Navigational fuzzy logic control of an autonomous vehicle

Neil Eugene Hodge

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/879
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700  800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
NOTE TO USERS

The original manuscript received by UMI contains pages with indistinct print. Pages were microfilmed as received.

This reproduction is the best copy available

UMI
NAVIGATIONAL FUZZY LOGIC CONTROL
OF AN AUTONOMOUS VEHICLE

by

Neil Eugene Hodge
Bachelor of Science
University of Nevada, Las Vegas
1995

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Department of Mechanical Engineering
University of Nevada, Las Vegas
May 1998
The Thesis prepared by

Neil Eugene Hodge

Entitled

Navigational Fuzzy Logic Control of an Autonomous Vehicle

is approved in partial fulfillment of the requirements for the degree of

Master of Science

Examination Committee Chair

Dean of the Graduate College

Graduate College Faculty Representative
ABSTRACT

Navigational Fuzzy Logic Control of an Autonomous Vehicle

by

Neil Eugene Hodge

Dr. Mohamed Trabia. Examination Committee Chair
Professor of Mechanical Engineering
University of Nevada, Las Vegas

The control of an autonomous passenger vehicle to reach a target amid static and dynamic obstacles is presented. To accommodate various model and environmental uncertainties, fuzzy logic is used in designing the vehicle’s controller. The controller evaluates sensor information and outputs signals to change the vehicle’s steering and throttle angles. To emulate human behavior, the controller is divided into separate modules. Each module deals with a specific navigational problem such as target steering, target throttle control, cornering throttle control, collision avoidance steering, and collision avoidance throttle control. Several simulation examples are included.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCCION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2 VEHICLE MODEL</td>
<td>8</td>
</tr>
<tr>
<td>CHAPTER 3 FUZZY LOGIC</td>
<td>15</td>
</tr>
<tr>
<td>CHAPTER 4 STEERING FUZZY CONTROL</td>
<td>24</td>
</tr>
<tr>
<td>Target Steering Fuzzy Module</td>
<td>28</td>
</tr>
<tr>
<td>Target Steering Fuzzy Rules</td>
<td>29</td>
</tr>
<tr>
<td>Target Steering Fuzzy Membership Sets</td>
<td>31</td>
</tr>
<tr>
<td>Final Orientation Control</td>
<td>35</td>
</tr>
<tr>
<td>Collision Avoidance Steering Fuzzy Module</td>
<td>38</td>
</tr>
<tr>
<td>Collision Avoidance Steering Fuzzy Rules</td>
<td>40</td>
</tr>
<tr>
<td>Collision Avoidance Steering Fuzzy Membership Sets</td>
<td>41</td>
</tr>
<tr>
<td>Bug Steering Fuzzy Module</td>
<td>44</td>
</tr>
<tr>
<td>Background</td>
<td>44</td>
</tr>
<tr>
<td>Description of Lumelsky's Bug2</td>
<td>46</td>
</tr>
<tr>
<td>Description of Sauerberger and Trabia's Modified Bug2</td>
<td>48</td>
</tr>
<tr>
<td>Description of Proposed Bug Fuzzy Module</td>
<td>51</td>
</tr>
<tr>
<td>Bug Steering Fuzzy Rules</td>
<td>55</td>
</tr>
<tr>
<td>Bug Steering Fuzzy Membership Sets</td>
<td>56</td>
</tr>
<tr>
<td>CHAPTER 5 THROTTLE FUZZY CONTROL</td>
<td>60</td>
</tr>
<tr>
<td>Target Throttle Fuzzy Module</td>
<td>60</td>
</tr>
<tr>
<td>Target Throttle Fuzzy Rules</td>
<td>60</td>
</tr>
<tr>
<td>Target Throttle Fuzzy Membership Sets</td>
<td>63</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cornering Throttle Fuzzy Module</td>
<td>68</td>
</tr>
<tr>
<td>Cornering Throttle Fuzzy Rules</td>
<td>68</td>
</tr>
<tr>
<td>Cornering Throttle Fuzzy Membership Sets</td>
<td>70</td>
</tr>
<tr>
<td>Collision Avoidance Throttle Fuzzy Module</td>
<td>73</td>
</tr>
<tr>
<td>Collision Avoidance Throttle Fuzzy Rules</td>
<td>73</td>
</tr>
<tr>
<td>Collision Avoidance Throttle Fuzzy Membership Sets</td>
<td>75</td>
</tr>
<tr>
<td>CHAPTER 6 SIMULATION EXAMPLES</td>
<td>80</td>
</tr>
<tr>
<td>Primary versus Bug Steering Control</td>
<td>80</td>
</tr>
<tr>
<td>General Example of Full Algorithm</td>
<td>85</td>
</tr>
<tr>
<td>Maze Tracking Case using Bug Steering Module</td>
<td>90</td>
</tr>
<tr>
<td>CHAPTER 7 CONCLUSIONS AND DISCUSSION</td>
<td>94</td>
</tr>
<tr>
<td>APPENDIX I DERIVATION OF EQUATIONS OF MOTION</td>
<td>98</td>
</tr>
<tr>
<td>APPENDIX II DERIVATION OF TRANSFORMATION EQUATIONS</td>
<td>102</td>
</tr>
<tr>
<td>APPENDIX III SCHEMATIC OF SIMULINK MODEL</td>
<td>108</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>111</td>
</tr>
<tr>
<td>VITA</td>
<td>115</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1 Free Body Diagram of Vehicle Model ............................................................... 9
Figure 2 Example Membership Sets ................................................................................... 18
Figure 3 Output Membership Set Scaling due to Implication ............................................. 21
Figure 4 Aggregated Fuzzy Output ................................................................................... 22
Figure 5 Schematic Diagram of the Fuzzy Controller ...................................................... 25
Figure 6 Example of SIG and PSIG Membership Functions ............................................. 26
Figure 7 Diagram Illustrating $\alpha$ and $\Delta \phi$ ............................................................ 29
Figure 8 Membership Sets for Input Variable $\alpha$ .......................................................... 32
Figure 9 Membership Sets for Input Variable $\Delta \phi$ ....................................................... 33
Figure 10 Membership Sets for Output Variable $\Delta \alpha$ ................................................ 34
Figure 11 Vehicle Approaching Target from an Arbitrary Direction ............................... 37
Figure 12 Vehicle Approaching Target from a Specified Direction ............................... 38
Figure 13 Sensing Field Configuration ............................................................................ 39
Figure 14 Membership Sets for Input Variable $d_o$ ......................................................... 42
Figure 15 Membership Sets for Input Variable $\Delta \phi_o$ .................................................... 43
Figure 16 Membership Sets for Output Variable $\Delta \alpha$ ................................................ 44
Figure 17 Path Generated by Lumelsky's Bug2 Algorithm ................................................. 47
Figure 18 Path Generated by Sauerberger's Path Planning Algorithm .............................. 49
Figure 19 Path Generated by Proposed Bug Implementation .......................................... 54
Figure 20 Membership Sets for Input Variable $d_o$ .......................................................... 57
Figure 21 Membership Sets for Input Variable $\Delta \phi_o$ .................................................... 58
Figure 22 Membership Sets for Output Variable $\Delta \alpha$ ................................................ 59
Figure 23 Membership Sets for Input Variable $v$ ............................................................ 64
Figure 24 Membership Sets for Input Variable $d$ ............................................................ 65
Figure 25 Membership Sets for Input Variable $\Delta v$ ....................................................... 66
Figure 26 Membership Sets for Output Variable $\Delta \delta_{gb}$ ............................................. 67
Figure 27 Membership Sets for Input Variable $v$ ............................................................ 70
Figure 28 Membership Sets for Input Variable $\Delta v$ ....................................................... 71
Figure 29 Membership Sets for Output Variable $\Delta \delta_{gb}$ ............................................. 72
Figure 30 Membership Sets for Input Variable $\rho$ .......................................................... 73
Figure 31 Membership Sets for Input Variable $v$ ............................................................ 76
Figure 32 Membership Sets for Input Variable $\Delta v$ ....................................................... 77
LIST OF TABLES

Table I Literature Based Parameters for Vehicle Model .......................... 13
Table II Empirically Determined Parameters for Vehicle Model ............ 14
Table III Fuzzified Input Values ......................................................... 19
Table IV Membership Set Name Abbreviations ................................... 27
Table V Target Coordinates and Tracking Order ................................. 36
ACKNOWLEDGMENTS

I wish to acknowledge several individuals for their contributions to my graduate work. First, I would like to thank my advisor and thesis committee chair, Dr. Mohamed Trabia, who helped and encouraged me throughout my graduate program and during the writing of my thesis, and generally went far and beyond the call of duty. I would like to thank Dr. Woosoon Yim for his unselfish contribution of new computer equipment, without which I would not have made nearly as much progress. I would also like to thank Dr. Robert Boehm and Dr. Sahjendra Singh for taking the time to sit on my committee. Finally, I would like to thank all of the people in my personal life who supported me during my graduate work, including my mother, my father, my brother, Charlie, and my best friend in the world, Andrea.
CHAPTER 1

INTRODUCTION

The potential applications of autonomous vehicles are widely varied and quite important. One possible application is in operations that must be performed in hazardous environments or situations. In outer space, mobile robots can be used for exploration of celestial bodies or repair of necessary equipment. Numerous functions could also be served in underwater operations, which, in addition to scientific research and hardware maintenance, could include commercial ventures and defense applications. Various sectors of the nuclear industry, such as the medical equipment business and the nuclear power industry, involve procedures that are hazardous to human health. One duty of law enforcement agencies is to investigate, retrieve, and disarm potentially explosive devices. This is an extremely dangerous job in which law enforcement officers are injured and killed each year. Using autonomous robots to retrieve and contain these devices is another application that would keep these officers out of danger. Due to the inherent nature of war, soldiers are routinely sent into life threatening scenarios. Autonomous robots could help safeguard soldiers in a number of ways, including use in anti-personnel mine detection and neutralization as well as general reconnaissance duties. The use of autonomous vehicles in all of these operations could prove quite beneficial.
Another general application that is ideal for autonomous vehicle control is the partial or total control of passenger automobiles. Every year, many people are injured and killed as a result of drivers not paying attention or falling asleep while driving. In addition, drivers can experience medical episodes (e.g., heart attacks and seizures) while driving, and may lose control of their vehicle. Alternately, even if a driver is healthy and aware, unexpected and dangerous behaviors on the part of other drivers may require responses in less time than the typical driver takes to react. In each of these cases, autonomous vehicle control could assist in reducing accidents. Another effect of the autonomous control of vehicles is to help relieve the traffic congestion that occurs by the alleviation of unnecessary stop-and-go behavior that occurs due to high vehicle density or accident scenes.

To design a controller for vehicle navigation, computer simulation can be a useful tool. The first step in simulation is constructing an appropriate model of the system to be controlled. Since an automobile is a complex, dynamic system, the details of modeling the vehicle are very important to the performance of the simulation. The principal work in this field is Wong [24], who published the first edition of his book, *Theory of Ground Vehicles*, in 1978. Wong developed the equations of motion for various types of vehicles, detailing aspects peculiar to each vehicle type, including road vehicles (two-axle and tractor-trailers), off-road vehicles, tracked vehicles, and air-cushioned vehicles. A model that has become widely used is a model that depicts a two-axle, four-wheel vehicle and is commonly known as the "bicycle" model. Since it uses only two wheels to represent a four-wheel vehicle, it neglects the lateral variations in the tire-road interface forces. Allen, Rosenthal, et al. [1] also proposed computer models to predict the steady
state and dynamic conditions of a vehicle using a minimal set of system parameters. They presented a clear progression in the development of the model, from the tire forces to the steady state analysis to the linear dynamic model and finally the nonlinear dynamic model. Brach [3] developed and discussed various dynamic models for use in computer simulations. He presented different methods for modeling tire forces, comparing the results of linear and nonlinear models. He also furnished a useful table of example values of the relevant vehicle parameters (such as mass, moment of inertia, cornering stiffnesses, etc.) for different types of vehicles. An interesting aspect of Brach's model is that he did not neglect lateral effects, i.e., his analysis considered all four of the vehicle's wheels, not just one front wheel and one back wheel, as in the bicycle model. Wheeler and Shoureshi [23] used Wong's model as the basis of their research, which focused on the use of fuzzy control to steer a vehicle around a series of static obstacles. Their model was made more realistic by limiting the total tire force vector, since the tires cannot provide an infinite amount of traction, either in acceleration or in braking. They also derived expressions for calculating the longitudinal braking and accelerating forces.

