2nd IFAC Conference on Intelligent Autonomous Vehicles, Helsinki University of Technology, Espoo, Finland, June 12-14, 1995

"Fuzzy Hierarchical Control for Autonomous Vehicles"

Christos Voudouris, Paul Chernett, Chang J. Wang, Vic L. Callaghan

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK.

Abstract

The subject of this paper is the control of autonomous vehicles. A hierarchical approach is studied in the context of fuzzy systems and a programming language for the mid to low level control of autonomous vehicles is described. The language, called FDTL (Fuzzy Decision Tree Language), is based on a computational model that combines fuzzy rule based control with the hierarchical nature of decision trees. Encouraging results are obtained from both the simulator and a real autonomous vehicle.

1. Introduction

Fuzzy rule based systems have been found useful in low-level control in applications ranging from kiln control to helicopter flight. The principle advantages cited for such systems are the ease of incorporating the knowledge of human experts into the prototype of a new controller and the robustness of the results.

The first seems to stem from the expression of the controller as a database of rules each of which maps the current state, at each iteration of the controller, to desired control outputs. The introduction of the notion of "fuzziness" makes the form of the rules closer to the intuitions of expert operators to whom the system developer must turn where the a dynamic predictive models is difficult to develop, as is often the case.

Traditional fuzzy systems only allow outputs to be fuzzy variables that normally map directly to the actuators. This tends to restrict its applicability to low-level systems. We extend the model to allow the outputs of rules to influence other fuzzy systems. In other words, the model becomes recursive, allowing systems to be expressed as hierarchies, nested to arbitrary depth. Complex behaviours can be decomposed, top-down, into simpler behaviours that can be designed independently.

The approach taken is based (but not dependent) on the Fuzzy Logic of Zadeh (Zadeh, 1965) and the fuzzy systems described in (Kosko, 1992a). Such fuzzy systems have been widely used to describe controllers that implement particular desired behaviours of robotic systems such as obstacle avoidance (Beom and Cho, 1992; Song and Tai, 1992) or navigation (Pin et al., 1992; Joo et al., 1993; Baxter and Bumby, 1993).

We use the rule combination feature provided by fuzzy inference to activate competing behaviours simultaneously but to differing degrees, thus providing a smooth transition between them. This fusion has previously been proposed in the case of obstacle avoidance and navigation by (Baxter and Bumby, 1993). However, they suggest a mechanism which lacks generality. Saffiotti et al. (1993) present a more general method based upon an activation scheme for the behaviours. They consider activation levels for behaviours implemented by fuzzy rulebases. The activation levels are determined by fuzzy meta-rules which arbitrate the dominance of different behaviours to the control outputs. Sugeno et al. (1993) move in the same direction considering fuzzy meta-rules that activate the flight modes of an unmanned helicopter at different degrees during a flight. Behaviours that are activated to various degrees using neural nets can be also found in Gachet et al. (1993). There, the emergent behaviour of an agent is defined as the weighted sum of the primitive behaviour outputs.

We further generalise this approach by taking up the proposal by Chang and Pavlidis (1977) to extend the classical decision tree such that all leaf nodes are assigned "degrees of activation" rather than only

one being selected. The final result is a controller in which complex behaviours can be decomposed in a top down manner into behaviours that can be developed independently.

2. The Computational Model

The processing stages in a classical fuzzy controller are fuzzification, fuzzy inference, rule combination, and defuzzification (Figure 1). Fuzzy inference and rule combination are often referred to jointly as fuzzy inference. Recent advances on this basic model incorporate rule strengths (weights) that provide an extra parameter for these systems (Kosko, 1992a; Cox, 1993). This extra parameter can be exploited by adaptive mechanisms or, as in our case, to build hierarchical structures. Fuzzy systems that incorporate rule weights are also known as FAMs (Fuzzy Associative Memories).

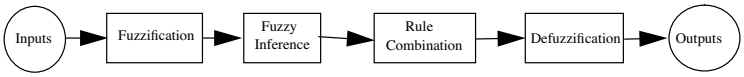

FIGURE 1. The processing stages in a fuzzy controller.

