A Fuzzy Logic Controller for Car-like Mobile Robots

Jacky Baltes
j.baltes@auckland.ac.nz

Robin Otte
rott001@student.auckland.ac.nz

Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand

Abstract

This paper describes a fuzzy logic controller for car-like mobile robots. It also introduces a simple heuristic that helps a designer in the specification of fuzzy input and output sets. The design of fuzzy rules follows intuitively from the design of the fuzzy input sets. In practical tests, this Fuzzy Logic controller resulted in greatly reduced errors and also resulted in a control law with 75% less control work than a traditional sliding mode controller.

1 Introduction

This paper describes the design of a fuzzy logic controller for car-like mobile robots. The intended application domain was the Aucklandianapolis robot competition at the University of Auckland [2].

The Aucklandianapolis race is a time trial around a letter A shaped race track (see Fig 7). The robots used in this domain are remote controlled off the shelf toy cars with proportional control for speed and direction. We developed a micro-controller based parallel port interface for the transmitter, which allows us to control the cars from a computer. The interface provides 64 different steering angles between -28 to 28 degrees and 64 different speed settings. However, since the toy car uses cheap D/A and A/D converters, both speed and direction control is very coarse and not all settings result in different actuator output.

Position and orientation feedback for the car is provided through a video camera mounted on the ceiling. Pictures are sampled at 50 fields/sec and the image processing is simplified through colored dots on the roof of the car. The error in the position is about 3cm and the error in the orientation is about 10 degrees. This setup is used for a variety of different tasks including RoboCup ([5]) and parallel parking.

Aucklandianapolis requires students to implement a path planner for car-like mobile robots ([3]). This path planner creates a path for the car to follow around the race track. The path created by most non-holonomic path planners consists of straight lines and full turns to the left or right. In other words, the path planner will never generate a plan that contains anything but a full turn. This motivated by a result from Reeds and Shepp which proves that the shortest length path for a non-holonomic vehicle contains only straight lines or full turns.

Given a path from the path planner, the toy cars must follow the path around the race track. Unfortunately, the lack of hardware support (e.g., shaft encoders) means that the cars will veer severely off track if not controlled properly.

Therefore, the path is handed to a controller whose job it is to keep the car on track. Firstly, we used a state of the art controller developed by Balluchi [1]. The controller is based on a sliding mode analysis and results in a bang bang controller, that is a controller that switches between two extreme control outputs. In the case of cars, the controller switches between full left and right turns. This behavior is called chattering and a smoothing function was applied to ameliorate this problem. Although the controller performs very well in simulations, it performs poorly in real world experiments. The fuzzy logic controller was developed to overcome some of the problems of the original controller, in particular the high control work.

Section 2 describes the kinematic model and the model used in the design of the controller. The design of the fuzzy controller is described in section 3. In particular the design of the fuzzy input and output sets. The fuzzy rule base is shown in section 4. Sec-

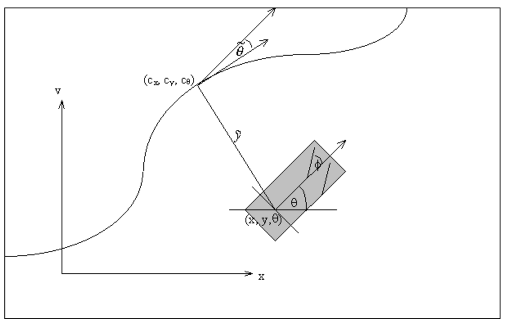

Figure 1: Kinematic Model of a Car

tion 5 shows the results of comparing the performance of the fuzzy logic controller and Balluchi's controller. The paper concludes with section 6.

2 Kinematic Model of the Car

This section describes the kinematic model of the rear wheel drive car as used in this paper (see Fig 1). The coordinates of the rear axle of the car are given by x and y. The orientation of the car is given as θ.

Coordinates c_x, c_y are the coordinates of the closest point to the car on the path. The slope of the path at point (c_x, c_y) is given by c_θ.

