Fuzzy Steering for Autonomous MCU-based Mobile Robotics

ROBERT T. CASEY, MIKE HENSLER
Department of Mathematics and Physics
DigiPen Institute of Technology
5001 150th Ave. NE; Redmond, WA 98052
USA
rcasey@digipen.edu, mhensle2@digipen.edu

Abstract: Small-scale autonomous vehicle navigation may involve the use of classical or "crisp" sets for control. In this paper, we discuss the successful application of fuzzy sets to such a vehicle's steering module, which results in an improvement in the vehicle's navigational capabilities.

Key-Words: Fuzzy sets, microcontroller, RC car, autonomous navigation, robot.

1 Introduction

Small-scale research with autonomous vehicles begins with a hardware platform. In our experiment, a modified RC truck provides the mobile framework, while a PIC18F452 microcontroller (MCU) serves as the data, logic, and I/O processing base [1][2]. Lastly, infrared sensors serve as the vehicle's perceptual link to its environment [7].

The aforementioned specifications limit the scope of our experimentation to the Reactive Paradigm, which has been described as a behavioral, rather than representative, architecture for robotic system building [3]. Our chosen architecture minimizes system memory requirements by eliminating the robotic localization, environmental mapping, and planning phases often found in a representative architecture. No internal representations of the external world are maintained. Instead, a tight coupling of sensors to actuators provides the basic building blocks of robotic actions: reflex-like behavior. Thus goal-oriented behavior is minimized: the final vehicle exhibits simple environmental exploration and obstacle avoidance as its highest level of behavior.

Once the various idiosyncrasies of hardware have been adequately addressed, the key challenge remains: to communicate to the hardware the instructions to attain autonomous navigation in the manner previously described.

2 Problem Formulation

Intelligent use of sensor data greatly facilitates autonomous navigation. However, these input data span a significant continuous domain. One approach to managing this complexity involves partitioning the inputs into three discrete domains: *near*, *mid*, and *far*. Any input within one of these domains will map to a singleton - a discrete steering value in the output range. This is the classical or "crisp" set approach; each steering output value has a membership of 0 or 1 with respect to each subset of the input domain. In Table 1, the numeral **1** indicates a member of the output belongs to a subset of the input, while a **0** indicates non-membership [5].

This "filtering" of sensor input data results in a loss of information and control: a vehicle utilizing such logic has trouble negotiating tight corners or corridors and exhibits somewhat jerky steering.

Classical Characteristic Functions

	Steering		
	Hard	Soft	None
Range			
Near	1	0	0
Mid	0	1	0
Far	0	0	1

Table 1: A simple Boolean mapping of inputs (percepts) to outputs (actuators)

3 Problem Solution

Our approach may be seen as a simplified version, in terms of hardware and software, of recent work by Valavanis, et al, in an effort to demonstrate the effectiveness of fuzzy sets on a small-scale autonomous vehicle [4]. One subset of Valavanis' experiment involved a fuzzy logic controller mapping

sensor range data to rotational velocities - varying degrees of turning. Our approach does the same. The application of fuzzy sets to the vehicle's translational velocity (speed) extends beyond the scope of this project.

3.1 Project Goals

Using the linguistic expressiveness of fuzzy sets, we wish to achieve the following:

1. Preservation of the distinctness of sensor data.
2. Continuous or near-analog steering.
3. Improved navigational responsiveness.

3.2 Mathematical Preliminaries

Our approach involves creating a group of membership functions describing the strength of sensor readings as well as the degree of turn executed by the front wheels. The fuzzy sets described by these membership functions will facilitate translation of the following linguistic algorithm into a workable instruction set for our robotic truck:

*The **farther away** an object,*
*the **less** the truck will turn.*
*The **closer** an object,*
*the **more** the truck will turn.*

Consider the following group of membership functions described by three interleaved triangles:

