BasketBallBot: education level development of a fuzzy controller for a linear motor under saturation limits

David L. McPherson
University of Tennessee at Chattanooga, pjp613@mocs.utc.edu

Follow this and additional works at: http://scholar.utc.edu/honors-theses
Part of the Electrical and Computer Engineering Commons

Recommended Citation
BasketBallBot: Education Level Development of a Fuzzy Controller for a Linear Motor under Saturation Limits

Author: David L. McPherson

Departmental Honors Thesis

The University of Tennessee at Chattanooga
Electrical Engineering
Project Director: Dr. Abdul R. Ofoli
Examination Date: 3/24/15

Names of Members of Examination Committee

[Signatures]

____________________________________
Project Director, Abdul R. Ofoli

____________________________________
Department Examiner, Thomas D. Loveless

____________________________________
Department Examiner, Raga Ahmed

____________________________________
Liaison, Departmental Honors Committee, Dr. Jeremiah Tate

____________________________________
Chair, Departmental Honors Committee
# Table of Contents

I. INTRODUCTION ............................................................................................................... 5

   A. Fuzzy Logic Controller Introduction ........................................................................ 7

   B. Paper Outline ............................................................................................................. 8

II. PROBLEM STATEMENT ................................................................................................. 9

III. SENSOR INTEGRATION ............................................................................................... 11

   C. Sonar Rangefinder ................................................................................................... 11
Sonar Rangefinder Interfacing ........................................................................................... 11
Sonar Rangefinder Usage .................................................................................................. 13
Sonar Rangefinder Testing ............................................................................................... 13

D. Encoder ........................................................................................................................... 17
Encoder Interfacing ............................................................................................................ 17
Encoder Usage ................................................................................................................... 19
Encoder Testing ................................................................................................................. 19

E. Autonomous, Camera Controlled Aiming ..................................................................... 20

IV. CONTROLLER DESIGN .............................................................................................. 21

F. Control Topology ........................................................................................................... 22

G. Fuzzy Logic Controller Designs .................................................................................... 24
Basic Linear Interpolation Controller (Controller A) ....................................................... 25
Smoothed Bang-Bang Controller (Controller B) .............................................................. 27
Smoothed 3-Point Switching Element Controller (Controller C) ..................................... 29

H. Linear Best Fit Designs ............................................................................................... 32

I. Controller Implementation Code .................................................................................... 33

V. SYSTEM MODELING FOR SIMULATION ................................................................... 33

J. Plant Model ................................................................................................................... 33
Abstract—Fuzzy logic seeks to express human modes of reasoning and decision making in a mathematical form. This is evident in its terminology such as “linguistic variables” defined over a “universe of discourse”. By taking human expressions such as “very high” or “pretty cold” and defining them in a mathematical context, expert operator knowledge can be transferred from verbal descriptions into automated control algorithms regardless of the operator’s familiarity with control systems. Because fuzzy logic is designed to be easily comparable with human thought, it makes an excellent first exposure to control systems concepts to high school and undergraduate students. Additionally, one of the barriers preventing widespread industry use of fuzzy controls is that the emerging workforce is not familiar enough with fuzzy controls to successfully operate a fuzzy system. This work will demonstrate the suitability of fuzzy controls for education at the undergraduate level through the development of BasketBallBot. BasketBallBot uses the educational platform distributed to high schools throughout the country through the FIRST (For Inspiration and Recognition of Science and Technology) Robotics Competition (FRC). Inexpensive sensors are added to the robot and interfaced using the easily accessible Arduino platform. The affordability of the sensors and prevalence of the computing and hardware platforms insure that this work could be recreated at other undergraduate institutions and even high schools. This paper will thoroughly describe the sensor integration process. It also describes a heuristic technique for developing fuzzy logic controllers and inference systems that does not require a high level of mathematics to use. This technique is employed to design several controllers
and a fuzzy inference system. These controllers’ performance is investigated through simulation and experiment. Finally, the fuzzy inference system is developed that prescribes the desired ball launch speed given the distance to the hoop.

I. INTRODUCTION

The concept for fuzzy logic and control originated with Dr. Lotfi Zadeh’s seminal work on fuzzy sets in 1965 [1]. However, it still has not gained widespread industry acceptance equal to that of classical controls. This is largely due to current field engineers’ and technicians’ unfamiliarity with fuzzy controllers and a preference to resort to the tried-and-true PID controllers. This problem should be easily overcome since fuzzy controllers are designed to mimic human experts’ thinking and be understood intuitively. Fuzzy controllers are ideal for controlling systems that are poorly modeled due to non-linearity or high complexity. This is because fuzzy controllers are designed using skilled operators’ or experts’ intuition to create a heuristic design.

Mixed fuzzy and PID controllers have been used in complex variable reluctance motors [2] due to this type of motors’ nonlinear flux hysteresis and saturation. This development used fuzzy logic as an adaptive tuner for the PID controller building on the heuristic tuning methods commonly used in industry. Liu and Song [2] also used the fuzzy controller in the loop alongside the PID controller to make a fuzzy-PID controller. This type of hybrid controller has also been used to control the temperature in large thermostatic systems [3] due to this systems’ inherent system lags.
Pure Fuzzy Logic Controllers (FLCs) have also been used to control nonlinear systems standalone from classical PID controllers. Mrad et. al leveraged FLC’s independence from modeling to create a robust industrial motor controller [4]. Ji et al. [5] used a FLC to solve the classic inverted pendulum problem.

Fuzzy controls have also seen a major introduction into industrial production applications. Jaiswal and Kumar [6] used a FLC to control a 3 degree of freedom robotic arm like those used in factories. Fuzzy controllers have also been used to perform precision screw fastening in industrial settings [7]. A current disadvantage of fuzzy controllers is the amount of processing required to compute the control output. Huang and Hu [8] seek to ameliorate this problem using the grey predictive algorithm’s prediction as an input rather than the directly sensed state. The issues facing fuzzy control are being removed through rigorous research. However, without a workforce trained and familiarized with the concept of fuzzy controllers, the industry will never be able to adopt the FLC as widely as it has the PID controller.

This work will implement a fuzzy controller on a control system used in the national high school education competition, the FIRST Robotics Competition. By implementing a fuzzy controller on educational hardware and software, this paper will provide a case study in fuzzy controls education. The CRIO controller used in this competition combines a low-level FPGA and a processor to afford both flexibility of inputs and computational power. This hardware is used in industry as well as education, making the control software developed in this research easily applicable to other fields. Finally,
National Instruments graphical programming language, LabView, will be used to allow easy dissemination of the fuzzy control scheme to high school FRC teams as well as other UTC students. This work will use a heuristic design methodology to design three different fuzzy designs. More classical linear controllers will also be investigated. These controllers will be compared in simulation to assess the best design as well as which controllers’ behavior could not be reproduced using classical controllers. The fuzzy controllers will then also be implemented on the robotic platform “BasketBallBot”. By doing this, this work will be a case study in how fuzzy controls can be taught to engineering undergraduates or even high school students using easily accessible educational controls hardware and software.

