THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ELECTRICAL ENGINEERING

AUTO-LABYRINTH

MATT BARANOSKI
FALL 2016

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Electrical Engineering
with honors in Electrical Engineering

Reviewed and approved* by the following:

Tim Kane
Professor of Electrical Engineering
Thesis Supervisor

Julio Urbina
Associate Professor of Electrical Engineering
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

As more and more industries turn to automation to help reduce costs and increase efficiency the demand for more robust control systems has increased. Historically these control systems only governed processes that could be represented mathematically but now the systems are my dynamic and do not always have a known underlying process. The Auto-labyrinth is a small-scale test bed for various control algorithms. It combines a system with an unknown underlying process with visual feedback to pose a unique challenge for any control algorithm. The project is based upon the Brio Labyrinth maze game and uses servo motors and a webcam to allow a computer to control the system. As a proof of concept the test bed is programmed with a PID controller to demonstrate the power of the PID controller as well as how the project can be used to test other algorithms. Although the project has some limitations, mainly an image acquisition delay from a USB 2.0 camera, it proves to be a unique test bed for control algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I'd like to thank my family and friends for supporting me throughout my academic career and this project. I'd like to thank Dr. Kane for making this project possible and encouraging me to pursue it.

Chapter 1

Introduction

The Auto-Labyrinth project creates a small-scale test bed for control algorithms based on the Brio Labyrinth game. The goal of the Brio Labyrinth game is for the user to guide the ball through the maze without letting it fall through any holes in the game board by tilting the board using the knobs on the side. The way the knobs tilt the board includes hysteresis and poses a unique challenge for any control algorithm. A few modifications have been made to the game to allow servo motors to control the knobs and a webcam to track the motion of the ball. All of the algorithms are programmed in LabVIEW and demonstrate the effectiveness of the test bed with a PID control algorithm. The first section will serve as background information on the ideas, theories and hardware used to implement the project. Subsequent sections will detail modifications made to the game, how the algorithms work as well as a discussion of the results and future work.

Figure 1 Completed Auto-Labyrinth

1.1: Control System Theory

Control system theory is the application of a feedback system to a dynamic process. The processes can vary greatly from vehicle navigation to ecosystems. In most cases the desired outcome of the controller is to control the process in a specified way. The system will always have a reference or desired outcome. This reference can be static like a home thermostat or changing like the navigation of an aircraft. The controller will use a sensor or group of sensors to gather feedback from the system. Using this feedback the controller will calculate the error between the current state of the system and the reference. The controller can then use the error information to change an input to the system to make the system come closer to the reference.

Control System Engineering is the design of a controller capable of eliciting a desired response from a given system. Typically, a controller is designed based on a mathematical representation of the system. This model will be a representation of how the input to a system affects the output. Knowing the underlying mathematical representation of the system allows engineers to design a control system that can efficiently manipulate the system.

1.2: PID controller

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

Figure 2 PID controller formula

At the heart of many industrial control systems lies the PID controller. A proportional-integral-derivative (PID) controller is a classic closed loop feedback mechanism. A PID controller uses a measured process variable and a desired set point to adjust a control variable. The PID controller does this by calculating the error, the difference between the process value and the desired set point. The controller adjusts the control variable over time to minimize this error. Common uses for the PID controller are control valves, dampers and thermostats, etc... but its application is limitless. A PID controller is very robust and broadly applicable since it relies on the measured process variable and not any knowledge of the underlying process of a system. The PID controller is tuned to elicit the desired response from a system even if it has an unknown underlying process.

The PID controller relies on a weighted sum to calculate the new control variable that minimizes the error. In the weighted sum shown in Figure 3, K_p, K_i and K_d are the gains of the proportional, integral and derivative terms respectively. When a PID controller is tuned, these coefficients are adjusted to elicit the desired system response characteristics. The block diagram in Figure 2 shows how each term of the PID controller is used in a closed loop feedback system. *P* denotes the present value of the error, *I* the past value of the error and *D* the possible future values of the error derived from the current rate of change of the error. Each of these terms has a

different effect on the control variable and the value of the gain coefficient of each term determines the extent of the effect it has on the control variable.