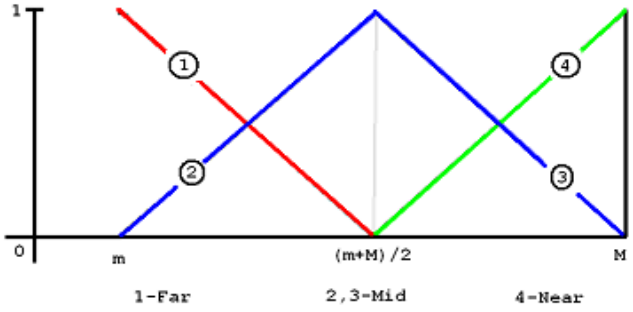

Fig.1 Geometric Representation of Four Simple Membership Functions for Range Input Data

with degrees of membership determined by the following analytic forms:

$$A : X \rightarrow [0, 1]$$

m represents the minimum value for sensor input, while **M** represents the maximum value.

For the domain:

$$\{ x \mid m \le x \le \frac{(m+M)}{2},\ x \in \mathbb{Z} \}$$

$$A_1(x) = \frac{(M+m-2x)}{(M-m)} \qquad (1)$$

$$A_2(x) = \frac{2(x-m)}{(M-m)} \qquad (2)$$

For the domain :

$$\{ x \mid \frac{(m+M)}{2} \le x \le M,\ x \in \mathbb{Z} \}$$

$$A_3(x) = \frac{2(M-x)}{(M-m)} \qquad (3)$$

$$A_4(x) = \frac{(2x-(M+m))}{(M-m)} \qquad (4)$$

In our fuzzy relation, we decided to simplify the translation by using a one-to-one mapping of the input membership functions to those of the output. Consider a few sample degrees of membership:

Fuzzy Membership Functions

	Steering		
	Hard	Soft	None
Range			
Near	0.75	0.25	0.00
Mid	0.10	0.80	0.10
Far	0.00	0.25	0.75

Table 2: Sample Degrees of Membership Relating Range Input Data to Steering Output Data

Compared with Table 1, Table 2 clearly provides more information and, as we demonstrate, accuracy.

3.3 Translation Process

A high-level view of the translation process marks a transformation of our sensor input values into unique or nearly unique steering output values. This process is as follows:

1. Instruct the PIC18F452 microcontroller's Analog-to-Digital converter to poll the sensor; this integral sensor reading represents our crisp input [6].
2. According to our chosen membership functions, fuzzify the crisp input to yield a few floating point degrees of membership; these represent our fuzzy input.
3. Using our fuzzy relation (one-to-one in this experiment), map the fuzzy input to the

fuzzy output, which will take the form of other, possibly similar, floating point degrees of membership.

4. Reverse step 2 by defuzzifying these degrees of membership for the fuzzy output to obtain a discrete integer value. This value represents our crisp output and the process is complete.

3.3.1 A Sample Translation

Let us assume that the CPU has polled the front left and right sensors and obtained values of 37 and 71, respectively. Our driving logic places higher priority upon the lateral sensor with the greater value, as increasing sensor values, in general, correspond to decreasing separation between vehicle and object. Thus we will proceed with the data provided by the right sensor. Hardware calibrations indicate the universe of discourse for our sensor values:

$$m_x = 15,\ M_x = 143$$
$$\{x \mid m_x \le x \le M_x,\ x \in \mathbb{Z}\} \qquad (5)$$

as well as that for our right turning values:

$$m_y = 92,\ M_y = 152$$
$$\{y \mid m_y \le y \le M_y,\ y \in \mathbb{Z}\} \qquad (6)$$

As in (1)-(4), $\mathbf{m_{x,y}}$ represents the minimum value and $\mathbf{M_{x,y}}$ represents the maximum value. Notice that the domain of x, our discrete sensor input value, prescribes the membership functions described by (1) and (2) , since

$$15 \le 71 \le 79$$

Setting up our data as follows:

$$\text{From (1):}\quad A_1(71) = \frac{(143 + 15 - 2\,(71))}{(143 - 15)}$$

$$\text{From (2):}\quad A_2(71) = \frac{2\,(71 - 15)}{(143 - 15)}$$

yields our fuzzy input:

