Successful autonomous mobile robot navigation requires controllers and navigation algorithms that are capable of navigating complex unstructured environments. These control algorithms must be able to successfully avoid densely spaced obstacles while traversing a relatively efficient but safe path. Navigation through a forest attracts special interest from the military community due to the lack of stealth and concealment that is evident when moving in open environments. Behavior based or reactive control systems address this task without the need for a priori information, which is often lacking or inaccurate. These systems deviate from the traditional Plan-Sense-Model-Act approach by eliminating modeling and attempting to make the most direct coupling between perception and action. The major components of behavior based systems are a set of independent parallel behaviors and an arbitration or fusion method for combining the behavioral responses to form a single control output. The arbitration problem is the most challenging to behavior based navigation, especially due to the increased demand of complex environments
that require the consideration of multiple behaviors, which may be conflicting. The major arbitration and fusion approaches are the subsumptive and the motor schemas approaches. Subsumptive systems switch between behaviors based on the situation while motor schemas based architectures combine behavior reactions with vector addition or a linear combination. The early approaches of behavior based systems used uni-valued behaviors in which behaviors respond to its stimulus by firing a single control command. Poor arbitration and fusion in these systems led to the development of multi-valued behavior systems in which each behavior responds to its stimulus by suggesting several control commands. Two approaches of multi-valued behavior systems have been proposed: the crisp logic approach and the fuzzy logic approach. The fuzzy logic approach is not only simple to program but also adds robustness to inaccurate sensor information and promotes smooth control when used properly.

This thesis presents the implementation of a multi-valued fuzzy behavior control system on a robotic vehicle. This algorithm was implemented on a Pioneer 2 mobile robot and tested in a forest like environment. During implementation, significant modifications were made to the original algorithm by redesigning the fuzzy rules to improve algorithm performance. The sensor reading and interpretation module was redesigned so that the laser range finder could be used instead of the initially assumed sonar sensors. Details of these modifications also are given in this thesis. In general these algorithmic modifications not only improved its navigation performance but also increased its overall robustness, reliability and computational efficiency.
IMPLEMENTATION OF MULTI-VALUED FUZZY BEHAVIOR
CONTROL FOR NAVIGATION IN CLUTTERED ENVIRONMENTS

By

DAMION D. DUNLAP

A thesis submitted to the
Department of Mechanical Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Summer Semester, 2004
The members of the Committee approve the thesis of Damion Donyell Dunlap defended on June 15th, 2004.

Emmanuel G. Collins, Jr.
Professor Directing thesis

Carl Moore, PhD
Committee Member

Dave Cartes, PhD
Committee Member

Majura Seleka, PhD
Committee Member

 Approved:

Chiang Shih, Chair
Department of Mechanical Engineering

Ching-Jen Chen, Dean, FAMU-FSU College of Engineering

Chanta Haywood, Dean, School of Graduate Studies and Research
To My Lovely Wife
I know it is probably traditional to thank your advisor first but I must first thank God for keeping me through this process and advising through the Holy Bible. I recognize that His strength is what has ultimately made this work possible.

I would like to express my sincere gratitude to my advisors Dr. Emmanuel Collins and Dr. Majura Seleka for guiding me through this process and for pushing me to fulfill the potential of this work. Dr. Collins your insight to the big picture was essential and I want to thank you for trusting me. This thesis would probably still be in my head or in a jumble of words if it had not been for the counsel of Dr. Seleka.

I must also thank Dr. Cartes and Dr. Moore for being committee members. Dr. Cartes is greatly appreciated for introducing me intelligent control and helping me to establish a firm foundation in this area while Dr. Moore’s personable manner has made him more than simply a professor.

I gratefully acknowledge the Army Research Laboratory and General Dynamics Robotic Systems for their support through the Collaborative Technical Alliance Program, Cooperative agreement DAAD 19-01-2-0012. Thanks must also be given to the Florida A&M University School of Graduate Studies and the GAANN fellowship for supporting my graduate studies.

Last for definitely far from least would like to thank my wonderful wife, Tracy Lavonne Dunlap whose support has lightened by burden and helped make the writing
process less stressful. I am grateful to my parents George Rufus Dunlap and Rose Seegars Dunlap for their unwavering support and for instilling in me, the importance of higher education.

I Love & Appreciate You All !!
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>Abstract</td>
<td>xii</td>
</tr>
<tr>
<td><strong>1. INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background of Robot Navigation</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Behavior Based Robotics</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 Behavior Based System Structure</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1.1 Command Arbitration</td>
<td>5</td>
</tr>
<tr>
<td>1.2.2 Design Approaches</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Behavior Based Architectures</td>
<td>8</td>
</tr>
<tr>
<td>1.3.1 Uni-valued Architectures</td>
<td>9</td>
</tr>
<tr>
<td>1.3.1.1 Subsumption Architectures</td>
<td>10</td>
</tr>
<tr>
<td>1.3.1.2 Motor Schemas Architectures</td>
<td>12</td>
</tr>
<tr>
<td>1.3.1.3 Uni-valued Fuzzy Behavior Architectures</td>
<td>13</td>
</tr>
<tr>
<td>1.3.2 Multi-valued Architectures</td>
<td>13</td>
</tr>
<tr>
<td>1.3.2.1 Planned Guided Reaction and the Distributed Architecture for Mobile Navigation (DAMN)</td>
<td>14</td>
</tr>
<tr>
<td>1.3.2.2 Multi-valued Fuzzy Behavior Architectures</td>
<td>15</td>
</tr>
<tr>
<td>1.4 Research Objectives and Thesis Outline</td>
<td>17</td>
</tr>
</tbody>
</table>
2. MULTI-VALUED FUZZY BEHAVIOR CONTROL FOR ROBOT NAVIGATION ........................................... 18
  2.1 Principle of operation of the proposed architecture ......................... 19
  2.2 The structure of the proposed architecture ................................. 20
    2.2.1 The heading control activity ........................................ 21

3. EXPERIMENTAL ACCOMMODATIONS .................................. 27
  3.1 Sensor Configuration .................................................. 28
  3.2 Fuzzy System Modifications ............................................ 30
    3.2.1 Command Block Elimination ....................................... 31
    3.2.2 Defuzzification Method ............................................ 33
    3.2.3 Limitations of Behavioral Influence ................................ 36
      3.2.3.1 Realm of Influence ........................................... 36
      3.2.3.2 Degree of Influence .......................................... 37
    3.2.4 Rule Base Determination .......................................... 38
      3.2.4.1 Relevant Input Information ................................... 39
      3.2.4.2 Visual Rule Determination ................................... 39
    3.2.5 Membership Function Shaping ..................................... 41
      3.2.5.1 Input Variable Terms ......................................... 41
      3.2.5.2 Output Variable Terms ....................................... 42
  3.3 From Discrete to Continuous: Command Action Execution .............. 44

4. EXPERIMENTAL PROCEDURE AND RESULTS ............................. 48
  4.1 Experimental Robotic Platform ......................................... 48
    4.1.1 Onboard Computer and Interfaces .................................. 49
    4.1.2 Range Sensors ....................................................... 50
    4.1.3 Localization Sensors ................................................ 51
4.2 fuzzyTECH .............................................................. 53
4.3 Experimental Procedure ......................................... 53
  4.3.1 Obstacle Description ........................................... 54
  4.3.2 Obstacle Configuration ........................................ 55
4.4 Results ................................................................. 55

5. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH ........................................... 63
  5.1 Conclusion ............................................................. 63
  5.2 Future Work .......................................................... 65

APPENDICES

A. FUZZYTECH DOCUMENTATION .................................... 67

B. C++ CODE ................................................................. 95
REFERENCES ............................................................... 104
BIOGRAPHICAL SKETCH ................................................ 107
# LIST OF FIGURES

1.1 (A) Classical AI (vertical) model. (B) Reactive (horizontal) model ... 4
1.2 Reactive Behavior Arbitration ......................................................... 5
1.3 Ethologically guided design approach .............................................. 6
1.4 Situational design approach ............................................................. 7
1.5 Experimentally driven design approach ............................................ 8
1.6 Uni-valued Behavior Architecture .................................................. 9
1.7 Layered Subsumptive Architecture ............................................... 10
1.8 Original AFSM used in the subsumptive architecture ....................... 11
1.9 Multi-valued Behavior Architecture ............................................... 14
2.1 The Proposed Fuzzy Architecture for Action Control ....................... 20
2.2 The Fuzzy Sets for the Heading ..................................................... 22
2.3 The Fuzzy Sets for the Measure of Relative Importance ..................... 22
2.4 Some of the Possible Obstacle Maps ............................................. 23
2.5 The Fuzzy Groups for the Measured Distances ................................. 24
2.6 Some of the possible right edge maps ........................................... 24
3.1 Sensor Configuration with undetected obstacle ............................... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Revised Sensor Region Configuration</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Revised Multi-valued Fuzzy Behavior Control Structure</td>
<td>32</td>
</tr>
<tr>
<td>3.4 Fuzzy output demonstrating the effect of compromise in traditional CoM</td>
<td>35</td>
</tr>
<tr>
<td>3.5 Turning away from goal required to avoid obstacles</td>
<td>38</td>
</tr>
<tr>
<td>3.6 Sensor Configuration with Input Term Overlay for VRD</td>
<td>40</td>
</tr>
<tr>
<td>3.7 Membership Functions for FL, L, LC, C, RC, R, and FR.</td>
<td>42</td>
</tr>
<tr>
<td>3.8 Membership Functions for VFL and VFR.</td>
<td>43</td>
</tr>
<tr>
<td>3.9 Membership Functions for $\Phi$.</td>
<td>43</td>
</tr>
<tr>
<td>3.10 Membership Functions for $\alpha_{14}, \alpha_{15}, \alpha_{23}, \alpha_{31},$ and $\alpha_{32}$.</td>
<td>44</td>
</tr>
<tr>
<td>3.11 Membership Functions for $\alpha_{22}, \alpha_{24},$ and $\alpha_{4j}, j = 1, 2 \ldots 5$.</td>
<td>44</td>
</tr>
<tr>
<td>3.12 Membership Functions for $\alpha_{13}$ and $\alpha_{33}$.</td>
<td>45</td>
</tr>
<tr>
<td>3.13 Membership Functions for $\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{25}, \alpha_{34},$ and $\alpha_{35}$.</td>
<td>45</td>
</tr>
<tr>
<td>3.14 Membership Functions for $\Delta \theta$.</td>
<td>46</td>
</tr>
<tr>
<td>4.1 Pioneer 2 Physical Dimensions and Swing Radius</td>
<td>49</td>
</tr>
<tr>
<td>4.2 Pioneer 2 With SICK Laser Range Finder</td>
<td>50</td>
</tr>
<tr>
<td>4.3 fuzzyTECH screen print in debug mode</td>
<td>54</td>
</tr>
<tr>
<td>4.4 Robot navigating through a dense obstacle field</td>
<td>56</td>
</tr>
<tr>
<td>4.5 Scenario 1</td>
<td>57</td>
</tr>
<tr>
<td>4.6 Scenario 2</td>
<td>58</td>
</tr>
<tr>
<td>4.7 Scenario 3</td>
<td>58</td>
</tr>
<tr>
<td>4.8 Scenario 4</td>
<td>59</td>
</tr>
<tr>
<td>4.9 Scenario 5</td>
<td>60</td>
</tr>
</tbody>
</table>
Successful autonomous mobile robot navigation requires controllers and navigation algorithms that are capable of navigating complex unstructured environments. These control algorithms must be able to successfully avoid densely spaced obstacles while traversing a relatively efficient but safe path. Navigation through a forest attracts special interest from the military community due to the lack of stealth and concealment that is evident when moving in open environments. Behavior based or reactive control systems address this task without the need for \textit{a priori} information, which is often lacking or inaccurate. These systems deviate from the traditional Plan-Sense-Model-Act approach by eliminating modeling and attempting to make the most direct coupling between perception and action. The major components of behavior based systems are a set of independent parallel behaviors and an arbitration or fusion method for combining the behavioral responses to form a single control output. The arbitration problem is the most challenging to behavior based navigation, especially due to the increased demand of complex environments that require the consideration of multiple behaviors, which may be conflicting. The major arbitration and fusion approaches are the subsumptive and the motor schemas approaches. Subsumptive systems switch between behaviors based on the situation while motor schemas based architectures combine behavior reactions with vector addition or a linear combination. The early approaches of behavior based systems used uni-valued behaviors in which behaviors respond to its stimulus by firing a
single control command. Poor arbitration and fusion in these systems led to the development of multi-valued behavior systems in which each behavior responds to its stimulus by suggesting several control commands. Two approaches of multi-valued behavior systems have been proposed: the crisp logic approach and the fuzzy logic approach. The fuzzy logic approach is not only simple to program but also adds robustness to inaccurate sensor information and promotes smooth control when used properly.

This thesis presents the implementation of a multi-valued fuzzy behavior control system on a robotic vehicle. This algorithm was implemented on a Pioneer 2 mobile robot and tested in a forest like environment. During implementation, significant modifications were made to the original algorithm by redesigning the fuzzy rules to improve algorithm performance. The sensor reading and interpretation module was redesigned so that the laser range finder could be used instead of the initially assumed sonar sensors. Details of these modifications also are given in this thesis. In general these algorithmic modifications not only improved its navigation performance but also increased its overall robustness, reliability and computational efficiency.
CHAPTER 1

INTRODUCTION

Recent research work in the autonomous mobile robot control seeks architectural solutions that provide performance robustness and reliability in a dynamic, uncertain and complex environment [29]. These architectures must provide a means to satisfy multiple, possibly conflicting goals subject to real time constraints. Additionally they must promote fault tolerance and provide for the safety of the robot vehicle and its surroundings. As the mobile robotics research advances from solutions laboratory scenarios towards real world 3-D, unstructured, uncertain, applications, the above-mentioned goals have become more and more desirable [5, 12]. On that front, behavior based systems have attracted more attention.