The control function is defined relative to the positional error ($\tilde{y}$), the orientation error ($\tilde{\theta}$), and the curvature of the path (R).

The positional error ($\tilde{y}$) is the distance between points (x, y) and (c_x, c_y). The orientation error is defined as $\tilde{\theta} = \theta - c_\theta$.

It is assumed that the path itself and its first derivative are continuous. However, the curvature of the path may be discontinuous.

3 Design of the Fuzzy Logic Controller

This section describes the design of our Fuzzy Logic controller for car-like mobile robots. It also discusses the heuristic that we used to determine the number of necessary fuzzy input and output sets.

3.1 Fuzzy Input Sets

In Fuzzy Logic, the input to the controller (curvature R, position error $\tilde{y}$, and orientation error ($\tilde{\theta}$) are converted into a series of Fuzzy Sets. The number and exact shape of these fuzzy sets critically determine the

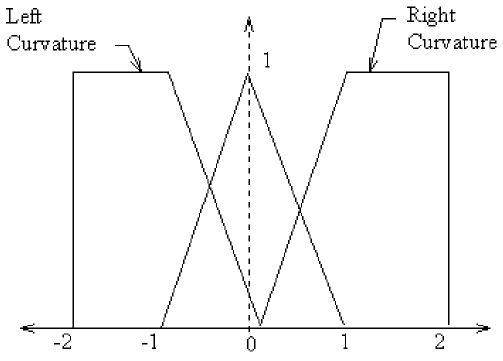

Figure 2: Fuzzy Input Set: Curvature

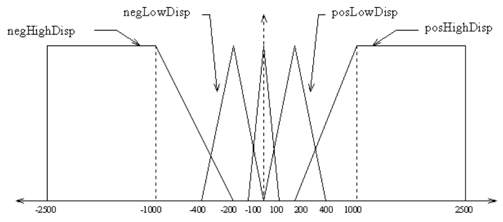

Figure 3: Fuzzy Input Set: Position Error

performance of the controller. In our work, we used the following heuristic:

> These fuzzy sets describe *qualitative* situations, that is situations in which the output of the controller is *qualitatively* different.

In other words, whenever the desired behavior (e.g., change from going straight to turning left or change from fast to medium speed) of the controller changes in an input situation, a fuzzy set is created to represent this case.

Figure 2 shows the fuzzification of the curvature set. The fuzzy set is normalized with respect to the minimum turn radius of the car, that is +/- 1 is a maximum turn to the right//left respectively. The limits of the fuzzy set allows path with circles that have a radius up to $\frac{1}{2}$ of the maximum turn radius.

The fuzzification of the positional error $\tilde{y}$ is shown in fig. 3. There are a total of five different fuzzy sets. Clearly it is desirable a set is needed for the car being on the line (zeroDisp). Assume that the car is far away from the path, then the desired behavior is to turn either right/left towards the path, drive straight towards the path, and the turn left/right to straighten out. Therefore, we require two extra sets on each side of the path.