From (1): $A_1(71) = 0.125$ (Far)
From (2): $A_2(71) = 0.875$ (Mid)

Our one-to-one fuzzy relation maps these degrees of membership (fuzzy input) to the same for the output functions (fuzzy output). We proceed with the defuzzification by setting A(x) = A(y) for either (1) or (2), due to symmetry (notice that the sum of the results from (1) and (2) will always be 1.0). Choosing (2):

$$0.875 = \frac{2\,(y - 92)}{(152 - 92)}$$

Solving for y and rounding to the nearest integer, we obtain y = 118, which represents a steering output value of turning soft right with a slight inclination towards the center, roughly 20 degrees right of center.

Fig. 2 illustrates the relation between the fuzzy sets for the percepts and those for the actuators, using data from the previous example.

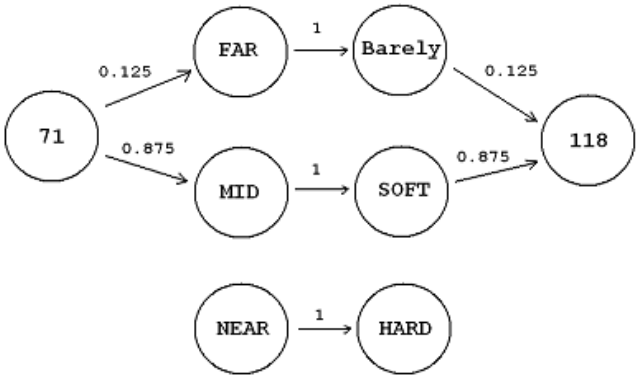

Fig. 2 Mapping Diagram Relating Sample Range Input to Steering Output

3.4 Engineering Considerations

Given this theoretical foundation, let us now address a few implementation details concerning hardware limitations.

3.4.1 Validity and Integrity of Data

Since one of our goals is to preserve the distinctness of sensor data, measures should be taken to insure the **accuracy** and **precision** of such data before passing it to our discerning membership functions. A challenge arises in that the electrical noise from multiple sensors grouped within a small radius occasionally leads to interference and inaccurate readings. We want to maximize the quality of these readings with minimal computational overhead.

In the end, we decided to take the arithmetic mean of two readings, which yielded an acceptable solution. While averaging more data points might superficially appear to provide even better sensor calibration, there are drawbacks:

1. The sensors have a listed range of 150 cm [7]. Taking more readings yields smaller improvements to the overall accuracy.
2. The high native torque of the vehicle required a cycling of the Pulse-Width-Modulated speed control. Management of this speed control involved cycle-sensitive counters and activation toggles. Taking more sensor readings per system loop meant taking

more time away from the speed management loop, resulting in a slower net speed. Additionally, too much time out of the speed management loop results in abrupt acceleration. Neither of these behaviors are acceptable.

3. While the sensors are taking readings, the vehicle's position is changing, thus invalidating the old data points.

After averaging the two sensor readings, the processor performed a final measure of quality control. If the reading was greater than the predefined maximum in domain (5), the data point's value was set equal to that maximum. If the reading was less than the predefined minimum in domain (5), the value was set equal to that minimum. The reasoning behind these adjustments will soon be clear.

3.4.2 Run-time Considerations

At the time of project implementation, no floating point library was readily available, making degree of membership calculations difficult. Even with such a library, the limited operational frequency, 1 MHz, of the 8-bit RISC CPU imposed significant limitations on the design of our control program written in the Microchip PIC18 assembly language [6]. The computationally expensive nature of performing fuzzification and defuzzification for *every* averaged sensor reading on such a platform could cripple the vehicle's navigation. Calculations for membership functions described by overlapping triangles might be feasible, but attempting to do the same with logistic curves or normal distributions would likely exceed the run-time capabilities of the MCU.