One of the current research challenges is in the action selection for behavior based systems, i.e., arbitration and command fusion. Two architectures have been proposed: one, which uses uni-valued behaviors and another that has multi-valued behaviors. Uni-valued fuzzy behavior systems have been implemented and tested on many platforms; however, the comprehensive fuzzy behavior system [27] has not
be implemented and tested. This thesis covers the modification done to improve the performance of the multi-valued fuzzy behavior system of [27] and its implementation on a Pioneer 2 robot with SICK laser range finder.

The chapter is organized as follows. Section 1.1 is an overview of robotic navigation while Section 1.2 addresses behavior based robotics. Several behavior based architectures are described in Section 1.3 including the architecture used for this implementation. The chapter is concluded with Section 1.4 which described the objectives of this research and outlines the remainder of this thesis.

1.1 Background of Robot Navigation

The field of robot navigation has developed along side the field of mobile robotics. Most of the first mobile robots addressed this problem with classical artificial intelligence (AI), a field whose birth is associated with the Dartmouth Summer Research Conference of 1955 [5]. Marvin Minsky is often seen as the father of this field [10] as he asserted that an intelligent machine would “build up within itself an abstract model of the environment in which it is placed [18].” This model would then be used to explore solutions to given problems including navigation. Classical AI attempted to recreate human cognition or thought by modeling and then planning before acting. This approach dominated robotics research for the next thirty years as robotic planning became increasingly dependant on a representative model and deliberate reasoning [5].
In 1987 at MIT Rodney Brooks asserted that cognition is a subjective fabrication of the observer and therefore can not be scientifically modeled and described planning as “just a way of avoiding figuring out what to do next [8].” Consequently behavior based robotics does not attempt to produce human cognition but to reproduce human behaviors and reactions. Brooks saw the value of animal behaviors that were completely ignored by the AI community in their preoccupation with human level intelligence. He recognized that animals, without cognition, are able to accomplish tasks well beyond the capabilities of mobile robotic technology of that time. Intellectual arguments concerning wether or not behaviors can be built up to produce human levels of intelligence dwindled when the focus of behavioral robotics shifted from producing systems that think intelligently to developing agents that can act intelligently. With the objectives more task oriented and less theoretical, behavior robotics aims to make the most direct link between perception and action [5] eliminating the environmental modeling process that is present in classical intelligent architectures.

1.2 Behavior Based Robotics

Behavior based robotics was introduced in [7] with one of the first well-formulated methodologies of the approach as an alternative to classical AI. Behavior based robotics break the navigation control problem into simple task-oriented units called reactive behaviors. Most of the early implementations of behavior based robotics used crisp logic. The growth of behavior based robotics has prompted the development
of various behavioral fuzzy methods that can handle the informational uncertainty common to real life robotic systems [3, 13, 17, 19, 22, 24, 25, 29]. One of the fundamental challenges facing the implementation of behavioral robotics is that of fusing the behavioral reactions when these reactions are conflicting.

1.2.1 Behavior Based System Structure

Reactive behaviors are control systems that make up the behavior based control system. The reactive behavior systems have proven to be very effective in accomplishing many of the complex tasks facing robotic systems today by decomposing these task into more simple well-defined subtasks. These behaviors can be implemented independently, which reduces behavioral interference and system complexity. This independence is due to their horizontal arrangement as illustrated in Figure 1.1(B) in comparison to the classical AI model approach in 1.1(A). These models are also referred to as parallel and serial respectively.

![Diagram of behavior based system structure](image)

**Figure 1.1.** (A) Classical AI (vertical) model. (B) Reactive (horizontal) model
1.2.1.1 Command Arbitration

Since reactive systems are comprised of independent behaviors, a mechanism is needed to determine one control command. This mechanism is known as command arbitration, which is a distinguishing characteristic as well as the toughest problem of reactive behavior systems. Figure 1.2 shows how each behavior sends information to the command fusion block which combines the information to create one output.

Various arbitration methods combine this information in fusion that makes the behaviors either compete or cooperate for control of the system. These methods will described in more detail in Section 1.3.

1.2.2 Design Approaches

Behaviors are the basic building block of reactive behavior systems and must be developed systematically. There are three prevalent approaches for designing reactive systems, which are ethologically guided, situational, and experimentally driven [5]. The chosen design approach is highly dependant on the design objectives as there is no universally favored method.
The ethologically guided approach is grounded in one of the foundational principles of behavior based robotics, with ethology being the study of animal behavior in natural environments. This approach uses available ethological information to develop a behavioral model, which is encoded into a robotic system and evaluated experimentally. The results of this evaluation are used to modify the behaviors as well as provide insight for biological experimentation as illustrated in Figure 1.3. These results benefit both the ethological and robotic research communities by providing a basis for the development of theories of animal behavior while constructing a basis for more intelligent robotic behavior.

The situational design approach is based on determining the appropriate response for each possible situation. These responses can be viewed as micro-behaviors since they follow the stimulus-reaction model, which can be combined to generate

Figure 1.3. Ethologically guided design approach
a behavioral structure. These behaviors, once encoded into the robotic system yield experimental results that lead to the improvement of the behavioral responses. Assuming situations can be unlimited, it is necessary to use sensory information to classify each situation to prevent the enumeration of every possible situation. This approach requires sufficient understanding of the interaction between the robot and its environment and sensors capable of providing situational awareness.

**Figure 1.4.** Situational design approach

In the experimentally driven design approach, behaviors are developed from the bottom-up as follows. A minimal system is exercised on the robot and the results lead
to system improvements. The existing behaviors are debugged and new behaviors are added to provide new competence or increased capabilities until the system meets design objectives. The design flow for this approach is illustrated in Figure 1.5 [5]. This methodology is highly dependant on real world experimentation as a simulation environment can not sufficiently model all the nuances of physical experimentation necessary for the success of this approach. Experimentally driven development enables relatively fast but incomplete preliminary results. These results can either show promise or invalidate the chosen behaviors as to avoid wasting research time on the development of complete systems that do not function properly.

![Experimentally driven design approach](image)

**Figure 1.5.** Experimentally driven design approach

### 1.3 Behavior Based Architectures

Implementation of behavior based control require determination of a means of expressing, encoding, and coordinating behaviors [5]. Hayes-Roth describes an
architecture as “... the abstract design of a class of agents: the set of structural components in which perception, reasoning, and action occur; the specific functionality and interface of each component, and the interconnection topology between the components [15].” Robotic architectures establish the foundation on which behaviors are constructed, defining the specific methods and software systems to be used. The main features of behavior-based architectures include their ability to directly couple perception and the control action, avoid of symbolic representation of the environment, and purposefully decompose the control objective into simple units (behaviors) [5]. Although behavior-based architectures follow the same general philosophy they differ greatly depending on their behavioral granularity (i.e. uni-valued or multi-valued), the design methodology used (i.e. ethological, situational, or experimental), and the command arbitration and fusion methods used [5]. Also implementation of behavior systems in dependant on the programming methods and software used.

1.3.1 Uni-valued Architectures

![Diagram of Uni-valued Behavior Architecture](image)

**Figure 1.6.** Uni-valued Behavior Architecture
Uni-valued behavior architectures contain behaviors that have singular responses; i.e., a single command is transmitted by each behavior for consideration in command arbitration as illustrated in Figure 1.6 [9]. Each behavior \( i, i = 1, 2, \ldots, n \) returns a command that will satisfy the purpose for which it was designed, based on the stimulus \( S_i \). For example, a \textit{go-to-goal} behavior may return a command to turn to a particular relative heading. The first approach to develop a single command for the whole system was Brooks subsumptive method. Following limitations of this approach, Arkin proposed the motor-schemas approach. These methods developed into their respective architectures in addition to their various fuzzy logic extensions that will be discussed in the next sections.

### 1.3.1.1 Subsumption Architectures

![Layered Subsumptive Architecture](image)

**Figure 1.7.** Layered Subsumptive Architecture

The task of choosing which behavior to activate was first handled by using the subsumption arbitration structure that layered behaviors from the bottom up and assigned priority to the behaviors, activating the higher priority behavior as seen in Figure 1.7 [7]. In the subsumption structure bottom-layer behaviors like collision avoidance are the most basic and have the highest priority while top-level behaviors
like goal seeking would only be activated when lower-layer behaviors are satisfied. Each behavior is represented by an augmented finite state machine (AFSM), shown in Figure 1.8. The AFSMs are arranged in the layered structure and use the mechanisms of inhibition and suppression to influence other behaviors and the control output. Suppression prevents an input stimulus from reaching behavioral module while inhibition prevents the behavioral reaction from effecting the control output and a reset mechanism restores the behavioral module to its original state. These mechanisms are used to implement a priority based system of behavior competition. Subsumptive architectures are typically developed experimentally using rule based behavior response encoding and AFSMs as a programming method although newer methods use Behavior Language [5].

Although subsumptive architectures enable behavioral layers to run independently, they do not support modularity. Upper layers interfere with lower ones and can not be designed independently which is a major downfall [14]. The subsumptive structure also forces prioritization which can lead to artificial arbitration schemes [14]. For example, right and left edge tracking behaviors would have to be given
different priorities making the control arbitrarily prefer one direction over another. This strict prioritization is necessary because the subsumptive architecture does not have a conflict resolution mechanism for behaviors with equal priority.

1.3.1.2 Motor Schemas Architectures

Schemas have been used to model human behaviors since the eighteenth century; they have been used in neurophysiological and psychology theories by modeling the interaction between the brain and the mind [5]. Norman and Shallice [20] used schemas to adaptively switch between cooperating and competing behaviors while Arbib [4] was the first to apply schema theory to the field of robotics. Arbib defines schemas as “an adaptive controller that uses an identification procedure to update its representation of the object being controlled [4].” Schemas are very similar to neural networks but allow robotic behaviors to be encoded at a “courser granularity,” meaning that schemas have a larger range of competence than a neural network [5].

The development of schema vector-based motion planning strategies and vector-based potential fields paved the way for Arkin’s motor schemas application to mobile robot navigation [5]. Motor schemas architectures use a linear combination fusion method where the behavioral weights parallel the synaptic weights of neural networks.

Schema architectures are inherently parallel and modular unlike subsumptive architectures which greatly increases code reusability but the main downfall is in the arbitration method. It is possible that the linear combination of two feasible control actions will result in an infeasible control action. The likelihood of this
occuring increases significantly with more dense obstacle configurations making this architecture impractical for navigation in cluttered environments.

1.3.1.3 Uni-valued Fuzzy Behavior Architectures

The use of fuzzy logic in the subsumptive and schematic structures resulted in what is known as uni-valued fuzzy behavior architectures. These are hierarchical and adaptive switching methods [19, 26] that use fuzzy logic to determine the degree of applicability of each fuzzy behavior at any instant. Adaptive switching is an extension of subsumptive architectures where fuzzy control is used to determine the degree of applicability for each behavior [26, 19]. The robotic reaction is then drawn from the behavior that has a higher degree of applicability, while other behaviors were discarded. Hierarchical methods use fuzzy logic to calculate the behavioral weights used by motor schemas architectures where the system output is obtained as a linear combination [3, 6, 13, 19, 29]. Computer simulations show that these arbitration methods can fail when the robot is to navigate through densely spaced obstacles and the path edges are irregular [27]. These findings are in agreement with those of [21], where it was noted that the practice of discarding some of the behaviors can lead the robot to wrong paths and deadlocks.

1.3.2 Multi-valued Architectures

Uni-valued behavior architectures have proven to break down in the presence of increasingly complex tasks because the control result can not consider multiple behaviors without some arbitrary bias. Behaviors in uni-valued architectures also
fail because they assume they are the only behavior in the system, therefore causing behaviors to compete for control. Payton and Rosenblatt in their *Planned Guided Reaction* address the “selfishness” of these structures with “fine grained behaviors,” and establishing the foundation of multi-valued behavior architectures. These concepts were consolidated in Rosenblatt’s Distributed Architecture for Mobile Navigation (DAMN) and extended to fuzzy logic in the work of John Yen which will be discussed in further detail in the next sections. Multi-valued architectures are systems where behaviors return their degree of preference to several control alternatives instead of a single control action. The preferences for each alternative can then be evaluated to determine the best control action.

1.3.2.1 Planned Guided Reaction and the Distributed Architecture for Mobile Navigation (DAMN)

The Planned Guided Reaction and its derivative Distributed Architecture for Mobile Navigation (DAMN) allow each behavior to express its degree of relative interest in all of the available command alternatives by assigning it a numerical value [23]. The command fusion summed up the numerical values of the relative interests or
votes expressed by all behaviors for all command alternatives and chose the command
that attracted the most behaviors, i.e., “the winner takes all.” This voting approach
was successful in resolving conflicts and may be considered a cornerstone of conflict
resolution in behavioral robotics [27]. The DAMN architecture however has a major
downfall inherent to its voting system which is clearly shown in Table 1.1. In this
equation Control Alternative A would be chosen although Behavior 4 has a preference
of 0 for that alternative. In its original form the architecture also used crisp data
in all computations, which required accurate information to always be available to
the robot. Unfortunately, information about the environment is very uncertain and
crisp data computation cannot accurately capture it. This drawback was one of the
reasons why fuzzy approaches were sought.