A. Fuzzy Logic Controller Introduction

In fuzzy logic the continuous range \([0,1]\) of truth values is used to replace the discrete set \(\{0,1\}\) of classical logic. This replacement can be understood as allowing “shades of gray” in the truth of a statement. While a bang-bang controller (defined on crisp sets) will have step changes in control output, a fuzzy controller (defined over fuzzy sets) will have more smooth transitions. In classical set theory, a number either belongs to a set or does not. In fuzzy set theory, a number can “mostly” belong to a set or “sort of” belong to a set with the truth value of the statement “number x belongs to set M ” of 0.8 or 0.3 instead of just 1 or 0. The truth value of the statement “number x belongs to set M ” is called the membership degree of x to M and is denoted as \(\mu_M(x)\). In a fuzzy
logic controller, crisp sensor inputs are checked to determine their membership degrees to various input sets called the “antecedents” (in analogy to logical statements’ antecedents and consequents). The fuzzy truth of the antecedents is then transferred to the fuzzy truth of various consequent statements concerning the desired output value. This transference is implemented through a fuzzy inference system or FIS. The consequents are themselves fuzzy sets over values of the output variable v. The “defuzzification” process is used to obtain a crisp output from these fuzzy consequents. This crisp output is then applied to the plant.

B. Paper Outline
The rest of this paper is structured as follows. Section II defines the challenge BasketBallBot is meant to solve using fuzzy controls and LabView. Section III details the sensors integrated onto the robot platform to provide ample decision and control data for the robot to make a reliable basketball shot. Section IV lays out the control topology including constraints from the robot’s power system, a heuristic method for designing Mamdani type fuzzy controllers, and three fuzzy controller designs using this heuristic. In addition, Section IV sets up an investigation on a simple control algorithm that is tuned to approximate each of the three fuzzy designs. If equivalent performance can be achieved using the extremely simple algorithm version as the computationally complex fuzzy logical one, then that fuzzy design will be abandoned. Section V covers the shooter motor model and parameter extraction used to develop a simulation of the
controller alternatives. Section VI investigates the simulated responses of the controller designs to pick a preliminary best choice amongst the designs. Section VI also explores the replaceability of the fuzzy designs with the simpler algorithm explored in Section IV. Section VII validates that the controllers work practically on the robotic platform and corroborates Section VI’s results. Section VIII investigates two algorithms for taking the distance measurement to the hoop and determining the corresponding shoot speed for netting the basketball.

II. PROBLEM STATEMENT
The FIRST organization posts a yearly challenge to technically minded high school students across the nation. In 2012, this challenge was to create a tele-operated robot to collect foam basketballs and shoot them into a hoop from various positions on the playing court. The challenge was meant to inspire students to investigate and research robotics fundamentals and learn what a technical career may entail. This research takes a robot designed during that year (seen in Figure 1 and advances its challenge even further. Rather than having the robot shoot hoops under human control, this work seeks to enable the robot to aim, assess distance, and shoot the basketball all completely autonomously. This will be accomplished using camera vision for aiming (developed in a previous work at UTC), a sonar range finder for assessing the distance, and a fuzzy logic controller with encoder speed data feedback for shooting the basketball. The BasketBallBot constructed for the 2012 FIRST Robotics Competition...
(FRC) was designed to be tele-operated. Therefore to implement the autonomous control goals already stated, the research team also had to retrofit the BasketBallBot with sensors.

The BasketBallBot was built to be controlled to pick up a basketball and shoot it into a basketball hoop to score. The design was broken up into three subsystems: The collector, feeder, and shooter. The collector gathers the balls and provides them to the feeder. The feeder then gains control of the ball and lifts it to where the shooter is located. The shooter system then launches the ball towards the hoop. The shooter system is the focus of this research. The shooter consists of a DC motor connected to two 6 inch wheels. When a ball is fed next to the spinning wheels, the ball is launched along a curved guide out of the robot towards the goal.

The robot is controlled using National Instruments’ (NI) Compact Reconfigurable Input Output (CRIO) FPGA and processing unit. A cRIO 9403 32-channel digital I/O module will be used along with a FIRST provided Digital Sidecar (AndyMark-0866) to interface with the sonar range finder circuitry and quadrature encoder. It will be programmed using LabView, since this language is promoted across all the FIRST programs and is familiar to students through elementary school because of this. Using this graphical programming language will allow students to more easily learn about fuzzy controls using the code developed through this research as an example.
III. SENSOR INTEGRATION

C. Sonar Rangefinder

The BasketBallBot uses a Maxbotix XL-MaxSonar-EZ2 Sonar Range Finder (MB1220) to determine the distance from the robot’s shooter motor to the basketball hoop.

Sonar Rangefinder Interfacing

This range finder operates at 42kHz and communicates the sensed distance to the master controller using any one of several communication interfaces (serial, scaled analog, scaled pulsewidth). The scaled pulsewidth method due to its relative simplicity over serial and relative stability over the analog readings. As the sonar range finder constantly sends out ultrasonic (US) pulses and receives the echoes, it sends out a squarewave output whose square pulses’ durations are directly proportional to the currently read distance. Originally the researchers attempted to have the CRIO read the pulsewidth coming from the range finder. A local variable was used to record the previous value of the incoming signal. When the incoming signal changed and no longer equaled the previously stored value, the CRIO would note the time as a rise or fall time depending on the new signal value. When a fall time was observed the difference between the recorded rise time and fall time would be calculated as the pulsewidth. Unfortunately, the main loop executed at a rate on the order of a few milliseconds. This gross time sample was too large to distinguish the 59 microsecond steps in the incoming signal’s pulsewidth.
Instead a slave Arduino microcontroller is used to read the pulsewidth and report the pulsewidth to the master CRIO using the inter-integrated circuit (I2C) communication protocol. Since the Arduino is a lower level processor, it was easier for the student to program it to react to real-time events such as a pulse’s microsecond scale duration. Although alternative methods likely exist for the CRIO to read the ultrasonic sensor, this method was easy and fast to implement for a student already familiar with Arduino.

The master CRIO - slave Arduino circuit is described in Figure 2.

**Figure 2- Wiring Diagram for I2C Protocol Master-Slave Configuration and Electronics Wiring**

The Worcester Polytechnic Institute’s (WPI) LabView library was employed to handle the I2C communication between the CRIO and the Arduino. The code used to handle the
I2C interfacing will be shown and discussed further in Section IX when the ultrasonic sensor’s data will be fed into one of the algorithms briefly described below in the section “Sonar Rangefinder Usage”. The Arduino code for measuring the sonar rangefinder’s pulsewidth and reporting the reading to the master CRIO via the I2C protocol is in Appendix B.

Sonar Rangefinder Usage

The detected range from the sonar range finder is fed into one of two algorithms to determine the speed desired to shoot the ball into the hoop. One algorithm uses launch angle information and kinematics to derive an analytic formula for desired launch speed in terms of distance from shooter motor to the hoop’s center. The second algorithm uses a more empirical approach. The BasketBallBot is placed at a number of distances away from the hoop and the corresponding launch speed that consistently nets the basketballs is tabulated. This table is used as a lookup table for the BasketBallBot and uses fuzzy inference to interpolate between table values to find the approximate desired launch speed. The derivations for both methods will be described in more depth in Section IX.