3.4.3 The Lookup Table Solution

Lookup tables proved to be an effective solution to the run-time calculation challenge. All possible input-to-output mappings are precomputed using a helper application (Section 3.5). Once the desired membership functions and strengths of association have been entered, the program generates a data file of tabular values for the translated steering output. These data files are loaded into the CPU's data memory for run-time access. As the vehicle obtains averaged sensor readings at run-time, a quick subtraction is performed using the minimum sensor threshold to generate an index into the data table of steering values. Since this table is simply an array in data memory, one may now see the logic behind doing bounds-checking on sensor input data before using it as a table index. Attempting to read data outside the array results in undefined behavior.

Given safeguarded indices, lookup tables provide an efficient means of handling the translation from fuzzification to defuzzification, with constant *O(k)* complexity. Whether the tables contain mappings generated by membership functions described by triangles, trapezoids, bell curves, or sigmoid curves, access time remains constant.

The use of lookup tables also promotes code modularity. To test the effect of other membership functions, no core code changes need be made. Simply regenerating the data files using the *FuzzyGUI* and including these new data files within the project will immediately provide new functionality

The final benefit of lookup table use is the minimal run-time overhead incurred.

3.5 The FuzzyGUI

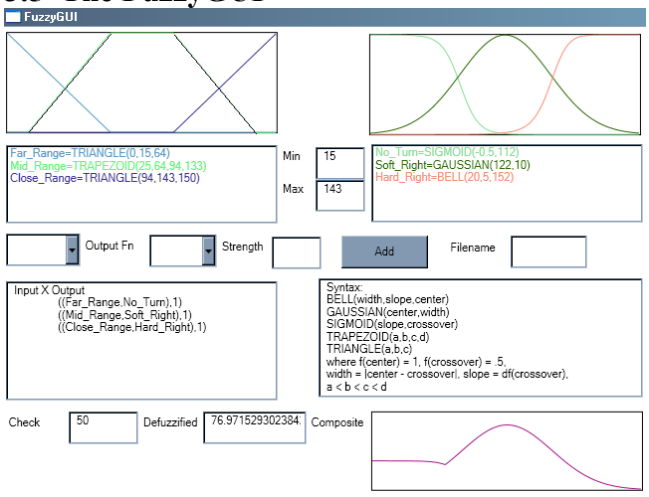

Fig. 3 Screenshot of the Fuzzy GUI
Upper Left: Input Membership Functions
Upper Right: Output Membership Functions
Lower Right: A Check on the Composite, Defuzzified Output

Written in C#, the FuzzyGUI (Fig. 3) utilizes the cutting edge Microsoft Visual Studio .NET 2005 IDE with .NET 3.0 Framework CTP and Extreme Optimization's Mathematics Library [8][9]. Created as a helper tool for this project, the FuzzyGUI provides a graphical display of membership functions to allow for a more concrete grasp as to their meaning. After we defined the valid input and output ranges, the application enabled us to define fuzzy sets using

triangular, trapezoidal, sigmoidal, bell, and Gaussian membership functions (as defined by [10]) over both domains. With the membership functions in place, we could then create fuzzy relationships from the input sets into the output sets. Making use of the built-in checking functionality, we previewed the behavior of the mapping by entering test points, at which point the FuzzyGUI would show us the composite membership function generated via relations into the output domain as well as the defuzzified output singleton using the centroid of area method. As soon as we felt satisfied with our choice of fuzzy sets and relations, we entered a filename and the FuzzyGUI automatically generated a file containing the precalculated lookup table. This greatly simplified the task of creating and testing the effects of various membership functions.