Many models use look-up tables or various simple functions to estimate tire parameters, such as the cornering stiffnesses. As these values are actually a function of the orientation of the tire with respect to the road, the friction coefficient between the tire and the road, and the temperature of the tire, they are constantly changing, and in a complex way. Würtenerberger and Isermann [25] proposed using neural networks to estimate some of the operating parameters critical to the performance of the model. They used neural network techniques to dynamically adjust these parameters. Byrne and Abdallah [4] presented a practical implementation of Wong's derivations. Along with a
compact form of Wong's equations, they provided values of all of the vehicle parameters for a specific vehicle.

There are two main goals of any autonomous vehicle: reaching the target and avoiding collision with obstacles. To accomplish this, the controller must be able to adjust the steering angle and the gas and brake pedal angles. Many researchers have studied various aspects of the vehicle navigation control problem. Much of the work in this area is concentrated on the use of traditional control methods. For example, Nesulescu, Kim, et al. [17] implemented a controller using a feedback nonlinear controller. They used kinematic and dynamic models and implemented the control of the vehicle in an unknown environment. Byrne and Abdallah [4] used model reference adaptive control to automatically control the lateral position of a vehicle on a road. Hassoun and Laugier [6] used a potential field approach along with a feedback loop to generate the motion commands for a vehicle. They simulated the vehicle using a kinematic model. The controller was designed to avoid obstacles and to perform full road following (as opposed to just lateral control, or just centering the vehicle in the lane). Bétoume and Campion [2] used kinematic and dynamic models along with an output feedback linearization technique to insure trajectory tracking within a specified tolerance.

Ground vehicle navigation is an area where human performance has proven to be reliable. Drivers typically respond to sudden changes in their environment in a very short time. Even though other control methods may be used to control a vehicle, fuzzy logic has proven to be a good substitute for human experience. This has been proven by the successful implementation of fuzzy logic in the control systems of many consumer
products that have traditionally relied upon human input. These products include washing machines, auto-focusing systems in still photo and video cameras, and automatic automobile transmissions. In its most basic form, fuzzy logic is a method that allows the relationship between a group of independent quantities and their corresponding dependent quantities to be conveniently defined by linguistic terms. One of the main advantages of fuzzy control is its ability to incorporate human knowledge and experience, via language, into the relationships among the given quantities.

Thus, the problem of vehicle navigation toward a target amid obstacles presents an ideal application for fuzzy logic. Recently, researchers have started considering the use of fuzzy logic control in the problem of autonomous vehicle navigation. Rosa and Garcia-Alegre [20] proposed a fuzzy controller that can modify the speed and the steering of a mobile robot to steer it at a fixed distance along a wall. Using a kinematic model and a variety of walls, including cubic splines and line segments with 90 degree intersections, they designed a fuzzy controller with a minimum of rules to accomplish the task. Maeda et al. [14] also applied fuzzy logic to the problem of steering and speed control of an autonomous mobile robot. They implemented it using a two-wheel drive robot. The environment in which the control scheme was tested was limited to straight-line paths with a minimum number of 90 degree turns. Hessburg and Tomizuka [7] incorporated fuzzy logic in lateral vehicle control to solve the lane following problem. They used a dynamic model simulation as well as real-world implementation in order to evaluate their control system. The experiment was run on a road with multiple curves interspersed by straight segments. The curves were not sharp turns, but gentle ones. Lee and Wang [10] proposed the use of fuzzy logic to assist in obstacle avoidance. Their simulation included
both static and moving obstacles. Wheeler and Shoureshi [23] studied the case of a vehicle on the handling track and controlled the vehicle using fuzzy logic. They simulated the vehicle using a modified form of Wong's bicycle model. The handling track consisted of a set of pylons (known static obstacles) that the vehicle must navigate at high speeds (i.e., around 50 miles per hour).

The objective of this research is to design a control system for an automobile using fuzzy control. Observation shows that drivers tend to separate the various functions of driving. For example, most drivers conceptualize speed and direction separately. Within direction control, the driver is attempting to simultaneously accomplish two objectives: steering toward a target and avoiding obstacles. This separation of objectives is the basis of the distributed fuzzy controller design of the control system described in this research. This control system should emulate the following human behaviors:

- Starting the vehicle moving from a complete stop, and stopping it when it gets sufficiently close to the target.
- Slowing the vehicle when it approaches an obstacle, and speeding it up as it continues past the obstacle.
- Slowing the vehicle when its turning radius decreases (i.e., the tighter the turn, the lower the speed).
- Steering the vehicle toward the target.
- Steering the vehicle around any obstacle to avoid a collision.
The remainder of this thesis is divided into several chapters. The second chapter describes the nonlinear model of the vehicle dynamics. The third chapter gives a general description of fuzzy logic control. The fourth and fifth chapters detail the controllers designed in this research project. The sixth chapter presents examples showing the operation of the controller. The final chapter presents conclusions and recommendations for further work.
CHAPTER 2

VEHICLE MODEL

Many researchers use computer simulation to aid in the design of control systems. In order for the simulation to be a useful tool, an appropriately accurate computer model of the system to be controlled must be implemented. Thus, the modeling of the system is quite important. The pioneering work in this field is that of Wong [24], with the first edition of his book entitled *Theory of Ground Vehicles* published in 1978. One of the most important and widely used models developed by Wong is the dynamic response model for a four-wheeled vehicle. Wong’s model is still used by most researchers who need an accurate model of a four-wheel ground vehicle. His model is known as the “bicycle” model since lateral variations are neglected; thus, the model has only two wheels. To derive the equations, Wong used an orthogonal set of axes attached to the center of gravity of the vehicle, with the x-axis in the vehicle’s longitudinal direction and the y-axis in the vehicle’s lateral direction. His free body diagram is shown in Figure 1.
Ultimately, Wong derived the equations of motion, which must be solved simultaneously. The equations are

\[ m(\ddot{x} - \dot{y} \dot{\theta}) = F_u \cos \delta, + F_{sr} - F_{ur} \sin \delta, \]  \hspace{1cm} (1)

\[ m(\ddot{y} + \dot{x} \dot{\theta}) = F_{sr} + F_{ur} \cos \delta, + F_u \sin \delta, \]  \hspace{1cm} (2)

\[ I \ddot{\theta} = L_r F_{ur} \cos \delta, - L_s F_{sr} + L_r F_u \sin \delta, \]  \hspace{1cm} (3)

or, in matrix form,

\[
\begin{bmatrix}
  m & 0 & 0 \\
  0 & m & 0 \\
  0 & 0 & I
\end{bmatrix}
\begin{bmatrix}
  \ddot{x} \\
  \ddot{y} \\
  \ddot{\theta}
\end{bmatrix}
+ \begin{bmatrix}
  -m \dot{y} \dot{\theta} \\
  m \dot{x} \dot{\theta}
\end{bmatrix}
= \begin{bmatrix}
  F_u \cos \delta, + F_{sr} - F_{ur} \sin \delta, \\
  F_{sr} + F_{ur} \cos \delta, + F_u \sin \delta, \\
  L_r F_{ur} \cos \delta, - L_s F_{sr} + L_r F_u \sin \delta,
\end{bmatrix}
\]  \hspace{1cm} (4)
where

- the accelerations in the x, y, and θ directions are denoted by $\ddot{x}$, $\ddot{y}$, and $\ddot{\theta}$, respectively.
- the velocities in the x, y, and θ directions are denoted by $\dot{x}$, $\dot{y}$, and $\dot{\theta}$, respectively.
- the force in the longitudinal direction on the front tire is $F_{ur}$.
- the force in the lateral direction on the front tire is $F_{v}$.
- the force in the longitudinal direction on the rear tire is $F_{ur}$.
- the force in the lateral direction on the rear tire is $F_{v}$.
- the steering angle is $\phi$.
- the mass is $m$.
- the mass moment of inertia is $I$.
- the length from the center of gravity to the front axle is $L_1$, and
- the length from the center of gravity to the rear axle is $L_2$.

The full derivation of the system of dynamic equations is presented in Appendix 1. To solve for the appropriate quantities, the equations are transformed into a first order system. Then they are solved for $x$, $y$, and $\theta$. However, this displacement is the local displacement, i.e., it is only the displacement for the current time step with respect to the vehicle, not the total displacement of the vehicle with respect to a fixed coordinate system. To get the total displacement, each local displacement must be transformed into the global coordinate frame and then all of the local displacements must be summed.
together to get the global position at any point in time. The derivations of the transformation equations are given in Appendix II.

The tires are modeled as nonlinear springs, and are described by a saturation operation on the total tire force vector. This operation is defined by the following expressions:

\[ F_i = \frac{F_i}{\|F_i\|} \min \left( \|F_i\|, F_{r,\text{max}} \right) \]

\[ F_r = \frac{F_r}{\|F_r\|} \min \left( \|F_r\|, F_{r,\text{max}} \right) \]

The lateral tire forces, \( F_i \) and \( F_r \), are functions of the tire slip angles \( \alpha \) and the cornering stiffness \( C_{\alpha} \). The lateral tire forces can be calculated using the following expressions:

\[ F_{il} = 2C_{il} \alpha_i \]

\[ F_{ir} = 2C_{ir} \alpha_r \]

The tire slip angles are functions of the steering angle and of the velocities. The slip angles are defined below:

\[ \alpha_i = \delta_i - \tan^{-1} \left( \frac{L_i \dot{\theta} + \dot{y}}{\dot{x}} \right) \]

\[ \alpha_r = \tan^{-1} \left( \frac{L_r \dot{\theta} - \dot{y}}{\dot{x}} \right) \]

The axial tire forces, \( F_x \) and \( F_{xr} \), are dependent on the angle of the gas pedal, \( \delta_{\text{th}} \). This model uses the convention that a positive gas pedal angle represents the driver pushing on
the gas pedal, and a negative gas pedal angle represents the driver pressing on the brake. As such, there are two sets of axial tire force equations. For $\delta_{gb} < 0$:

$$F_{sl} = 0.7K_s \delta_{gb} \quad (11)$$

$$F_{tr} = 0.3K_s \delta_{gb} \quad (12)$$

The power train is assumed to be a rear wheel drive. Thus, for $\delta_{gb} > 0$:

$$\ddot{F}_{sl} = 0 \quad (13)$$

$$\tau_g \ddot{F}_{tr} = K_\tau \delta_{gb} - \ddot{F}_{tr} \quad (14)$$

For the implementation of this model, various constants, which are properties of the specific vehicle being modeled, needed to be found or determined experimentally. For this model, the parameters for a GMC S-15 Blazer are used. As such, some of the parameters were found in Byrne and Abdallah [4]. The values used from Byrne and Abdallah are given in Table I.
TABLE 1

Literature Based Parameters for Vehicle Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>1727 kg</td>
</tr>
<tr>
<td>$L_L$</td>
<td>1.17 m</td>
</tr>
<tr>
<td>$L_2$</td>
<td>1.42 m</td>
</tr>
<tr>
<td>$C_{af}$</td>
<td>47.000 N/rad</td>
</tr>
<tr>
<td>$C_{ar}$</td>
<td>47.000 N/rad</td>
</tr>
<tr>
<td>$I$</td>
<td>2867 kg-m²</td>
</tr>
</tbody>
</table>

However, the preceding table does not address all of the necessary constants. Specifically, the values of $K_a$, $K_b$, and $T_e$ are not given. All of these constants relate the gas pedal angle to the forces on the tires, and thus the response times of the vehicle. They were determined empirically, by varying the values and comparing the model starting and braking times to known starting and braking times. The constants were then adjusted until the model times matched the known times. The values determined are shown in Table II.
### TABLE II

Empirically Determined Parameters for Vehicle Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_c$</td>
<td>8.500 N/rad</td>
</tr>
<tr>
<td>$K_h$</td>
<td>17.000 N/rad</td>
</tr>
<tr>
<td>$\tau_g$</td>
<td>1 N/rad-s</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 3

FUZZY LOGIC

As was stated previously, fuzzy logic is basically a method that allows the relationship between a group of independent quantities and their corresponding dependent quantities to be conveniently defined by linguistic terms. The basis of fuzzy logic in natural language is the underpinning of many of its advantages in the application to control systems. One of the main advantages of fuzzy control is its ability to incorporate human knowledge and experience, via language, into the relationships among the given quantities.

Fuzzy logic was originally developed by Zadeh in 1965. The original notion was that not all sets of items in the universe can be described by Boolean logic. For example, the following statement is well defined:

Saturday is a day of the week.

And the following statement is also well defined:

Computer is a day of the week.
In each case, the previous statements can be easily and totally answered using binary logic, either with the response true or the response false. However, each of the following statements may or may not be true:

- A 20 year old person is old.
- A 40 year old person is old.
- A 60 year old person is old.
- An 80 year old person is old.