Our model only departs from the FAM theory, as given by Kosko (1992a), in allowing the consequent of a rule to be an entire fuzzy ruleset (fuzzy system) as well as a fuzzy set (Voudouris,1993). In this case the "firing level" of the rule will have a "scaling" effect on the weights of the consequent fuzzy system. This recursion allows fuzzy systems to be expressed hierarchically as fuzzy decision trees. Internal nodes of the tree contain decision making rules where the leaf nodes are traditional fuzzy systems mapping inputs to outputs. In the context of autonomous vehicles the leaf nodes are the behaviours (Brooks, 1986) of the autonomous vehicle. Fuzzy decision making takes place in internal nodes. Activation propagates from the root to all the leaf nodes activating all behaviours to various degrees.

The rule combination process combines these contributions from the various leaf nodes, instead of choosing between them as in other inference schemes. We have employed the "weighted additive" combination scheme as presented by Kosko (1992a, 1992b) but extended to multiple rulebases. In the simple case, where all the leaf nodes operate on the same output, Z, the resulting fuzzy set, C, (it is actually a distribution) in the domain of Z from this inter-rulebase combination is given by the weighted sum of the individual rule contributions C_{ij}' where i indexes the rulebase and j the particular rule of the i-th rulebase (EQ 1).

$$C = \sum_{i=1}^{n} \sum_{j=1}^{m_i} w_{ij} \cdot C_{ij}' \qquad \text{(EQ 1)}$$

The fuzzy set C is usually normalized to [0,1]. The rule weights w_{ij} differ from rulebase to rulebase as a result of the fuzzy decision tree scheme which activates fuzzy systems to different degrees by scaling the weights of their rules. The output value is given by producing a normal "crisp" value from the distribution C. Various methods have been proposed for this (see for example Klir & Folger, 1988). We employ fuzzy centroid defuzzification also known as COA (centre of area) defuzzification. The reader is referred to Voudouris (1993) for a fuller account.

Figure 2 presents the fuzzy decision tree architecture. It is a feed forward system. The incoming information from the sensors together with internal variables (goal positions, battery level, damage detectors, state variables etc.) are fed to the tree. The tree, in a feed forward manner, spreads activation from the root to the leaf nodes. The inputs of all nodes (leaf and nonleaf) are external to the tree. It is actually possible for outputs to be fed back to the tree forming feedback loops although this avenue is yet to be explored by us.

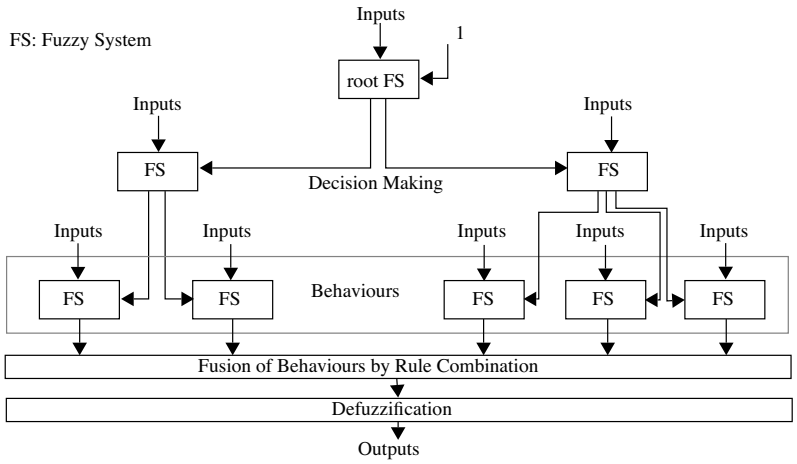

FIGURE 2. The Fuzzy Decision Tree Architecture.

3. The Language

FDTL is a language in which fuzzy decision trees are described. The language is a tool for programming robots and intelligent systems in general. The C code produced by the compiler of the language can be executed in a variety of platforms as embedded or independent programs. We use FDTL programs as embedded controllers in both a mobile vehicle and its simulator.