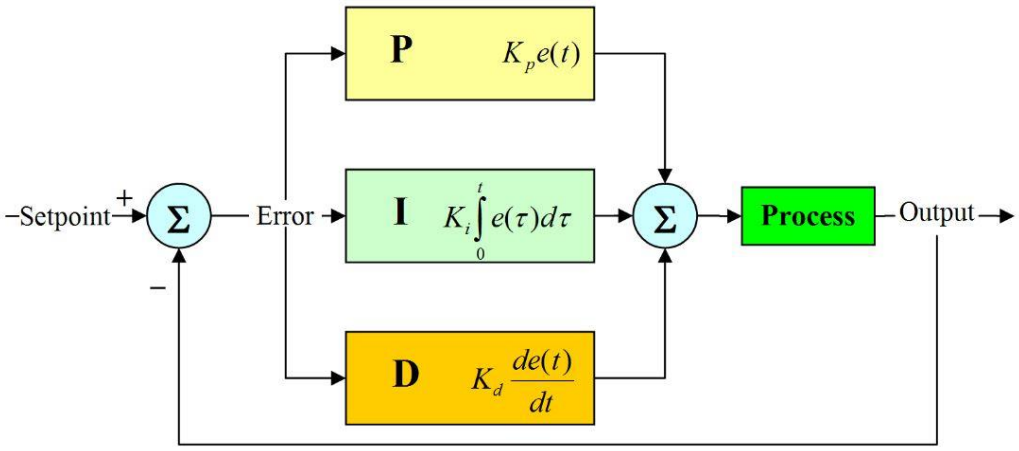

Figure 3: PID block diagram

The application of the PID controller to the Auto-labyrinth is ideal because the underlying process is unknown but the ball's location can be easily measured. In this system, the measured process variable is the ball's location, the set points are pre-programmed turns and safe zones on the game board outlining the desired path through the maze and the control variable is the duty cycle of the PWM signal controlling the motors position. The PID controller can only control one variable and the ball's location is a 2D value with an X and Y component. In the Auto-labyrinth the X and Y components are split and processed separately with their own PID controller to overcome this limitation and take advantage of separability.

1.3: Computer vision

All control systems require one or more feedback mechanisms, such as a vision system. The term computer vision is a very broad, all encapsulating field spanning the acquisition, processing, analysis and interpretation of high dimension data from real world situations. This

information helps to make decisions about the contents of the images and execute an appropriate response given the present information. The goal of the field of computer vision is to recreate the ability of the human eye to view a situation, understand the image and choose an appropriate response. Computer vision encompasses all forms of using images to provide data about a situation. Some common examples are object identification, video tracking, image restoration, event detection and quality checking.

The Auto-Labyrinth will use computer vision to acquire an image, process it and track the ball's movement. The Auto-Labyrinth uses a webcam as a sensor to gather the image and processes it in LabVIEW. Object tracking is based on a matching algorithm that compares a template image to the captured image and locates similar features to the template on the captured image. This comparison can be made using many different protocols. Some of the most common are a color match, geometric match and pattern match. The color match compares the color of the template image to the color of the captured image. Geometric matching builds a profile of the template based on any geometric shapes and locates them in the captured image. A pattern matching algorithm is similar to geometric matching but builds a profile based on any patterns in the template image.

1.4: Motion estimation

Modern control and navigation systems all have an inherent delay in processing data from sensors. The systems need to estimate the sensor data at the time of processing which is not always the time it was acquired. In the context of the Auto-Labyrinth, the image the algorithm is processing does not match what is occurring on the game board because of a delay in acquiring

the image. In order to compensate for this the motion estimation algorithm has to predict where the ball is based on the most recent image and the previous image. The simplest form of this algorithm is comparing the balls location on the current image to that of the next most recent image. From this comparison, an estimated location can be calculated. This simple algorithm assumes the ball will follow the exact same trajectory from one image to the next and that the trajectory is linear. This is not exactly how the ball moves on the board but it is a good approximation of the motion. There are many other more complex motion estimation algorithms such as the Kalman filter but in the context of the Auto-Labyrinth the simple estimator is adequate since it allows the ball to successfully navigate the maze.