None of the previous statements can be absolutely evaluated as true or false. Whether none, some, or all of the previous statements are true depends on many things. It depends on the person’s situation, i.e., whether the person evaluating the statements is 10 years old or 100 years old. It may depend upon the time period, i.e., a person from 2000 years ago would probably evaluate these statements differently from a person alive today. It may depend on other random factors, such as occupation. A librarian would probably evaluate these statements differently than a Secret Service agent on the Presidential detail. Even with all of the previous factors fixed, each statement may be truthful to some degree. One person may say that a 40 year old person is a little old, a 60 year old person is moderately old, and an 80 year old person is very old. This example shows that varying degrees of truthfulness can typically be assigned to many statements that people make every day. Another way of saying this is that the set of old people is a fuzzy set, i.e., that its members are not clearly defined. And this varying degree of truthfulness is the foundation of fuzzy logic. Specifically, fuzzy logic refers to the fact that the
truthfulness of a statement can have any value between 0 and 1, inclusive, and thus is fuzzy (i.e., not crisp). For more information on fuzzy logic, see [5] or [26].

Even though fuzzy logic is, in and of itself, quite an interesting concept, it does not directly address the problem of automatic control. This problem is solved by a method of converting fuzzy logic from a purely linguistic system to a type of translator that can create a set of mathematical equations that can be easily implemented on a computer. This method will be called fuzzy logic control (or FLC) in this thesis. FLC has five steps that must be followed to allow it to be implemented as an algorithm in a computer program:

1. Input fuzzification
2. Application of logical operators
3. Implication
4. Aggregation
5. Output defuzzification

Each of these steps will be discussed in the following paragraphs.

The first step in the FLC process is that of input fuzzification. The inputs to the system are crisp (absolute) numerical values, which will be converted to truth values using the membership sets. The membership sets are really just curves which determine what truth value can be assigned to any given input value. Figure 2 shows a possible set of membership sets for a fuzzy quantity.
The curve shapes (or membership functions) in this example are arbitrary and, in general, can be of any type, including triangular, trapezoidal, Gaussian, or sine. The truth value (hereafter denoted by $\mu$) can be found by going to the appropriate value on the x axis, tracing a line up to the membership function curve defining the fuzzy set, and then tracing a line left to find the truth value. To be noted is the fact that all input variables have truth values with respect to all membership sets, even if that value is 0. This is demonstrated in Table III.
TABLE III

Fuzzified Input Values

<table>
<thead>
<tr>
<th>Input value</th>
<th>( \mu_{\text{small}} )</th>
<th>( \mu_{\text{medium}} )</th>
<th>( \mu_{\text{large}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.25</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>13.75</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Looking at an input value of 6.25, it is noted that the input has a 0.5 truth value for the small membership set, a 0.5 truth value for the medium membership set, and a 0 truth value for the large membership set. This might be stated linguistically as “6.25 is kind of small, kind of medium, but not large at all.”

After all of the inputs have been fuzzified, i.e., truth values have been obtained, the next step is to combine all of the input values to a single value. This is really the application of the fuzzy operators, the fuzzy version of the standard logical operators and and or. From the human perspective, the rule base controls exactly how the input values are combined. The rule base is a list of if-then type rules describing the input-output relationship. A possible rule might be

If (input1 is small) and (input2 is large) then (output is medium).
Using the truth values for input values of 6.25 and 13.75 yields

\[ \text{If } (0.5) \text{ and } (0.5) \text{ then (output is medium)}. \]

Evaluation of the fuzzy version of \textit{and} and \textit{or} is variable and dependant on the preference of the person building the system. For this project, \textit{and} was defined as a simple product, and \textit{or} was chosen to be computed using the probabilistic OR function, which is defined as

\[
\text{probor}(a, b) = a + b - ab
\]  

(15)

Thus, the previous rule evaluates to

\[ \text{If } (0.25) \text{ then (output is medium)}. \]

The next step in the process is \textit{implication}, which is sometimes called inference. This is the procedure by which the antecedent (\textit{if} part of the rule) effects the consequent (\textit{then} part of the rule). Again, there are various methods for accomplishing this, but the method used in this research is scaling. For the previous rule evaluation, the medium membership set would be scaled by a factor of 0.25. Assuming that the output membership sets are identical to the input membership sets, the result is shown in Figure 3, where the dotted line shows the original membership set and the solid line shows the scaled membership set.
The fourth step in the process is aggregation. This is simply the act of creating a total fuzzy set describing the total output of all of the rules based on the output of each rule. Like some of the other steps, this may be implemented numerically in a number of ways. This research uses the max method. All this method does is go through the fuzzy output for all rules and determine the largest scale factor for each membership function. Thus, if the maximum small membership function scale factor is 0.75, the maximum medium membership function scale factor is 0.25, and the maximum large membership function scale factor is 1, then the aggregate fuzzy output would look like Figure 4.
The final step is to *defuzzify* the aggregate fuzzy output to get a numeric output. Again, many methods can be used, but the one used in this paper is the centroid method. This method finds the centroid of the closed geometric shape that is the aggregate fuzzy output. The crisp value obtained is actually the *x* value of the centroid. The *x* coordinate of the centroid of a composite shape (a shape with any number of simple shapes, including rectangles and triangles) is given by

\[
\bar{x} = \frac{\sum A_i \bar{x}_i}{\sum A_i}
\]

(16)

where

- \( i \) is the total number of simple areas.
- \( A \) is the area of each simple area, and
- \( \bar{x} \) is the global centroid coordinate of each simple area.

**Figure 4** Aggregated Fuzzy Output
For each simple area, the local centroid can be found with

\[
\bar{x} = \frac{\int x \, dA}{A}
\]

(17)

For the aggregated fuzzy set shown in Figure 4, the output value is 11.4 (assuming the upper bound of the large membership set is 20).

This chapter shows the flexibility of FLC, with respect to its linguistic representation and its computational implementation. All of these factors combine to make a FLC an ideal tool to implement human knowledge of a system that is very complex. The next two chapters describe the implementation of fuzzy logic that is used in this research to control a passenger vehicle.
In an effort to utilize human knowledge and experience most efficiently in the design of the controller, the driving tasks were divided and a fuzzy controller was designed for each of them. The driving tasks were divided as follows: target steering, target throttle, cornering throttle, collision avoidance steering, and collision avoidance throttle. An additional controller was implemented to assist in the incorporation of the Bug algorithm.

The actual implementation of the controllers in MATLAB's Simulink simulation environment grouped the controllers by function, i.e. all of the steering controllers were grouped together and all of the throttle controllers were grouped together. Figure 5 shows a schematic of the inputs and outputs of the various fuzzy modules.
The operating parameters, which will be described here, were common to all of
the fuzzy modules. The membership functions in all of the fuzzy modules in this thesis
are of the predefined MATLAB types sigmoid (SIG) and product of sigmoid (PSIG).
The SIG membership function is defined as follows: given the parameters \(a\) and \(c\).

\[
f(x) = \frac{1}{1 + e^{-a(c-x)}}
\]  

Figure 5 Schematic Diagram of the Fuzzy Controller
Given the above definition, the parameter $a$ is analogous to a slope value and $c$ is analogous to an $x$-intercept. The PSIG membership function is just the product of two SIG membership functions. The SIG membership function is open ended, somewhat like a step function, and thus was used for the boundary membership functions (such as positive large). The PSIG is a closed shape, somewhat like a triangle, and so was used for the inner membership functions (for example, the zero membership functions).

Figure 6 Example of SIG and PSIG Membership Functions

Figure 6 shows an example of the SIG and PSIG membership functions. Three functions are shown: a SIG open to the left (SIG2), a SIG open to the right (SIG1), and a product of the two SIG functions, which is closed. The parameters are as follows: $a_1 = 2.5$, $c_1 = 3$, $a_2 = -$.
0.75, and $c_5 = 7$. All of the memberships set plots displayed in this thesis show angular quantities in radians. The logical operator and was implemented using the product method and the logical operator or was implemented using the probabilistic OR function. Implication was performed using the product function. aggregation was performed using the maximum function, and defuzzification was performed using the centroid function. All of the parameters listed here are described in depth in Chapter 3. The naming of the membership sets is summarized in Table IV.

**TABLE IV**

Membership Set Name Abbreviations

<table>
<thead>
<tr>
<th>Membership set name</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>Z, ZE</td>
</tr>
<tr>
<td>Positive</td>
<td>PO, P</td>
</tr>
<tr>
<td>Negative</td>
<td>NE, N</td>
</tr>
<tr>
<td>Small</td>
<td>S, SM</td>
</tr>
<tr>
<td>Medium</td>
<td>M, ME</td>
</tr>
<tr>
<td>Large (big)</td>
<td>L, L.A, B</td>
</tr>
</tbody>
</table>

Examples of varying complexity were used to manually tune the membership sets and the rules of the various modules. The complexity of the examples was varied by
altering the initial vehicle position and orientation, the target vehicle position and orientation, the number and configuration of static obstacles, and the path and speed of dynamic obstacles (which can represent other vehicles, people, or any other moving object in the vehicle’s environment). As the controller is divided into two primary sections, steering and throttle control, this thesis is divided that way as well, with Chapter 4 addressing steering control and Chapter 5 addressing throttle control.

**Target Steering Fuzzy Module**

The objective of this module is to steer the vehicle toward a target from its current location. The inputs to this module are the steering angle $\alpha$ and the angle to the target $\phi$. The steering angle $\alpha$ and angle to target $\phi$ are shown in Figure 7. The value of the steering angle used in all modules is the value from the previous time step. The output of this module is the change of steering angle $\Delta\alpha$. 
Target Steering Fuzzy Rules

The two basic goals of this module are as follows:

- If the target is to the right of the vehicle, the vehicle should turn to the right. The more the target is to the right, the sharper the right hand turn.

- If the vehicle needs to turn right, the amount of right turn added is dependent upon the current amount of right turn.
Note that the number of goals of the module is the same as the number of input variables. This is because each goal can be resolved into an input-output relationship. Looking at several of the rules that define part of the input-output relationship:

- If \( \alpha \) is Z and \( \Delta \phi \) is NB then \( \Delta \alpha \) is NM: This rule says that if the vehicle is moving along a straight path and the target is to the right of the vehicle then the correction to the steering angle should be to the right.

- If \( \alpha \) is PB and \( \Delta \phi \) is Z then \( \Delta \alpha \) is NM: This rule says that if the vehicle is turning to the left and the target is straight ahead then the correction to the steering angle should be to the right.

- If \( \alpha \) is Z and \( \Delta \phi \) is Z then \( \Delta \alpha \) is Z: This rule says that if the current steering angle is zero and the target is straight in front of vehicle then the correction to the steering angle should be zero.

The full rule base of this module can be summarized as follows:

- If \( \alpha \) is PM or PB and \( \Delta \phi \) is NB then \( \Delta \alpha \) is NB

- If \( \alpha \) is Z or PM or PB and \( \Delta \phi \) is NM then \( \Delta \alpha \) is NM

- If \( \alpha \) is Z and \( \Delta \phi \) is NB then \( \Delta \alpha \) is NM

- If \( \alpha \) is PB and \( \Delta \phi \) is Z then \( \Delta \alpha \) is NM

- If \( \alpha \) is NB or NM and \( \Delta \phi \) is NB or NM then \( \Delta \alpha \) is Z

- If \( \alpha \) is PM or PB and \( \Delta \phi \) is PM or PB then \( \Delta \alpha \) is Z
• If (\(\alpha\) is NM or Z or PM) and (\(\Delta \phi\) is Z) then (\(\Delta \alpha\) is Z)

• If (\(\alpha\) is NB) and (\(\Delta \phi\) is Z) then (\(\Delta \alpha\) is PM)

• If (\(\alpha\) is Z) and (\(\Delta \phi\) is PM or PB) then (\(\Delta \alpha\) is PM)

• If (\(\alpha\) is NB or NM) and (\(\Delta \phi\) is PM or PB) then (\(\Delta \alpha\) is PB)

Target Steering Fuzzy Membership Sets

From this point forward, all fuzzy variables with symmetric membership sets will
be discussed by observing only the positive membership sets. Looking at the
membership sets for \(\alpha\) and \(\Delta \phi\), shown in Figures 8 and 9, the following is observed:

• Z is from negative 20 degrees to 20 degrees with a peak at zero degrees

• PM is from zero degrees to 90 degrees and is relatively crisp

• PB is from zero degrees to 180 degrees and is relatively crisp
Figure 8 Membership Sets for Input Variable $\alpha$
The inner bounds of the positive sets are all at zero due to the gradual manual tuning that was performed on them. This also accounts for the lack of a "small" membership set, which was eliminated during the fine tuning phase.

