splits the rules in fuzzy instructions and stores them in the base of instructions. A rule is transformed in a sequence of three types of fuzzy instructions: *fuzzify, infer* and *defuzzify* in the following way: Each condition is translated in an instruction *fuzzify*, each conclusion in an instruction *infer* and finally each control output in an instruction *defuzzify*.

In the execution mode, for control application, the three instructions, *fuzzify, infer* and *defuzzify* are performed. The instruction *fuzzify* calculates the degree in which the current value of the condition variable fits its linguistic values. For instance, if the current value of the temperature into the electric blast furnace is 1450ºC, the instruction *fuzzify* will give the degrees of membership to the all linguistic value *high, medium,*…etc of the variable *temperature.*

The instruction *infer* calculates the degree of truth of a rule, combining the degrees of its conditions. In the case of a fuzzy controller this truth-value is the degree in which the rule influences the control action. The inference procedure, as usual, propagates this degree of truth from the antecedent to the conclusions of the consequent. This degree is stored in the elements of the base of variables associated with the output variables appearing in the rule consequent.

The form of the antecedent referred before implies that the instruction *infer* calls for lower level instructions to calculate the intersection, the aggregation and the complement of conditions.

The instruction *defuzzify* provides the numeric value of an output variable, for instance the value of the angle that the steering wheel has to turn. The instruction *defuzzify* weighs all the rules involved to obtain the concrete action.

1.4. User instructions

The ORBEX programmer has to know neither its internal language nor its data structures, he has to know the following four user instructions only: *fzzinit* (...), *fzzinput* (...), *fzzrules, fzzoutput* (...). The user interface is a sequence of user instructions.

The instruction *fzzinit* (...) sets the ORBEX in load mode and creates the base of variables and the base of instructions of the problem. This instruction is the first and the unique of this type in the user interface. It has one parameter, the name of the file containing the problem description.

The instruction *fzzinput* (...) fuzzifies the current value of the input appointed by the parameter. There are as many instructions *fzzinput* as inputs have the problem, the order is not relevant but they have to be behind the instruction *fzzinit*.

The instruction *fzzrules* does the fuzzy inferences. There is only one and has to be behind the last instruction *fzzinput*.

The instruction *fzzoutput* (...) obtains the value of the output appointed by the parameter. These instructions are the last of the user interface and there will be as instructions as outputs involved in the problem.

Except the instruction *fzzinit* the other are done in execution mode.

An example of a user interface can be:

fzzinit (control of vehicles)
fzzinput (distance)

fzzinput (angle)
...
fzzrules
fzzoutput (steering wheel)
fzzoutput (pedal)
...

1.5. Controller implementation in ORBEX

The research projects of the IAI have determined the first real implementation of the ORBEX model, fuzzy controllers for electric cars. The concrete problem is to control the direction, the brake and the accelerator of a CITROËN BERLINGO van augmented with a differential global position system (DGPS). The car circulates by private paths where the conventional traffic is forbidden. The set of these paths is named ZOCO3 and it simulates the streets of an urban district.

In this application the ORBEX hosts the fuzzy controllers which determine in each control cycle the positions of the steering wheel, the brake and the accelerator. In this way the car moves along the desired trajectory with the adequate velocity, in function of the current car position and orientation provided by the DGPS.

2. Application: Control of unmanned cars

The aim of the COVAN project is to develop the basic techniques in order for cars to execute some maneuvers by themselves. Eventually the final objective of this project to substitute the driver but, by now is to free him of some tedious tasks like parking, reversing, moving in platoon, to help handicapped people and to provide automatic guiding vehicles in some dangerous applications like circulating in pollutioned or burning zones. Besides, there are many applications in which substituting the driver is possible and convenient.

The experimental zone, ZOCO, is a set of paths with a total length of about 1 km. and a width of 6 meters. These dimensions allow that two cars move in parallel. In order to allow that a car route could be determined symbolically road segments and crosses have been identified by names. In this way the user define the desired trajectory by a sequence of names, that has to be mapped in the real positions of ZOCO. So the controller has to know a map of the experimental zone. This map is built with the data provided by a DGPS that has been moved along the experimental zone.

The real experiments have been done with two electric cars, CITROËN BERLINGO, in which, besides the DGPS; have been installed one motor to act on the steering wheel and an electronic accelerator, which by now acts as break also. A CIRYX industrial computer boarded on car hosts the functional model of the ORBEX Finally a radio communications allows to pass programs, trajectories and positions among the vehicles that move in the zone.

[3] ZOCO: CICYT IN 960118

The figure 1 represents the control hardware. The boarded computer receives by a serial line the data of the DGPS, other line connects the accelerator controller that acts in voltage and receives the measure of the velocity from the tachometer. Lastly a KELVIN board that acts a motor de 100 w. embedded in the steering wheel controls the direction.