1.3.2.2 Multi-valued Fuzzy Behavior Architectures

The first fuzzy implementation of the fine grained behavior was presented in
[32]. This implementation used only two fuzzy behavioral reactions, i.e., “allowed”
or “disallowed.” The robot followed the direction that had highest “allowed” fuzzy
value. While this structure was sufficient for simple tasks, it lacks features that
ensure smoothness in robot motion. By distinguishing between only two values, i.e.,
the allowed and the disallowed directions, this method can cause the robot to behave

<table>
<thead>
<tr>
<th>Command Alternative</th>
<th>Behavior 1</th>
<th>Behavior 2</th>
<th>Behavior 3</th>
<th>Behavior 4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 1.1. Multi-valued voting architecture failure
erratically by jumping from one allowed direction to another allowed direction of highest fuzzy value. This erratic behavior is expected because the method does not consider partial values such as “may be allowed”. The major contribution of this implementation is the development of the Center of Largest Area (CLA) defuzzification method, discussed further in Section 3.2.2, that allows behaviors to compete and cooperate at the same time.

This thesis uses multi-valued fuzzy behavior design and fusion for developed by Selekwa and Collins [27], which removes the effects of “selfishness.” This was accomplished by denying individual behaviors the ability to control the robot by themselves without regard to the interests of other behaviors. Each behavior independently determines its relative preference to all possible fuzzy command alternatives, and a separate command fusion unit evaluates the preferences for all behaviors to select one command that best satisfies all behaviors. This approach ensures that no behavior is ignored by forcing cooperation instead of competition. The development of this method was highly inspired by the fine grained behaviors of [21] and adds the additional features that [32] lacked. The command output of the system is a result of a compromise made between the behaviors which makes the system more robust, not only against sensor and environmental uncertainty, but also against changes in behavioral priorities.
1.4 Research Objectives and Thesis Outline

The main objective of this research was to implement the multi-valued fuzzy behavior system for navigation control of a Pioneer 2 robot. The robot is equipped with a SICK laser range finder and navigates in cluttered environments. To achieve this objective several modifications were made to improve the performance of the original algorithm and adapted to use the laser range finder measurements instead of the sonar sensors. This thesis presents the algorithmic modifications that were made and the final implementation of the multi-valued fuzzy behavior algorithm for navigation control in cluttered environments.

Chapter 2 presents in further detail the navigation algorithm as presented in [?]. The modifications needed for physical implementation of the algorithm and increases of computation efficiency are presented in Chapter 3. The robotic platform, software tools, and experimental procedures used are described in Chapter 4 as well as presenting the experimental results. Chapter 5 summarizes the the work of the thesis and makes recommendations of future improvements.
CHAPTER 2

MULTI-VALUED FUZZY BEHAVIOR
CONTROL FOR ROBOT NAVIGATION

This chapter begins by pointing out differences between behaviors and what will be called control activities. The standard behavior of a system is its reaction to a particular stimulus or environment. Robot control behaviors are elementary units that lead the robot towards achieving specific goals. Examples of well known robot behaviors include goal seeking, obstacle avoidance, path tracking, and corridor following. Different behaviors may share the same sensory information but yield different reactions. Control activities are the actions that are taken by the behaviors in achieving their goals. These actions involve changing particular control parameters such as the steering angle or the vehicle speed. The act of changing one control parameter is what is referred to here as a control activity. Two control activities are sufficient for robot navigation: speed control, and heading control.
2.1 Principle of operation of the proposed architecture

Figure 2.1 shows the proposed fuzzy behavioral architecture for one control activity. Each control activity is implemented through two blocks; the advisory block and the command block. The control command is done centrally by the command block. The available control command alternatives are known to both the advisory and the command block. The advisory block has several behaviors, each of which is always active. Each behavior determines the relative importance of each of the available command alternatives in satisfying its control objectives. The command block accepts all responses from the advisory block about the available command alternatives. The responses from all behaviors for each of the command alternatives are fused to get the resultant preference by using an intersection operation. The command alternative that has the highest resultant preference is the one that best satisfies all behaviors; it is eventually picked by the command block and sent to the robot actuators.

Note that the stimulus to and the outputs of each behavior are expressed by fuzzy sets. Each behavior uses fuzzy rules to determine the relative importance of each command alternative as fuzzy sets. The cumulative relative importance of each command alternative for all behaviors is found by a fuzzy ‘AND’ operation.

In implementation, the command alternatives are defined as fuzzy sets over the universe of discourse (UD) of the reactions in each control activity. In achieving their behavioral objectives, the individual behaviors do not choose a single fuzzy
Figure 2.1. The Proposed Fuzzy Architecture for Action Control

set from this UD but rather, they express the importance of each. This structure
enables information from all behaviors to reach the command block which chooses
the command that best fits the interests of all behaviors.

2.2 The structure of the proposed architecture

This section gives a more detailed description of how each control activity is
implemented for an ATRV-Jr. robot under the proposed structure. The robot uses
sensor information to determine its way to the target. It is assumed that this mission
requires five behaviors: (1) goal-seeking, (2) obstacle avoidance, (3) the left edge
tracking, (4) right edge tracking, and (5) overturning avoidance. The details of these
behaviors are given in the next subsections.

20
2.2.1 The heading control activity

The heading control activity controls the heading direction of the robot. This requires four behaviors to be satisfied: (1) obstacle avoidance, (2) left edge tracking, (3) right edge tracking, and (4) goal seeking. Its control command is the heading angular change $\Delta \theta$. If the heading angle $\theta$ can be controlled by varying it from $\theta_R$ on the right to $\theta_L$ on the left, then the command alternatives for the heading control activity are fuzzy sets defined on the range $[\theta_R, \theta_L]$ where right turns are assumed negative. An odd number of symmetric fuzzy sets are required. The simplest case requires three fuzzy sets; however, smoother performance requires more fuzzy sets. In the simulations reported by this paper, five fuzzy sets shown in Figure 2.2 were used. The fuzzy labels in this figure are: Large Right Turn (LRT), Slight Right Turn (SRT), No Turn (NT), Slight Left Turn (SLT), and Large Left Turn (LLT). Each behavior $i$ assigns a relative importance to each command alternative $j$ by some parameter $\alpha_{i,j} \in [0, 1]$; the more the value, the more the importance. This parameter is also expressed by fuzzy sets on the interval [0,1]. Any reasonable number of fuzzy sets can be used as shown in Figure 2.3, where four fuzzy sets are used with the linguistic symbols: Not Acceptable (NA), Favored (F), and Highly Favored (HF). Detailed description of each behavior is given below.

**The obstacle avoidance behavior:** The obstacle avoidance behavior uses measurements from the FD1, FD2, and FD3 sensors to determine the possible movements in the forward direction. It becomes active only when an obstacle is
Figure 2.2. The Fuzzy Sets for the Heading

Figure 2.3. The Fuzzy Sets for the Measure of Relative Importance

observed, i.e., the minimum of these measurements is less than some fixed valued. These measurements are used to build a rough map of the obstacle ahead of the robot in order to determine the best robot movement as illustrated in Figure 2.4. In this Figure, measurements in scenario (a) indicate more preference to the left turn than other movements while scenario (b) indicates left and right turns are preferred equally. Scenario (c) is an exact opposite to scenario (a).

The measurements are fuzzified into three fuzzy groups as in Figure 2.5. The fuzzy rules for obstacle avoidance are designed such that the robot can avoid obstacles more smoothly without making sudden turns. The general form of these rules for forward motion is:

\[
\text{IF} \ (FD_1 \text{ and } FD_2 \text{ and } FD_3) \ \text{THEN} \\
(\alpha_{1,1} \text{ and } \alpha_{1,2} \text{ and } \alpha_{1,3} \text{ and } \alpha_{1,4} \text{ and } \alpha_{1,5}). \\
\tag{2.1}
\]

Similarly, for backward motion the rules are
IF $(RD_1 \text{ and } RD_2 \text{ and } RD_3)$ THEN

$$(\alpha_{1,1} \text{ and } \alpha_{1,2} \text{ and } \alpha_{1,3} \text{ and } \alpha_{1,4} \text{ and } \alpha_{1,5}).$$  \hfill (2.2)

The right and the left edge tracking behaviors: The edge tracking behaviors are active only when the robot is within some specified distance from an edge. The stimuli to the right edge tracking behavior during the forward movement are the sensor measurements $FRS$, $FRM$, and $FRV$. The corresponding stimuli for the reverse motion are the measurements $RRS$, $RRM$, and $RRV$. These measurements are used to build a map of the left edge as illustrated in Figure 2.6. The measurements
Figure 2.5. The Fuzzy Groups for the Measured Distances represented by scenario (a) indicate the tendency to turn to the left while the measurements of scenario (b) indicate the tendency to turn to the right.

Figure 2.6. Some of the possible right edge maps

In order to reduce the number of fuzzy rules required to handle these measurements, the measurements are combined as

\[
F_1 = 0.5(FRV + FRM), \quad F_2 = 0.5(FRM + FRS) \quad (2.3)
\]

for forward motion and

\[
F_1 = 0.5(RRV + RRM), \quad F_2 = 0.5(RRM + RRS) \quad (2.4)
\]
for backward motion. In practice, these values are non-negative. As such, any number of fuzzy groups can be used for fuzzification of $F_1$ and $F_2$ similar to those in Figure 2.5. The right edge tracking behavior is implemented by fuzzy rules of the form:

$\textbf{IF} \ (F_1 \text{ and } F_2) \ \textbf{THEN}$

$(\alpha_{2,1} \text{ and } \alpha_{2,2} \text{ and } \alpha_{2,3} \text{ and } \alpha_{2,4} \text{ and } \alpha_{2,5}).$ \hspace{1cm} (2.5)

The left edge track behavior has a similar structure where sensors FLS, FLM, FLV, RLS, RLM, and RLV are used. Its fuzzy rules are of the form

$\textbf{IF} \ (F_1 \text{ and } F_2) \ \textbf{THEN}$

$(\alpha_{3,1} \text{ and } \alpha_{3,2} \text{ and } \alpha_{3,3} \text{ and } \alpha_{3,4} \text{ and } \alpha_{3,5}).$ \hspace{1cm} (2.6)

**The goal seeking behavior:** The goal seeking behavior directs the robot to a specific predefined target. It is assumed that a proper electronic compass is available and the heading direction measured by this compass is $\theta$. If the direction of the goal is $\psi$, then the difference $\Phi$ between the robot heading direction and the goal direction can be computed as

$$\Phi = \psi - \theta, \hspace{1cm} (2.7)$$

The objective of the goal seeking behavior is to make this difference as small as possible, taking the available path into consideration. Its stimuli are $\Phi$ and the estimated frontal distances to the nearest path edges $D_1$ and $D_2$ defined respectively as

$$D_1 = 0.5(FRM + FRV), \quad D_2 = 0.5(FLM + FLV). \hspace{1cm} (2.8)$$

Symmetric fuzzy groups for $\Phi$ as those shown in Figure 2.2 and any number of fuzzy groups for $D_1$ and $D_2$ are needed; however, three groups such as those shown in Fig.
2.2 are sufficient. The behavior is implemented by fuzzy rules of the form

\[
\text{IF } (\Phi, D_1, D_2) \text{ THEN } (\alpha_{4,1} \text{ and } \alpha_{4,2} \text{ and } \alpha_{4,3} \text{ and } \alpha_{4,4} \text{ and } \alpha_{4,5}).
\]

(2.9)

**The command block:** The command block receives proposals \(\alpha_{i,j}\) that are forwarded by individual behaviors \(i\) for each control command alternative \(j\). These proposals are fused by the command block by using the intersection operation

\[
\alpha_j = \bigcap_i \alpha_{i,j}.
\]

(2.10)

Each \(\alpha_j\) is a measure of the importance of each command alternative \(j\). The final control command \(\Delta \theta\) is determined by the command block using fuzzy rules of the form

\[
\text{IF } (\alpha_1 \text{ and } \alpha_2 \text{ and } \alpha_3 \text{ and } \alpha_4 \text{ and } \alpha_5) \text{ THEN } (\Delta \theta).
\]

(2.11)
CHAPTER 3

EXPERIMENTAL ACCOMMODATIONS

Direct implementation of a Multi-Valued Fuzzy Behavior (MVFB) as presented in Section ?? was not possible due to various differences between the experimental platform and the simulation procedure. These differences are a direct result of progressing from a discrete idealistic simulation environment to continuous time implementation with a fixed sensor configuration and more complex obstacle configurations. Exploration of means by which to accommodate for these differences also lead to major changes in the system structure. Many of the changes were prompted by an understanding the intended purpose of various system components. These components were then constrained to operate within that purpose utilizing only necessary operations. Consequently the computational efficiency of the algorithm was significant improved along with increased system performance.
3.1 Sensor Configuration

Although the original sensor configuration for active ranging was effective in simulation, it was impractical for physical implementation. Under that configuration obstacles were detected by measuring the distance to the closest obstacle along lines emanating from sensors mounted on the perimeter of the robot. The physical realization of such a sensor is that of a 1-D sensor type such as a simple laser range finder or an optical triangulation sensor. Such a configuration allows small obstacles that lie out of the line of sight of the sensor to pass undetected as illustrated in Figure 3.1. In this case, the navigation would behave as if the obstacle were not there and would lead to the robot hitting it. In the simulation results, this scenario was most likely not observed due to the increased scale of the obstacle size with respect to the size of the robot itself. In implementing the algorithm, the active ranging device was a 2-D sensor and will be discussed in further detail in Section 4.1.2.