Looking at the membership sets for $\Delta \alpha$, shown in Figure 10, the following is observed:

- $Z$ is from negative 15 degrees to 15 degrees with a peak at zero degrees
- PM is from zero degrees to 50 degrees with its peak between 10 degrees and 30 degrees
- PB is from 35 degrees to 65 degrees with its peak between 50 degrees and 65 degrees
Notice that this time, each of the sets seems to have a moderate amount of overlap. However, there is still not a "small" set, which is due to the same cause as before. It may be said that a 45 degree change in the steering angle in one time step (0.01 seconds) is not reasonable; however, it should be noted that only six rules (out of 25) trigger the "big" membership sets. Thus, large values of $\Delta \alpha$ will always be mitigated by the amount of smaller output that is triggered.

**Figure 10** Membership Sets for Output Variable $\Delta \alpha$
Final Orientation Control

Even though it was not implemented using fuzzy logic, the final orientation control was implemented within the target steering calculations, and so is discussed here. The final orientation calculations are totally independent of the fuzzy steering modules: that is, the fuzzy steering modules continue to control the vehicle as they otherwise would while the final orientation algorithm is operational. The method of achieving the final orientation is quite simple, and effectively requires fooling the target steering fuzzy module. The algorithm only affects the steering of the vehicle if the final orientation feature is activated by specifying a final orientation angle in the program. Essentially, the final orientation consists of setting up "virtual" target coordinates that are based on the desired final orientation. Strategically placed virtual target coordinates fool the controller into guiding the vehicle to the correct orientation. The premise is that the vehicle will always steer toward the target coordinates; thus, if the virtual coordinates are chosen properly, the final orientation of the vehicle can easily be controlled. The ordering of target coordinates when the final orientation algorithm is shown in Table V.
In Table V, $X_{\text{global}}$ and $Y_{\text{global}}$ are the actual target coordinates and $\theta$ is the desired final orientation. By creating the sequence of target points shown, the algorithm guides the vehicle to the target with the correct final orientation.

Figure 11 shows the vehicle approaching the actual target at a 45 degree orientation, but only because that is the angle the vehicle happens to be approaching from.
Figure 11 Vehicle Approaching Target from an Arbitrary Direction

On the other hand, Figure 12 shows the vehicle approaching the target from a specified direction: in this case, 90 degrees. The final orientation algorithm first creates T1\textsubscript{virtual}, which the vehicle drives straight towards. When the vehicle is close enough, the algorithm creates T2\textsubscript{virtual}, and again the vehicle drives toward it. Finally, the algorithm allows the vehicle to detect the actual target, and the vehicle goes straight toward it. Again, it is useful to note that T1\textsubscript{virtual} is 20 meters from the target and T2\textsubscript{virtual} is 10 meters from the target.
Figure 12 Vehicle Approaching Target from a Specified Direction

Collision Avoidance Steering Fuzzy Module

The objective of this module is to steer the vehicle away from obstacles, both static and dynamic. It is assumed that the vehicle is equipped with a sensing system to determine the distance and direction of obstacles. The sensing system may be a single sensor mounted on a rotating platform on the front of the vehicle or may be a battery of
sensors arrayed around the vehicle's front section and pointed along regular intervals. In this thesis, the buffer zone radius will be denoted by $r_b$, and is generally assigned a value of 20 meters. The configuration of the buffer zone is shown in Figure 13.

![Figure 13 Sensing Field Configuration](image)

The sensor system inputs to the controller only two bits of information in order to simplify the analysis. These inputs are the nearest distance to an obstacle $d_{\text{min}}$ and the
angle of this distance with respect to the vehicle $\Delta \phi$. As Figure 13 illustrates, the sensor may not return the *actual* closest point and angle. This is because the sensing system used for this research is not continuous: rather, the sensor takes $n_t$ samples over its 180 degree sweep. The output of the module is the steering correction $\Delta \alpha$.

Collision Avoidance Steering Fuzzy Rules

Since the locations of the obstacles are not known before the vehicle starts traversing the environment, the module uses a right-hand-turning rule whenever it confronts an obstacle. The two basic goals of this module are as follows:

- The closer the vehicle is to the obstacle (linear distance), the more extreme the evasive maneuver (the larger the steering correction).
- The direction of the obstacle with respect to the front of the vehicle determines the magnitude and direction of the steering correction.

Again, note that the number of goals of the module is the same as the number of input variables. At this point, it would be useful to look at a few of the rules to observe the relation between the rules and the goals of this fuzzy module:

- If $(d_o \text{ is SM})$ and $(\Delta \phi_o \text{ is PS})$ then $(\Delta \alpha \text{ is NM})$: This rule says that if the obstacle is from 10 to 20 meters away and is from zero to 60 degrees off to the left of the vehicle, the steering correction will be to the right and will be severe.
• If \((d_{o} \text{ is LA})\) then \((\Delta \alpha \text{ is ZE})\): This rule says that if the obstacle is more than 20 meters away from the vehicle then no steering correction is made, regardless of its direction.

The rules of this module can be summarized as follows:

• If \((d_{o} \text{ is SM or ZE})\) and \((\Delta \phi_{o} \text{ is ZE or PS or PM})\) then \((\Delta \alpha \text{ is NM})\)

• If \((d_{o} \text{ is LA})\) then \((\Delta \alpha \text{ is ZE})\)

• If \((d_{o} \text{ is SM})\) and \((\Delta \phi_{o} \text{ is NS})\) then \((\Delta \alpha \text{ is PS})\)

• If \((d_{o} \text{ is ZE})\) and \((\Delta \phi_{o} \text{ is NS})\) then \((\Delta \alpha \text{ is PM})\)

• If \((d_{o} \text{ is SM or ZE})\) and \((\Delta \phi_{o} \text{ is NM})\) then \((\Delta \alpha \text{ is PM})\)

Collision Avoidance Steering Fuzzy Membership Sets

The membership sets for this module are shown in Figures 14, 15, and 16.

Looking at the membership sets for \(d_{o}\), shown in Figure 14, the following is observed:

• \(Z\) is from zero meters to 20 meters with its peak at zero meters

• \(SM\) is from zero meters to 20 meters with its peak at 18 meters

• \(LA\) is from 20 meters to 80 meters with its peak from 23 meters out to 80 meters
Figure 14 Membership Sets for Input Variable $d_n$

The range from zero to 20 meters is filled with the zero and small sets, with zero decreasing as $d_n$ increases and small increasing as $d_n$ increases. If the obstacle is more than 20 meters from the vehicle, it is considered to be far enough away to be negligible.

Looking at the membership sets for $\Delta \phi_n$, shown in Figure 15, the following is observed:

- **ZE** is from zero degrees to 30 degrees with its peak at 0 degrees
- **PS** is from zero degrees to 60 degrees with its peak at 30 degrees
- **PM** is from 30 degrees to 180 degrees with its peak at all values past 60 degrees
The medium set, which is everything past 60 degrees from the front of the vehicle, is viable: anything more than 60 degrees from the front of the vehicle can be treated in the same manner. Finally, looking at the membership sets for $\Delta \phi$, shown in Figure 16, the following is observed:

- **ZE** is from zero degrees to five degrees with its peak at zero degrees
- **PS** is from zero degrees to 20 degrees with its peak at between five and 15 degrees
- **PM** is from 20 degrees to 60 degrees with its peak at all values past 30 degrees
Figure 16 Membership Sets for Output Variable $\Delta \alpha$

The sets of $\Delta \alpha$ for this module are the same shape and relative size as the sets for $\Delta \alpha$ for the previous module. The only difference is that all of the sets have been scaled down, i.e., the bounds are smaller. The similarity is indicative of the fact that the relative set shapes and sizes provide a smooth steering response.

**Bug Steering Fuzzy Module**

**Background**

In certain configurations, the primary fuzzy module can get stuck in a repeating loop, never reaching the target and never determining that the target is unreachable; thus.
the controller would keep trying to reach the target forever. As a backup to the primary fuzzy controller, which regulates obstacle avoidance steering, the Bug module was implemented. The Bug module is based on the Bug2 algorithm proposed by Lumelsky [11]. This module, as opposed to all of the others, is not designed to emulate human behavior, but rather to guarantee a successful path to the target in the situation that the primary steering module is not able to.

Much of Lumelsky's research focused on the application of maze theory to the path planning capabilities of a "point automaton." Lumelsky compared various path-planning algorithms in order to define a set of criteria from which the best path planning algorithm might be chosen for a given type of environment. The environment that Lumelsky studied was typically a complex series of obstacles (a maze) between the start point and the target point. The criteria he used were the ability of the algorithm to get to the target and the path length. Ultimately, he concluded that Bug2 has "unbounded worst-case performance", i.e. in very rare cases. Bug2 can steer the vehicle in an unending loop, and thus will not reach the target at all. But these are rare cases, and in more common mazes, the path generated by Bug2 is significantly shorter than those generated by the other algorithms he studied. This is due to the fact that some of the other algorithms, specifically Bug1, in order to guarantee better worst-case performance, will actually search the whole maze, if necessary, to reach the target; thus, the Bug1 approach, although systematic and reliable, is somewhat inefficient. Given this information, the backup algorithm for this control system was modeled after Bug2. This is because it was
assumed that the obstacle configurations for this research would be relatively simple: thus, \textit{Bug2} should always reach the target and would provide better performance.

\textbf{Description of Lumelsky's Bug2}

Several definitions are necessary to describe the \textit{Bug2} algorithm. The first is the "desired" path: this is a straight line between the starting point and the target point (hereafter called the ST line). The hit and leave points are the points where the ST line intersects with the obstacles. The hit points are denoted by $H_i$ and $L_i$ denotes the leave points, where $i$ defines the point number. Initially, $i=1$ and $L^n=S$. The algorithm follows the list of rules below:

1) From point $L^{i-1}$, move along the ST line toward the target until one of the following occurs:

a) The target point is reached and the procedure stops.

b) An obstacle is encountered, and a hit point $H_i$ is defined. Go to step 2.

2) Turn left and follow the obstacle boundary until one of the following occurs:

a) The target is reached and the procedure stops.

b) The ST line is met at a point $A$ such that the distance $AT < HT$ and the line $AT$ does not cross the current obstacle at the point $A$. Define the leave point $L_i=A$ and let $i=i+1$. Go to step 1.
c) The autonomous robot returns to $H^t$ and thus traverses a closed loop without having defined the next hit point $H^{t-1}$. The vehicle or the target is trapped and cannot be reached; the procedure stops.

The execution of these rules is demonstrated in Figure 17.

Figure 17 Path Generated by Lumelsky's Bug2 Algorithm

Note that the path of the vehicle follows Lumelsky's rules, except for the second time the vehicle reaches the point $H^2$. If the rules above (taken directly from [11]) are literally interpreted, the vehicle would assume that it is trapped and would stop. However, one of
the plots in [11] showed this same situation: the vehicle passing $H$ going back to $L$ and instead of turning left at $L$, going straight through the ST line and then resuming the execution of the rules. Obviously, Lumelsky left a rule out of the previous list.

Description of Sauerberger and Trabia's Modified Bug2

Another approach was proposed by Sauerberger and Trabia [11] for use with an autonomous omnidirectional vehicle. In their algorithm, the following steps are followed:

1) The vehicle attempts to go straight to the target.

2) If the vehicle encounters an obstacle, it goes into the "efficient approach"
   a) Upon encountering the obstacle, turn left.
   b) Constantly check the difference between the target heading and the obstacle's boundary heading. If the magnitude of the angle to the target is smaller than the magnitude of the angle of the wall, break from the obstacle and go toward the target.

3) Constantly check the loop condition. This consists of checking to see if the magnitude of the vehicle's orientation has increased by more than 360 degrees. If so, then the control algorithm assumes the vehicle is stuck and reverts to the Bug2 algorithm.
a) The single modification in Sauerberger’s backup Bug2 implementation is that the start point of the S_T line (Sauerberger’s ST line) is the vehicle’s position when Bug2 becomes active, which, in general, will be a different point than the one from where the vehicle started.

The execution of these rules is shown in Figure 18.