Adding a Kalman filter to the Auto-Labyrinth control algorithm would allow a higher degree of accuracy in predicting the motion of the ball. The Kalman filter is an estimation algorithm that relies on an approximate model of the system and any inputs to the system as well as measurements. A Kalman filter has two main implementations; estimating the actual values of a noisy measurement given a noise model or estimating an inaccurate measurement by predicting the measurement using a system input and model of the system. In both instances the Kalman filter can produce more accurate values than a single measurement alone. The filter continually estimates an unknown value over time and compares it to a measurement to make very accurate predictions.

1.5: Data Flow Programming

Data Flow programming was developed by Jack Dennis at MIT in the 1960's and is a programming structure that addresses many of the limitations of the traditional linear

programming style. Linear programming style is utilized by C, Java and many other text-based languages. The basic structure of linear programming is text statements that elicit a response. The statements execute sequentially as they are listed from the first line to the last. This introduces the concept of the state of the program: what data is available right now, what operations have executed and what operations still need to occur. Although linear programming has its strengths for various programming needs when manipulating numerical data, it becomes cumbersome to program.

Dataflow programming addresses this problem by executing all instructions simultaneously. When the program starts all commands or statements execute, the only limit on the execution is the availability of inputs to each statement. A command can execute only once it has data at each of its inputs. This leads to the name dataflow programming; data is all that controls the flow or execution of the program. The ability to execute multiple commands simultaneously make dataflow programming conducive to parallel processing without the addition of excess code. Parallel processing takes advantage of the multiple cores on modern processors allowing dataflow programs to execute quicker than a similar program without parallel processing.

LabVIEW is a dataflow programming language and enjoys the many benefits of such. In addition, LabVIEW is a graphical programming environment making the connections and the flow of data very easy to see and debug. The Auto-labyrinth uses mostly algorithms that manipulate and gather numerical data so LabVIEW is an excellent choice for programming. LabVIEW's parallel processing ability is used in both the ball tracking algorithm and the PID controllers. The interfacing toolkits in LabVIEW make it easy to interface with the devices as

well. The only downside is that the system needs a computer to function, which is adequate for a test bed but adds to the total cost of the project.

1.6: Servo Motors

The Auto-Labyrinth uses two servo motors to control the tilt of the game board. Servomotors are typically DC motors used in a wide array of applications that demand precise rotations. Servomotors control most RC cars, airplanes and other vehicles but the use of these devices extends into many other applications from airplanes to electronics to factory automation

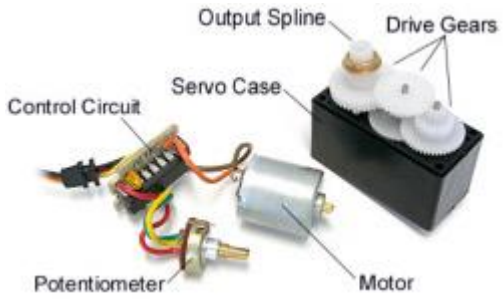

Figure 4: Inside view of a servo motor

and cars. Just about all robots employ servomotors to create movement. Servos are widely used because of their simplicity, reliability and low cost.

A DC servomotor is comprised of a small DC motor, a series of gears, a potentiometer and circuit board that controls the motor. The motor spins at a high speed but with little torque, the gears are arranged to slow down the motors rotation and increase the torque it can produce on the output shaft of the servo. On the final gear of the servo, a potentiometer is connected. The resistance of the potentiometer changes depending on the rotation of the final gear and inherently the output shaft of the servo. The potentiometer connects to the circuit board that controls the electricity to the DC motor depending on the input signal as well as the potentiometer position.

The circuit board compares the potentiometers resistance to the input signal of the servo and determines if the output shaft is in the correct position and how much it needs to move if it is not. This design allows servomotors to be precise but also very cheap, especially when the gears are made of plastic.

There are a few different types of servomotors available for purchase; positional rotation servos, continuous rotation servos and linear servos. Positional rotation servomotors are the most common and the type employed in the Auto-labyrinth. The output shaft of a positional rotation servo rotates 180 degrees with physical stops on each side of the rotation. The control signal adjusts the physical position of the output shaft of the motor. Continuous rotation servos are very similar but they can rotate indefinitely in either direction and the control signal adjusts the speed and direction of rotation rather than the position. The final type, the linear servo, are very similar to positional rotation servos but have an additional mechanism to allow it to have a back and forth rotation instead of a circular one. These are less common and harder to find.