Sonar Rangefinder Testing

To assess whether the installed ultrasonic range finder worked correctly, the robot was placed at four distances away from a wall. The ultrasonic ranger’s output was polled for approximately 4 seconds at these four locations. The average distance reading was calculated and compared to the true distance as read by a tape measure.
The ultrasonic ranger was polled for 4.2 seconds at each of four distances. The tape measured distances are recorded in Table I. The average ultrasonic found range for the 4.2 second interval is tabulated besides the column of the tape measured distance. This time of 4.2 seconds collects 211 ultrasonic reading samples for one distance. The percent error between the two methods of measuring the distance is calculated in the rightmost column.

Table I: Ultrasonic Ranger vs. Tape Measure Results

<table>
<thead>
<tr>
<th>Tape Measured Distance (cm)</th>
<th>Averaged Ultrasonic Measured Distance (cm)</th>
<th>Percent Error (%)</th>
<th>Variance in Ultrasonic Sample (cm^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>141</td>
<td>0.7%</td>
<td>1.00</td>
</tr>
<tr>
<td>234</td>
<td>241</td>
<td>2.9%</td>
<td>143.48</td>
</tr>
<tr>
<td>276</td>
<td>277</td>
<td>0.4%</td>
<td>9.50</td>
</tr>
<tr>
<td>368</td>
<td>372</td>
<td>1.1%</td>
<td>14.78</td>
</tr>
</tbody>
</table>

Table I shows that the average percent error between the ultrasonic sensors and the ground truth (as represented by the tape measure reading) is 1.3%.
Note that the tape measured 234 cm case has an especially poor signal. This signal has 6 significant spikes in the output whereas the other signals have no spikes. These spikes can be caused by the ultrasound pulses missing the desired target for measuring distance from. This in turn can be caused by the ultrasonic sensor being poorly positioned or aimed. Even in this poor case, however, the percent error was only 2.9%.

The individual ultrasonic output traces for the 4.2 second interval are included below. The average value (tabulated in Table I) has low percent errors with the value obtained via the tape measure. The ultrasonic reading varies with respect to time. The value of this variance trends to be larger with larger distances as seen in Table I.
Although the ultrasonic range finder’s output varies slightly with respect to time, the standard deviation is a small percent of the total reading (at the largest 4.97%). The errors caused by the fluctuating deviations can be filtered out by utilizing a moving average.
The variance in the ultrasonic distance signal is small, but can also be dealt with using an average. By taking the average of a sample of ultrasonic readings, the percent error between true distance and sensed distance can be kept below 3% as shown from the results in Table I. Thus a valuable estimate of the distance from the sensor to a large object can be obtained from the ultrasonic sensor data. This legitimizes its use in the BasketBallBot system.

D. Encoder

A US Digital E4P, 250 counts/revolution quadrature encoder is attached to the shooter motor to sense the shooting speed for use in feedback control.

Encoder Interfacing

The Worcester Polytechnic Institute’s (WPI) robotics library for FIRST LabView was used to interface with the quadrature encoder. The quadrature encoder signals can wire directly into the digital sidecar’s pins as shown in Figure 2. The encoder directly measures absolute wheel angular position. The encoder data is then manipulated according to the simplified block diagram in Figure 5. This block diagram is executed inside a program loop that executes every 20 milliseconds.

![Figure 5: Block Diagram for Encoder Filtering](image)
At the end of each of the program loops that contain the encoder code, the currently read value for encoder position is shifted into the shift register.

Angular velocity can be calculated from this reading by subtracting a prior reading from the current one and dividing by the time period between the two readings. This will return the average speed over the time period. This averaging will effectively perform a filtering function on the velocity simultaneously with calculating the velocity. The LabView code for interfacing with the encoder and conditioning the data (as dictated from the algorithm in Figure 5) is shown in the Appendix A.

The BasketBallBot design provided to the research team lacked an encoder on the shooter system. Standard FIRST encoders are designed to mount onto 1/4” shafts, while the BasketBallBot’s shooter motor shaft was 3/8”. This required fabrication of a mechanical coupler to step down the shaft diameter from 3/8” to 1/4”. This coupler was additively manufacture, also known as 3D printed. A mount was designed and additively manufactured to hold the encoder in place over the coupler as shown in Figure 6.
Encoder Usage
The raw encoder counts from the US Digital encoder are used as the base angular unit to maintain
the highest level of angular resolution available in an integer format. The US Digital E4P encoder
has 250 ticks on its optical wheel, making the conversion from encoder count to revolutions 250
counts/revolution. This conversion will be verified in the following encoder testing subsection.

The encoder speed data will be used to implement a feedback control loop to maintain
a consistent speed while throwing basketballs. Multiple designs for the controller
process will be discussed in Section V.

Encoder Testing
A marker was placed on one point on the rim of the shooter wheel. The wheel was
turned approximately 10 revolutions using visual tracking of the marker to assess current
wheel rotation. The encoder’s read wheel rotation in encoder counts was divided by 10
to produce the average wheel rotation counts per revolution. The resulting conversion factor corroborated the nominal conversion of 250 counts/revolution with a 2% error. This error is likely due to the crude visual estimation method used in assessing whether the wheel had turned 10 revolutions.

With the encoder rotational distance data confirmed as valid, the encoder speed data calculated from it can be considered valid as well. The encoder distance reading from the loop occurring 100 milliseconds prior is subtracted from the current encoder distance reading and then divided by the 100 millisecond time delay. This produces the average encoder speed for the 100 milliseconds between the earlier reading and the current one. This moving average removes unwanted spikes and noise sourcing from the encoder itself, although noise due to the mechanical system’s vibrations are still present.

### E. Autonomous, Camera Controlled Aiming

The Axis M1011 Camera is integrated with the CRIO to allow machine vision applications. A previous team of undergraduate UTC researchers implemented a basic machine vision algorithm for aligning the robot’s shooter system with the goal. This system uses the National Instruments Vision Assistant to compare edges and color content of the current camera image with a reference image of the basketball hoop seen straight on and centered. The robot continues to pivot until what the Axis camera sees matches the reference image indicating the robot has the hoop centered in front of it. After this pivoting stage, program control is handed over to the ball shooting procedure which is the main thrust of this paper.
IV. CONTROLLER DESIGN

Three fuzzy controller designs will be implemented to control the shooter motor’s speed and compared. The encoder data will be used for feedback control. Three designs for fuzzy logic controllers are designed to provide smoothed versions of bang-bang, 3-point switching element, and linear (P) controllers. These three designs picked are by no means optimal, but instead represent three random samples from the possible design space of fuzzy controllers to demonstrate the ease of heuristic design and the flexibility of the architecture.

Although fuzzy controllers provide flexibility in characteristic surface shape and therefore controller behavior, it is valid to consider whether this same flexibility could be afforded by linear gains (P controllers) with saturation limits. If the characteristic surface for the fuzzy controller can be closely approximated by a line or plane, then the computation required for fuzzy controllers is not validated since it could be replaced by simple algebraic multiplication. To test whether the same response could be obtained using a simpler linear control algorithm, best fits for each of the three fuzzy controller designs were crafted. The fuzzy controller is compared to the linear best fit by observing its response in simulation.
F. Control Topology

The controller designs were placed into a control topology as shown in the simulation diagram in Figure 7. The controller was placed in series with an accumulator before having the output voltage fed into the motor plant. The reasoning for adding the accumulator is due to the motor’s steady state speed being roughly proportional to the input voltage. A P controller would drop the input voltage to zero when the error between desired speed and observed speed is zero. Since voltage is proportional to speed, the speed will drop to zero causing the error to grow again. Instead the controller output should be linked to acceleration, not speed. The change in voltage is proportional to acceleration (ignoring time constants and corresponding delays). This link between control output and change in voltage can be attained by adding an accumulator in series with the controller.