3.1 Grammar

The grammar of FDTL in EBNF notation is given below (some lexical details are omitted for brevity):

```
<system> ::= system <identifier> begin <inputs><outputs><rules>end.
<inputs> ::= inputs begin <var_declarations> end;
<outputs> ::= outputs begin <var_declarations> end;
<var_declarations> ::= <var_declaration> {<var_declarations>}
<var_declaration> ::= numeric <identifier> '{' <fuzzy sets> '}' |
                      symbolic <identifier> '{' <values> '}'
<fuzzy sets> ::= {<fuzzy set}
<fuzzy set> ::= triangular <identifier><number><number><number>; |
                trapezoidal <identifier><number><number><number><number>;
<values> ::= {value}
<value> ::= <string>;
<rulebases> ::= rulebases begin <rb_declaration> end;
<rb_declaration> ::= <identifier> begin <rules> {<rb_declaration>} end
<rules> ::= <rule> {<rule>}
<rule> ::= if <conditions> then <actions>;
<conditions> ::= <condition> {and <condition>}
<condition> ::= <left operand><relop><right operand>
<left operand> ::= <identifier>
<right operand> ::= <identifier> | <number> | <string>
<relop> ::= < | <= | == | != | > | >=
<actions> ::= <action> {and <action>}
<action> ::= <identifier><assignop><term>
<term> ::= <identifer> | <string>
```

3.2 An Example

In order to give a flavour of FDTL we will describe the most complex of the controllers that we have implemented and which we call GC. The controller is for an autonomous vehicle that steers by differential velocities of it's two driving wheels. The vehicle is also assumed to be equipped with ultrasonic

proximity sensors together with other sensors that can give the angle and distance to various goals. The overall task is to collect and dispose of garbage, while avoiding obstacles, and moving to a recharging point, if the battery condition is low, and to a service point if damage is detected.The fuzzy decision tree for GC is presented diagrammatically in Figure 3. Each node represents a complete rulebase. All leaf nodes in this example have only two outputs, the velocities of each wheel. Where a rulebase outputs to an internal node the output value will act to "scale" the relative contribution of the rulebase at that node. For instance, root of the tree spreads activation progressively to Obstacle Avoidance as the vehicle approaches obstacles. Otherwise, it spreads activation to the right-subtree and so on.

Thus any of the leaves can potentially have an effect on the output velocities but this effect is moderated by the activation level of its ancestor nodes.

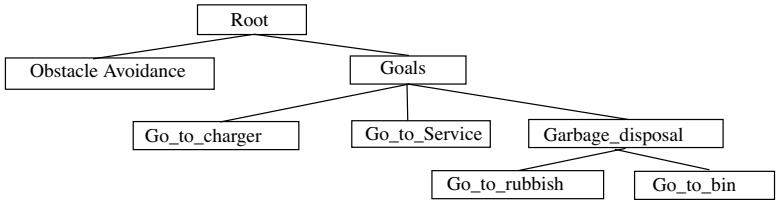

FIGURE 3. The fuzzy decision tree for the garbage collection task, GC.

3.3 Variable Declarations

The first step in expressing this in FDTL is to define each of the input and output variables. In our case we have many input variables, readings from the ultrasonic sensors, the minimum proximity reading, the relative angles and distances from the various goals, the state of the battery and damage detectors, and so on. Most of these are represented as fuzzy variables as described in Mamdani (1977). For example the FDTL definition for *d,* one of the distance sensor readings might be:

```
numeric d
 {
  trapezoidal VN 0   0    80  120;     // Very Near
  triangular  NE 80  120  160;         // Near
  trapezoidal FR 120 160  255 256;     // FAR
 }
```

(Characters between '//' and end of line are comments). Which could be represented graphically as

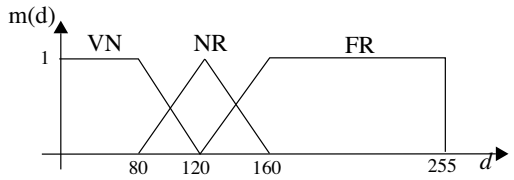

FIGURE 4. The fuzzy sets an input variable *d* representing the reading of an ultrasonic sensor.