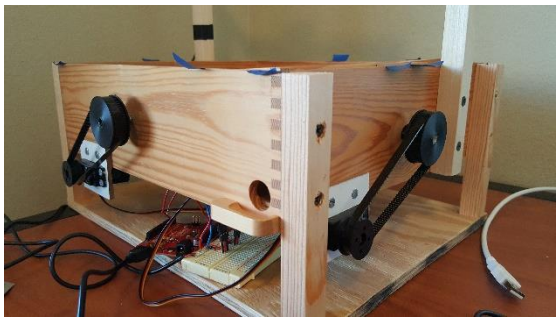

Figure 5 Two Independent Servo motors and control mechanisms

The Auto-labyrinth utilizes two positional rotation servos, one on each of the control knobs for the Labyrinth game board as shown in Figure 5. Each servo controls the tilt of the game board on one of its two axes. A 5v DC source and a control signal power the servos. The control signal is a Pulse Width modulated signal with an amplitude of 5v. The pulse width modulated (PWM) signal has a minimum pulse, maximum pulse and period of repetition. The

signal is 0 except for a pulse of 5v that comes each period and the width or duration of this pulse determines the position of the servo. The servos expect a pulse every 20ms, the period. The pulse will be between 1ms and 2ms. A 1ms pulse corresponds to the 0 degree position, 1.5ms the 90 degree position and 2ms the 180 degree position as shown in Figure 6.

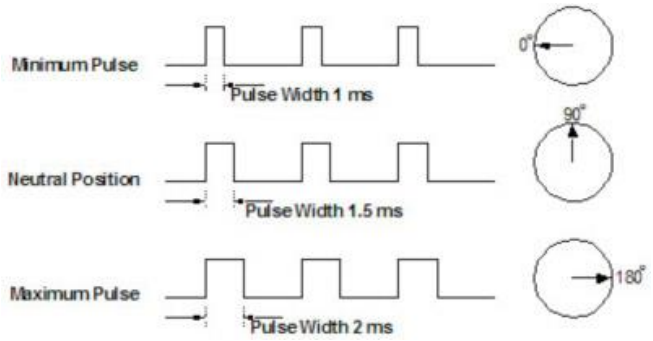

Figure 6: PWM servo Motor control

Chapter 2

Hardware Implementation

In order to build a control algorithm test bed, a dynamic system for the controller to manipulate is required. The Brio® Labyrinth game is used as this dynamic system. The game features many nuances and intricacies for the control system to overcome. Before the board could be used it required the addition of a few components to create the control system. Motors were added to the game controls to allow the computer to tilt the game board. An Arduino board facilitates communication between the computer and the motors. The last addition is a camera mounted above the board so the computer can track the position of the ball.

2.1: Game board

The Brio® Labyrinth game itself had only a few small modifications made to it. The project started with the goal of keeping the game as close to stock as possible so it could also be played by hand without the control system. The knobs that came on the board were flimsy and weak, they were replaced with pulley wheels that were roughly the same size but would allow a micro XL timing belt to fit on them. The game utilizes a rod connected to each knob that turns a string mechanism inside the game and tilts the board on its respective axis. This design was not ideal for a control system but was kept since it adds hysteresis to the system and poses a challenge. Turning the knob 10 degrees in one direction and back does not necessarily return the board to the same tilt angle. The rods spun infinitely in one direction and not in the other since the string would slip on the rod. This caused the motors to not be able to tilt the board in one direction after a few minutes of operation. To overcome this problem sticky textured tape was

wrapped around the rod where the string winds around it. This prevented the string from slipping on the rod and allowed the motors to always have control over the game board tilt without removing the hysteresis.

In addition to the above changes to the tilting system of the game, the board itself needed to be slightly modified. In testing, the ball kept falling through the holes before the system could learn the characteristics of the balls motion. Since an automated mechanism to restart the ball at the beginning of the maze was not practical, the holes in the board were taped over to avoid the problem. Above the game board a wooden support structure is added to mount the camera in a way that would keep the board in focus and not capture other areas around the game. This support structure is anchored to the base of the game and does not tilt with the board.