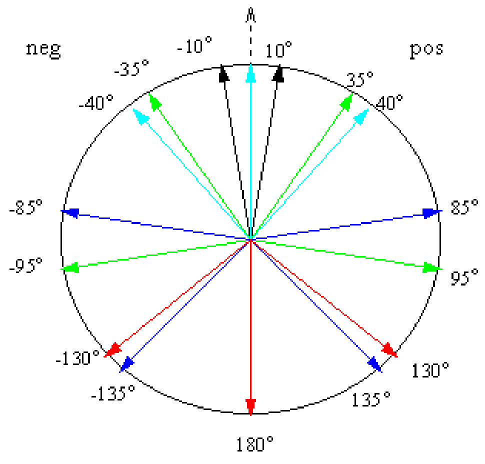

Figure 4: Fuzzy Input Set: Orientation Error

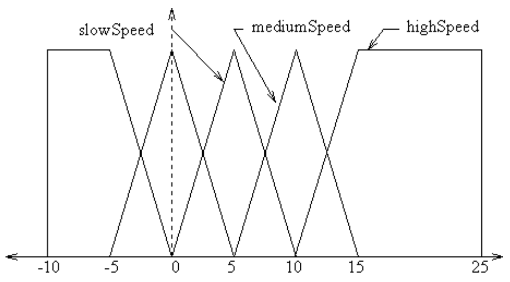

Figure 5: Fuzzy Output Set: Speed

Similar reasoning leads to the design of the fuzzy sets for the orientation error shown in fig. 4. A total of nine sets are needed to describe all different cases. The main reason is that if the orientation error is less than 90 degrees, a turn to the left is appropriate. If the error is more than 90 degrees, turning to the right leads to faster recovery.

3.2 Fuzzy Output Sets

There are two outputs of the current controller: (a) speed and (b) direction. The current hardware supports 64 settings for the speed and 64 for directions, of which only five were used. Figures 5 and 6 show the fuzzy sets for speed and direction respectively.

A fuzzy controller uses rules over fuzzy sets to compute a fuzzy set for the desired output. A crisp output value is then computed from this fuzzy set. This step is called de-fuzzification. In this research, we used the well known centroid de-fuzzification method which uses the centre of gravity as the crisp output value.

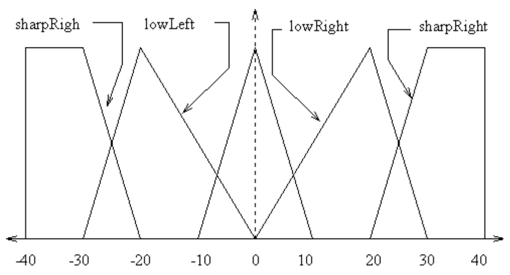

Figure 6: Fuzzy Output Set: Steering

4 Fuzzy Rule Base

Given these fuzzy input sets, a fuzzy controller uses a set of fuzzy rules to specify the desired control behavior. After the design of the fuzzy input and output sets, the design of the fuzzy rules is straight forward. There are a total of $9 * 5 * 3 = 135$ possible different input configurations. For each of these input configurations, a rule was specified to indicate the desired speed and directional settings.

Of course, the number of necessary rules could be greatly reduced if the symmetry of the situations is taken into account. For example, if the car follows a straight line, then any control decision that is made when the car is to the right of the line (positive displacement error) with a small orientation error is identical to the symmetric control decision (switching right with left and vice versa) when to the left of the line (negative displacement error). There are also symmetric control decisions when following a circle to the left or to the right. For example, the case where the car is to the left of a left circle (negative displacement) and facing the circle (positive orientation error) facing it is symmetric to the case of being to the right of a right circle (positive displacement) and facing it (negative orientation error).

The number of rules could further be reduced if non-relevant inputs are ignored. For example, when having an orientation error of less than +90 degrees and following a straight line, the control decision is always to turn full right independent of exactly how far the car is away from the line. The current implementation of the FL controller would not benefit greatly from a reduced rule set, so this option was not pursued further in this research.

The following paragraphs give some examples of the rules that were used by the fuzzy controller. The antecedent of each rule consists of a possible value for each fuzzy input variable (Curvature, Displacement, Orientation error) and the conclusions specifies the desired speed and direction settings for this configuration.

straightLine & zeroDisp & zeroAngle
⇒ **fast & straight** (Rule 1)
In this case, the car is following a straight line and is close to the line and in the correct orientation, which is an almost ideal situation. The controller makes the car go straight ahead at full speed.

straightLine & zeroDisp & posLowAngle
⇒ **medium speed & lowLeft** (Rule 2)
Here the car is following a straight line and close to the line, but is slightly turned to the right. In this case, the controller turns only gently to the left and proceeds at medium speed.

straightLine & zeroDisp & posHighAngle
⇒ **medium speed & sharpLeft** (Rule 3)
This rule is similar to rule two, but here the car is facing 90 degrees away from the line. In this case, the controller steers sharp left and uses medium Speed.