BasketBallBot’s current power supply system is using two 12V, 5.1 A power supplies placed in parallel. This results in roughly 120W of power available for the robot. When the motor pulls more power than allowed the voltage drops which turns off the motor. Two saturation blocks are placed in the control topology to avoid having the motor pull too much power.

Due to the saturation functions, there is a cap on the magnitude of the controller output. Even the largest available control output, often results in a sluggish response. To make the most efficient use of this control output headroom, the controller input is scaled to allow the largest observed error to map to the largest
available control output. The largest observed error will be when the motor is at a standstill and the error is equal to the commanded speed. The controller’s input error is divided by the magnitude of the speed command resulting in a percent error being the input for the controllers. Thus, the controllers are designed to map an input percent error of magnitude 1.0 to the highest allowed control output.

Since the control output headroom is such an important variable, it is often retuned to attempt to maximize the allowable control output. To avoid having to redesign the fuzzy controllers output each time the maximum allowable output is changed, the controllers instead map to a range between -1.0 and 1.0. This output is then scaled so that it fits within the headroom.

Combining all these control architecture design considerations yields the simulation diagram shown in Figure 7. This diagram is rendered in Mathworks’ Simulink software. This simulation software was originally used to simulate the system, until the LabView simulation discussed in Section V superseded it.

**Figure 7-Simulation Diagram for Shooter System Control Architecture (Rendered in Simulink)**
G. Fuzzy Logic Controller Designs

Three fuzzy logic controllers were designed that formed smoothed versions of bang-bang and 3-point switching element controllers as well as an approximate P controller. The smoothing is caused by fuzzy logic controller’s innate interpolation and is what causes fuzzy controls to be so easy and intuitive to design.

The controllers designed in this work were designed using an intuitive understanding of the Mamdani control architecture. Mamdani controllers use input sets with triangular characteristic functions. The triangular sets are extensional hulls of crisp points under a particular similarity relation. This means these sets represent the linguistic terms “around $x$” or “similar to $y$” where $x$ and $y$ are crisp values. The definition of “around” or “about” is defined by the similarity relation used. When a side of the triangle slants down sharply it means that a small move away from $x$ results in a larger dissimilarity to $x$ since proximity counts. When a side of the triangle has a gradual slope it means that distance does not impact that measurement’s similarity to the reference $x$ much at all. In the extreme case a leg can be flat meaning that distance is inconsequential over that leg’s range and all values are basically equal to $x$ there. This can result in trapezoidal membership functions as well that contain flat tops to represent equality to the reference. The similarity-based input sets are then mapped to similarity-based output sets forming the fuzzy logical rule:

“If input speed is around $x$ then output change in voltage should be around $y$”
When the input speed equals $x$ the output will equal $y$ as intended. However, the Mamdani controller will also interpolate between rules to give outputs for values not specified by any one rule. More overlapping between input sets will allow for more areas of interpolation and more “fuzziness”. Output sets with broader triangular bases make the rules more likely to be “fudged on a bit” making the output more smooth. Conversely, output sets with narrower triangular bases are more tightly followed. In the extreme case an output set with infinitely narrow base is defined only for one numerical value and is termed a “singleton”. These output sets represent crisp output values.

This intuitive understanding was used to construct the three heuristic designs described below.

**Basic Linear Interpolation Controller (Controller A)**

First a controller with operation similar to a classical P controller was constructed. This demonstrates how fuzzy controllers’ capabilities encompass that of classical controls as well as expands its flexibility to more nonlinear characteristics. Note that this approximate P controller does not approximate the optimal gain, and may be underdamped or overdamped. The input fuzzy sets mapped values “around” an anchor value $w$ to output fuzzy sets that defined values “around” an output $v$. The values for $v$ are linearly spaced through the input range of percent errors [-1.0,1.0]. The values for $w$ are likewise linearly spaced through the output range of output changes in voltage [-1.0,1.0]. The similarity relations were defined such that the fuzzy sets overlapped the neighboring sets completely, allowing for strong interpolation throughout the entire
range. The fuzzy sets for this design are shown in Figure 8 and Figure 9. Note that the range is between -1.0 and 1.0, but the set definitions extend beyond this range. The extensions beyond the range bounds are for mathematical convenience only and the output never extends beyond its range.

![Figure 8-Fuzzy Input Sets for Fuzzy Linear Controller](image)

**Figure 8-Fuzzy Input Sets for Fuzzy Linear Controller**

![Figure 9-Fuzzy Output Sets for Fuzzy Linear Controller](image)

**Figure 9-Fuzzy Output Sets for Fuzzy Linear Controller**

The resulting characteristic curve for this fuzzy controller is shown in Figure 10.
Figure 10—Characteristic Curve for Fuzzy P Controller

Note that the characteristic curve is almost linear with an $R^2$ value of 0.9994 to a linear fit. A linear characteristic curve corresponds to a simple gain for the controller plant which is also known as a “P Controller”. This linear behavior is expected since the fuzzy controller maps linearly spaced points to linearly spaced points. In Section VI the similarity of this controller’s performance to a classic P controller will be investigated further via their responses.

Smoothed Bang-Bang Controller (Controller B)

Due to limited power and the resulting need for saturation functions (described in Control Topology subsection), the controller output is limited to keep below a certain value. To maximize speed of response, it seems prudent utilize the highest available output magnitude as much as possible. In response to this design heuristic, a sort of fuzzy bang-bang controller was designed. The input fuzzy sets (shown in Figure 11) were designed to have total overlap and a single triangle function designating values...
“around 0”. Two trapezoidal input sets include the rest of the numbers as being described by the linguistic variables “too high” or “too low”.

![Figure 11: Fuzzy Input Sets for Fuzzy Bang-Bang Controller](image1)

**Figure 11-Fuzzy Input Sets for Fuzzy Bang-Bang Controller**

The output fuzzy sets are all singletons (fuzzy sets that define a single crisp output value as representing the output for that fuzzy set), with “too high” mapping to 1.0 voltage rate, “around 0” mapping to 0 voltage rate, and “too low” mapping to -1.0 voltage rate. These fuzzy sets are shown in Figure 12.

![Figure 12: Fuzzy Output Sets for Fuzzy Bang-Bang Controller](image2)

**Figure 12-Fuzzy Output Sets for Fuzzy Bang-Bang Controller**

The characteristic surface created by these sets and the rules described above is shown in Figure 13.
The characteristic surface in Figure 13 is similar to a bang-bang controller, but has a transition region around 0 which smooths the transition between one state and the other.

**Smoothed 3-Point Switching Element Controller (Controller C)**

This design takes the heuristic design for the bang-bang controller and adds another heuristic: decrease acceleration when the error approaches zero. This heuristic seeks to minimize oscillations and overshoot. This can be accomplished by having a small slope on the characteristic curve as it crosses through the origin. This corresponds to a small gain on a linear controller that will result in more damped behavior. The bang-bang design does not accomplish this small slope near the origin (as seen in Figure 13) as it quickly slopes through the origin to transfer to the other state (from “too high” to “too low” or vice versa).
To combat this large slope shortcoming, the rule pertaining to the inputs “around 0” needs to be changed. The input fuzzy set for “around 0” in Figure 11 is enlarged to encompass more values as shown in Figure 14.