Figure 7: Game board with modifications

2.2: Vision System

Every control system needs feedback to know the effects of its previous actions. The Auto-Labyrinth utilizes a vision based feedback system. The vision system uses a webcam to track the location of the ball and use that as the feedback of the control system. Figure 7 depicts the design of the game with the camera mounted above. The webcam in use on the system is a Logitech C500 webcam which was chosen because it was recycled from a previous project and provided an adequate picture resolution. The webcam is adjusted so the resolution of the images captured is 480x600. This allowed the images to process faster as well as locate the ball quicker and more accurately. A series of trials, conducted at all the possible resolutions, found this to be the ideal balance between image size and picture quality. The webcam at this resolution functions well but lacks an autofocus feature that would ensure the clearest picture regardless of the image resolution.

Throughout the project the vision system presented the biggest challenge. The efficiency of the system hinges on locating the ball and determining which way to tilt the board to keep it on track and follow the maze path in as little time as possible. The algorithms used are all streamlined and simplified as much as possible to reduce the computational delay. Paired with the reduced image resolution the computational delay becomes negligible. The delay in the system comes from capturing an image into LabVIEW for processing. LabVIEW comes with great tools to make capturing images very simple to set up but these algorithms are not nearly as quick as assumed. Without the ability to streamline the capture of the image, this became the limiting factor in the efficiency of the system.

2.3: Motors

To facilitate the ability of the computer to tilt the game board two HiTec servo motors are attached to the control knobs of the game. Servomotors are ideal for this situation because they can programmatically have their position set and hold the position as long as they have the control signal. Stepper motors were considered for this project but ultimately not used because they required the addition of a motor encoder and would take longer to adjust the position. The servos are a very efficient and powerful option that can be controlled by a variety of LabVIEW compatible devices. The servos used are HS422 motors with a speed of .16s/60º. This was a good balance of power, speed and cost for this application.

2.4: Communication system

A servomotor's position is set by a Pulse Width Modulated (PWM) signal. To make the motors move a PWM signal is required and in this case that signal needed to have a 5v amplitude and pulse width range of 1ms to 2ms as discussed in Chapter 1.6. At first a NI MyDAQ device was used to create this signal from LabVIEW code. The MyDAQ created an analog output with a pulse train programmatically set by LabVIEW. This worked well to control the motors but the rate of change of the PWM signal was very slow. The MyDAQ has a software-timed clock which took too long to modify. The MyDAQ was swapped out for an Arduino Uno chip on the Spark Fun Red board. The Arduino uses a hardware-timed clock making changes to the PWM signal much quicker to execute. The Arduino also allowed more control over the USB connection settings. The Arduino is set up to change at an interval of 1ms. This is plenty fast for this project since the acquisition of an image into LabVIEW requires 10ms. This reduced the

overall cost of the project as well. The MyDAQ is $150 and the Arduino was only $20. The MyDAQ was required for a previous class so using it as a starting point was logical.

Chapter 3

Algorithms

The majority of the work on this project lies in the interface and control algorithms programmed in LabVIEW. The algorithms capture an image from the camera, analyze it to find the position of the ball, relate that position to the desired position of the ball for that part of the maze and create a motor control signal to get the ball to go to the desired location. LabVIEW made much of the programming easier than other languages since it has many built in interfacing features.

3.1: Sensor Interface

The first step in building the algorithms for the Auto-labyrinth was to interface the sensors, the webcam and the motors. LabVIEW has a built-in interfacing system for USB webcams called IMAQdx. This palette streamlined the interfacing with the webcam, opening a connection to the camera and grabbing images from it. Grabbing images from the camera rather than snapping allows the acquisition of images to be much quicker and more frequent. Grabbing images keeps the communication to the camera open and allows the capture of an image without having to set up a new session each capture. Using the maximum resolution of the camera allowed the accurate tracking of the ball from a high-quality image. After some testing, changing the image quality to about half the max resolution of the camera (480x600) made the image acquisition and processing quicker as discussed in Chapter 2.2.