A non-fuzzy numeric variable can be declared simply by including no fuzzy sets. Its value can then be used in the rulebases directly and will not be modified by the system.

The third possibility is to declare an input as "symbolic". These are analogous to the enumerated types of PASCAL (or the enums of C) and are intended for inputs that switch abruptly between a relatively small number of states. In the example the state of the "has_rubbish" detector could be expressed as:

```
symbolic hold_rub // carry rubbish
  {
   'yes';
   'no';
  }
```

3.4 Leaf Rulebase Declarations

The rules of leaf rulebases connect fuzzy input variables to fuzzy output variables. In FDTL a couple of rules that are part of a rulebase expressing obstacle avoidance behaviour might be:

```
ObstacleAvoidance
   begin
      if d1==VN  and d2==VN  and d3==VN  and d4==VN then Vleft:=NS and Vright:=NS;
      if d1==VN  and d2==NE  and d3==NE  and d4==VN then Vleft:=PS and Vright:=PS;
   end
```

The input variables d1 - d4 represent ultrasound readings pointing at -90°, -45°, 45°, and 90° with respect to the front of a vehicle. The fuzzy output variables Vleft and Vright represent the velocity of the two driving wheels. The first rule above is meant to say that if the vehicle finds obstacles “very near” on all sides it should back up “slowly” (Vleft and Vright “Negative Small”). The second rule says that if we have obstacles “very near” on either side but only “near” to the front then we should move forwards slowly. Of course, these being fuzzy variables the meaning of the first rule could perhaps be rephrased as “on this control cycle contribute ’backing up’ behaviour to the final effective behaviour to the extent to which d1 - d4 are reading ’very near’. Of course the second rule should be interpreted similarly

To make this clearer imagine that the vehicle was approaching a cul-de-sac at the end of a corridor. Initially d3 and d4 would be mostly in the “near” region and only just in the “very near” region. The first rule would only “fire” to a small degree. The negative velocities of the “backing up” behaviour would only have the effect of slightly reducing the positive velocities appropriate when the obstruction has just come into view. As the obstruction comes closer the vehicle will slow down and eventually come to a stop when the equilibrium point is reached. At no point would there be abrupt changes in velocity. The proportionality between inputs an outputs necessary for stability in the controller would be preserved. In a real rulebase the second rule would perhaps more usefully rotate the vehicle about its axis.

3.5 Building the Tree

So far all has been standard fuzzy control. However we now come to the internal nodes that will mediate in a similar way between entire rulebases. The representation is entirely uniform with what has gone before and should give the reader little trouble to understand. The FDTL for the overall GC rulebase is given below (omitted rules denoted by ’...’):

```
rules
   begin
      GC
         begin
            if dmin==NEAR  then ObstacleAvoidance;
            if dmin==FAR   then Goals;
               ObstacleAvoidance
                  begin
                     ...
                  end
               Goals
                  begin
                     if energy<=20 and damage=='yes' then go_to_charger;
                     if energy>20  and damage=='yes' then go_to_service;
                     ...
                     if energy>20  and damage=='no'  then do_jobs;
                        go_to_charger
                           begin
                              ...
                           end
                        go_to_service
                           begin
                              ...
                           end
                        do_jobs
                           begin
                              if hold_rub=='yes' then go_to_bin;
                              if hold_rub=='no'  then go_to_rub;
                                 go_to_bin
                                    begin
                                       ...
                                    end
                                 go_to_rub
                                    begin
                                       ...
                                    end
                              end // end of do jobs
                  end // end of Goals
         end // end of GC
   end; // end of rules
```

4. Results

Our test vehicles (VME based, VxWorks OS) are equipped with only 4 ultrasound range finders and a very simple infrared system that can find the bearing and identify a small set of beacons (Figure 5).