![Figure 14-Input Fuzzy Sets for Smoothed 3-Point Switching Element Controller]

Instead of a singleton specifying a crisp input of zero, the output fuzzy set for the region “around 0” will be a triangular set denoting “about zero”. This change will make the “around 0” rule more easily interpolated between, creating a larger transition zone than that observed in the smoothed bang-bang controller. The new output fuzzy sets for this controller are shown in Figure 15.

![Figure 15-Fuzzy Output Fuzzy Sets for the Smoothed 3-Point Switching Element Controller]
The characteristic curve for this controller design is shown below in Figure 16.

**Figure 16**-**Characteristic Curve for Smoothed 3-Point Switching Element Controller**

Note that in this controller the transition region “around 0” is much larger. This third region between the constant outputs of -1.0 and 1.0 forms a region where the system can settle. Because of this 3 region format, this controller is similar to a 3-point switching element controller with very smooth transitions between states.

Because of the new third region “around 0”, the characteristic curve’s slope at the origin is much smaller. In a linear controller this would correspond to decreasing the proportional gain to reduce oscillations. However, this design also moves uses the full control output headroom allowing for aggressive action when the percent error is large, and gentler action when the percent error is small.
H. Linear Best Fit Designs

The three designs outlined above represent the flexibility of fuzzy controllers. However, since saturation functions are already present in the control topology, some of these controllers can be approximated via the simple gains (thought of as P controllers). To test whether the complexity of the fuzzy computation is necessary, best fits for the controllers using only P controllers and the saturation function were constructed. The characteristic curve of a P controller is a line, and with the saturation function the characteristic curve’s magnitude can be clamped to be below a fixed value. For constructing a best-fit P controller design, the section not saturated at -1.0 or 1.0 will be fit with a line and the slope of this line will become the P controller’s gain. These P controllers do not represent the optimal P controller design, but instead represent the closest simplification of the fuzzy controller designs A, B, and C discussed above. Whether the fuzzy controller contributes any behavioral difference over the simple linear function will be investigated via simulation.

Figure 10 shows that Controller A has an almost linear characteristic curve, so the corresponding linear controller fits closely. The line fits with a gain of 1.0 between the input percent error and output fraction of full control effort.

Figure 13 shows the characteristic curve for the smoothed bang-bang controllers. The smoothed transition zone in this controller can be fitted with a line extending from the input-output pairs (-0.01,-1.0) to (0.01,1.0). This line has a slope of 100 which becomes the P controller’s gain. All input values with magnitudes greater than 0.01 will be cutoff.
by the saturation function resulting in the behavior similar to a bang-bang controller away from 0.

Similarly we can attempt to fit a line to Figure 16 extending between the input-output pairs (-0.4, -1.0) and (0.4, 1.0). This results in a slope of 2.5 for the P controller’s gain.

I. Controller Implementation Code
The fuzzy controllers were implemented using National Instruments’ PID and Fuzzy Logic Toolkit. The Fuzzy System Designer was used to build the membership functions for the input and output sets seen in Figure 8, Figure 12, and Figure 15. The linear best fit controllers were implemented using a simple multiplication block. To summarize the control architecture a simplified block diagram of the control algorithm is shown in Figure 17. The blue input blocks at the left in the block diagram are outputs from the encoder filter and sonar rangefinder-speed determination algorithms described in Figure 5 and Figure 31, respectively.

![Figure 17: Block Diagram for Control Algorithm](image)

The LabView implementation of this algorithm is in the Appendix A.

V. SYSTEM MODELING FOR SIMULATION
J. Plant Model
The shooter system’s motor dynamics are defined in Eq. 1.
The state variables are the motor speed ($\omega$) and current ($i$). The electrical parameters are resistance ($R$), inductance ($L$), and back electromotive force constant ($K_e$). The mechanical parameters are the motor damping ($c$), moment of inertia ($I$), and torque constant ($K_T$). The model’s inputs are the motor terminal voltage ($v(t)$) and load torque ($T_d$). The input voltage is controlled by the accumulator and controller as seen in Figure 7. The load torque is set to zero in this simulation. The motor model described in Eq.1 is simulated in LabView. The LabView implementation of Equation 1 can be viewing in Appendix A. The input voltage can be connected to a constant voltage source to simulate a step response or connected to the control algorithm described in Subsection I to simulate the controllers’ performance as will be investigated in Section VII.

K. Parameter Extraction

The motor datasheet’s stall torque and no load speed were used to obtain the mechanical and electrical parameters defining the shooter motor. Taking the first row in Eq. 1 to the steady state makes $\dot{i} = 0$. This equation can then be solved for steady state current. This steady state current can be multiplied by the torque constant $k_T$ to find the motor generated torque. This equation can be further simplified by assuming no electromagnetic losses which means that $k_T = k_e = k$.

$$T_M = \frac{k}{R} (v - k\omega)$$  \hspace{1cm} (Eq. 2)
Under stall conditions, the voltage will be the rated voltage and the speed will be zero. This creates:

\[ T_{\text{stall}} = \frac{k}{R} v \]  

(Eq. 3)

Under no load conditions and neglecting the damping counter-torque the motor torque will be zero simplifying Eq. 2 to:

\[ 0 = \frac{k}{R} (v - k\omega_{nl}) \]  

(Eq. 4)

Eqs. 3 and 4 can be solved for the resistance \( R \) and constant \( k \) as:

\[ k = \frac{V}{\omega_{nl}} \]  

(Eq. 5)

\[ R = \frac{v^2}{\omega_{nl} T_{\text{stall}}} \]  

(Eq. 6)

The damping constant can be calculated using the no load current as derived by Mathworks in [9] as:

\[ c = k \frac{i_{nl}}{\omega_{nl}} \]  

(Eq. 7)

The static motor parameters can be obtained via Eqs. 5, 6, 7 and datasheet parameters, however the dynamic parameters cannot be obtained by the steady state measurements from the datasheet.

An initial approximation for these parameters can be obtained by assuming values for the time constants. The mechanical and electrical time constants, \( c/I \) and \( R/L \), from a
similarly sized motor were taken to be the time constants for the shooter motor. The mechanical time constant was taken to be 4.614 seconds. The electrical time constant was taken to be 0.136 seconds. The moment of inertia and the inductance were solved from the time constants using the values for $R$ and $c$ derived from Eqs. 6 and 7. The parameters extracted using this method are described in Table II.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>0.1068 $\Omega$</td>
</tr>
<tr>
<td>$L$</td>
<td>0.0145 $H$</td>
</tr>
<tr>
<td>$c$</td>
<td>1.05e-4 $kg \frac{m^2}{s}$</td>
</tr>
<tr>
<td>$I$</td>
<td>4.84e-4 $kg \frac{m^2}{s}$</td>
</tr>
<tr>
<td>$k_T = k_E \Rightarrow k$</td>
<td>0.0216 $N \frac{m}{A}$</td>
</tr>
</tbody>
</table>

The first estimation for these parameters shown in Table II will be refined using experimental information concerning this robot’s particular motor as described next.