![Figure 18 Path Generated by Sauerberger’s Path Planning Algorithm](image-url)
Note that the original ST line is left in Figure 18 only for reference. Also, it may be useful to note here that Sauerberger's primary algorithm is somewhat similar to the Bug2, i.e., it exhibits the same object following characteristic but uses a different process to decide when to break from the obstacle. At point $A$, the vehicle encounters an obstacle and turns left. The vehicle tracks the obstacle until point $B$, where the target is closer than the wall (in the angular direction). At this point, it breaks from the obstacle and goes straight toward the target. It is useful to note that Sauerberger's algorithm generates a straight line between $B$ and $B'$, while Lumelsky's algorithm would have continued to track the obstacle, turned left at the ST line, and then turned left at the next obstacle, thus generating a much longer path between $B$ and $B'$. Next, the vehicle tracks the obstacle until point $C$, where, again, the target heading is smaller than the wall heading, and thus the vehicle steers straight toward the target. At the next obstacle it again turns left. It tracks the wall until the point $S_T$. At this point, the algorithm recognizes that the vehicle has traversed 360 degrees, and reverts to the Bug2 algorithm. Bug2 then steers the vehicle successfully to the target. It is useful to note that, again, the vehicle is able to cross the $S_T$ Line, even though Sauerberger does not explicitly state how it does so. The fact that it can do this is based on the assumption that the backup algorithm is strictly Bug2 (with the single exception stated), and thus must have such a provision.
Description of Proposed Bug Fuzzy Module

As has been previously stated, the previous implementations of Bug2 have exhibited the following characteristics:

- Used to control a point automaton / omnidirectional vehicle (both)
- Used as a primary path-planning algorithm (Lumelsky/Sauerberger uses modified version)
- Used as a backup path planning algorithm (Sauerberger uses original version)

However, in this research, the Bug algorithm is used exclusively as a backup algorithm to the primary steering controller. Thus, a criterion was necessary to determine when the Bug steering module needed to be switched on. The controller designed in this research uses a modified version Sauerberger's criterion to determine if the backup algorithm needs to be activated. That is, the Bug module will activate if the orientation of the vehicle is more than 360 degrees away, in either direction, from the angle of the ST line. This can be expressed as the following:

\[ |\theta_{\text{vehicle}} - \theta_{\text{ST}}| > 2\pi \] \hspace{1cm} (19)

For example, if the ST line were at 45 degrees, the vehicle would have to traverse a closed loop, and have an orientation of 405 degrees, before the Bug2 algorithm would activate.

Once the trigger condition has been met, the Bug algorithm will perform the following tasks:
1) Initially, the algorithm guides the vehicle straight toward the target.
   a) If the finishing criteria (discussed below) are satisfied, then turn off the Bug2
      algorithm and go straight to the target.
   b) If the vehicle gets to within $r_b$ meters of an obstacle, the vehicle will turn to the
      right, putting the obstacle on the left of the vehicle.

2) The vehicle will proceed to follow the boundary of the obstacle, always keeping the
   obstacle on its left.
   a) If the finishing criteria are satisfied, the vehicle will go straight to the target.
   b) The vehicle encounters the ST line. The algorithm will cause the vehicle to stay
      on one side of the ST line. This will aid the vehicle in reaching the target. When
      the vehicle approaches the ST line, it treats it as an obstacle, turning to the right
      and keeping the line on its left.
   c) If the vehicle does manage to get on the opposite side of the ST line, behave as if
      nothing had changed (i.e., track all obstacles, keeping them on the left side of the
      vehicle) and cross the ST line at the first opportunity. Then, continue as described
      above.

All of the finishing criteria must be simultaneously satisfied for the controller to allow the
vehicle to go to the target, and they are as follows:
• The vehicle must be within 20 meters of the target. If the vehicle is more than 20 meters from the target, an obstacle could lie between the vehicle and the target and be undetectable by the vehicle's sensors.

• The vehicle must have "line of sight" to the target. This is based on Sauerberger's criterion, and is implemented by comparing the magnitude of the angle to the target to the magnitude of the angle to the nearest obstacle.

The final customization of the Bug module is to account for the type of vehicle involved. The rules and sets that implement the module have accounted for the fact that a "real" passenger vehicle cannot make a zero radius turn. Thus, the module begins the appropriate steering adjustments far enough ahead of time to allow the vehicle to make the necessary turns without colliding with obstacles or crossing the ST line. An approximate path generated by the Bug module is shown in Figure 19.
Figure 19 Path Generated by Proposed Bug Implementation

It is initially useful to note that the vehicle has a right hand turning rule. Given driving customs in the United States, it seems more logical that a human driving a vehicle would turn to the right, all other things being equal. It may also be noted that the changes that allow the algorithm to work for a real vehicle, as opposed to an omnidirectional vehicle, are not shown, as they do not significantly modify the vehicle's behavior with respect to its overall path. Initially, the vehicle behaves just as Lumelsky's. However, it quickly has the opportunity to get on the "wrong" side of the ST line, which it does. However, it continues to track the obstacles, and as soon as it can cross back to the "correct" side, it

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
does. It then continues on to the end of the maze, and breaks free when the finishing conditions are met, driving straight to the target.

The switching of the various controller components necessary to make the Bug algorithm work is done with Simulink elements. The fuzzy module used in the Bug algorithm controls the obstacle tracking characteristics. The inputs to the Bug module are the distance to the obstacle $d_o$ and the angle to the obstacle $\Delta \phi_o$. The output of this module is the correction in the steering angle $\Delta \alpha$.

**Bug Steering Fuzzy Rules**

As the rules in this module allow the vehicle to track obstacles, they are very similar to those in the collision avoidance steering module. The rules of this module can be summarized as follows:

- If $(d_o, \text{is ZE or SM})$ and $(\Delta \phi_o, \text{is LR})$ then $(\Delta \alpha, \text{is PS})$
- If $(d_o, \text{is ZE})$ and $(\Delta \phi_o, \text{is LS})$ then $(\Delta \alpha, \text{is NS})$
- If $(d_o, \text{is ZE})$ and $(\Delta \phi_o, \text{is LF or FR})$ then $(\Delta \alpha, \text{is NM})$
- If $(d_o, \text{is ZE})$ and $(\Delta \phi_o, \text{is RF or RS or RR})$ then $(\Delta \alpha, \text{is PM})$
- If $(d_o, \text{is SM})$ and $(\Delta \phi_o, \text{is LS})$ then $(\Delta \alpha, \text{is ZE})$
- If $(d_o, \text{is SM})$ and $(\Delta \phi_o, \text{is LF})$ then $(\Delta \alpha, \text{is NS})$
• If \( (d_o \text{ is SM}) \) and \( (\Delta \phi_o \text{ is FR}) \) then \( (\Delta \alpha \text{ is NM}) \)

• If \( (d_o \text{ is SM}) \) and \( (\Delta \phi_o \text{ is RF or RS or RR}) \) then \( (\Delta \alpha \text{ is PM}) \)

• If \( (d_o \text{ is ME}) \) and \( (\Delta \phi_o \text{ is LR or LS}) \) then \( (\Delta \alpha \text{ is PS}) \)

• If \( (d_o \text{ is ME}) \) and \( (\Delta \phi_o \text{ is LF or FR}) \) then \( (\Delta \alpha \text{ is NS}) \)

• If \( (d_o \text{ is ME}) \) and \( (\Delta \phi_o \text{ is RF or RS or RR}) \) then \( (\Delta \alpha \text{ is PM}) \)

• If \( (d_o \text{ is LA}) \) then \( (\Delta \alpha \text{ is ZE}) \)

Bug Steering Fuzzy Membership Sets

The membership sets for this module are shown in Figures 20, 21, and 22.

Looking at the membership set for the distance to the obstacle \( d_o \), shown in Figure 20, the following is observed:

• \( Z \) is from zero meters to two meters with its peak at zero meters

• \( SM \) is from two meters to four meters with its peak at three meters

• \( ME \) is from four meters to 20 meters and is approximately crisp

• \( LA \) is from 20 meters to 100 meters and is approximately crisp
It should be noted that even though the rules of this module are quite similar to the previous one, the membership sets are quite different. The significant modification of the membership sets was necessary to modify the module for use in the Bug algorithm.

Looking at the membership set for the angle to the obstacle $\phi$, shown in Figure 21, the following is observed:

- FR (front) is from negative 30 degrees to 30 degrees and is relatively crisp
- LF (left front) is from 30 degrees to 60 degrees and is relatively crisp
- LS (left side) is from 60 degrees to 120 degrees and is relatively crisp
- LR (left rear) is from 120 degrees to 180 degrees and is relatively crisp
Figure 21 Membership Sets for Input Variable $\phi$.

Again, even though the previous variable covers the same range of values as its predecessor, the number of membership functions and their shapes are quite different. Looking at the membership sets for $\phi$, shown in Figure 22, it can be seen that the number and relative shape of the membership sets is exactly the same as in the previous module. The only difference is that the bounds are different; thus, all of the sets have been scaled appropriately. In the previous module the bounds were [-0.9, 0.9] and in this module they are [-1.5, 1.5].
Figure 22 Membership Sets for Output Variable $\Delta \xi$
CHAPTER 5

THROTTLE FUZZY CONTROL

Chapter 5 complements Chapter 4, in that it describes the throttle-control portion of the controller. The three throttle controllers are target throttle control, cornering throttle control, and collision avoidance throttle control. Each module is described in the following subsections.

Target Throttle Fuzzy Module

The objective of this module is to speed up the vehicle to reach the target and to slow it down to stop at the target. The inputs to this module are the speed of the vehicle $v$, distance to target $d$, and the change of speed from the previously measured value $\Delta v$. The module has one output, which is the change in the gas pedal/brake angle $\Delta \theta_{gb}$.

Target Throttle Fuzzy Rules

The goals of this module are as follows:
• To modify the velocity of the vehicle based on the current velocity of the vehicle.

• To increase the speed of the vehicle if the target is far away and decrease the speed if the target is near.

• To modify the velocity of the vehicle based on the change in velocity (acceleration) of the vehicle.

Looking at a few of the rules, as follows, demonstrates the implementation of the goals:

• If \( (v \text{ is SM}) \) and \( (d \text{ is ZE}) \) and \( (\Delta v \text{ is ZE}) \) then \( (\Delta \delta_{gb} \text{ is NS}) \): This rule says that if the velocity is in the neighborhood of 16 kilometers per hour and the distance is less than five meters and the acceleration is zero then slow the vehicle down some.

• If \( (v \text{ is ME}) \) and \( (d \text{ is LA}) \) and \( (\Delta v \text{ is PO}) \) then \( (\Delta \delta_{gb} \text{ is ZE}) \): This rule says that if the velocity is medium (10 to 15 meters per second) and the distance is more than 200 meters and the acceleration is already positive then do not modify the gas pedal angle.

• If \( (v \text{ is SM}) \) and \( (d \text{ is LA}) \) and \( (\Delta v \text{ is ZE}) \) then \( (\Delta \delta_{gb} \text{ is PB}) \): This rule says that if the velocity is about 16 kilometers per hour and the target is over 200 meters away and the acceleration is zero then increase the velocity of the vehicle.

Finally, the full set of rules of this module can be summarized as follows:

• If \( (v \text{ is ME}) \) and \( (d \text{ is E}) \) and \( (\Delta v \text{ is ZE or PO or NE}) \) then \( (\Delta \delta_{gb} \text{ is NB}) \)

• If \( (v \text{ is ME or LA}) \) and \( (d \text{ is SM}) \) and \( (\Delta v \text{ is PO}) \) then \( (\Delta \delta_{gb} \text{ is NB}) \)

• If \( (v \text{ is LA}) \) and \( (d \text{ is ZE}) \) and \( (\Delta v \text{ is ZE or PO}) \) then \( (\Delta \delta_{gb} \text{ is NB}) \)
• If \((v \text{ is } \text{ZE or SM}) \text{ and } (d \text{ is } \text{ZE}) \text{ and } (dv \text{ is } \text{PO})\) then \((\Delta \delta_{gb} \text{ is NS})\)

• If \((v \text{ is SM}) \text{ and } (d \text{ is SM}) \text{ and } (dv \text{ is } \text{PO})\) then \((\Delta \delta_{gb} \text{ is NM})\)

• If \((v \text{ is SM}) \text{ and } (d \text{ is SM}) \text{ and } (dv \text{ is } \text{NE})\) then \((\Delta \delta_{gb} \text{ is ZE})\)

• If \((v \text{ is ME}) \text{ and } (d \text{ is SM}) \text{ and } (dv \text{ is } \text{ZE})\) then \((\Delta \delta_{gb} \text{ is NM})\)

• If \((v \text{ is ME}) \text{ and } (d \text{ is SM}) \text{ and } (dv \text{ is } \text{NE})\) then \((\Delta \delta_{gb} \text{ is NS})\)

• If \((v \text{ is LA}) \text{ and } (d \text{ is ZE or SM}) \text{ and } (dv \text{ is } \text{NE})\) then \((\Delta \delta_{gb} \text{ is NS})\)

• If \((v \text{ is LA}) \text{ and } (d \text{ is LA}) \text{ and } (dv \text{ is } \text{PO})\) then \((\Delta \delta_{gb} \text{ is NB})\)

• If \((v \text{ is ZE}) \text{ and } (d \text{ is SM}) \text{ and } (dv \text{ is } \text{PO})\) then \((\Delta \delta_{gb} \text{ is ZE})\)

• If \((v \text{ is SM}) \text{ and } (d \text{ is SM}) \text{ and } (dv \text{ is } \text{ZE})\) then \((\Delta \delta_{gb} \text{ is NS})\)

• If \((v \text{ is LA}) \text{ and } (d \text{ is LA}) \text{ and } (dv \text{ is } \text{ZE})\) then \((\Delta \delta_{gb} \text{ is ZE})\)