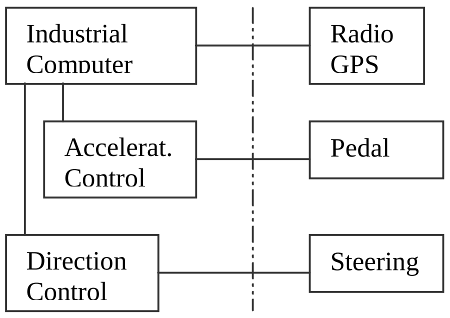


Figure 1. Control hardware

2.1. Basic control of a vehicle

The model of the world that uses the car is reduced to a reference line, the desired trajectory; the deviations of the real positions of the car from this reference are the lateral displacement and the angle between the car orientation and trajectory of reference [7]. These two variables, displacement and angle, joint to the velocity of reference are used as inputs in the control strategy. The outputs or control variables are the pedal and the steering wheel. The definition of variables and the rules to control the direction and the acceleration of car are the following:

```
Velocity       {reference -2 0 0 2 }
Angle          {zero -20 0 0 20}
Displacement   {negative -100 -100 -2 0; positive 0 2 100 100}
Pedal          {up -100 -100 -1 0; down 0 1 100 100}
Steeringwheel  {negative –100 -100 -1 0; positive 0 1 100 100}

if velocity more than reference then pedal up
if velocity less than reference then pedal down
if angle less than zero then steering wheel positive
if angle more than zero then steering wheel negative
if angle zero and displacement positive then steering wheel negative
if angle zero and displacement negative then steering wheel positive
end
```

2.2. First experiments with cars

Before doing real experiments some trajectories have been simulated in a Silicon Graphics station [6]. The simulation programs have the kinematic model of cars and the map of the conduction zone; it shows the cars moving while keeping the traffic rules. The velocity, the acceleration and the power consumed depend on the control rules. It is interesting to remark that the simulation recommended changing the initial control rules and the parameters of the membership functions.
The simplest maneuvers realized have been: to drive in straight line and to turn. In particular:

1) To go and return with different speed along the Zadeh Avenue. The figures 3 and 4 show the answers of the velocity controller when the car stops while it is going uphill and downhill respectively.

2) To go along the route: Zadeh 220, Mercator, Torres Quevedo, Enrique el Navegante, Juanelo, Mercator, Torres Quevedo, Azarquiel, Zadeh 121. The figure 5 shows the real car trajectory.

The results, recorded in video, allow checking the behavior of the control system, for instance they show that the control brakes the car when a turn is near.

These first experiments have leaded to augment the number of input variables and rules. For instance the acceleration has to be taken into account. This variable is calculated by numerical differentiation.

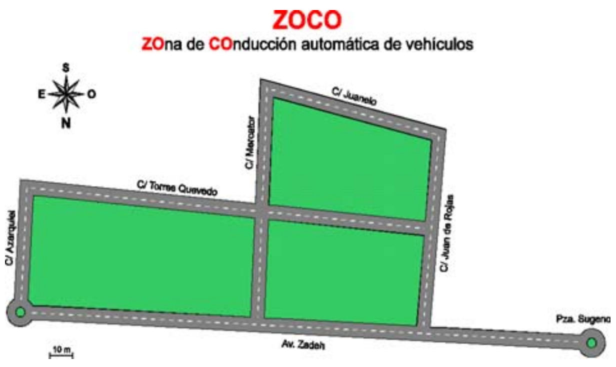
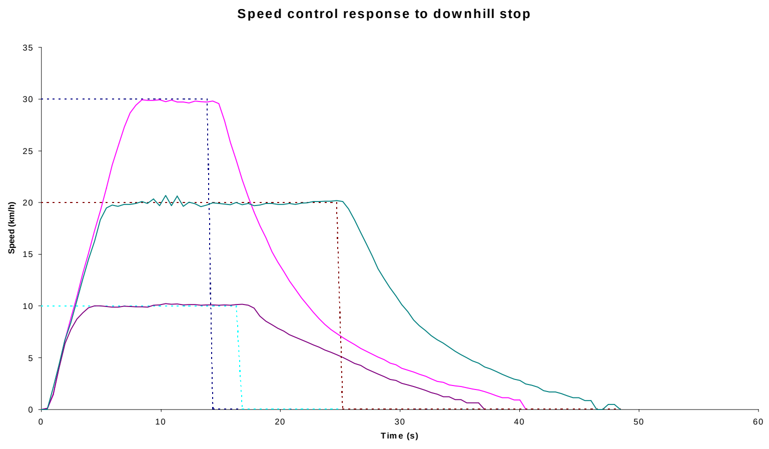