**L. Experimental Comparison**

A step voltage of 2.4V was applied to the motor terminals in both simulation and experiment. The resulting responses are shown below in Figure 18. The first simulation used the initial parameters extracted and defined in Table II.
The initial simulation’s voltage step response did not match with the experimental step response in two ways: the steady state speed and rise time. To amend the steady state speed, Eq. 5 was repurposed to define $k$ in terms of the step voltage and experimentally obtained no load steady state speed as:

$$k = \frac{v}{\omega_{nl}} = \frac{2.4 V}{2531.9 \frac{counts}{sec} \left(\frac{2\pi \text{ rad}}{250 \text{ counts}}\right)} = 0.0377 \frac{V}{\text{rad/s}} = 0.0377 \frac{N \cdot m}{A}$$

The experimental no load speed used above ($2531.9 \frac{counts}{sec}$) was obtained by taking an average over the speed response’s steady state section. Not all of this steady state sample data is shown in Figure 18 for the sake of concision.

To amend the rise time, the dynamic constants have to be adjusted. The motor’s moment of inertia $I$ was slowly increased until the experimental and adjusted simulation
responses matched. The final value response obtained using this hand calibration method is also plotted in Figure 18. The value of $l$ settled upon to accomplish this fit was

$$l = 0.02 \, kg \, m^2.$$

VI. SIMULATION RESULTS

M. Fuzzy Controller Comparison

The shooter motor plant model derived in Section V was constructed in LabView and attached to the LabView controller implementation discussed in Subsection I to form a full simulation. All three fuzzy controller designs discussed in Section IV were tested in simulation. Their responses are plotted below in Figure 19.

![Figure 19 - Comparison of Fuzzy Controllers' Responses](image)

All three designs achieved similar rise times, but Controllers A and B exhibited undesirable sustained oscillations. Controller C’s oscillations damped out over time.
Controller B’s oscillations had sharper peaks and troughs like the bang-bang controller from which it is heuristically derived. It did succeed in rounding off the peaks due to the smoothing of the transition zone between “too high” and “too low”.

The frequency of oscillations for different controller designs are (in descending order): B, A, C. Controller C was the only design that settled to within 2% of the command speed within the time frame of 20 seconds. Thus, this was the controller chosen to implement on BasketBallBot’s shooter motor.

N. Interchangeability between Fuzzy and Closest Linear Fit
The fuzzy controllers simulated above require assessing the membership degree of the error to all the input fuzzy sets, clipping the output sets by the corresponding membership degrees, aggregating the clipped output sets, and defuzzifying the aggregate set. This requires more processor overhead than simple multiplication and limit saturation checking. If the fuzzy controller behaves similarly to a controller constructed using multiplications and saturation limits, then the fuzzy controller design should be abandoned as realistically requiring too much overhead for equivalent performance. However, these controller designs still demonstrate how fuzzy controllers can be designed to incorporate elements from many other nonlinear and linear control architectures.

To investigate the interchangeability problem, the linear fits to the fuzzy controllers discussed in Subsection H were simulated and the responses compared to those obtained
by the fuzzy controllers. Controller A and its linear approximation (with gain of 1) were simulated and the resultant responses plotted as in Figure 20.

**Figure 20 - Controller Design A: Fuzzy and Linear Comparison**

The fuzzy and linear controllers for Design A overlapped closely for the first two oscillations, but a mismatch in frequency separated the peaks after two oscillations. The linear controller also exhibited a slight amount of damping in its oscillations making the linear controller superior for Design A. This close match between fuzzy computed linear characteristic curve and directly computed linear gain should be expected. Thus, the fuzzy controller using Design A should be abandoned considering how easily equivalent performance could be obtained.

Design B’s closest approximation linear controller (with gain of 100) was simulated and compared to the fuzzy version. The responses are shown in Figure 21.
The fuzzy and linear versions of Design B behave similarly. Thus, Design B should be abandoned as not providing significant behavioral difference from a simple multiplication and saturation.

Finally the fuzzy and linear approximation for Design C were simulated and plotted in Figure 22.
The linear best fit of Design C had very different behavior from the original fuzzy version. This is because the linear controller could not form a flat region “around 0” which is the key characteristic of this controller design. Thus, controller Design C cannot be approximated with just a multiplication and clipping operation.

Overall fuzzy controller designs A and B were found to be superseded by behaviorally equivalent algorithms that only required multiplication and saturation limit checks/clipping. Thus, although the fuzzy architecture could be used to create linear and bang-bang like characteristics, in the end these functions are best created using actual linear gains and limit checks rather than using the fuzzy “jack of all trades”. The fuzzy architecture’s real strength is in combining these two elements to create a characteristic surface with both linear and saturated regions without extra computation. This sort of merging is used in design C which has several saturated and sloped regions merged together into one characteristic curve. Controller design C could not be replaced by an algorithm of this form, and so its computational overhead is justified. Controller design C may be “replaceable” by a piecewise linear function (gain scheduling), but that type of replacement for the algorithm is not investigated here.

VII. EXPERIMENTAL VERIFICATION OF CONTROLLERS

Subsection M concluded that Controller design C was the best amongst the fuzzy designs sampled here. To verify this simulation result, all three designs were integrated
onto the CRIO using the actual encoder data (processed according to Subsection D) as an input and the shooter motor plant.

The three fuzzy controller designs were tested on the system under a step command of -4000 counts/sec as in the Simulation. The resulting responses are plotted in Figure 23.

![Figure 23-Fuzzy Controller Experimental Responses](image)

The same pattern of relative frequencies between the three controller designs exists in the experimental responses as appeared in the simulated responses. Controller C still settles the most quickly which confirms Subsection M’s result.

A slight amount of system vibration resulted in some noise for the controllers. Controllers C and A were able to tolerate the noise and keep their responses consistent with the simulated predictions. In contrast, Controller B’s oscillations were predicted to have constant magnitude but instead randomly increase and decrease as mechanical noise impacts the acceleration.
The experimental responses confirm again that controller design C is indeed the best fuzzy controller design. It also shows that fuzzy controllers can be implemented on real time control environments and successfully programmed onto embedded controllers like the CRIO.

VIII. SHOOT SPEED DETERMINATION ALGORITHMS

With the speed now controlled to a constant value, the next concern for BasketBallBot’s design is the choice of command speed. For different distances from the robot to the hoop, different speeds must be imparted to the basketball to insure it lands in the hoop. BasketBallBot will use the distance measurement to the hoop obtained by the sonar rangefinder as an input to an algorithm to compute the correct launch speed. The basketball launch speed will be related to the shooter motor’s speed and the desired motor command speed will be obtained.

Two algorithms were investigated for determining the appropriate launch speed. One method uses the kinematics of a freefalling particle to approximate the basketball’s flight and calculates the desired initial speed and corresponding shooter motor speed. The other method uses trials to obtain the appropriate shooter motor speed for four different distances from the hoop. Then fuzzy logic’s inherent interpolation is leveraged to interpolate between the four distance-shooter speed pairs and provide a shoot speed for any input distance between the four distances. In this setting a fuzzy controller is termed a “fuzzy inference system” instead.
0. Empirical Relation with Fuzzy Inference System

BasketBallBot was positioned to lie along a ray centered on the hoop and extending perpendicular from the backboard. The robot was angled so that the ball would travel towards the hoop. The success of the basketball shot was sensitive to the proper positioning of the robot. For these experiments BasketBallBot was positioned by hand, but ultimately the machine vision pivoting system developed by Brady et. al would accomplish this autonomously.