straightLine & posLowDisp & negHighAngle
⇒ **medium speed & sharpRight** (Rule 4)
In this case, the car is off to the right of a straight line and is facing towards the line. Then the controller turns sharp to the right, to straighten out onto the line.

straightLine & posHighDisp & negHighAngle
⇒ **medium speed & straight** (Rule 5)
When being far off to the right of a straight line and facing the line, then go straight ahead at medium speed to approach the line. Notice that as the car approaches the line, rule 4 will become more and more applicable, which will gently turn the car to the right.

leftCircle & zeroDisp & zeroAngle
⇒ **medium speed & sharpLeft** (Rule 6)
This rule is applicable if the car is on a circle to the left. In this case, if the car is facing in the right direction, then the controller turns full to the left, since all circles are at full turning radius.

leftCircle & posLowDisp & negLowAngle
⇒ **medium speed & lowLeft** (Rule 7)
The controller steers slightly to the left at medium speed when just outside of a left circle and slightly facing towards the circle. The curvature of the circle should catch up with car.

leftCircle & negLowDisp & zeroAngle
⇒ **medium speed & lowLeft** (Rule 8)
If the car is slightly inside of a left circle and is facing in the correct direction, the car drives only gently to the left to approach the maximum turn radius circle. This is the most important aspect in the design of the controller. Balluchi's controller would turn full to the right in this situation to approach the circle as quickly as possible. However, since there is a notable delay in turning from full left to full right, this means that the car will cross the circle with a 90 degree orientation error. This larger orientation error is very difficult to recover from.

The examples above show that the design of the rules is straight forward. The best speed and direction decision for the individual cases is obvious in all but the most severe error cases.

The power of the fuzzy controller arises from the fact that the individual rules are activated in parallel. For configurations that fall between rules, the control decision is based on a mixture of the control decision for the individual rules as is shown in the example of rule 4 and 5 above.

5 Evaluation

The performance of the FL controller was tested both in simulation as well as in the real world. The performance of the controller was compared to a traditional sliding mode controller developed by Balluchi. This bang-bang controller is prone to chattering, so a smoothing function was applied to the control output.

The choice of the smoothing function critically determines the practical performance of the controller. We tried a number different functions. The one that worked best and the one that was used in the evaluation would prevent sudden control changes when the car was close to the path (small positional and orientation error).

The Aucklandianapolis race track was used as the test path. In both evaluations (Simulation and real world), the actual track of the vehicle and the positional and orientation errors were logged as well as the change in control output. The later is an indication of how much work the controller must do to complete the task.

Although Balluchi's controller performed very well in the simulation, it performed poorly when driving a real car. The main problem is that a sudden change from a full left turn to a full right turn is impossible in reality. The fastest lap time that could be achieved with Balluchi's controller was 1 min. 30 sec. Even this rather slow performance was achieved only after careful tuning of the parameters for the actual race.

Figure 7 shows the track of the car. Further analysis showed that the biggest problem of the controller are: (a) over-steering along a straight line and (b) turning too far away from a circle. In case (a), the delay in the actual controller results in the car having

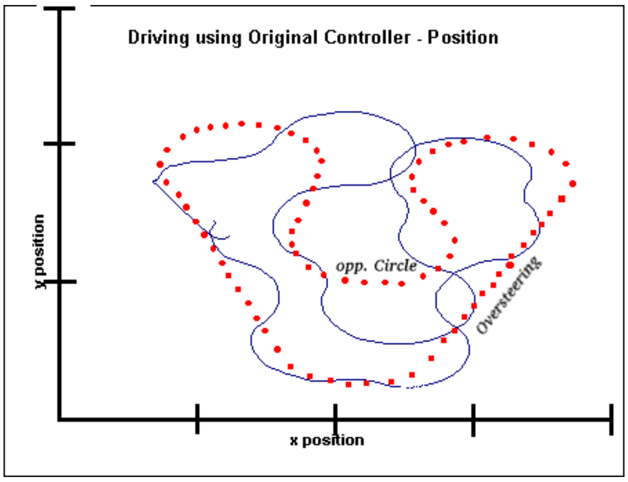

Figure 7: Track of Balluchi's controller. The dotted line is the race track. The track clearly shows two problems: (a) over-steering on the straight, and (b) turning too sharp away from a circle.