• If \((v \text{ is ZE}) \text{ and } (d \text{ is ZE or SM}) \text{ and } (dv \text{ is } \text{ZE or NE})\) then \((\Delta \delta_{gb} \text{ is PS})\)

• If \((v \text{ is SM}) \text{ and } (d \text{ is ZE}) \text{ and } (dv \text{ is } \text{ZE or NE})\) then \((\Delta \delta_{gb} \text{ is NS})\)

• If \((v \text{ is ME}) \text{ and } (d \text{ is LA}) \text{ and } (dv \text{ is ZE or NE})\) then \((\Delta \delta_{gb} \text{ is PS})\)

• If \((v \text{ is ME}) \text{ and } (d \text{ is LA}) \text{ and } (dv \text{ is } \text{PO})\) then \((\Delta \delta_{gb} \text{ is ZE})\)

• If \((v \text{ is LA}) \text{ and } (d \text{ is LA}) \text{ and } (dv \text{ is } \text{NE})\) then \((\Delta \delta_{gb} \text{ is PS})\)

• If \((v \text{ is ZE or SM}) \text{ and } (d \text{ is LA}) \text{ and } (dv \text{ is } \text{ZE or PO or NE})\) then \((\Delta \delta_{gb} \text{ is PB})\)

• If \((v \text{ is ZE}) \text{ and } (d \text{ is ME}) \text{ and } (dv \text{ is } \text{ZE or PO or NE})\) then \((\Delta \delta_{gb} \text{ is PB})\)
• If \( v \) is SM and \( d \) is ME and \( \Delta v \) is ZE then \( \Delta \delta_{\text{th}} \) is ZE

• If \( v \) is SM and \( d \) is ME and \( \Delta v \) is PO then \( \Delta \delta_{\text{th}} \) is NS

• If \( v \) is SM and \( d \) is ME and \( \Delta v \) is NE then \( \Delta \delta_{\text{th}} \) is PS

• If \( v \) is ME and \( d \) is ME and \( \Delta v \) is ZE then \( \Delta \delta_{\text{th}} \) is ZE

• If \( v \) is ME and \( d \) is ME and \( \Delta v \) is PO then \( \Delta \delta_{\text{th}} \) is NM

• If \( v \) is ME and \( d \) is ME and \( \Delta v \) is NE then \( \Delta \delta_{\text{th}} \) is PS

• If \( v \) is LA and \( d \) is SM and \( \Delta v \) is ZE then \( \Delta \delta_{\text{th}} \) is NS

• If \( v \) is LA and \( d \) is ME and \( \Delta v \) is ZE or PO then \( \Delta \delta_{\text{th}} \) is NS

• If \( v \) is LA and \( d \) is ME and \( \Delta v \) is NE then \( \Delta \delta_{\text{th}} \) is ZE

Target Throttle Fuzzy Membership Sets

Looking at the membership sets for the speed \( v \), shown in Figure 23, the following is observed:

• ZE is from zero meters per second to two meters per second with its peak at zero meters per second

• SM is from zero meters per second to 10 meters per second with its peak between two and six meters per second
• ME is from five meters per second to 20 meters per second with its peak between 10 and 15 meters per second

• LA is from 15 meters per second to 25 meters per second with its peak between 20 and 25 meters per second

![Membership Sets for Input Variable v](image)

**Figure 23** Membership Sets for Input Variable v

It can be noted here that the sets for the fuzzy variable \( v \) are only defined for positive values of velocity. This indicates the limitation of the current controller to process negative velocities. It is also useful to note the upper bound on the LA membership set: this indicates that the maximum speed of the vehicle is 25 meters per second. Looking at the membership sets for the variable \( d \), shown in Figure 24, the following is observed:

• ZE is from zero meters to five meters with its peak at zero meters
• SM is from five meters to 25 meters with its peak between 10 and 20 meters

• ME is from 25 to 200 meters with its peak between 30 and 150 meters

• LA is from 100 meters to 500 meters with its peak between 175 and 500 meters

The medium set extends to 200 meters (with a more gentle slope on the right side) because the speed in the LA set can be up to 25 meters per second (90 kilometers per hour). Since this is a totally autonomous control, the speed controllers are designed to be conservative: thus, the reduction in speed due to the distance to the target begins early, to avoid the potential for collision. Looking at the membership sets for the variable \( \Delta v \), shown in Figure 25, the following is observed:
- ZE is from zero meters per second to 0.01 meters per second with its peak at zero meters per second

- PO is from zero meters per second to 0.1 meters per second with its peak between 0.01 and 0.1 meters per second

**Figure 25** Membership Sets for Input Variable Δv

The only information that needs to be obtained from this variable is whether the speed is constant, increasing, or decreasing. As such, the membership sets are almost crisp.

Finally, the membership sets for the variable Δδ_{θh}, shown in Figure 26, exhibit the following characteristics:

- NB is from -12 degrees to negative eight degrees and is relatively crisp
• NM is from negative eight degrees to negative three degrees and is relatively crisp
• NS is from negative three degrees to -0.5 degrees and is relatively crisp
• ZE is from -0.5 degrees to 0.5 degrees with its peak at zero degrees
• PS is from 0.5 degrees to two degrees and is relatively crisp
• PB is from two degrees to six degrees and is relatively crisp

Figure 26 Membership Sets for Output Variable $\Delta \delta_{\text{eq}}$

Here it is useful to note that the membership sets of the variable $\Delta \delta_{\text{eq}}$ are not symmetric around zero. This is indicative of the fact that the vehicle needs to be able to stop in less time than it takes to accelerate. It is also shown that there are more negative sets than positive ones. This allows finer control over braking than accelerating.
Cornering Throttle Fuzzy Module

The objective of this module is to slow down the vehicle when it is turning. The inputs to this module are the speed of the vehicle \( v \), the radius of curvature of the vehicle's path \( \rho \), and the change of speed from the previous measured value \( \Delta v \). The module has one output, which is the change in the gas pedal/brake angle \( \Delta \delta_{gb} \).

Cornering Throttle Fuzzy Rules

The goals of this module are as follows:

- To modify the velocity of the vehicle based on the current velocity of the vehicle.
- To increase the speed of the vehicle if the radius of curvature is very large and decrease the speed of the vehicle as the radius of curvature decreases.
- To modify the velocity of the vehicle based on the change in velocity (acceleration) of the vehicle.

Using several of the rules to demonstrate the enactment of the goals:

- If \( (v \text{ is ME}) \text{ and } (\rho \text{ is ZE}) \text{ and } (\Delta v \text{ is } Z) \) then \( (\Delta \delta_{gb} \text{ is NS}) \): This rule says that if the velocity is 16 kilometers per hour and the vehicle is in a very tight turn and the acceleration is constant then slow the vehicle down.
• If \( (\rho \text{ is LA}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \): This rule says that if the radius of curvature is infinite (i.e., the vehicle is driving along a straight path) then do not attempt to change the velocity.

All of the rules of this module are shown below:

• If \( (v \text{ is SM}) \) and \( (\rho \text{ is ZE}) \) and \( (\Delta v \text{ is P}) \) then \( (\Delta \phi_{gh} \text{ is NS}) \)

• If \( (v \text{ is ME}) \) and \( (\rho \text{ is ZE}) \) and \( (\Delta v \text{ is Z or P}) \) then \( (\Delta \phi_{gh} \text{ is NS}) \)

• If \( (v \text{ is LA}) \) and \( (\rho \text{ is ZE}) \) and \( (\Delta v \text{ is N or Z or P}) \) then \( (\Delta \phi_{gh} \text{ is NS}) \)

• If \( (v \text{ is LA}) \) and \( (\rho \text{ is SM}) \) and \( (\Delta v \text{ is P}) \) then \( (\Delta \phi_{gh} \text{ is NS}) \)

• If \( (v \text{ is ZE}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (\rho \text{ is LA}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (v \text{ is SM}) \) and \( (\rho \text{ is ZE}) \) and \( (\Delta v \text{ is N or Z}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (v \text{ is SM}) \) and \( (\rho \text{ is SM}) \) and \( (\Delta v \text{ is Z or P}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (v \text{ is ME}) \) and \( (\rho \text{ is ZE}) \) and \( (\Delta v \text{ is N}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (v \text{ is ME}) \) and \( (\rho \text{ is SM}) \) and \( (\Delta v \text{ is N or Z or P}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (v \text{ is LA}) \) and \( (\rho \text{ is SM}) \) and \( (\Delta v \text{ is N or Z}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)

• If \( (v \text{ is SM}) \) and \( (\rho \text{ is SM}) \) and \( (\Delta v \text{ is N}) \) then \( (\Delta \phi_{gh} \text{ is ZE}) \)
Cornering Throttle Fuzzy Membership Sets

The membership sets of $v$, $\Delta v$, and $\Delta \delta_{\theta}$ are the same as used in the previous module, and are shown in Figures 27, 28, and 29.

![Membership Sets for Input Variable $v$](image)

**Figure 27** Membership Sets for Input Variable $v$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 28 Membership Sets for Input Variable $\Delta v$
Looking at the membership sets for the radius of curvature $\rho$, shown in Figure 30, the following can be observed:

- **ZE** is from zero meters to five meters with its peak at zero meters
- **SM** is from five meters to 10 meters with its peak at 7.5 meters
- **LA** is from 10 meters to 100 meters (relatively infinity) and is relatively crisp
Figure 30 Membership Sets for Input Variable $\rho$

**Collision Avoidance Throttle Fuzzy Module**

This module is necessary to ensure slowing down of the vehicle as it navigates near obstacles. The inputs to this module are the vehicle speed $v$, distance to obstacle $d$, and the change of speed from the previous measured value $\Delta v$. The output of the module is the change in the gas pedal/brake angle $\Delta \delta_{gb}$.

**Collision Avoidance Throttle Fuzzy Rules**

The goals of this module are as follows:
• To change the velocity based on the current value of velocity.

• The increase the speed if there are no obstacles in the vicinity of the vehicle and decrease the speed if there are obstacles in the vicinity of the vehicle.

• To change the velocity based on the current value of acceleration.

Looking at a few of the rules to examine the implementation of the goals:

• If \((d_o \text{ is LA})\) then \((\Delta \delta_{gh} \text{ is ZE})\): This rule says that if the distance to the obstacle is large then do not modify the velocity.

• If \((v \text{ is ZE})\) and \((d_o \text{ is ZE})\) and \((\Delta v \text{ is ZE})\) then \((\Delta \delta_{gh} \text{ is PS})\): This rule says that if the velocity is zero and the distance to the obstacle is zero and the acceleration is zero then speed up the vehicle, but only slightly.

• If \((v \text{ is LA})\) and \((d_o \text{ is SM})\) and \((\Delta v \text{ is ZE})\) then \((\Delta \delta_{gh} \text{ is NS})\): This rule says that if the velocity is between 64 and 80 kilometers per hour and the obstacle is between 3 and 20 meters away and the acceleration is zero then slow down the vehicle.

The full rule set of this module is shown below:

• If \((v \text{ is ME or LA})\) and \((d_o \text{ is ZE})\) and \((\Delta v \text{ is ZE or PO})\) then \((\Delta \delta_{gh} \text{ is NL})\)

• If \((v \text{ is ME})\) and \((d_o \text{ is SM})\) and \((\Delta v \text{ is PO})\) then \((\Delta \delta_{gh} \text{ is NL})\)

• If \((v \text{ is ZE or SM})\) and \((d_o \text{ is ZE})\) and \((\Delta v \text{ is PO})\) then \((\Delta \delta_{gh} \text{ is NS})\)

• If \((v \text{ is SM})\) and \((d_o \text{ is SM})\) and \((\Delta v \text{ is NE or ZE})\) then \((\Delta \delta_{gh} \text{ is ZE})\)

• If \((v \text{ is SM})\) and \((d_o \text{ is SM})\) and \((\Delta v \text{ is PO})\) then \((\Delta \delta_{gh} \text{ is NS})\)
- If \( v \) is ME and \( d_o \) is ZE and \( \Delta v \) is NE then \( \Delta \delta_{sh} \) is NS

- If \( d_o \) is LA then \( \Delta \delta_{sh} \) is ZE

- If \( v \) is ZE and \( d_o \) is SM and \( \Delta v \) is PO then \( \Delta \delta_{sh} \) is ZE

- If \( v \) is ZE and \( d_o \) is ZE or SM and \( \Delta v \) is NE or ZE then \( \Delta \delta_{sh} \) is PS

- If \( v \) is SM and \( d_o \) is ZE and \( \Delta v \) is NE or ZE then \( \Delta \delta_{sh} \) is PS

- If \( v \) is ME and \( d_o \) is SM and \( \Delta v \) is ZE then \( \Delta \delta_{sh} \) is NS

- If \( v \) is ME and \( d_o \) is SM and \( \Delta v \) is NE then \( \Delta \delta_{sh} \) is ZE

- If \( v \) is LA and \( d_o \) is ZE and \( \Delta v \) is NE then \( \Delta \delta_{sh} \) is NS

- If \( v \) is LA and \( d_o \) is SM and \( \Delta v \) is NE or ZE then \( \Delta \delta_{sh} \) is NS

- If \( v \) is LA and \( d_o \) is SM and \( \Delta v \) is PO then \( \Delta \delta_{sh} \) is NL

Collision Avoidance Throttle Fuzzy Membership Sets

The membership sets for \( v \), \( \Delta v \), and \( \Delta \delta_{sh} \) are the same as in the previous module, and are shown in Figures 31, 32, and 33.
Figure 31 Membership Sets for Input Variable $v$
Figure 32 Membership Sets for the Input Variable $\Delta v$
Figure 33 Membership Sets for Output Variable $\Delta \delta_{ch}$

Observing the membership sets for the input variable distance to the obstacle $d_{in}$ shown in Figure 34, the following can be observed:

- **ZE** is from zero meters to two meters with its peak at zero meters
- **SM** is from two meters to 20 meters and is relatively crisp
- **LA** is from 20 meters to 100 meters and is relatively crisp
Figure 34 Membership Sets for Input Variable $d_o$.
CHAPTER 6

SIMULATION EXAMPLES

To check the validity of the ideas proposed in this thesis, the fuzzy controller was used to guide the vehicle around obstacles. The MATLAB program was run with different simulation configurations. Each example configuration was chosen to display certain characteristics of the controller or to contrast various parts of the controller.