In order to eliminate the possibility that front obstacles can be wrongly detected as a side obstacle in close proximity to the robot and to transform the sensor readings in accordance with the algorithm, the sensor inputs were grouped into a total of nine regions. Initially all nine of these regions were represented by equal sized cones of 20° each. This orientation had to satisfy two conditions:

1. At a minimum safe distance, \( d_s \), between the robot and the side obstacles the width of the six side regions must represent a width, \( w_t \), that is traversable by the robot.
2. The three front sensor regions combined must represent a width traversable by the robot.

The latter condition was simplified further; which lead to the front regions being rectangular with an overall width that is slightly wider than the actual robot in order to provide some safety margin. The conic side sensor regions were widened to overlap; the sensor regions are illustrated in Figure 3.2. The necessary included angle of the cone, $\theta_s$, for side sensor regions can be calculated as:

$$\theta_s = 2 \arcsin\left(\frac{w_t}{2d_s}\right),$$

which yields $\theta_s = 35^\circ$ for a $w_t/d_s$ ratio of 0.6.

Through experimentation, it was also observed that the larger these regions are, the more likely it is that the algorithm will not be able to discern a traversable path.
even when one actually exists. The sensor inputs for larger regions tend to indicate that an excessively wide space is needed in order for the proposed direction to be traversable. The opposite is also true with making the regions smaller resulting in the algorithm more likely to attempt a direction that is not in fact traversable. Therefore the sensor region size becomes a major factor in the tuning process of the algorithm.

Figure 3.2. Revised Sensor Region Configuration

3.2 Fuzzy System Modifications

After modification of the sensor inputs, preliminary experimentation demonstrated the need to revise the fuzzy system. The complexity of the navigation was significantly increased by the use of much smaller obstacles as well as more densely populated environments. This added complexity exposed issues that lead to the addition of linguistic terms, and reshaping of the fuzzy inputs and outputs, and most importantly the elimination of the command block. To facilitate the tuning
process, the fuzzy input and output terms were normalized so that they could be
scaled to modify the system performance without having to change the fuzzy system
membership functions. The range sensor values, $r_s$, used as inputs to the fuzzy
system were the normalized values of the measured range, $r$. The normalization
used the robot length, $l_r$ and a properly chosen input scale factor, $s_i$ such that

$$ r_s = \frac{s_i}{l_r} r. $$

Similarly, the system output change in heading angle, $\Delta \theta_s$ was a normalized value
of the actual heading change, $\Delta \theta$ where

$$ \Delta \theta_s = \Delta \theta s_o, $$

and $s_o$ is a properly chosen factor.

3.2.1 Command Block Elimination

Upon producing initially unfavorable results for relatively simple configurations,
the purpose of the command block was evaluated. At least for these circumstances,
the behaviors that comprise the advisory block were functioning properly but the
command block was sending commands to the robot actuators that caused the
navigation to fail. The purpose of the command block was to take the importance of
all the behaviors, $\alpha_j$, which are crisp numbers and determine the resulting $\Delta \theta$ that
satisfies all the behaviors. This was described in more detail in Section 2.2.1. In
essence, the command block determines the activation level of each of the available
command alternatives and selects the command with the highest activation level.
Evaluation of the command block revealed that the level of activation of command $j$ can be determined directly as the value of the intersection $\alpha_j$, making the command block unnecessary. This eliminates the need to execute the $3^5 = 243$ rules of the command block and fuzzyify its inputs, $\alpha_j$ representing a significant computational improvement in the algorithm. The elimination of the command block removes the conflict resolution ability of the system when there are command alternatives with equally high suitability; however, this capability will be restored with the use of a special defuzzification method that will be discussed further in Section 3.2.2. The structure of the control system after elimination of the control block is shown in Figure 3.3. The control is now comprised of the advisory block, a fusion block as well as a defuzzification block, which is separate because it is special for these types of systems.

Figure 3.3. Revised Multi-valued Fuzzy Behavior Control Structure
3.2.2 Defuzzification Method

The defuzzification method must be chosen such that it gives a best compromise between the proposed control actions (PCAs) as well as being computationally efficient. This compromise also forces the defuzzification to be continuous and inherently have a degree of smoothness. Defuzzification methods like Mean of Maximum (MoM) give the most plausible result but can only evaluate one output term at a time [16]. The discontinuity of MoM is also unfavorable causing an arbitrarily small change in inputs to cause the output decision to change by significantly large amounts. For mobile robot navigation this method is likely to produce an erratic response similar to that of uni-valued behavior systems since it does not attempt to find a compromise between the output terms. To ensure smoothness of the robot response it was necessary to use a centroid defuzzification method; however defuzzification methods that involved the online calculation of areas like Center of Area (CoA) are computationally inefficient. The numerical integration that is utilized in CoA can cause the method to take over a thousand times longer than singleton centroid methods [16]. Fast CoA eliminates the online calculation of areas that slows the traditional CoA, by calculating the areas at compilation of the C++ code [16]. Fast CoA can take into account multiple output terms and find a good compromise; the precomputed areas effectively become scaling factors that make the method mathematically the same as Center of Maximum (CoM). The CoM defuzzification method is a special case of the centroid defuzzification method that takes into account the discontinuity
of the output fuzzy set by determining the centroid of the maximum segment of the
output fuzzy set. The major difference between Fast CoA and CoM is that the scaling
factors for CoM can be defined explicitly while those of Fast CoA are determined by
the areas, therefore much care must be taken to accommodate for these areas when
modifying the output variable. Consequently CoM is the defuzzification method that
best satisfies all of the system preferences and can be calculated for singleton sets as

\[ u = \frac{\sum_{k=1}^{n} DOF_k \cdot M_k}{\sum_{k=1}^{n} DOF_k \cdot B_k}, \]

where \( B_k \) is degree of scaling of the rule \( k \), \( M_k \) is the moment of \( B_k \), and \( DOF_k \)
is the degree of fulfillment of the \( k^{th} \) rule [30]. The moment of \( B_k \) is the product \( B_k \)
it’s location along the \( \mu \) axis, \( LOC_k \). For the heading control output defuzzification,
\( B_k = 1.0 \) and \( DOF_k = \alpha_k \) therefore, the output \( \Delta \theta \) becomes

\[ \Delta \theta = \frac{\sum_{k=1}^{5} \alpha_k \cdot LOC_k}{\sum_{k=1}^{5} \alpha_k}, \]

so that the only remaining tuning parameter for each output term is \( LOC_k \) [30].
This greatly simplifies determining their shape since the only parameter that affects
the control output is the center of the maximum of each term.

The major concern with the defuzzification approach is that it may lead to
an undesirable result when there are two or more viable control alternatives.
This situation arises frequently in a densely populated obstacle configuration with
relatively small obstacles such as a forest. For instance the obstacles may be
configured such that it is desirable to turn either right or left but not straight as
demonstrated in Figure 3.4. In order to prevent the defuzzification from resulting in an undesirable control command, either the rules or output terms are often biased in a particular direction. This biasing is often made a priori and arbitrarily without taking into account the current situation.

Figure 3.4. Fuzzy output demonstrating the effect of compromise in traditional CoM

In order to avoid this biasing as a means to compensate for this shortcoming while maintaining the favorable qualities of CoM defuzzification, the Center of Largest Area (CLA) method was used [32]. This method divides the fuzzy output terms into separate groups whenever an intermediate term falls below a certain threshold. This prevents the output from falling in a region that represents an unsuitable control action. When no terms fall below that threshold there is only one group and CLA behaves just as CoM. The group with the largest degree of suitability, \( \alpha^* \), is then defuzzified using the standard CoM which makes the best compromise between the control alternatives within that group.
\[ \alpha_g^* = \sum_{k=m_g}^{n_g} \alpha_k, \]  

(3.1)

where \( m_g \) and \( n_g \) are the first and last output terms respectively in group \( g \).

### 3.2.3 Limitations of Behavioral Influence

The success of the MVFBC hinges on preventing individual behaviors from being able to dictate the control action without consideration of the other behaviors. This requires setting limitations on the influence of each behavior. These limitations should be characterized according to the control objective of each behavior and can be enforced through determination of the input and output terms available to each behavioral rule block. Determination of these terms can be done by evaluating the intent of each behavior while keeping in mind the nature of the fusion that is necessary to obtain a final control action.

#### 3.2.3.1 Realm of Influence

Firstly the realm of influence of each behavior must determined. This is based upon an understanding of how the behavior results will be combined to determine the suitability of each PCA. Since the behavioral outputs are combined using an intersection operation, it only takes one behavior to reduce the suitability of a PCA. Based on this information, a behavior should not take any action that can reduce the suitability of a PCA that it does not have information about. For example, the right edge tracking behavior is only given sensor information about the obstacles.
on the right side, which does include the front to some extent based on the sensor configuration as illustrated in Figure 3.2. Therefore the right edge tracking rule block should not return a suitability other than 1.0 for any PCA concerning turning to the left. Similar cases prevail for the left edge tracking and front avoidance behaviors.

3.2.3.2 Degree of Influence

In the case of the goal seeking behavior, the issue is not necessary the realm of influence but the degree of that influence. Since the goal seeking behavior is only given information concerning it’s location with respect to the goal and not obstacle information, it should not be able to eliminate any PCA by returning a linguistic term of not acceptable (NA) or the suitability of 0.0. The goal seeking is then relegated to determine the degree to which PCAs are acceptable instead of deciding wether or not those PCAs are in fact acceptable. It must also be noted that another input term is added to preserve the fidelity for the behavior, after the removal of the NA term. To allow the goal seeking behavior to declare a PCA not acceptable creates a major problem in many complex obstacle configuration. In these situations the least favorable PCA according to the goal seeking can possibly be the only feasible control output. Figure 3.5 shows such a circumstance where navigation of the only traversable path requires turning significantly far away from the goal direction.

In these kinds of situations, if the goal seeking were able to declare PCAs not acceptable, there would either be no acceptable PCAs or the robot would be forced to go in a direction of extremely low suitability that would eventually lead to failure. This failure occurs because the goal seeking behavior is allowed to respond as if the
only traversable path is blocked. Limiting the degree of influence of this behavior allows the navigation of more complex obstacle fields.

3.2.4 Rule Base Determination

After having dealt with the major structural issues regarding the fuzzy system, it was necessary to evaluate the system rule base. Upon evaluation, it proved more simple to completely rewrite the rule blocks for all four behaviors. The necessity of such drastic changes are mostly due to the drastic changes in the nature of the sensor inputs; this is logical since the rules are formulated based on how the input fuzzy variables represent the physical obstacle configuration. It is not possible to modify the very nature of the system inputs without having to drastically change the rule base in order to obtain similar results. Modifications to the type and scale of the obstacles are also a major contributor to the inadequacy of the original rule block.
3.2.4.1 Relevant Input Information

It is important that each behavior have relevant and necessary information. This information should be sufficient to satisfy its control objective and unnecessary information should be eliminated. Along this line of reasoning, the sole purpose of the goal seeking algorithm is to reach the goal. This behavior in and of itself should totally disregard obstacles and sensor inputs regarding such information, therefore making the Nearest Edge input unnecessary. The only essential information is a relative angle to the goal position; the relative distance to goal also is but not essential. The use of the Nearest Edge input into the goal seeking behavior represents an attempt to incorporate objectives beyond the scope of the behavior. If these objectives prove necessary, which they did not in experimentation, then they would function better as their own behavior. The elimination of the Nearest Edge input removes ten rules but more importantly alleviates the burden of tuning the various parameters needed for its calculation. The other three behavioral rule blocks have all the relevant inputs that are needed to determine their control objective.

3.2.4.2 Visual Rule Determination

Complete selection of the input and output terms of the behavioral rule blocks are prerequisites to determination of the actual rules that compose each rule block. This task was completed in Sections 3.2.3, and 3.2.4.1. The compartmentalization of the control objective into behaviors as well as limiting the total number of input terms greatly simplifies rule determination.
For the obstacle avoidance behaviors there are three input variables with three terms each, resulting in $3^3$ rules. This relatively small number of rules decreases the difficulty of manual rule determination. Due to the highly visual nature of the behaviors, visual rule determination (VRD) is the most intuitive approach for the shaping the rule base. The sensor configuration as illustrated in Figure 3.2 was overlain with the circular regions representing the range sensor input terms as shown in Figure 3.6 in order to facilitate the utilization of VRD. The range sensor inputs are described in more detail in Sections 3.1. The sensor configuration without the region overlap was used for simplicity, as the final configuration can complicate rule determination.