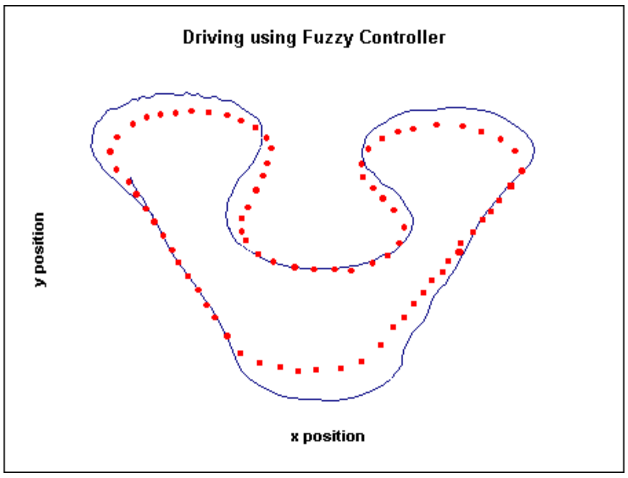

Figure 8: Track of the Fuzzy Logic controller. Dotted line is the race track.

to make large corrections since the turning angle is maintained well until the car crosses the line. In case (b), there are three cases where the controller makes a full right/left turn in a maximum left/right turn since the car was slightly inside the desired track. Consequently, the circular track is crossed at almost a 90 degree angle from it is very difficult to recover.

Figure 8 shows the resulting track when the same track is driven with the FL controller. The FL controller achieved a lap time of 0 min 40 secs, which is more than twice as fast as Balluchi's controller. Furthermore, even at this higher speed, the FL controller follow the path much more closely than Balluchi's controller. It gently approaches the line and then follows the path from there. Also the path of the car is much smoother throughout the whole lap.

Figure 9 compares the positional error of Balluchi's original controller and the FL controller. As can be seen, the FL controller has not only less error on average, but it also has a much smaller maximal error (Balluchi's controller 64cm, FL controller 22cm), which means that it always stays within one car width of the desired track.

An analysis of the orientation errors (see Fig. 10) during the tests and the total amount of control work done shows similar results, but is omitted here to save space. The FL controller has smaller average and maximum orientation errors. Also the total control work for the FL controller is only 22 as opposed to 88 for Balluchi's controller.

6 Conclusion

This paper describes the design of a fuzzy logic controller for car-like mobile robots and shows that it performs significantly better than a state of the art controller based on traditional control theory. Furthermore, it introduces a heuristic that can guide a designer in the design of the fuzzy input and output sets. The design of these fuzzy sets is critically important to the success of the Fl controller.

As shown in section 4, the design of the fuzzy rules follows intuitively from the design of the fuzzy input and output sets. A case by case analysis is used to determine the control output for the individual configurations defined by the fuzzy input sets.

Currently, we are continuing work on improving the FL controller and comparing it to more recent control designs that use look ahead [4]. We are using the Aucklandianapolis as a stepping stone towards our goal of building a soccer team for the RoboCup. In the soccer domain, it is doubtful whether look ahead will be useful, since paths change very quickly due to the highly dynamic nature of the domain.

References

[1] A. Balluchi, A. Bicchi, A. Balestrino, and G. Casalino. Path tracking control for dubin's cars. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.