FIGURE 5. One of the test vehicles

Steering is by differential velocity of the two driving wheels. We also have a simulator for this vehicle. Simulated vehicles can also find the range and bearing of a number of targets and detect various internal "states" such as damage, possession of rubbish, and battery charge. We have implemented a compiler for the language whose output is ANSI standard C. Both the simulator and the real vehicles run executables compiled from identical sources.

We have implemented a controller that combines goal-seeking and obstacle avoidance behaviour both on the simulator and on a real vehicle. The behaviour was sufficiently similar to give us confidence in our simulator. In both environments the vehicle consistently moved smoothly to the goal only deviating from its path enough to clear the obstacles with a small margin of safety.

The vehicle was able to react quickly to avoid an experimenter suddenly standing in front of the vehicle. This in fact exceeded our expectations since the vehicle is equipped with only four simple ultrasonic range finders and we have put very little effort into making the implementation efficient.

Oscillations at corners and in cul-de-sacs were avoided by simply always turning to the right in such situations. However there are no inherent properties of the system that can ensure the absence of such oscillations since the model is essentially for a purely reactive controller and has no internal state although feedback is theoretically possible (See section 2.). However since we are interested in extending the FAM model to operate at higher levels in architectures for autonomous control addressing this issue is a high priority.

We have also implemented the "GC" controller described above on the simulator only. As can be seen from the simulator output in Figure 6, the path of the vehicle (which moved to the targets in the speci-

fied order) was smooth and took efficient trajectories to the targets while keeping a safe distance from the obstacles.

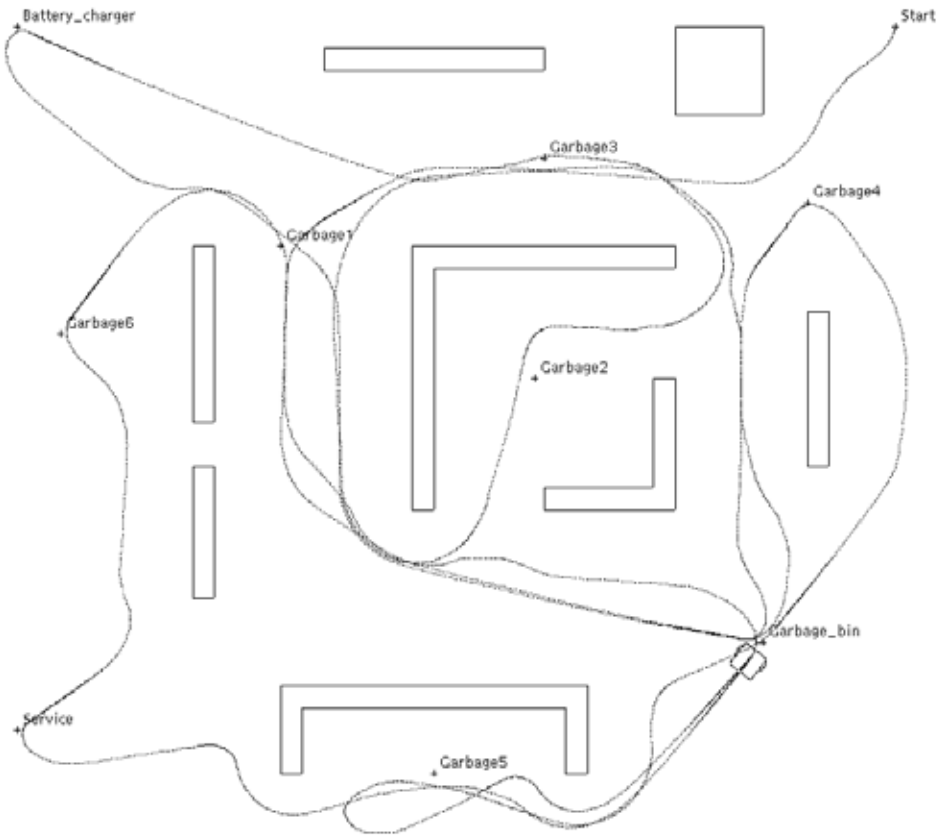

FIGURE 6. The simulator running the GC controller.