Primary versus Bug Steering Control

The first three examples presented are designed to show the performance differences between the standard steering controller and the Bug controller that was implemented. The conditions for all of the steering control examples are an initial position of (0, 0), an initial orientation of 0 degrees, a final position of (175, 0), and an unspecified final orientation. The difference between the examples shown is the control scheme. Figure 35 shows the results using the Bug module to control the vehicle. The times in Figure 35 are $t_0 = 0$ seconds, $t_1 = 15$ seconds, $t_2 = 25$ seconds, and $t_3 = 100$ seconds.
As the Bug module is a backup system, and is not typically active, the execution of this simulation required the Bug module to be artificially activated, so that it was active from the very beginning. It is useful to note here that the path of the vehicle in this example follows the steps outlined in the previous chapter regarding the Bug module. To reiterate:

- Initially, the Bug algorithm will guide the vehicle straight toward the target.
- If the vehicle gets to within 10 meters of an obstacle, the vehicle will turn to the right, placing the obstacle on the left of the vehicle.
• The vehicle will proceed to follow the boundary of the obstacle, always keeping the obstacle on its left.

• The vehicle will stay on the one side of the ST line. This assures that the vehicle will not get “lost” (make any more 360 degree turns) again. If the vehicle approaches the ST line, it treats it like an obstacle, turning to the right and keeping the line on its left.

Once the target is within “sight” of the vehicle and the vehicle is within 20 meters of the target, the vehicle will leave the obstacle and go straight to the target.

The next simulation, illustrated in Figure 36, demonstrates the abilities of the primary steering controller. The times in the plot are $t_0 = 0$ seconds, $t_1 = 15$ seconds, $t_2 = 25$ seconds, and $t_3 = 75$ seconds.
Figure 36 Results Using Primary Steering Control

Notice that the primary controller is quite efficient, as it never allows the vehicle to drive among the obstacles, which would produce a longer path. An important aspect of this is the distance that the controller keeps between the vehicle and the obstacle. For this case, the controller has a 20 meter buffer zone. Comparing the Bug steering module to the primary steering control is useful here. The main modification that allows the Bug control to behave properly is the fact that it tries to keep the vehicle 3 meters away from the target. Since the primary control keeps the buffer radius at 20 meters, and the obstacles are substantially less than 40 meters apart, the vehicle is never allowed to enter between the obstacles.
To allow for this difference, one final simulation was performed, using the primary controller algorithm, but reducing the buffer radius to 5 meters. The times in Figure 37 are $t_0 = 0$ seconds, $t_1 = 15$ seconds, $t_2 = 25$ seconds, and $t_3 = 89$ seconds.

![Figure 37 Results Using Primary Steering Controller with Five Meter Buffer Zone](image)

Note that this time, the primary controller does direct the vehicle to drive between the obstacles. However, the controller produces a shorter path length than the Bug module, and finishes the trip in about 10 percent less time. Thus, given a comparable buffer zone, the primary fuzzy control, which was designed based on human experience, seems to have the same order of robustness as the bug control.
General Example of Full Algorithm

The next two simulations involve various features. The common conditions are an initial position is (0. 0), an initial orientation of 0 degrees, a final position of (100. 60). The first example here has a final orientation that is dependent upon the direction of approach. Also, the static obstacles in both cases are identical. The times for the first simulation, shown in Figures 38 and 39, are $t_0 = 0$ seconds, $t_1 = 10$ seconds, $t_2 = 15$ seconds, $t_3 = 20$ seconds, $t_4 = 37.5$ seconds, and $t_5 = 75.75$ seconds.
Figure 38 Position Results of the First General Example
Figure 39 Velocity Results of the First General Example

The position plot shows that the controller keeps the vehicle 20 meters from the obstacle at all times, with little variance. It can also be seen that the controller does not specify the final orientation. The velocity plot shows the velocity versus time. At times $t_1$, $t_2$, and $t_3$, the vehicle's speed is in the 5 meters per second range. This is because the sensors are detecting obstacles at the edge of the buffer zone. Even though the steering control is successful in keeping the appropriate distance between the vehicle and the obstacle, the throttle controller senses the proximity of an obstacle, and thus keeps the speed down. At time $t_4$, the speed is being decreased even more as the vehicle is in a turn and as it nears
the target. Finally, the velocity drops below 1 meter per second, due to the fact that the vehicle is quite near (less than 20 meters) the target.

The second general case simulation involves several variations from the first case. The first difference is that the final orientation is specified to be 90 degrees. Also, an additional obstacle has been added, but it is dynamic, rather than static. The times for the second simulation, shown in Figures 40 and 41, are $t_0 = 0$ seconds, $t_1 = 10$ seconds, $t_2 = 15$ seconds, $t_3 = 20$ seconds, $t_4 = 45$ seconds, and $t_5 = 99.5$ seconds.

![Figure 40 Position Results of the Second General Example](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 41 Velocity Results of the Second General Example

The position plot shows quite different behavior from the first general case. Notice that the controller keeps the same amount of space on either side of the vehicle, thus maximizing the buffer zone to the left and the right of the vehicle simultaneously, despite the fact that it can not achieve the desired 20 meter buffer zone. After obstacles no longer surround the vehicle (on both left and right sides), it returns to its default behavior of keeping 20 meters between the vehicle and the nearest obstacle. As opposed to the first case, the controller now does specify the final orientation, which is set to be 90 degrees. As can be seen, the vehicle is pointing straight up by the time it reaches the target. The velocity plot for the second case is reasonably similar to the first. Again, at times $t_1$, $t_2$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
and $t_1$, the vehicle's speed is in the 5 meters per second range. However, in this case, the velocity is decreasing. This is due to the greater intrusion of obstacles into the buffer zone and on both sides of the vehicle. As in the first case, at time $t_1$, the speed is being decreased due to the turning radius and the closeness of the vehicle to the target. Finally, the velocity drops below 1 meter per second, due to the fact that the vehicle is quite near (less than 20 meters) the target, just as in the first case.

**Maze Tracking Case using Bug Steering Module**

The final example case to be presented in this report will be on which will demonstrate the backup steering capabilities. The conditions for the simulation are an initial position of (0, 0), an initial orientation of 0 degrees, a final position of (100, 100) and a final orientation of 0 degrees. The results for this example are shown in Figures 42 and 43. The times depicted in the figures are $t_0 = 0$ seconds, $t_1 = 15$ seconds, $t_2 = 25$ seconds, $t_3 = 75$ seconds, $t_4 = 140$ seconds, and $t_5 = 145$ seconds. $t_6 = 170$ seconds, $t_7 = 198$ seconds, and $t_8 = 217.8$ seconds.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 42 Position Result of Maze Tracking Example
Initially, when there are no obstacles in its vicinity, the vehicle attempts to go straight to the target. Once it gets to the closest obstacle, it goes to the right, attempting to avoid it. When the vehicle senses an opening in the obstacle, it attempts to enter; however, as the opening is only 10 meters wide, the controller will not allow the vehicle to enter due to the 20 meter buffer zone. The vehicle then continues around the obstacles, attempting to find a clear path to the target. Once its orientation has deviated from the ST angle by 360 degrees, the Bug module is triggered. This occurs between times $t_1$ and $t_2$. At this point, again, the vehicle attempts to go straight to the target. Once it senses the obstacle, it turns right, keeping the obstacle at a distance of about 3 meters to the left of the vehicle. It
follows the wall of the maze until about t. At this point, the vehicle is within 20 meters of the target and the vehicle can see the target. Also, the final orientation control switches on here. Thus, the vehicle goes as straight to the target as possible while maintaining a course that will result in a final orientation of 0 degrees. Initially, the velocity plot is almost a straight line upward. As the vehicle sees no obstacles, it attempts to reach its maximum allowed speed (about 25 meters per second). This is cut short as the vehicle detects the first obstacle in its path. While traversing about the boundary of the maze and with a finite turning radius, the vehicle settles to a speed of about 2.5 meters per second. At t, the controller attempts to increase the velocity again, as there are no obstacles within 5 meters of the vehicle. But while tracking the maze, and with obstacles only 3 meters away, the final speed settles to just over 1 meter per second.
CHAPTER 7

CONCLUSIONS AND DISCUSSION

This paper presents a fuzzy logic control system for a two-axle vehicle. The dynamic model of the vehicle incorporates some nonlinearities to ensure a sufficiently realistic simulation. The fuzzy controller is broken into several modules that represent the distributed way in which humans deal with driving tasks. These modules are grouped into steering tasks and throttle tasks.

The steering tasks, presented in Chapter 4, include target steering, collision avoidance steering, and bug steering. The target steering is designed to steer the vehicle toward the target. The collision avoidance steering is designed to make the vehicle avoid collisions with static and dynamic obstacles. As the total steering angle is a summation of the output of each steering module, the collision avoidance steering is given a higher weight than the target steering, so that when the vehicle is near an obstacle, the collision avoidance steering will be able to significantly affect the behavior of the vehicle. The Bug steering module is meant to replace the target steering module in the event that it becomes confused and can not guide the vehicle to the target. Lumelsky used Bug2 as an
exclusive path planning tool, while Sauerberger used a modified Bug2 as a primary path planner and the original Bug2 as the backup. This controller uses its Bug module exclusively as a backup system. The criterion used to turn on the Bug module on is based on Sauerberger and consists of checking the vehicle's global orientation and making sure that it does not change by more than 360 degrees. The criteria used to turn the Bug module off are partially modeled after the criteria used by Sauerberger. They consist of 1) making sure that the vehicle is within 20 meters of the target, and 2) making sure that the vehicle has a "line of sight" to the target. Finally, the Bug module in this research accounts for the physical properties of a passenger vehicle, allowing it enough time and space to make the necessary maneuvers. Both Lumelsky and Sauerberger assume omnidirectional vehicles, which perform maneuvers that would be impossible for a passenger vehicle.

The throttle tasks are presented in Chapter 4. The throttle tasks include target throttle, cornering throttle, and collision avoidance throttle. The total throttle angle is a function of the summation of the output of each of the modules: again, the collision avoidance throttle has a scaling factor to allow its influence to be easily adjusted.

Simulation shows that the fuzzy controller successfully guides the vehicle in the proper way toward the target, while avoiding all obstacles placed in its path and keeping the speed in the desired range. Although the velocity values obtained may seem small, they are that way intentionally: a very conservative approach was taken in this regard. The logic is "slower is safer". The structure of fuzzy logic control (i.e., rules and membership sets, linguistically defined and based on human experience) makes the
design of the control system much more convenient than it would be if conventional control methods were being used, which justifies the use of fuzzy logic.

Several tasks were left out of this controller, such as the ability of the controller to back the vehicle up, i.e., to have a negative local x velocity. Being able to back up could be a very useful tool in the ventures of maze traversal and final orientation. In real life, a person may put his car into reverse in various situations, and that capability is very useful. Another improvement that could be implemented is adjustment of the Bug fuzzy module. Even though it performs well in the environment assumed in this research, modifications that would allow it to perform more robustly in a wider variety of situations would be useful. In general, the manual tuning performed on the controller yielded quite good results. However, occasionally, precise tuning of a fuzzy controller is sometimes difficult to do manually. This is the case with respect to the target throttle fuzzy module. The oscillatory nature of certain parts of the velocity response would indicate that a more systematic approach is necessary. Thus, a more methodical optimization of the fuzzy modules presented here should be explored. Although this could involve optimization of the rules, the membership sets, or both, it seems that systematic modification of the membership sets holds the most promise. Some type of self-adjusting algorithm, i.e., a neural network, may be the most effective route to achieving this goal. Finally, the control algorithm presented here should eventually be implemented experimentally to determine its practical viability.