3.6 Findings

The fuzzy relations between sensor input data and steering output data fulfilled our first project goal (Sect 3.1) of the preservation of distinctness of sensor data. This happened at compile-time. At run-time, compared to the algorithm defined by crisp sets, the fuzzy algorithm enabled much smoother, near-analog, steering for the robotic vehicle, thus marking the achievement of our second project goal. A few hardware issues involving the steering servos pulled the vehicle slightly to the left, but these issues were invariant under software. Several stress tests were executed in which the vehicle was sent down a tight corridor of obstacles (Fig. 4). The fuzzy-controlled

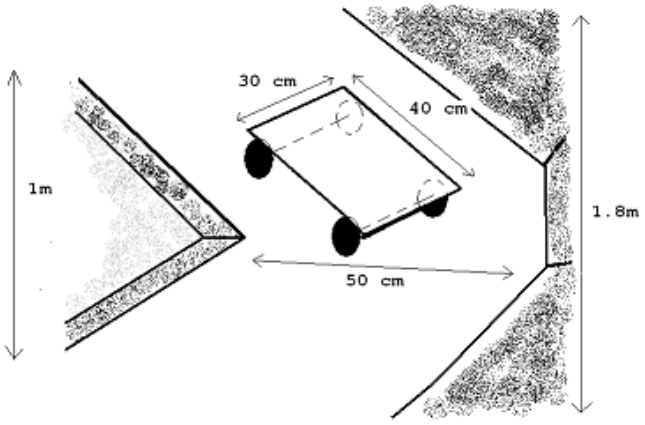

Fig. 4 Diagram of the Track: Stress Test

module yielded more successful passes down the corridor than the classical set-driven module. In many instances, the "crisp" version of the vehicle would get snagged on the ninety-degree corner due to excessively sharp turning, unlike the vehicle using fuzzy sets. This navigational improvement completed our third and final project goal.

What was curious about the results of the experiment was the perceived "confidence level" of the vehicle. Anthropomorphically speaking, the truck seemed more "skittish" with crisp sets, veering abruptly away from obstacles. With fuzzy sets, however, the truck seemed to approach these obstacles aggressively, then smoothly follow their outline.

4 Conclusion

The success of our modest goals in these simple experiments has illustrated that, even with the limited computational resources provided by an 8-bit microcontroller, the concepts behind fuzzy sets offer a simple yet effective way to improve significantly the quality of control over robotic vehicle navigation.

Our triangular membership functions highlight only the tip of the fuzzy set iceberg. Much could be done to improve the quality of the membership functions used for the purposes described in this paper. First, experiments could examine the effects of logistic curves and normal distributions. Next, mappings between sensors and actuators might more tightly couple input memberships to the specifications listed on the sensor datasheet, yielding a better correlation between A/D voltage output values and physical distance to an object. Finally, the techniques of knowledge acquisition (KA) or statistical modeling might help build membership functions yielding improved navigational responsiveness [5].

Dedicated hardware performing the fuzzy translations might ease the computational burdens on the microcontroller and provide a fruitful field of experimentation in computer engineering.

Acknowledgments

The authors would like to thank Dr. Michael Aristidou for his mentoring during this project as well as Dr. Charles Duba for his technical contributions.

References:

[1] http://www.traxxas.com/products/electric/rustler2006/trx_rustler_intro.htm

[2] http://www.microchip.com

[3] R.R. Murphy, *Introduction to AI Robotics*, The MIT Press, 2000.

[4] K. Valavanis, L. Doitsidis, M. Long, R.R. Murphy, A Case Study of Fuzzy-Logic-Based Robot Navigation, *IEEE Robotics & Automation Magazine*, Vol. 13, No. 3, 2006, pp. 93-107.

[5] G. Klir, B. Yuan, *Fuzzy Sets and Logic: Theory and Applications*, Prentice Hall PTR, 1995.

[6] H. Huang, *PIC Microcontroller: An Introduction to Software and Hardware Interfacing*, Delmar Learning, 2005.

[7] GP2Y0A02YK, Sharp Electronics Corp., web.mit.edu/6.270/www/contestants/handouts/acc_ir_dist_001.pdf , 4/05/2006.

[8] http://msdn.microsoft.com/vstudio/

[9] http://www.extremeoptimization.com

[10] J.-S.R. Jang, C.-T.Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Language*, Prentice Hall, 1997.