![Sensor Configuration with Input Term Overlay for VRD](image)

**Figure 3.6.** Sensor Configuration with Input Term Overlay for VRD
The terms C (Close), M (Medium), and F (Far) separate ranges of distance measurements and are crisp representations of the range sensor input terms. Of course the actual input terms are not crisp regions but the level of complication needed to add actual trapezoidal membership functions was unnecessary for VRD. The size of each region in this figure is for visualization only and must be tuned later to fit the purpose of the control system. Then using the visual regions of Figure 3.6, the intended control action for each set of possible input terms can be logically determined by an individual who is capable of expertly navigating the configuration. This is done systematically for each behavior while making a conscience attempt to maintain smoothness. Smoothness is promoted by avoiding rule sets that allow the output to change drastically from among behaviors and input terms. For example, a rule should not say that one $PCA_3$ is HF while the PCA directly next to it $PCA_4$ is NA. Similar cases are true concerning input terms as well.

### 3.2.5 Membership Function Shaping

After having fully determined the input and output variables and their terms as well as establishing the behavioral rule blocks it was then necessary to shape the fuzzy membership functions. These membership functions must be shaped according to the purpose of the term that it represents.

#### 3.2.5.1 Input Variable Terms

Determination of the shapes of the input terms is more specific than for the output terms since the defuzzification method of the output only accounts for the
maximum point of each membership function, $LOC_k$ as described in Section 3.2.2.
For simplicity, trapezoidal membership functions were chosen for all the obstacle avoidance input terms while the orientation input also uses triangular shapes. The shapes of the membership functions were tuned primarily by experimental trial-and-error while the VRD described in Section 3.2.4.2 helped define a starting point. Increasing the amount of overlap between input membership functions helped to allow the activation of multiple rules more often, which resulted in more smooth control. This overlap was increased between the small (S) and medium (M) terms for all range sensors except VFR and VRL, which have a wider medium (M) region as illustrated in Figures 3.7 and 3.8. The $\Phi$ input terms are closely grouped as shown in Figure 3.9 to provide more fine control for smaller changes in angle.

![Membership Functions for FL, L, LC, C, RC, R, and FR.](image)

**Figure 3.7.** Membership Functions for FL, L, LC, C, RC, R, and FR.

### 3.2.5.2 Output Variable Terms

The output terms ($\alpha_{ij}$) need only to be defined by an ordered pair of $LOC_k$ and maximum degree of membership, which is a fuzzy singleton. However in the fuzzy logic software utilized for the implementation, *fuzzyTECH*, these membership
functions are represented by other shapes and converted into singletons during code compilation. Because of the limitation of behavioral influence discussed in Section ?? membership functions for each behavior are different. Figures 3.10 through 3.13 show the different membership functions for the $\alpha_{ij}$s.

The membership functions shown in Figure 3.12 are subject to the same limitations and are the result of shifting Figure 3.11 to the right; this gives the navigation a slight preference to not turn which reduces the control effort. Due to the realm of influence limitations discussed in Section 3.2.3.1, the output terms illustrated in Figure 3.13 only have a HF term.
Figure 3.10. Membership Functions for $\alpha_{14}, \alpha_{15}, \alpha_{23}, \alpha_{31}$, and $\alpha_{32}$.

Figure 3.11. Membership Functions for $\alpha_{22}, \alpha_{24}$, and $\alpha_{4j} j = 1, 2 \ldots 5$.

The membership functions for $\Delta \theta$ shown in Figure 3.14 are simply a representation as this final output variable is defuzzified outside of the fuzzyTECH software. This term is therefore actually represented by an ordered pair with maximum membership locations in the same places as in the figure.

3.3 From Discrete to Continuous: Command Action Execution

In simulation, execution of the command action is relatively simple mostly due to the discrete nature of the problem. It uses a simple discrete time kinematic model of the robot with fixed time step. This makes the determination of the
next robot position and orientation a simple calculation, which was not so simple in implementation. Decisions must be made that allow the robot to execute the command action accurately and reliably. The control action is in terms of a change in angle, $\Delta \theta$ which could either be sent directly to the robot or translated into a turning rate, $\dot{\theta}$ (or $\omega$) for a particular time period, $\Delta t$ where

$$\omega = \dot{\theta} = \frac{\Delta \theta}{\Delta t}. \quad (3.2)$$

This could possibly produce favorable results but is highly dependant on being able to accurately control $\omega$ and $\Delta t$. The time step, $\Delta t$ can be forced to remain
constant but this causes $\omega$ to be proportional to $\Delta \theta$, which becomes a problem since faster rotational velocities induce more slip that drastically decreases the accuracy of robot localization. Slower rotations also seem to waste time. A piecewise continuous combination of fixed and variable time rate was conceived but considered not worth the calculation time especially since a direct execution of the control command was available. Direct heading control is a much more simple approach, especially since the robot interface contains a function that determines whether or not the heading command is completed.

Initial implementation of direct execution approach provided unfavorable results because the check heading was not properly utilized. Since the algorithm did not wait until the heading change was completed, additional control commands were determined at relatively the same location. These commands were then executed after they were no longer situationally relevant. It was then resolved that the algorithm wait until one control output has been completely before the calculation of another command. The only concern with this constraint is that the resulting
execution speed of the navigation is limited by the robot’s maximum turning rate, $\omega_{max}$. This physical limitation is attributed to the selected robotic platform and does not significantly slow the algorithm.
CHAPTER 4

EXPERIMENTAL PROCEDURE AND
RESULTS

This chapter presents the experimental apparatus, procedures, and results. It is divided into four sections. Section 4.1 describes the experimental platform including the vehicle dimensions, computer, and sensors. The fuzzyTECH software and its use are discussed in Section 4.2. Section 4.3 provides detail of the experimental procedure while Section 4.4 describes the final experimental results.

4.1 Experimental Robotic Platform

The revised MVFB for robot navigation developed in Chapter 3 was implemented on the Pioneer 2 Mobile Robot Platform manufactured by ActivMedia Robotics. This is a differentially driven platform is configured with two drive wheels and one swivel caster for balance. It has a 44cm x 38cm x 22cm aluminum body with two 16.5cm diameter drive wheels and a swing radius of 26cm [2]. Each wheel is driven independently by a motor with 19.5:1 gear ratio which enables the robot to drive at
a maximum speed of 1.2 m/s and climb a 25% grade [2]. The following subsections present an overview of the onboard computer system and the navigation sensors.

![Figure 4.1. Pioneer 2 Physical Dimensions and Swing Radius](image)

### 4.1.1 Onboard Computer and Interfaces

The robot is equipped with an embedded 850 MHz Pentium III onboard computer running RedHat Linux version 7.2. The platform is also includes a Hitachi HS8-based micro-controller with 8 digital inputs, 8 digital outputs, and 1 dedicated A/D port [2]. There is also a 802.11b PCMCIA wireless receiver installed for TCP/IP connectivity. User I/O is configured in the packet structure and is accessible through ARIA (ActivMedia Robotics Interface Application), which is ActivMedia’s API (Application Programming Interface) and is written completely in the OO (Object Oriented) paradigm. ARIA can be run in single or multi threads at the control level of the user’s choice; it also has the capability to dynamically control vehicle velocity, heading, and relative heading as well as poll range and localization sensors [1]. What
most makes ARIA a useful tool for quick robot implementation is the comprehensive reference documentation.

4.1.2 Range Sensors

In order for the robot to perceive its environment, range measurement sensors are needed. These sensors measure the closest distance to an obstacle in a particular direction or in a particular region. The Pioneer 2 is equipped with sonar sensors as well as a laser range finder or laser radar system for range measurement. Because of its increased accuracy and resolution the laser range finder is used in this control system. Laser range finders operate on the same principle as sonar except that they transmit laser pulse beams that can be reflected off an internal mirror in various directions to create a scan. The laser range finder used is the SICK LMS 200 as

![Figure 4.2. Pioneer 2 With SICK Laser Range Finder](image)
shown in Figure ??.

This sensor has a resolution of 10mm, a typical measurement accuracy of ±15mm, 180° scanning angle, and 10m typical measured distance range [11]. Measurements can be made for scan angles as small as 0.25° and that can be composed into rectangular and cone shaped regions [11].

4.1.3 Localization Sensors

The main object of the navigation algorithm is to reach a goal position without while avoiding obstacles. You can not get where your going if you do not know where you are. In other words, reaching to goal position would not be possible if the robot had no knowledge of its current position with respect to some coordinate system wherein the goal position is defined. This information is provided by localization sensors. Typically localization sensors include Global Positional Systems (GPS), Inertial Navigation Systems (INS), an electronic compass, etc. Localization on the Pioneer 2 robot was achieved by using wheel encoders. Each motor on the mobile robotic platform is equipped with a 500 tick encoder [2]. These measure the change in orientation of the motors in increments of 1/500 of a rotation. This information after gear ratios have been factored in provide the change in orientation of each wheel. The orientation information is differentiated to provide wheel velocities which can be used in conjunction with the vehicle kinematic model to calculate the vehicles relative position and orientation.

This method uses a simple kinematic model of the robot and does not account for wheel slip. The degree of slip is highly surface dependant and the use of only encoders
for localization will lead to surface dependant performance. The encoder information is also used as feedback for the low level speed and heading control which causes slip to impact the obstacle avoidance aspect of the algorithm as well. For example, a change heading control command of 30° sent to the robot may result in a turn of 25° because of slip. This inaccurate execution of a control command can cause the control system to fail.

The good thing is that the error due to heading control is not cumulative, which is true because each control command is based on the current relative obstacle locations and does not rely on accurate execution of prior commands. This fact is due to the reactive nature of the control and its lack of dependency on preplanned paths and environmental modeling. The localization error is accumulated but is path dependant, meaning that although the localization error exists, the algorithm will be able to reach the same point when having to execute relatively the same turns. These two facts enable the implemented algorithm to be tuned for various surfaces by manipulating the steering scale factor and goal distance tolerance. The use of INS (Inertial Navigation System) would alleviate the additional tuning needed due to this surface and path dependance. INS systems use various technologies to measure accelerations in principle directions. These accelerations can then be integrated to obtain significantly more accurate relative position and orientation information.
4.2 fuzzyTECH

The fuzzy control system must be programmed in C++ in order to run on the robot using the ARIA interface. In this implementation, the C++ code to implement different fuzzy logic functions was generated using fuzzyTECH software from INFORM Software Corporation. This software was selected as an alternative to the more common, Matlab Fuzzy Logic Toolbox because of its automatic C code generation and the more user friendly graphical interface. The automatic code generation saves valuable research time as virtually no programming is necessary to implement fuzzy control. The software also generates .m files for integration into the Matlab simulation environment. The fuzzy systems can be modified with a standard text editor as well as graphical spreadsheet and matrix rule block editors. Figure 4.3 is a typical screen shot showing a fuzzy system, its inputs, outputs, and rule block in the fuzzyTECH environment. The fuzzyTECH Online Edition adds the ability to monitor and modify the fuzzy system online through a TCP/IP connection. This capability is very useful in the tuning process since it allows the user to view how current inputs effect the control output. FuzzyTECH has all the necessary input and output variable types as well as all the necessary aggregation operators.

4.3 Experimental Procedure

In order to validate the preliminary simulation results and test the capabilities of the navigation algorithm in more complex environments, physical experimentation is necessary. For each experimental case the robot is set at a particular start point
and the goal is defined in either cartesian or polar coordinates with respect to the start position.

4.3.1 Obstacle Description

A dense forest was chosen as the experimental environment with tree like shapes being the obstacles. These obstacles are very difficult to navigate through because they are relatively small and because several densely configured obstacles can effectively form one obstacle. Trees were simulation by 2’ long by 2”, 3”, and 4” diameter PVC pipe sections. The height is important because the SICK laser only measures distances along a plane about 18 inches above the ground. The pipe
diameters scale appropriately to the vehicle size and accurately depict the trunks of trees. The different sizes also test the ability to detect and avoid smaller obstacles.

4.3.2 Obstacle Configuration

The configuration of these obstacles must be chosen carefully. Firstly it is important for each obstacle configuration to have at least one traversable path. This is due to the fact that the current navigation does not incorporate a speed controller and therefore does not have the capability to stop when there is no path. The number of traversable paths is decided by the objective of the experiment. An obstacle configuration with only one traversable path is the most difficult because the robot must be able to identify and navigate that one path. The existence of multiple paths can serve to illustrate the decision making of the algorithm by forcing the robot to choose one path direction as apposed to another. These paths are considered traversable by being wide enough not only for the robot but must also enable the robot to negotiate the path by providing room for the robot to make appropriate turns. Obstacle configurations with single and multiple paths are used in experimentation to demonstrate the capabilities of the MVFB algorithm where typical algorithms fail. Figure 4.4 shows the robot navigating an obstacle field.

4.4 Results

Depiction of experimental results of physical implementation is best represented in video format. Since that medium is not available for graphical representation the obstacle configurations were mapped and the localization data (X,Y,θ) recorded. The
Figure 4.4. Robot navigating through a dense obstacle field
following figures were generated in Matlab based on the recorded information and illustrate actual results. They are not simulation results but a depiction of physical experiments conducted on the Pioneer 2 mobile robot. The goal region represents the tolerance from the desired goal position that defines a successful navigation of the obstacle field. The scenarios progress from simple test cases that were used for development and tuning to more complex obstacle configurations.

Scenario 1 has only one obstacle directly in front of the robot. The goal is located along the line formed by the robot start position and the obstacle. The robot chose to avoid the obstacle by going to the right even though the feasibility of both directions are relatively identical. This illustrates the algorithms ability to resolve conflicts between opposing behaviors. Scenario 2 represents the same obstacle configuration except with the goal position moved slightly to the left which results in the robot
avoiding the obstacle in that direction. This shows that the algorithm does not prefer a particular direction except when the alternatives have equal feasibility as well as demonstrating the consideration of the goal seeking behavior while avoiding obstacles.