The robot was placed at four distances along the ray as read by the sonar rangefinder: 145 cm, 195 cm, 235 cm, and 295 cm. At each distance, the appropriate shooter motor command speed was iteratively tuned until the robot could net at least 75% of shots attempted. The tuned command speeds for the corresponding distances are tabulated below in Table III.

Table III: Iteratively Tuned Command Speeds for Increasing Shot Distances

<table>
<thead>
<tr>
<th>Sonar Ranger Distance (cm)</th>
<th>145</th>
<th>195</th>
<th>235</th>
<th>295</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed Magnitude (counts/sec)</td>
<td>6200</td>
<td>6400</td>
<td>6600</td>
<td>6900</td>
</tr>
</tbody>
</table>

The same heuristic design method methodology employed in Section V was used to create a fuzzy inference system to interpolate between these four values. The ruleset...
stated that when the sonar ranger’s distance is about one of the test points the command speed should be about the corresponding tuning. The input and output fuzzy sets for this fuzzy inference system are shown below in Figure 24 and Figure 25.

**Figure 24-Fuzzy Input Sets for Fuzzy Inference System**

![Fuzzy Input Sets for Fuzzy Inference System](image)

**Figure 25-Fuzzy Output Sets for Fuzzy Inference System**

These fuzzy sets resulted in the characteristic curve shown in Figure 26.
**Figure 26—Characteristic Curve for Fuzzy Inference System**

### P. Kinematics Method

**Kinematics**

The planar equations of motion for a particle (assuming constant acceleration and the particle’s initial position is the origin) are:

\[
x = v_0 \cos(\theta) t + \frac{1}{2} a_x t^2
\]

(Eq. 8)

\[
y = v_0 \sin(\theta) t + \frac{1}{2} a_y t^2
\]

(Eq. 9)

Where \(v_0\) is the initial basketball speed and \(\theta\) is the launch angle. Assuming zero acceleration in the x-direction, only gravitational acceleration in the negative y-direction, and assessing at the moment the ball hits the backboard turns Eqs.8 and 9 into:

\[
d = v_0 \cos(\theta) t_f
\]

(Eq. 10)

\[
h = v_0 \sin(\theta) t_f - \frac{1}{2} g t_f^2
\]

(Eq. 11)
Where \( g \) is the acceleration due to gravity, \( t_f \) is the time when the basketball hits the backboard, \( h \) is the distance upwards the ball must travel to hit the backboard at the desired height, and \( d \) is the distance from the launch position to the backboard. The distance \( d \) will be obtained from the sonar ranger’s measurement by subtracting off the distance from the sonar ranger’s position to the launch position. On BasketBallBot, the distance from sensor to shooter is 30 cm.

Eq. 8 can be solved for the final time \( t_f \) as:

\[
t_f = \frac{d}{v_0 \cos(\theta)} \\
\text{(Eq. 12)}
\]

Since \( t_f \) is not a parameter determined by the shooting problem it should be eliminated from the equations. This can be done by substituting Eq.12 into Eq. 11 as:

\[
h = v_0 \sin(\theta) \frac{d}{v_0 \cos(\theta)} - \frac{1}{2} g \left( \frac{d}{v_0 \cos(\theta)} \right)^2 = d \tan(\theta) - \frac{1}{2} g \left( \frac{d}{v_0 \cos(\theta)} \right)^2 \\
\text{(Eq. 13)}
\]

The parameters \( h \) and \( d \) are given by desired shooting behavior and ultrasonic sensor data, respectively. The gravitational acceleration \( g \) is a constant. Thus, the free variables in Eq. 11 are the variables \( v_0 \) and \( \theta \). We seek to find a fixed value for these variables or a relation between the variable and shooter motor command speed.
Launch Angle Equation

The basketball’s angle of travel at launch, θ, was measured using image processing in MATLAB on camera footage of the basketball being launched. The individual frames of the video from around the time of launch were converted into the LAB color space. The LAB color space replaces the RGB color space and has a higher sensitivity to color changes numerically than RGB does. The pixels of these frames were then tested to see whether their A content was above a threshold and the B content was below another threshold. These thresholds are tuned to distinguish the orange color of the basketball out of the background. This thresholding process creates a binary image with white pixels where the thresholds were cleared, and black where they were not. An example of the original frames during the basketball launch are shown in Figure 27, while the same frames but processed are shown in Figure 28.
FIGURE 27-ORIGINAL IMAGE FOR LAUNCH FRAMES EXAMPLE
Figure 28-Binary Image for Launch Frames Example

The binary image clearly illuminates the basketball against the background. The centroid of the largest object (the basketball) in each binary frame is tracked across all frames. The change in centroid position is calculated for each frame and the angle of this change is recorded. After this processing, the angle change for the frame when the ball leaves the shooter motor is tabulated.

This processing was performed on two videos under the same launch speed. Three different launch speeds were tested to investigate whether launch angle is dependent on launch speed. The launch angles for these six videos are tabulated below in Table IV:

Table IV: Launch Angle and Corresponding Commanded Launch Speed

<table>
<thead>
<tr>
<th>Rotational Speed (counts/sec)</th>
<th>4000</th>
<th>6000</th>
<th>7000</th>
</tr>
</thead>
</table>

51
| Shot Angle in Video #1 (degrees) | 23.25 | 38.65 | 43.4 |
| Shot Angle in Video #2 (degrees) | 18.1  | 38.9185 | 48.3 |
| Average Angle (degrees)         | 20.675 | 38.78425 | 45.85 |

The average launch angle between the two videos is calculated in the bottom row of Table IV. The variance around this average is roughly ±2.5 degrees for the 4000 and 7000 counts/sec shots. The variance is ±0.13 degrees for the 6000 counts/sec shots. Graphing the average shot angle against commanded launch speed results in Figure 29.

![Graph showing launch angle as a function of command speed](image)

**Figure 29 - Launch Angle as a Function of Command Speed**

The trend line fitted to the data in Figure 29 fits well with an $R^2$ value of 0.9966. Thus, launch angle is a function of command speed as:

$$\theta = 0.0085w - 12.986 \quad \text{(Eq. 14)}$$

Where $w$ is the commanded launch speed in encoder counts per second.
Launch Speed-Motor Speed Relation
The basketball is launched by making contact with a pair of wheels affixed to the shooter motor’s axle. Thus, the speed of the basketball will be the same as the speed of a point of the rim of the shooter motor’s attached wheels. Thus the launch speed will be:

\[ v_o = \omega r \]  \hspace{1cm} (Eq. 15)

Where \( r \) is the wheel’s radius and \( \omega \) is the wheel speed at launch in radians per second. When the basketball makes contact with the shooter wheel’s rim a disturbance torque acts on the controller. This slows the wheel down temporarily from the speed the controller accelerated the wheel to. The controller resists this deceleration, but ultimately the speed will fall. To find how much the speed falls by, a series of ball launching experiments were conducted with different initial wheel speeds commanded by the controller. The speed the motor dropped to during the disturbance was recorded. The results of these tests are recorded in Table V.