5. Conclusions

We believe that we have successfully demonstrated that Kosko's FAM model for fuzzy rule based controllers can be extended to form a basis for a practical system for programming medium to low level controllers for autonomous vehicles.

We make no claims for high efficiency in the running controller although the speed of execution of our initial implementation on the real vehicle was pleasing. We intend to explore the inherent parallelism of the model in a multi-processor implementation.

The most immediate thrust of our research will be to integrate FDTL sourced controllers into a complete architecture for autonomous vehicles. We see this as happening in two ways. Firstly we would like to introduce internal state to our model together with "actions" that can operate on that state. In this way perhaps a task like map-making could be expressed as a further "behaviour" of the system. The avenue that we are exploring is in an interface to higher level planners, possibly written in a language that is built on an entirely different model. We have already ported PROLOG to our vehicles for this purpose.

6. References

Baxter J.W. and Bumby J.R. (1993) Fuzzy Logic Guidance and Obstacle Avoidance Algorithms for Autonomous Vehicle Control. Proceedings of the 1st International Workshop on Intelligent Autonomous Vehicles, Southampton, April 1993, pp. 259-264

Beom H.R. and Cho H.S. (1992) A Sensor-based Obstacle Avoidance Controller for a Mobile Robot Using Fuzzy Logic and Neural Network. Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '92), Vol. 2, pp. 1470-1475.

Brooks R.A. (1986) A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, RA-1, March 1986, pp. 14-23.

Chang R.L.P. and Pavlidis T. (1977) Fuzzy Decision Tree Algorithms. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 7, No. 1, pp. 28-35.

Cox E. (1993) Adaptive Fuzzy Systems. IEEE Spectrum, February 1993, pp. 27-31.

Gachet D. et al. (1993) Neural Network Approaches for Behavioural Control of Autonomous Systems. Proceedings of the 1st International Workshop on Intelligent Autonomous Vehicles, Southampton, April 1993, pp. 330-334.

Joo Y. H. et al. (1993) Intelligent Navigation Control of An Autonomous Mobile Robot. Proceedings of the 1st International Workshop on Intelligent Autonomous Vehicles, Southampton, April 1993, pp. 259-264.

Klir G.J and Folger T.A (1988), Fuzzy Sets, Uncertainty and Information, Englewood Cliffs: Prentice-Hall

Kosko B. (1992a) Neural Networks and Fuzzy Systems: a Dynamical Systems Approach to Machine Intelligence. Englewood Cliffs: Prentice-Hall.

Kosko B. (1992b) Fuzzy Systems as Universal Approximators. Proceedings of the IEEE International Conference on Fuzzy Systems, San Diego, California, March 1992, pp.1153-1162.

Mamdani E.H. (1977) Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Systems. IEEE Transactions on Computers, Vol. 26, No. 12, pp. 1182-1191.

Pin F.G. et al. (1992) Using Custom-Designed VLSI Fuzzy Inferencing Chips for the Autonomous Navigation of a Mobile Robot. Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '92), Vol. 2, pp. 790-795.

Saffiotti A. et al. (1993) Blending Reactivity and Goal-Directedness in a Fuzzy Controller. Proceedings of the Second IEEE International Conference on Fuzzy Systems, San Francisco, California, March 28 - April 1, 1993, pp. 134-139.

Song K.T. and Tai J.C. (1992) Fuzzy Navigation of a Mobile Robot. Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '92), Vol. 1, pp. 621-627.

Sugeno et al. (1993) Fuzzy Hierarchical Control of An Unmanned Helicopter. Proceedings of the International Fuzzy Systems and Applications conference (IFSA'93), Korea.

Voudouris C. (1993) Fuzzy Hierarchical Control for Autonomous Mobile Robots. MSc dissertation, Department of Computer Science, University of Essex, 1993.

Zadeh L.A. (1965) Fuzzy Sets. Information and Control 8, pp. 338-353.