Scenario 3 shows the ability of the robot to identify and traverse the only gap wide enough for the robot to pass. This is very difficult because the robot has to choose its initial direction properly and be able to accurately gauge the obstacle spacing. It must be noted that this is accomplished without actually measuring the distance between the obstacles but by the sensor region sizing as described in Section 3.1.

One of the key features of this algorithm is its affinity to find the shortest distance. Scenarios 4 and 5 represent the same obstacle configuration except that scenario 5 was executed after the removal of one obstacle, which creates a shortest path to the
Figure 4.6. Scenario 2

Figure 4.7. Scenario 3
goal. As such the path taken in Scenario 5 is shorter than that in Scenario 4. These experiments also show how the algorithm will make an effort to traverse the most direct path to the goal position.

Scenarios 6 and 7 are more complex situations that illustrate the ability to navigate very small gaps and turn away from the goal when necessary to avoid obstacles. This ability remains even when relatively close to the goal. For these cases, the goal region has to be enlarged due to the accumulated localization error which becomes larger for longer paths.

Scenarios 8 and 9 represent a more sparse obstacle configuration that is more typical to an actual forest. There are various traversable paths but the robot navigates the most direct path to the goal. These scenarios have the same obstacle configuration with different goal locations. In scenario 9 turning to the right is more
Figure 4.9. Scenario 5

Figure 4.10. Scenario 6
favorable than turning to the left from an obstacle avoidance perspective but the
contribution of the goal seeking behavior makes turning left the resulting control
action.
Figure 4.12. Scenario 8

Figure 4.13. Scenario 9
CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

This chapter summarizes the implementation process of multi-valued fuzzy behavior control (MVFBC) and the experimental results obtained navigating through densely populated environments as well as making recommendations for future improvements to the algorithm. The summary is contained in Section 5.1, and the recommendations are in Section 5.2.

5.1 Conclusion

Behavior robotics has demonstrated various successful methods for robot navigation. These methods fail in cluttered environments due to the increasing necessity to accommodate the needs of all behaviors simultaneously. This thesis demonstrates through implementation, the significant contribution of multi-valued fuzzy behavior control to the resolution of this issue. MVFBC addresses informational uncertainty with fuzzy logic control while using multi-valued behaviors to allow the control command to satisfy the needs of all behaviors.
Major changes were made to simplify the control structure. Removal of the control block lead to the characterization of the disjoint center of maximum (DCoM) defuzzification method as well as substantial improvements in the computational efficiency of the algorithm. A systematic method was developed to limit the influence of behaviors. Aspects of this method force each behavior to operate in its information space, which prevents the behaviors from unnecessarily conflicting with one another. This method also prevents behaviors from disallowing particular proposed control actions while maintaining their ability to determine their preference for those control actions.

Several changes were also made to accommodate implementation on the Pioneer 2 mobile robot. Range sensor inputs were reconfigured and sized analytically based on robot parameters, which enabled the detection and avoidance of smaller and more densely configured obstacles. The control output was also modified for more direct execution of the command action using the ARIA interface.

Results illustrate the success of MVFBC in both simple and complex circumstances. These scenarios demonstrate that MVFBC works in situations where typical uni-valued control would either be unsuccessful or highly inefficient. The Pioneer 2 was able to navigate complicated paths barely wide enough to be traversable while considering the location of the goal. This implementation has proven MVFBC to be a major contribution towards behavioral conflict resolution by allowing behaviors to be effectively fused without the biasing inherent to subsumptive navigation algorithms and erratic behavior generated by the “winner take all” approaches. The
defuzzification method developed for MVFBC (DCoM) also ensures that two feasible solutions can not be combined to form an infeasible solution as is possible with linear combination methods.

5.2 Future Work

Reactive behaviors alone are not sufficient for autonomous mobile robot navigation. Although the behavior base of MVFBC can be expanded to increase the capabilities of the algorithm, scenarios exist where behavior based approaches would become inefficient if not ineffective. For example, if the robot had to navigate out of a room with one exit to reach a goal inside an adjoining room, deliberative approaches would be more efficient than purely reactive approaches. Under deliberative approaches the robot could simply identify the exit and proceed to the goal after exiting the room whereas a purely reactive controller would wander around the room until it found the exit. Incorporation of deliberative capabilities along with reactive capabilities into a hybrid control architecture would be the most natural progression of this work which would require some changes.

Although the implementation of MVFBC shows promising results, several improvements would increase the performance of the algorithm. Firstly the implementation of the speed controller is necessary. Currently the robot speed is fixed for simplicity and to minimize the number of tuning variables; it is set to be slow enough to navigate the most complicated environments and is therefore limited. Implementation of the speed controller would greatly decrease the navigation time.
as well as incorporate an emergency stop feature needed both for safety and to prevent damage to the vehicle in real forest environments.

The goal seeking behavior and turn control are implemented using wheel encoder sensor measurements, which are shown to have significant errors due to wheel slip. These errors can be greatly reduced by using measurements from an inertial navigation system (INS) which would virtually eliminate the localization error and allow accurate execution of each control command. Use of the INS will make the algorithm more robust and therefore more useful in real applications.

Reliability and accuracy would also be increased if the algorithm had a more detailed sensorial representation of the obstacle configuration. This could be added by increasing the number of sensor regions, which would result in the need for additional rules. The computational effects of these additions with respect to the original algorithm would be minimal in light of the improvements made from the removal of the rule block.

The proposed improvements would significantly increase the number of tuning variables present in the control. In order to deal with the increasing complexity of the controller, the utilization of adaptive tuning would be especially useful. This would most likely be accomplished using a genetic algorithm since it has already proven to be successful in tuning the fuzzy parameters of similar systems [31].
This appendix ...
1 General Information

Author: Damion Dunlap
Created: Tuesday, July 01, 2003
Print Date: Monday, June 14, 2004

Edition
Edition Name: fuzzyTECH 5.54k Professional Edition
Neuro Modul: NeuroFuzzy add-on Module installed

.1 List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Center</td>
</tr>
<tr>
<td>FL</td>
<td>Far Left</td>
</tr>
<tr>
<td>FR</td>
<td>Far Right</td>
</tr>
<tr>
<td>L</td>
<td>Left</td>
</tr>
<tr>
<td>LC</td>
<td>Left Center</td>
</tr>
<tr>
<td>R</td>
<td>Right</td>
</tr>
<tr>
<td>RC</td>
<td>Right Center</td>
</tr>
<tr>
<td>VFL</td>
<td>Very Far Left</td>
</tr>
<tr>
<td>VFR</td>
<td>Very Far Right</td>
</tr>
<tr>
<td>Compute MBF</td>
<td>Compute Membership Function (Fuzzification Method)</td>
</tr>
<tr>
<td>CoM</td>
<td>Center of Maximum (Defuzzification Methode)</td>
</tr>
<tr>
<td>BSUM</td>
<td>Bounded Sum Fuzzy Operator for Result Aggregation</td>
</tr>
<tr>
<td>MIN</td>
<td>Fuzzy Operator for AND Aggregation</td>
</tr>
<tr>
<td>MAX</td>
<td>Fuzzy Operator for OR Aggregation</td>
</tr>
<tr>
<td>GAMMA</td>
<td>Compensatory Operator for Aggregation</td>
</tr>
<tr>
<td>PROD</td>
<td>Fuzzy Operator for Composition</td>
</tr>
<tr>
<td>LV</td>
<td>Linguistic Variable</td>
</tr>
<tr>
<td>MBF</td>
<td>Membership Function</td>
</tr>
<tr>
<td>RB</td>
<td>Rule Block</td>
</tr>
</tbody>
</table>
2 Multi-valued Fuzzy Behavior Control for Navigation in Cluttered Environments

.1 Project Description

<table>
<thead>
<tr>
<th>Input Variables</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Variables</td>
<td>20</td>
</tr>
<tr>
<td>Intermediate Variables</td>
<td>0</td>
</tr>
<tr>
<td>Rule Blocks</td>
<td>4</td>
</tr>
<tr>
<td>Rules</td>
<td>344</td>
</tr>
<tr>
<td>Membership Functions</td>
<td>85</td>
</tr>
</tbody>
</table>

Table 1: Project Statistics

.2 System Structure

The system structure identifies the fuzzy logic inference flow from the input variables to the output variables. The fuzzification in the input interfaces translates analog inputs into fuzzy values. The fuzzy inference takes place in rule blocks which contain the linguistic control rules. The output of these rule blocks are linguistic variables. The defuzzification in the output interfaces translates them into analog variables.

The following figure shows the whole structure of this fuzzy system including input interfaces, rule blocks and output interfaces. The connecting lines symbolize the data flow.
Variables

This chapter contains the definition of all linguistic variables and of all membership functions. Linguistic variables are used to translate real values into linguistic values. The possible values of a linguistic variable are not numbers but so called 'linguistic terms'.

For example:
To translate the real variable 'temperature' into a linguistic variable three terms, 'cold', 'pleasant' and 'warm' are defined. Depending on the current temperature level each of these terms describes the 'temperature' more or less well. Each term is defined by a membership function (MBF). Each membership function defines for any value of the input variable the associated degree of membership of the linguistic term. The
membership functions of all terms of one linguistic variable are normally displayed in one graph. The following figure plots the membership functions of the three terms for the example 'temperature'.

![Membership Function of 'temperature'](image)

A 'temperature' of 66 °F is a member of the MBFs for the terms:

cold  to the degree of 0.8  
pleasant to the degree of 0.2  
warm  to the degree of 0.0  

Linguistic variables have to be defined for all input, output and intermediate variables. The membership functions are defined using a few definition points only.

The following tables list all variables of the system as well as the respective fuzzification or defuzzification method. Also the properties of all base variables and the term names are listed.

### Inputs

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>X</td>
<td>Norm_Distance 0</td>
<td>4</td>
<td>4</td>
<td>S M L</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FL</td>
<td>X</td>
<td>Norm_Distance 0</td>
<td>4</td>
<td>4</td>
<td>S M L</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>FR</td>
<td>X</td>
<td>Norm_Distance 0</td>
<td>4</td>
<td>4</td>
<td>S M L</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>X</td>
<td>Norm_Distance 0</td>
<td>4</td>
<td>4</td>
<td>S M L</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LC</td>
<td>X</td>
<td>Norm_Distance 0</td>
<td>4</td>
<td>4</td>
<td>S M L</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PHI</td>
<td>X</td>
<td>Norm_Angle -1</td>
<td>1</td>
<td>0</td>
<td>LN NZ P LP</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Variable Name</td>
<td>Type</td>
<td>Unit</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
<td>Term Names</td>
</tr>
<tr>
<td>----</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>7</td>
<td>R</td>
<td>R</td>
<td>Norm_Distance</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>S M L S M L</td>
</tr>
<tr>
<td>8</td>
<td>RC</td>
<td>RC</td>
<td>Norm_Distance</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>S M L S M L</td>
</tr>
<tr>
<td>9</td>
<td>VFL</td>
<td>VFL</td>
<td>Norm_Distance</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>S M L S M L</td>
</tr>
<tr>
<td>10</td>
<td>VFR</td>
<td>VFR</td>
<td>Norm_Distance</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>S M L S M L</td>
</tr>
</tbody>
</table>

Table 2: Variables of Group “Inputs”

Fuzzification Methods
- Compute MBF
- Look up MBF
- Categorical Variable
- Display
- Fuzzy Input

## Outputs

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>alpha_11</td>
<td>alpha_11</td>
<td>Suitability</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>HF</td>
</tr>
<tr>
<td>12</td>
<td>alpha_12</td>
<td>alpha_12</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>HF</td>
</tr>
<tr>
<td>13</td>
<td>alpha_13</td>
<td>alpha_13</td>
<td>Suitability</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>A</td>
</tr>
<tr>
<td>14</td>
<td>alpha_14</td>
<td>alpha_14</td>
<td>Suitability</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>NA A F HF</td>
</tr>
<tr>
<td>15</td>
<td>alpha_15</td>
<td>alpha_15</td>
<td>Suitability</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>NA A F HF</td>
</tr>
<tr>
<td>16</td>
<td>alpha_21</td>
<td>alpha_21</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>HF</td>
</tr>
<tr>
<td>17</td>
<td>alpha_22</td>
<td>alpha_22</td>
<td>Suitability</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>A</td>
</tr>
<tr>
<td>18</td>
<td>alpha_23</td>
<td>alpha_23</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>NA A F HF</td>
</tr>
<tr>
<td>19</td>
<td>alpha_24</td>
<td>alpha_24</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>alpha_25</td>
<td>alpha_25</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>HF</td>
</tr>
<tr>
<td>21</td>
<td>alpha_31</td>
<td>alpha_31</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>NA A F HF</td>
</tr>
<tr>
<td>22</td>
<td>alpha_32</td>
<td>alpha_32</td>
<td>Units</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>NA</td>
</tr>
</tbody>
</table>
Table 3: Variables of Group "Outputs"

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>alpha_33</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>alpha_34</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>alpha_35</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>alpha_41</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>alpha_42</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>alpha_43</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>alpha_44</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>alpha_45</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

Defuzzification Methods

- Center of Maximum (CoM)
- Mean of Maximum (MoM)
- Center of Area (CoA)
- Hyper CoM
- Fuzzy Output

The default value of an output variable is used if no rule is firing for this variable. Different methods can be used for the defuzzification, resulting either into the 'most plausible result' or the 'best compromise'.