<table>
<thead>
<tr>
<th>Speed (cts/sec)</th>
<th>-4000</th>
<th>-6000</th>
<th>-7000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trial A</td>
<td>-1567.5</td>
<td>-3600</td>
<td>-4470</td>
</tr>
<tr>
<td>Trial B</td>
<td>-1820</td>
<td>-3467.5</td>
<td>-4245</td>
</tr>
<tr>
<td>Trial C</td>
<td>-1732.5</td>
<td></td>
<td>-4282.5</td>
</tr>
<tr>
<td>Trial D</td>
<td>-1850</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>-1742.5</td>
<td>-3533.75</td>
<td>-4332.5</td>
</tr>
</tbody>
</table>
The average post-disturbance speed was plotted as a function of initial speed in Figure 30. A linear trend line was fit to the data with an \( R^2 \) value of 0.9992.

\[
y = 0.8679x + 1715.4
\]

Figure 30 - Post Disturbance Launch Speed as a Function of Initial Wheel Speed

Thus the post-disturbance launch speed is a function of the original, controlled speed as:

\[
w_s = 0.8679w + 1715.4 \quad \text{(Eq. 16)}
\]

Where \( w_s \) is the post-disturbance launch speed in encoder counts per second.

Converting this post-disturbance launch speed from counts per second to radians per second results in the rotational frequency needed in Eq. 15. Substituting this converted launch speed from Eq. 16 into Eq. 15 produces

\[
v_o = \frac{r(0.8679w + 1715.4)(2\pi)}{250} \quad \text{(Eq. 17)}
\]
Analysis

Eqs. 13, 14, and 17 are combined to yield:

\[ h = d \tan(0.0085w - 12.986) - \frac{g}{2} \left( \frac{250d}{r(0.8679w + 1715.4)(2 + \pi) \cos(0.0085w - 12.986)} \right)^2 \]  

(Eq. 18)

Eq. 18 must be solved for the desired command speed \( w \). Unfortunately due to the \( w \) terms inside the trigonometric functions this equation is poorly analytically tractable. A constant value for \( \theta \) could be assumed to remove the \( w \) factors from the trigonometric functions. The empirical approach discovered that the appropriate speeds are contained within the 6000 to 7000 counts/sec range. The average of the 6000 and 7000 counts/sec launch angles could be taken as this constant value. However, this process must be saved for a future work.

Q. Implementation

The empirical algorithm employing the fuzzy inference system was chosen due to its completeness. In addition, it promised another avenue for exploring fuzzy logic.

This algorithm and its fuzzy system were implemented in LabView and programmed onto the real-time CRIO. A simplified block diagram describing the algorithm is shown in Figure 31. Note that the block diagram in Figure 31 repeats after the fuzzy inference system’s computation is complete. This computation can only execute again after another 2 second sampling has completed. Thus, the command velocity is updated every 2 seconds in a loop. The output from this algorithm is then fed into the control algorithm described in Figure 17.
Figure 31-Block Diagram for Sonar Rangefinder Usage in Shoot Speed Determination

The ultrasonic sensor is polled every 200 milliseconds and the 10-point moving average is taken every 2000 milliseconds (or 2 seconds). When the 10-pt moving average is updated, the resulting average is fed into the fuzzy inference system designed in Subsection O to obtain the appropriate shooter motor speed for the current distance. The LabView code for communicating with the slave Arduino via I2C and processing the sonar rangefinder data (as shown in Figure 31) is in the Appendix A.

Using this algorithm, the robot successfully netted 80% of 15 test shots attempted. The combination of fuzzy controller to keep the shot speed consistent and fuzzy inference algorithm to compute the correct shot speed allowed the robot to successfully shoot basketball hoops.
IX. CONCLUSION

Fuzzy logic controllers are increasing in industrial relevance. To equip the upcoming generation of engineers to understand fuzzy controllers, test projects and curriculum need to be synthesized and tested. This project focuses on using tools readily available to high school and undergraduate students (FIRST Robotics Competition hardware, LabView, and Arduinos) to demonstrate fundamental controls concepts and even advanced techniques such as fuzzy controls. With the robotic sensor foundation set through this project, other undergraduate students at University of Tennessee at Chattanooga’s (UTC) electrical engineering department can explore fuzzy controls for themselves by tweaking the designs laid out in this paper. In addition, the design concepts and techniques demonstrated in this paper can be disseminated to the many local high schools UTC assists with robotic education through the FIRST Robotics Competition.

The variety in shape of fuzzy logic controllers’ characteristic curves was demonstrated through simulation. Two fuzzy controller designs’ step responses were shown to be similar to bang-bang and linear controllers. This flexibility allows a wide range of design strategies, ideal for a learning environment. Although fuzzy logic is flexible enough to reproduce linear and bang-bang like controllers, the amount of compute time required for a fuzzy rendition of these controllers is greater than using the original controller. Thus when looking for one of these types of controllers, it is best to use the original rather than a fuzzy facsimile. However, fuzzy controllers can smoothly combine
these types of controllers to gain the best of both types. This fusion is the strength of fuzzy controllers.

The three heuristic designs considered here were compared. The fuzzy fusion of bang-bang and linear characteristics was found to provide the best response out of the three. This controller was implemented on the BasketBallBot system which successfully shot hoops using this controller.

Fuzzy logic was used in another aspect as the means for deducing the appropriate ball launch speed given a particular distance for the ball to travel. In this sense, fuzzy logic was used to interpolate between experimentally obtained distance-launch speed pairs that netted the basketball with 80% accuracy. This alternative use again demonstrated the flexibility of the fuzzy logic system.

This flexibility in design combined with the heuristic approach for design makes fuzzy logic and fuzzy controllers ideal as a platform for exploring control theory in education. The same platform can be used for both controllers and interpolation algorithms. This platform can be applied to inexpensive sensors and educational hardware with great success. BasketBallBot is a model of how fuzzy logic is ideal for educational robotic exploration.
X. ACKNOWLEDGMENT

The authors would like to acknowledge Brady, McNabb, Broadstone, and Borden for their work on the camera aiming software for BasketBallBot. Thanks to Dr. Ahmed and Dr. Loveless for sitting on the review committee and providing feedback on drafts. Huge thanks to Dr. Ofoli for guiding this research so kindly and informatively; without your scope on the project I would have wandered terribly. Thanks to Brian MacCleery from National Instruments for providing us with the NI software license. Many thanks to the UTC robotics team for lending their tools and when it was needed most.

XI. REFERENCES


Appendix A: LabView Block Diagrams

Quadrature Encoder Reading:

Control Algorithm Implementation:
Motor Simulation Implementation:

Sonar Rangefinder Processing:
Appendix B: Slave Arduino Code

```c
#include <Wire.h>

void setup() {
  // put your setup code here, to run once:
  Wire.begin(84);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
  Serial.begin(9600);
  pinMode(7,INPUT);
}

long sensorValue=0;

void loop() {
  // put your main code here, to run repeatedly:
  sensorValue = pulseIn(7,HIGH)/58;
  Serial.print(sensorValue);
  Serial.println(" cm");
  delay(100);
}

// Function for handling receives *OBSOLETE*
void receiveEvent(int howMany)
{
  while(Wire.available())
  {
    int c = Wire.read();
    Serial.print("received! - ");
    Serial.println(c);
  }
}

// Function for handling master’s requests for sensor data
void requestEvent()
{
  uint8_t Buf[2];
  Buf[0] = (uint8_t) sensorValue;
  Buf[1] = (uint8_t) (sensorValue >> 8);
  Wire.write(Buf,2);
}
```