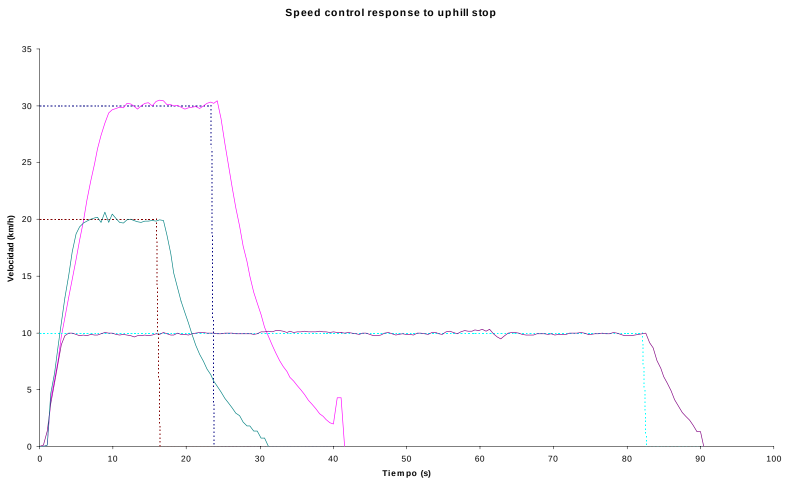
Figure 2. Realistic map of the conduction zone

3. Conclusions

The work done until now shows the efectiveness of fuzzy controllers, even the simplest ones, to create guiding systems which can helped, or even sustitute, people in boring, dangerous or unwealthy situations. These tools can deal with tasks of parking, going back, maintaining the distance in platoons, etc.

One of the main reasons to have chosen fuzzy control to guide vehicles is that, in the future, taking advantage of the power and flexibility of fuzzy reasoning, it will be possible to embed fuzzy controllers into higher level systems to perform complex tasks. For instance to integrate fuzzy controllers into a global traffic control system, to coordinate a fleet of autonomous cars charged of transporting materials in a restricted zone, etc.





Figures 3 and 4. Answers to velocity steps

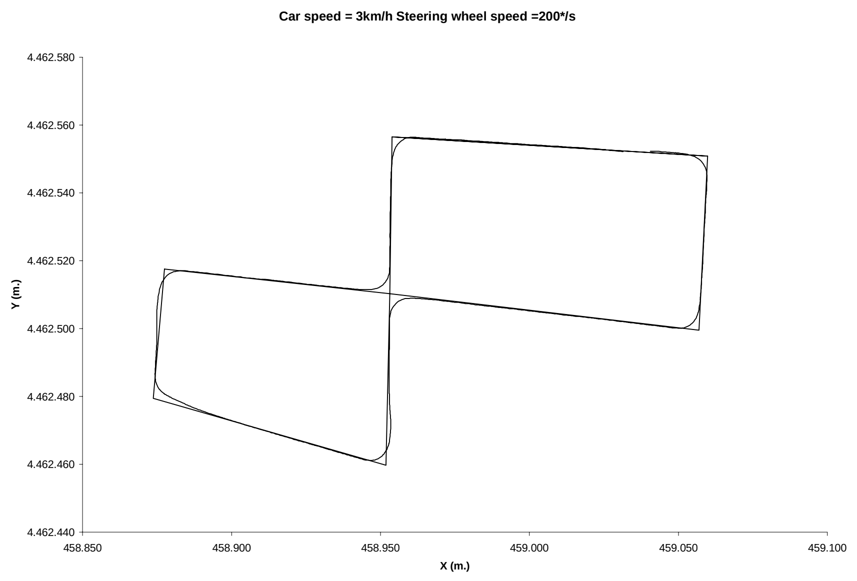


Figure 5. A real trajectory followed by the car

References

[1] R. García Rosa, T. De Pedro. Un modelo de coprocesador borroso. *Actas del ESTYLF'96, VI Congreso Español sobre Tecnologías y Lógica Fuzzy*, 1996.

[2] R. García Rosa, T. De Pedro. Modeling a fuzzy coprocessor and its programming language. *Mathware and Soft Computing*, vol V, nº. 2-3, pp 167-174, 1998.

[3] R. García Rosa, T. De Pedro. A powerfull model for a fuzzy coprocessor. *Proc. of the CLCA'98 and IFAC'98* 1998.

[4] R. García Rosa, T. De Pedro, M.T. de Andrade. Parallel inference engine for fuzzy controllers. *Cybernetics and Systems. An International Journal*, vol 25, nº 2, pg 359-371, 1994.

[5] R. García Rosa, T. De Pedro, A. Rosetti. Fuzzy driving strategies for cars in several traffic situations. *IFAC Sympusium on Intelligent Autonomous Vehicles*, 1998.

[6] R. García Rosa, T. De Pedro. Automatic car drivers. *31st ISATA International Sympusium on Automotive Technology and Automation*, 1998.

[7] M. Sugeno, Murofushi. Fuzzy algoritmic control of model car by oral instructions, *2nd IFSA Congress*, Tokyo.