The 'best compromise' is produced by the methods:
- CoM (Center of Maximum)
- CoA (Center of Area)
- CoA BSUM, a version especially for efficient VLSI implementations

The 'most plausible result is produced by the methods:
- MoM (Mean of Maximum)
- MoM BSUM, a version especially for efficient VLSI implementations

3 Left

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>VFL</td>
<td></td>
<td>Norm_Distance</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FL</td>
<td></td>
<td>Norm_Distance</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Variable Name</td>
<td>Type</td>
<td>Unit</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
<td>Term Names</td>
</tr>
<tr>
<td>----</td>
<td>---------------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Norm_Distance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Variables of Group "Left"

<table>
<thead>
<tr>
<th>Fuzzification Methods</th>
<th>Defuzzification Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute MBF</td>
<td>Center of Maximum (CoM)</td>
</tr>
<tr>
<td>Look up MBF</td>
<td>Mean of Maximum (MoM)</td>
</tr>
<tr>
<td>Categorical Variable</td>
<td>Center of Area (CoA)</td>
</tr>
<tr>
<td>Display</td>
<td>Hyper CoM</td>
</tr>
<tr>
<td>Fuzzy Input/Fuzzy Output</td>
<td>Force</td>
</tr>
</tbody>
</table>

4 Center

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>RC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Variables of Group "Center"

5 Right

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>VFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>FR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>
### Table 6: Variables of Group “Right”

<table>
<thead>
<tr>
<th>#</th>
<th>Variable Name</th>
<th>Type</th>
<th>Unit</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
<th>Term Names</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nce</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td>L</td>
</tr>
</tbody>
</table>

#### Fuzzification Methods
- Compute MBF
- Look up MBF
- Categorical Variable
- Display
- Fuzzy Input/Fuzzy Output

#### Defuzzification Methods
- Center of Maximum (CoM)
- Mean of Maximum (MoM)
- Center of Area (CoA)
- Hyper CoM
- Force

#### 6. Input Variable “C”

![Figure 2: MBF of "C"](image)

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1), (0.5, 1), (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0), (0.5, 0), (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1), (3, 0), (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0), (2.5, 0), (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>

**Table 7: Definition Points of MBF “C”**

#### 7. Input Variable “FL”
**Figure 3: MBF of "FL"**

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>

*Table 8: Definition Points of MBF "FL"*

**8 Input Variable "FR"**

**Figure 4: MBF of "FR"**

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>

*Table 9: Definition Points of MBF "FR"*
9 Input Variable "L"

![Figure 5: MBF of "L"](image)

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>

Table 10: Definition Points of MBF "L"

.10 Input Variable "LC"

![Figure 6: MBF of "LC"](image)

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>
Table 11: Definition Points of MBF "LC"

.11 Input Variable "PHI"

![Figure 7: MBF of "PHI"](image)

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN</td>
<td>linear</td>
<td>(-1, 1) (-0.2, 1) (-0.1, 0) (1, 0)</td>
</tr>
<tr>
<td>N</td>
<td>linear</td>
<td>(-1, 0) (-0.2, 0) (-0.1, 1) (0, 0)</td>
</tr>
<tr>
<td>Z</td>
<td>linear</td>
<td>(-1, 0) (-0.1, 0) (0, 1) (0.1, 0)</td>
</tr>
<tr>
<td>P</td>
<td>linear</td>
<td>(-1, 0) (0.0, 0) (0.1, 1) (0.2, 0)</td>
</tr>
<tr>
<td>LP</td>
<td>linear</td>
<td>(-1, 0) (0.1, 0) (0.2, 1) (1, 1)</td>
</tr>
</tbody>
</table>

Table 12: Definition Points of MBF "PHI"

.12 Input Variable "R"

![Figure 8: MBF of "R"](image)
<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>

Table 13: Definition Points of MBF “R”

.13 Input Variable “RC”

![MBF of “RC”](image)

Figure 9: MBF of “RC”

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (2, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (2, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4, 1)</td>
</tr>
</tbody>
</table>

Table 14: Definition Points of MBF “RC”

.14 Input Variable “VFL”
Table 15: Definition Points of MBF "VFL"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (1.5, 0) (4, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 1) (0.5, 1) (1.5, 0) (4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (1.5, 1) (2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 0) (0.5, 0) (1.5, 1) (2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
</tbody>
</table>

Table 16: Definition Points of MBF "VFR"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>linear</td>
<td>(0, 1) (0.5, 1) (1.5, 0) (4, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 1) (0.5, 1) (1.5, 0) (4, 0)</td>
</tr>
<tr>
<td>M</td>
<td>linear</td>
<td>(0, 0) (0.5, 0) (1.5, 1) (2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 0) (0.5, 0) (1.5, 1) (2.5, 1) (3, 0) (4, 0)</td>
</tr>
<tr>
<td>L</td>
<td>linear</td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0, 0) (2.5, 0) (3, 1)</td>
</tr>
</tbody>
</table>

.15 Input Variable "VFR"

.16 Output Variable "alpha_11"
Figure 12: MBF of "alpha_11"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 17: Definition Points of MBF "alpha_11"

Output Variable "alpha_12"

Figure 13: MBF of "alpha_12"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 18: Definition Points of MBF "alpha_12"

Output Variable "alpha_13"
Figure 14: MBF of "alpha_13"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.9, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 19: Definition Points of MBF "alpha_13"

Output Variable "alpha_14"

Figure 15: MBF of "alpha_14"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>linear</td>
<td>(0, 1) (1, 0)</td>
</tr>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 20: Definition Points of MBF "alpha_14"

Output Variable "alpha_15"
Figure 16: MBF of "alpha_15"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>linear</td>
<td>(0, 1) (1, 0)</td>
</tr>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 21: Definition Points of MBF "alpha_15"

.21 Output Variable "alpha_21"

Figure 17: MBF of "alpha_21"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 22: Definition Points of MBF "alpha_21"

.22 Output Variable "alpha_22"
Figure 18: MBF of "alpha_22"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 23: Definition Points of MBF "alpha_22"

.23 Output Variable "alpha_23"

Figure 19: MBF of "alpha_23"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>linear</td>
<td>(0, 1) (1, 0)</td>
</tr>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 24: Definition Points of MBF "alpha_23"

.24 Output Variable "alpha_24"
### Figure 20: MBF of "alpha_24"

### Table 25: Definition Points of MBF "alpha_24"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

### Output Variable "alpha_25"

### Figure 21: MBF of "alpha_25"

### Table 26: Definition Points of MBF "alpha_25"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

### Output Variable "alpha_31"
Table 27: Definition Points of MBF "alpha_31"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>linear</td>
<td>(0, 1) (1, 0)</td>
</tr>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Figure 22: MBF of "alpha_31"

.27 Output Variable "alpha_32"

Table 28: Definition Points of MBF "alpha_32"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>linear</td>
<td>(0, 1) (1, 0)</td>
</tr>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Figure 23: MBF of "alpha_32"

.28 Output Variable "alpha_33"
Figure 24: MBF of "alpha_33"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) 0.7, 1 (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) 0.9, 1 (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) 1, 1 (1, 1)</td>
</tr>
</tbody>
</table>

Table 29: Definition Points of MBF "alpha_33"

Figure 25: MBF of "alpha_34"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) 1, 1</td>
</tr>
</tbody>
</table>

Table 30: Definition Points of MBF "alpha_34"

Output Variable "alpha_35"
Figure 26: MBF of \textit{"alpha}_35\textit{"}

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

*Table 31: Definition Points of MBF \textit{"alpha}_35\textit{"}*

31 Output Variable "alpha\_41"

Figure 27: MBF of \textit{"alpha}_41\textit{"}

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

*Table 32: Definition Points of MBF \textit{"alpha}_41\textit{"}*

32 Output Variable "alpha\_42"
Figure 28: MBF of "alpha_42"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 33: Definition Points of MBF "alpha_42"

Output Variable "alpha_43"

Figure 29: MBF of "alpha_43"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Table 34: Definition Points of MBF "alpha_43"

Output Variable "alpha_44"
Figure 30: MBF of "alpha_44"

Table 35: Definition Points of MBF "alpha_44"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Output Variable "alpha_45"

Figure 31: MBF of "alpha_45"

Table 36: Definition Points of MBF "alpha_45"

<table>
<thead>
<tr>
<th>Term Name</th>
<th>Shape/Par.</th>
<th>Definition Points (x, y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>linear</td>
<td>(0, 0) (0.4, 1) (1, 0)</td>
</tr>
<tr>
<td>F</td>
<td>linear</td>
<td>(0, 0) (0.7, 1) (1, 0)</td>
</tr>
<tr>
<td>HF</td>
<td>linear</td>
<td>(0, 0) (1, 1)</td>
</tr>
</tbody>
</table>

Rule Blocks
The rule blocks contain the control strategy of a fuzzy logic system. Each rule block confines all rules for the same context. A context is defined by the same input and output variables of the rules.

The rules' 'if' part describes the situation, for which the rules are designed. The 'then' part describes the response of the fuzzy system in this situation. The degree of support (DoS) is used to weigh each rule according to its importance.

The processing of the rules starts with calculating the 'if' part. The operator type of the rule block determines which method is used. The operator types MIN-MAX, MIN-AVG and GAMMA are available. The characteristic of each operator type is influenced by an additional parameter.

For example:

MIN-MAX, parameter value 0 = Minimum Operator (MIN)
MIN-MAX, parameter value 1 = Maximum Operator (MAX)
GAMMA, parameter value 0 = Product Operator (PROD)

The minimum operator is a generalization of the Boolean 'and'; the maximum operator is a generalization of the Boolean 'or'.

The fuzzy composition eventually combines the different rules to one conclusion. If the BSUM method is used all firing rules are evaluated, if the MAX method is used only the dominant rules are evaluated.

.1 Rule Block "Front_Avoid"

<table>
<thead>
<tr>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation:</td>
</tr>
<tr>
<td>Parameter:</td>
</tr>
<tr>
<td>Result Aggregation:</td>
</tr>
<tr>
<td>Number of Inputs:</td>
</tr>
<tr>
<td>Number of Outputs:</td>
</tr>
<tr>
<td>Number of Rules:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>LC</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>
### Rule Block "Goal_Seeking"

**Parameter**
- Aggregation: MINMAX
- Parameter: 0.00
- Result Aggregation: MAX
- Number of Inputs: 1
- Number of Outputs: 5
- Number of Rules: 25

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LN</td>
<td>1.00</td>
</tr>
<tr>
<td>N</td>
<td>1.00</td>
</tr>
<tr>
<td>Z</td>
<td>1.00</td>
</tr>
<tr>
<td>P</td>
<td>1.00</td>
</tr>
<tr>
<td>LP</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 38: Rules of the Rule Block "Goal_Seeking"

### Rule Block "Left_Edge_Tracking"

**Parameter**
- Aggregation: MINMAX
- Parameter: 0.00
- Result Aggregation: MAX
- Number of Inputs: 3
- Number of Outputs: 5
- Number of Rules: 135

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>S</td>
<td>L</td>
</tr>
<tr>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>S</td>
<td>L</td>
</tr>
</tbody>
</table>

Table 39: Rules of the Rule Block "Left_Edge_Tracking"
Table 39: Rules of the Rule Block "Left_Edge_Tracking"

4 Rule Block "Right_Edge_Tracking"

Parameter
Aggregation: MINMAX
Parameter: 0.00
Result Aggregation: MAX
Number of Inputs: 3
Number of Outputs: 5
Number of Rules: 135
Table 40: Rules of the Rule Block "Right_Edge_Tracking"

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>L S M</td>
<td>1.00 F 1.00 HF 1.00 A 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L S L</td>
<td>1.00 HF 1.00 HF 1.00 A 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L M S</td>
<td>1.00 NA 1.00 A 1.00 F 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L M M</td>
<td>1.00 F 1.00 HF 1.00 F 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L M L</td>
<td>1.00 HF 1.00 HF 1.00 F 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L L S</td>
<td>1.00 NA 1.00 A 1.00 HF 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L L M</td>
<td>1.00 F 1.00 HF 1.00 HF 1.00 HF 1.00 HF</td>
</tr>
<tr>
<td>L L L</td>
<td>1.00 HF 1.00 HF 1.00 HF 1.00 HF 1.00 HF</td>
</tr>
</tbody>
</table>

.5 Settings

Global Options
- Base Variable Data Type: Double Precision
- Computation Options: Fast CoA

Code Generator Options
- Settings: Public Input and Output
- Online Code
- Comments
- Online Code Password Protection
- Compatibility: ANSI C

Online Options
- Communication Channel: OCC (Online Communication Channel): C:\Program Files \INFORM\fuzzyTECH 5.5\Fttcip.dll
- Refresh Time: 55 ms
- Timeout: 1100 ms
APPENDIX B

C++ CODE

The C++ code is ...
This program implements a Multi-valued Fuzzy Behavior Control System for Navigation in Cluttered Environments on a Pioneer 2 mobile robot equipped with a SICK laser range finder.