The research presented in this paper successfully implemented a fuzzy logic controller in conjunction with a model that represented a two-axle ground vehicle. This
system could be a stepping stone to the eventual practical implementation of a fully autonomous vehicle, including all of the capabilities described herein, and others, including backing up and self tuning of the controller.
APPENDIX I

DERIVATION OF EQUATIONS OF MOTION

The purpose of this appendix is to present the complete derivations of the equations of motion of the vehicle. To begin the derivation, a force balance will be done in the \( x \) and \( y \) directions and a moment balance will be done in the \( \theta \) direction. The expressions that result are

\[
\sum F_x = F_{u_x} \cos \delta_x - F_{v_x} \sin \delta_x + F_{w_x} = ma_x, \tag{20}
\]

\[
\sum F_y = F_{u_y} \sin \delta_x + F_{v_y} \cos \delta_x + F_{w_y} = ma_y, \tag{21}
\]

\[
\sum M_z = -F_{w_z} l_z + (F_{u_z} \sin \delta_x + F_{v_z} \cos \delta_x) l_z = I_z a_z, \tag{22}
\]

The previous expressions have mass, acceleration, and force terms. The mass is constant, no matter the frame of reference. The force/moment terms were taken in terms of the reference frame attached on the car. The acceleration, however, is absolute, i.e. with respect to a fixed frame of reference. Thus, to get analogous terms, the global accelerations need to be found in terms of the frame of reference attached to the car. A diagram depicting the velocity components at time steps \( t \) and \( t + \Delta t \) is shown in Figure 44.

98
The first step is to get the global acceleration terms \( a_x \) and \( a_y \) into a different form.

Thus,

\[
A_x = \frac{\Delta v_x}{\Delta t} \quad (23)
\]

\[
A_y = \frac{\Delta v_y}{\Delta t} \quad (24)
\]

Restating the numerators of each of the above expressions yields the following
\[ a_x = \frac{\dot{x}(t + \Delta t) - \dot{x}(t)}{\Delta t} \]  

(25)

\[ a_y = \frac{\dot{y}(t + \Delta t) - \dot{y}(t)}{\Delta t} \]  

(26)

The next step shows the numerators of the fractions in terms of the velocities at time \( t \) and time \( t + \Delta t \) and the angle between the vehicle at the two times:

\[ a_x = \frac{((\dot{x} + \Delta \dot{x})\cos \Delta \theta - (\dot{y} + \Delta \dot{y})\sin \Delta \theta) - \dot{x}}{\Delta t} \]  

(27)

\[ a_y = \frac{((\dot{y} + \Delta \dot{y})\cos \Delta \theta + (\dot{x} + \Delta \dot{x})\sin \Delta \theta) - \dot{y}}{\Delta t} \]  

(28)

Expanding out each of the above expressions yields the following:

\[ a_x = \frac{(\dot{x}\cos \Delta \theta + \Delta \dot{x}\cos \Delta \theta - \dot{y}\sin \Delta \theta + \Delta \dot{y}\sin \Delta \theta) - \dot{x}}{\Delta t} \]  

(29)

\[ a_y = \frac{(\dot{y}\cos \Delta \theta + \Delta \dot{y}\cos \Delta \theta + \dot{x}\sin \Delta \theta + \Delta \dot{x}\sin \Delta \theta) - \dot{y}}{\Delta t} \]  

(30)

Assuming that \( \Delta \theta \) is very small (i.e., \( \cos \Delta \theta = 1 \) and \( \sin \Delta \theta = \Delta \theta \)) and that all of the second order terms are very small and can be neglected, the results are

\[ a_x = \frac{(\dot{x} + \Delta \dot{x} - \dot{y}\Delta \theta + \Delta \dot{y}\Delta \theta) - \dot{x}}{\Delta t} = \frac{(\Delta \dot{x} - \dot{y}\Delta \theta)}{\Delta t} = \ddot{x} - \dot{y}\dot{\theta} \]  

(31)

\[ a_y = \frac{(\dot{y} + \Delta \dot{y} + \dot{x}\Delta \theta + \Delta \dot{x}\Delta \theta) - \dot{y}}{\Delta t} = \frac{(\Delta \dot{y} + \dot{x}\Delta \theta)}{\Delta t} = \ddot{y} + \dot{x}\dot{\theta} \]  

(32)

Note that each of the above expressions has, in the end, two terms. For each one, the first is the linear term and the second is the rotational term. After finding the global
accelerations in terms of their linear and rotational components, the final form of the
equations of motion can be expressed as

\[ \begin{align*}
  m(\ddot{x} - \dot{y}\dot{\theta}) &= F_{u} \cos \delta_{u} - F_{u} \sin \delta_{u} + F_{u} \\
  m(\ddot{y} + \dot{x}\dot{\theta}) &= F_{u} \sin \delta_{u} + F_{u} \cos \delta_{u} + F_{u} \\
  I_{z}\ddot{\theta} &= -F_{u}L_{z} + (F_{u} \sin \delta_{u} + F_{u} \cos \delta_{u})L_{u}
\end{align*} \]  

(33) (34) (35)

The previous expressions yield the accelerations \( \ddot{x}, \ddot{y}, \) and \( \ddot{\theta} \). Integrating each of these
quantities twice yields the displacement in the local frame. The next appendix will derive
the transformation equations.
The purpose of this appendix is to present the complete derivations of the equations which transform the local displacements, calculated by the equations of motion, into global displacements. In general, the vehicle can move in the x', y', and θ' directions all at once. Note that the prime (') frame is the local frame. Also, in the actual simulation, the magnitude of each local displacement (x', y', and θ') is incrementally small. To begin, the reader can assume that the initial position of the car is (x₀, y₀, θ₀) (i.e., the position before the initial time step). So, in the global frame, before the time step,

\[
\begin{bmatrix}
x \\ y 
\end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}
\]

(36)

After the time step, the global position is a function of the initial position and the change in position during the time step. Thus,

\[
\begin{bmatrix}
x \\ y 
\end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x_{step} \\ y_{step} \end{bmatrix}
\]

(37)

where \( x_{step} \) and \( y_{step} \) represent the local displacements which occur during the time step that have been transformed into the global frame. The next step is to develop the
expressions that convert the local displacements to global displacements. The expressions are

\[
\begin{align*}
\begin{bmatrix} x_{\text{wp}} \\ y_{\text{wp}} \end{bmatrix} &= \begin{bmatrix} x' \cos \theta_n - y' \sin \theta_n \\ x' \sin \theta_n + y' \cos \theta_n \end{bmatrix} \\
(38)
\end{align*}
\]

where \( x' \) and \( y' \) are the local displacements during the first time step and \( \theta_n \) is the global angle of the vehicle before the first time step, which corresponds to the angle at the end of the previous (zero) time step. Now, substituting equation 38 into equation 37 yields

\[
\begin{align*}
\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} \cos \theta_n & -\sin \theta_n \\ \sin \theta_n & \cos \theta_n \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \\
(39)
\end{align*}
\]

The final step is to convert the term for use for all time. The expression is

\[
\begin{align*}
\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \sum_{i=1}^{n} \begin{bmatrix} \cos \theta_{i-1} & -\sin \theta_{i-1} \\ \sin \theta_{i-1} & \cos \theta_{i-1} \end{bmatrix} \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} \\
(40)
\end{align*}
\]

where \((x, y)\) is the global position, \((x_n, y_n)\) is the initial position, \( x'_i \) and \( y'_i \) are the local displacements at time step \( i \) and \( \theta_{i-1} \) is the global angular displacement at time step \( i-1 \). and \( i \) is the index for the current time step, which goes from 1 to \( n \). where

\[
n = \frac{\text{total elapsed time}}{\Delta t} \\
(41)
\]

For the simulations run in this research project, \( \Delta t = 0.01 \) seconds. The second part of this derivation is to find the expression that describes the total angular position. Before the time step in the figure, the angular position of the vehicle is

\[
\theta = \theta_n \\
(42)
\]
Again, after the time step, the global angular position is a function of the initial position and the change in angular position during the time step. Thus,

$$\theta = \theta_i + \theta_{\text{step}}$$  \hspace{1cm} (43)

where $\theta_{\text{step}}$ represents the local angular displacement that occurred during the time step that has been transformed into the global frame. The next step is to convert the local angular displacement to the appropriate global angular displacement. The direction of $\theta_{\text{step}}$ will be in the same direction as the steering angle of the front wheels. Thus

$$\theta_{\text{step}} = \text{sign}(\delta_{\text{step}}) \|\theta_{\text{step}}\|$$  \hspace{1cm} (44)

where the function $\text{sign}(x)$ returns the sign of $x$ in the form of 1 if $x>0$, -1 if $x<0$, and 0 if $x=0$, and $\theta'$ is the angular displacement in the local frame. The next step is to find the magnitude of $\theta'$. Given the fact that each incremental displacement will be very small, the magnitude of the angle $\theta'$ can be represented using the following relation:

$$s = \rho \theta'$$  \hspace{1cm} (45)

Thus,

$$\theta_{\text{step}} = \text{sign}(\delta_{\text{step}}) \frac{s_i}{\rho_{i}}$$  \hspace{1cm} (46)

where $s$ is the arc length and $\rho$ is the radius of curvature during the first time step.

Expressions must now be found for $s$ and $\rho$ in terms of the local displacements. Using the approximation that all of the quantities involved are very small, the arc length can be approximated as a straight line. Thus.
and $\rho$, the radius of curvature, is calculated as

$$\theta_{\text{up}} = \text{sign}(\delta_{11}) \left| \frac{\sqrt{x'^2 + y'^2}}{\rho} \right|$$  \hspace{1cm} (47)

Since both the numerator and denominator are the square root of squares, the value of the magnitude is always positive. The quantities in the previous expression are all shown in Figure 45.
The total expression to calculate $\theta$ for the first time step is

$$
\theta = \theta_0 + \left( \text{sign}(\delta_{i_1}) \right) \frac{\sqrt{x_{i_1}^2 + y_{i_1}^2}}{\sqrt{L_z^2 + \left( \frac{L_z}{\tan(\delta_{i_1})} \right)^2}}
$$

(49)

Just as before, $x'$ and $y'$ are the local displacements during this time step, as is $\delta_i$. Again, the final step is to convert equation 49 for use for all time. The expression is...
\[ \theta = \theta_0 + \sum_{i=1}^{i=n} \left( \text{sign}(\delta_i, \frac{\sqrt{x_i^2 + y_i^2}}{L_i + \left( \frac{L_{z_i}}{\tan(\delta_i)} \right)} \right) \]  

where \( i \) and \( n \) are the same as previously defined.
APPENDIX III

SCHEMATIC OF SIMULINK MODEL

The purpose of this appendix is to graphically present the structure of the Simulink model created to test the ideas presented in this report. The following series of block diagrams display how the Simulink model was organized. All user created files that were called that were external to the Simulink file (*.m and *.fis files) are shown in square brackets.

![Block Diagram of the Main Program](image)

**Figure 46** Block Diagram of the Main Program
Figure 47 Block Diagram of the System Model Subprogram

Figure 48 Block Diagram of the Fuzzy Steering Subprogram
Figure 49 Block Diagram of the Fuzzy Throttle Subprogram
REFERENCES


1996.


VITA

Graduate College
University of Nevada, Las Vegas

Neil Eugene Hodge

Local Address:
2836 Belmont Drive
Henderson, NV 89014

Home Address:
2210 Maple Avenue #102
Evanston, IL 60201

Degrees:
Bachelor of Science, Mechanical Engineering, 1995
University of Nevada, Las Vegas

Special Honors and Awards:
NASA Scholarship, 1993 and 1994
Alumni Scholarship, 1996 and 1997

Publications:
N. Hodge and M. Trabia. "Navigational Fuzzy Logic Control of an Autonomous Vehicle." International Association of Science and Technology for Development International Conference on Control, May 1997

Thesis Title: Navigational Fuzzy Logic Control of an Autonomous Vehicle

115
Thesis Examination Committee:
   Chairman. Dr. Mohamed Trabia. Ph.D.
   Committee Member. Dr. Woosoon Yim. Ph.D.
   Committee Member. Dr. Robert Boehm. Ph.D.
   Graduate Faculty Representative. Dr. Sahjendra Singh. Ph.D.