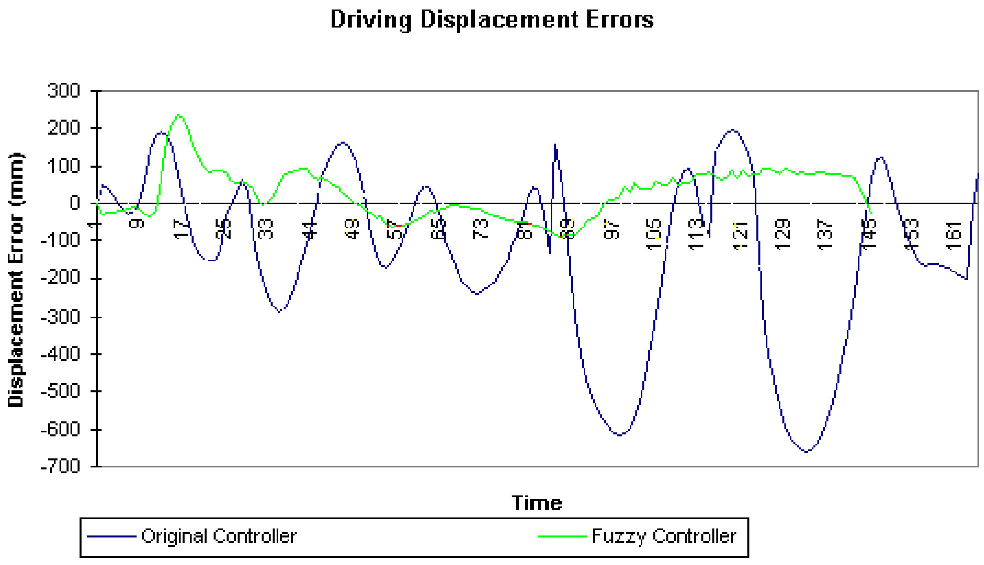

Figure 9: Distance error in centimeters for the Aucklandianapolis

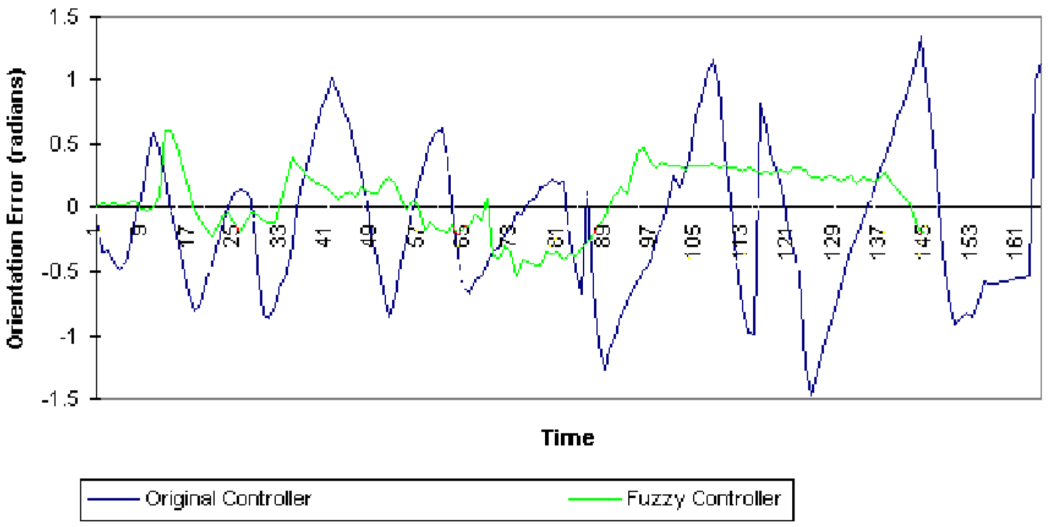

Figure 10: Orientation error in degrees for the Aucklandinanapolis

[2] Jacky Baltes. Aucklandianapolis homepage. WWW, February 1998. http://www.tcs.auckland.ac.nz/~jacky/teaching/-courses/415.703/aucklandianapolis/index.html.

[3] Antonio Bicchi, Giuseppe Casalino, and Corrado Santilli. Planning shortest bounded-curvature paths for a class of nonholomic vehicles among obstacles. *Journal of Intelligent and Robotic Systems*, 16:387–405, 1996.

[4] Markus Egerstedt, X. Hu, and A Stotsky. Control of a car-like robot using a dynamic model. In *Proceedings of the 1998 IEEE Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[5] Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I.* Springer Verlag, 1998.