//Include fuzzyTECH generated files and libraries
extern "C" { 
#include "FT_ST_IN.h"
}
#ifdef FT_ONLINE
extern "C" { 
#include "online.h"
}
#endif
#include "Aria.h"
#include <stdio.h>
#include <math.h>
#include <sys/time.h>

//Function prototypes
double min(double dist[],int size);
int absmaxloc(double dist[],int size);
double abs(double value);

int main()
{
    //Declare variables that may be read from user
    double steer_scale,vel_scale,init_vel,input_scale,igoaldist,igoalangle;
    steer_scale=35;
    vel_scale=0;
    input_scale=0.7;

    printf("Enter the initial velocity(cm/s): ");
    scanf("%lf",&init_vel);
    printf("Enter the initial goal distance(mm): ");
    scanf("%lf",&igoaldist);
    printf("Enter the initial goal angle(deg ccw+): ");
    scanf("%lf",&igoalangle);

    **************************** Pioneer 2: Aria Interface Code ****************************

    // whether to use the sim for the laser or not, if you use the sim for the laser
    // for the robot, you have to use the sim for the robot too
    bool useSim = false;
    // robot
    ArRobot robot;
    // the laser
    ArSick sick;
    // sonar device
    ArSonarDevice sonar;
    // connection
ArDeviceConnection *con;
// Laser connection
ArSerialConnection laserCon;
ArTime start;

// configure the laser before we make the logger
/*sick.configureShort(useSim, ArSick::BAUD38400, ArSick::DEGREES180,
ArSick::INCREMENT_HALF);*/
sick.configure(0, 0, 0, ArSick::BAUD38400, ArSick::DEGREES180,
ArSick::INCREMENT_HALF);

// mandatory init
Aria::init();
//printf("Pausing 5 seconds so you can disconnect VNC if you are using
// it.
\n");
//ArUtil::sleep(5000);
// attach the laser to the robot
robot.addRangeDevice(&sick);
// if we're not using the sim, make a serial connection and set it up
if (!useSim)
{
    ArSerialConnection *serCon;
    serCon = new ArSerialConnection;
    serCon->setPort();
    con = serCon;
}
// if we are using the sim, set up a tcp connection
else
{
    ArTcpConnection *tcpCon;
    tcpCon = new ArTcpConnection;
    tcpCon->setPort();
    con = tcpCon;
}
if (ArModuleLoader::load("libArInertial", &robot, NULL, true) == 0)
{
    printf("Loaded the base inertial library\n");
    if (ArModuleLoader::load("ISense_Mod", &robot,(void ")2", true)==0)
    {
        printf("The ISense inertial module has been loaded and should
be correcting heading now.\n");
    }
    else if (ArModuleLoader::load("ISIS_Mod", &robot,(void *)ArUtil::COM2,
false) == 0)
    {
        printf("The ISIS inertial module has been loaded and should be
correcting heading now.\n");
    }
}
// set the connection on the robot
robot.setDeviceConnection(con);
// try to connect, if we fail exit
if (!robot.blockingConnect())
{
    printf("Could not connect to robot... exiting\n");
Aria::shutdown();


97
return 1;
}
// if we're not using the sim, set up the port for the laser
if (!useSim)
{
    #ifdef WIN32
        laserCon.setPort("COM3");
    #else
        laserCon.setPort("/dev/ttyS2");
    #endif
    sick.setDeviceConnection(&laserCon);
}
// give p2os a command to make it respond to the on-microcontroller
// joystick port
//robot.comInt(ArCommands::JOYDRIVE, 1);

// This must be created after the robot is connected so that it'll
// get the right laser pos
//ArSickLogger logger(&robot, &sick, 300, 25, filename.c_str());
sick.setSensorPosition(0,0,0);

// Make a key handler, so that escape will shut down the program
// cleanly
ArKeyHandler keyHandler;

// Add the key handler to Aria so other things can find it
Aria::setKeyHandler(&keyHandler);

// Attach the key handler to a robot now, so that it actually gets
// some processing time so it can work, this will also make escape
// exit
robot.attachKeyHandler(&keyHandler);

// add a keydrive action so that the robot can be driven with a
// keyboard as well, toss in a joystick one too, just in case
// someone does get a joystick working on a robot somehow

/*ArActionJoydrive joydrive;
joydrive.setStopIfNoButtonPressed(false);
ArActionKeydrive keydrive;
robot.addAction(&joydrive, 100);
robot.addAction(&keydrive, 99);*/

// run the robot, true here so that the run will exit if connection lost
robot.runAsync(true);

// now that we're connected to the robot, connect to the laser
sick.runAsync();

// connect to the laser
if (!sick.blockingConnect())
{
    printf("Could not connect to SICK laser... exiting\n");
    robot.comInt(ArCommands::SOUND, 13);
    Aria::shutdown();
}
return 1;
}

// enable the motors, disable amigobot sounds
robot.comInt(ArCommands::SONAR, 1);
robot.comInt(ArCommands::ENABLE, 1);
robot.comInt(ArCommands::SOUND, 32);
robot.comInt(ArCommands::SOUNDTOG, 0);
robot.comInt(ArCommands::JOYDRIVE, 1);
// just hang out and wait for the end
//robot.waitForRunExit();

ArUtil::sleep(1000); //give the laser time to get a reading

/********************** Navigation Code Begins **************************/
double rob_length=520;  //Robot length (mm)
double rob_w=500;  //Robot width (mm): Used as tuning variable
int t=0;   //Execution loop timer
int tlimit=500;  //Maximum execution loops
double Dx,Dy,Dth,Th //Relative x,y,th w/r/t goal; Angle to goal
double tol=200;  //goal seeling tolerance for code termination
double goal[2]=igoaldist*cos(igoalangle*3.1416/180),
    igoaldist*sin(igoalangle*3.1416/180);
double goaldist=igoaldist;  //Distance to goal
double scale=input_scale/rob_length;  //Range sensor scaling factor
double current_vel=init_vel;  //Current velocity
double steer_scale2;   //Secondary steering scale factor
double max_reading=4000;  //Maximum distance for laser sensing regions
double vfl,fl,l,lc,c,rc,r,fr,vfr; //Range sensor variables
double THs;   //Scaled steering command for execution
double orient;  //Orientation of robot in global coordinates
int t2;   //Timer for command execution
double loc[5]=-1, -.5, 0, .5, 1; //Locations of output singletons
double fuzdth[5];  //Degrees of membership for defuzzification
double sum1[5];  //Defuzzification variable
double sum2[5];  //Defuzzification variable
double Dtheta;  //Normalized steering command
double A[5];  //Defuzzification variable: Areas
int k,m,n,index;  //Index variables
double deflimit=.2; //Defuzzification threshold to separate areas
double LRT,SRT,NT,SLT,LLT;  //Ouput command alternatives
long timel;   //Timer variable
FLAGS rv;   //Fuzzy system execution variable

#endif FT_ONLINE
initonline();
#endif
initft_st_in();

//Loop continuously until time limit or reach goal position
while (goaldist>tol && t<tlimit)
{
  t2=0;
  //Obtain range sensor measurements
vfr=sick.currentReadingPolar(-80,-45,NULL);
fr=sick.currentReadingPolar(-60,-25,NULL);
r=sick.currentReadingPolar(-45,-10,NULL);
rc=sick.currentReadingBox(250,-rob_w/6,max_reading,-rob_w/2,NULL);
c=sick.currentReadingBox(250,rob_w/6,max_reading,-rob_w/6,NULL);
lc=sick.currentReadingBox(250,rob_w/6,max_reading,rob_w/2,NULL);
l=sick.currentReadingPolar(10,45,NULL);
fl=sick.currentReadingPolar(25,60,NULL);
vfl=sick.currentReadingPolar(45,80,NULL);

//Prefuzzification measurement modifications
if(fr*scale<.7)
{
    r=0.5/scale;
    fr=0.5/scale;
}
if(r*scale<.7)
{
    fr=0.5/scale;
    r=0.5/scale;
    rc=0.5/scale;
}
if(l*scale<.7)
{
    lc=0.5/scale;
    l=0.5/scale;
    fl=0.5/scale;
}
if(fl*scale<.7)
{
    l=0.5/scale;
    fl=0.5/scale;
}
if(c*scale<.75)
{
    lc=c;
    rc=c;
}

//Scale range sensor inputs
vfl=vfl*scale;
fl=fl*scale;
l=l*scale;
lc=lc*scale;
c=c*scale*.9;
rc=rc*scale;
r=r*scale;
fr=fr*scale;
vfr=vfr*scale;

//Declare fuzzy system range inputs
VFL_ft_st_in=vfl;
FL_ft_st_in=fl;
L_ft_st_in=l;
LC_ft_st_in=lc;
C_ft_st_in=c;
RC_ft_st_in=rc;
R_ft_st_in=r;
FR_ft_st_in=fr;
VFR_ft_st_in=vfr;

//Obtain current position
Dx=goal[0]-robot.getX();
Dy=goal[1]-robot.getY();
//Calculate angle and distance to goal
Th=atan2(Dy,Dx);
goaldist=pow(pow(Dx,2)+pow(Dy,2),.5);

orient=robot.getTh()*(3.1416/180);  //Obtain robot orientation
Dth=Th-orient; //Calculate orientation to goal angle
PHI_ft_st_in=Dth/(3.1416);  //Normalize PHI by PI

rv=ft_st_in();  //Execute fuzzy system

//Collect output terms for intersection (min) operation
double alpha1[]={alpha11_ft_st_in, alpha12_ft_st_in,
alpha13_ft_st_in, alpha14_ft_st_in};
double alpha2[]={alpha21_ft_st_in, alpha22_ft_st_in,
alpha23_ft_st_in, alpha24_ft_st_in};
double alpha3[]={alpha31_ft_st_in, alpha32_ft_st_in,
alpha33_ft_st_in, alpha34_ft_st_in};
double alpha4[]={alpha41_ft_st_in, alpha42_ft_st_in,
alpha43_ft_st_in, alpha44_ft_st_in};
double alpha5[]={alpha51_ft_st_in, alpha52_ft_st_in,
alpha53_ft_st_in, alpha54_ft_st_in};

//Combine alphas to determine preferences for each command alternative
LRT=min(alpha1,4);
SRT=min(alpha2,4);
NT=min(alpha3,4);
SLT=min(alpha4,4);
LLT=min(alpha5,4);

//Increase steering scale when command due to alpha13
SLT==alpha4[3] && LLT==alpha5[3])
   steer_scale2=1.5;
else
   steer_scale2=1;

//Declare defuzzification variables
for(n=0;n<5;n++)
{
   sum1[n]=0;
   sum2[n]=0;
   A[n]=0;
}
k=0;
m=0;
fuzdth[0]=LRT;
fuzdth[1]=SRT;
fuzdth[2]=NT;
fuzdth[3]=SLT;
fuzdth[4]=LLT;

//CLA singleton defuzzification
while(k<5)
{
    while(k<5 && fuzdth[k] > deflimit)
    {
        sum1[m]+=fuzdth[k]*loc[k];
        sum2[m]+=fuzdth[k];
        A[m]=sum1[m]/sum2[m];
        k++;
    }
    k++;
    m++;
}

index=absmaxloc(sum2,5);
Dtheta=A[index];

THs=Dtheta*steer_scale*steer_scale2; //Scale steering output

//Execute Control Commands
robot.setVel(current_vel);
robot.setDeltaHeading(THs);
start.setToNow();

while (!robot.isHeadingDone(1) && t2 < 25)
{
    ArUtil::sleep(100);
    t2++;
}

if(timel < 2000)
    ArUtil::sleep(2000-timel);

# ifdef FT_ONLINE
    online();
# endif

# ifdef FT_ONLINE
    closeonline();
# endif

printf("Out of main loop !!!");

/*************Disconnect from Laser and Pioneer 2 **************/

sick.lock();
sick.disconnect();
//sick.setRobot(NULL);
sick.unlock();

// start the robot running, true so that if we lose connection the run stops
robot.run(true);
// now exit
Aria::shutdown();
return 0;
}  
/********************* Out of Main Loop *******************/
/************************ Function Definitions *****************/
double min(double dist[], int size)
{
  int t=1;
  double temp=dist[0];

  while (t<size)
  {
    if(temp>dist[t])
    {
      temp=dist[t];
      t++;
    }
  }
  return temp;
}

//Determines the index location of the maximum absolute value in a vector
int absmaxloc(double dist[], int size)
{
  double abs[10];
  for(int i=0;i<size;i++)
  {
    if(dist[i]<0)
      abs[i]=-dist[i];
    else
      abs[i]=dist[i];
  }
  int t=1;
  int loc=0;
  double temp=dist[0];
  while(t<size)
  {
    if(abs[t]>temp)
      {
        temp=abs[t];
        loc=t;
      }
    t++;
  }
  return loc;
}

double abs(double value)
{
  if(value >= 0)
    return value;
  else
    return -value;
}
REFERENCES


BIOGRAPHICAL SKETCH

Damion Donyell Dunlap

Damion Donyell Dunlap was born on September 30th, 1980 in Charlotte, North Carolina. After completing high school he attended Florida A&M University in Tallahassee Florida with a full academic scholarship to obtain his Bachelor’s Degree in Mechanical Engineering. During his undergraduate studies, Damion interned with Saturn Corporation and General Motor’s Racing Division and was inducted into the Tau Beta Pi engineering honor society. He graduated Suma Cum Laude in 2002 and decided to stay at Florida A&M for his graduate studies. Damion is a recipient of the Graduate Assistance in Areas of National Need (GAANN) fellowship and the Computer Science, Engineering, & Mathematics Scholarship (CSEMS). His research interests are primarily in the areas of intelligent control and mobile robotics.