The other sensor interfaced on this project was the motor system. Having two identical servomotors, one on each axis, two PWM signals are needed to control the motors. The signals have the same characteristics but the duty cycles will not be the same. A USB port cannot output a PWM signal on its own so an encoder is needed to create a physical PWM signal from a software description of it. First, a NI MyDAQ device was used to create the physical signals but later swapped out for a faster Arduino board as discussed in Chapter 2.4. The Arduino board is not a device that LabVIEW can natively control. An open source toolkit for LabVIEW from MakerHub provides an easy means of controlling the Arduino with LabVIEW. Using the Arduino toolkit, a serial communication channel is opened between LabVIEW and the Uno board. Within this channel the two PWM signals are specified on two different ports and the parameters of the signals are set. The algorithm allows the duty cycle of each to change independently and this facilitates the independent movement of each motor.

3.2: Image Rectification

The camera mount discussed in Chapter 2.3 anchors the camera very securely to the base but means it is not always perpendicular to the tilting game board. The camera and algorithms find the ball based on the pixel location on the camera but when the board tilts that pixel location can change, even if the ball does not move on the game board. The image the camera is capturing needs to be rectified and have a consistent axis system built on it to ensure an accurate description of the ball's location every time. To do this, the same object tracking algorithm used to find the ball (described in chapter 3.3) is used to locate the four corners of the game board and build an axis system based on those points. This facilitates the creation of an accurate square axis

system based off an image of the tilted board when it was not square. This algorithm will be much more accurate than a pixel based system since the image of the game board is always square and there is no variation in the location of the ball or the target points.

Rectifying the camera image and overlaying an accurate coordinate system requires a great deal of computing power. The greater computing power means a greater amount of time required to generate the rectified image. The delay caused by this step was too great to make it useful in the system. Although the image rectification system was accurate, it could not function in a real-time system. Removing the rectification algorithm and basing the ball's location as well as the guidance system off pixel locations reduced the computational delay for this step. The slight error in pixel location vs location on the board is taken into account in the guidance system programming. The guidance system accepts a range of pixel values rather than a specific location.

3.3: Ball Location and motion estimation

With the image acquired from the camera into LabVIEW the algorithm needs to extract the location of the ball so the board can be tilted to keep it on track to follow the path through the maze. LabVIEW has a toolkit of image processing tools that includes object tracking. For this system, the pattern matching algorithms proved to be the most accurate. The pattern matching algorithm uses a template image of the ball saved on the computer and builds a profile of the image including characteristics of the shape, curves and characteristics of the ball. The image of the game board captured from the camera is scanned pixel by pixel to find a profile matching that of the ball. The algorithm is set to find one match that exceeds 85% certainty. These settings

proved to be the most accurate in testing. LabVIEW is able to locate the ball but it takes a bit of time to process the image from the camera into LabVIEW. That time causes a delay in a real-time system so compensation for that delay is necessary.

The camera delay in the system causes the ball to be at a slightly different location than the image captured at the time the ball is located. In order to compensate for this delay, a motion estimation algorithm is necessary. This delay is described in detail in Chapter 2.2. A very basic motion estimation algorithm was utilized and adjusted to be effective. The estimation algorithm implemented uses the current location of the ball and the most recent past location of the ball to predict where it will be when the algorithm completes execution. The estimator finds the difference between the current and past location, divides that difference by two and adds it to the current location to find an estimate of the ball's location after the processing delay. The algorithm does this in both the X and Y direction independently to give a more accurate estimation. This system works well and overcomes the time delay of the system. However, it is not perfect and in certain isolated instances changes the balls target location to one that is not reachable but within one iteration of the algorithm, it is overcome.

3.4: Desired Path

The algorithm can locate the ball on the game board and compensate the location for the delay in acquiring the image from the camera. With the estimated location of the ball, the system needs to determine where the ball should be moving. To determine this location in the quickest way possible a map is built on the game board based on the image pixels that has target points at every curve along the desired path. Around each target point, a range of pixels is specified so

when the ball enters that range the target is moved to the next target along the path. This is illustrated in Figure 8 and overcomes any inaccuracies in pixel location caused by the board tilt. The algorithm is a case structure within a case structure. Each case structure is an axis and controlled by the compensated location of the ball on that axis. For each range around a target there is a new target. Based on the location, the algorithm finds the next target location which the PID controller can use as the set point to control each motor.

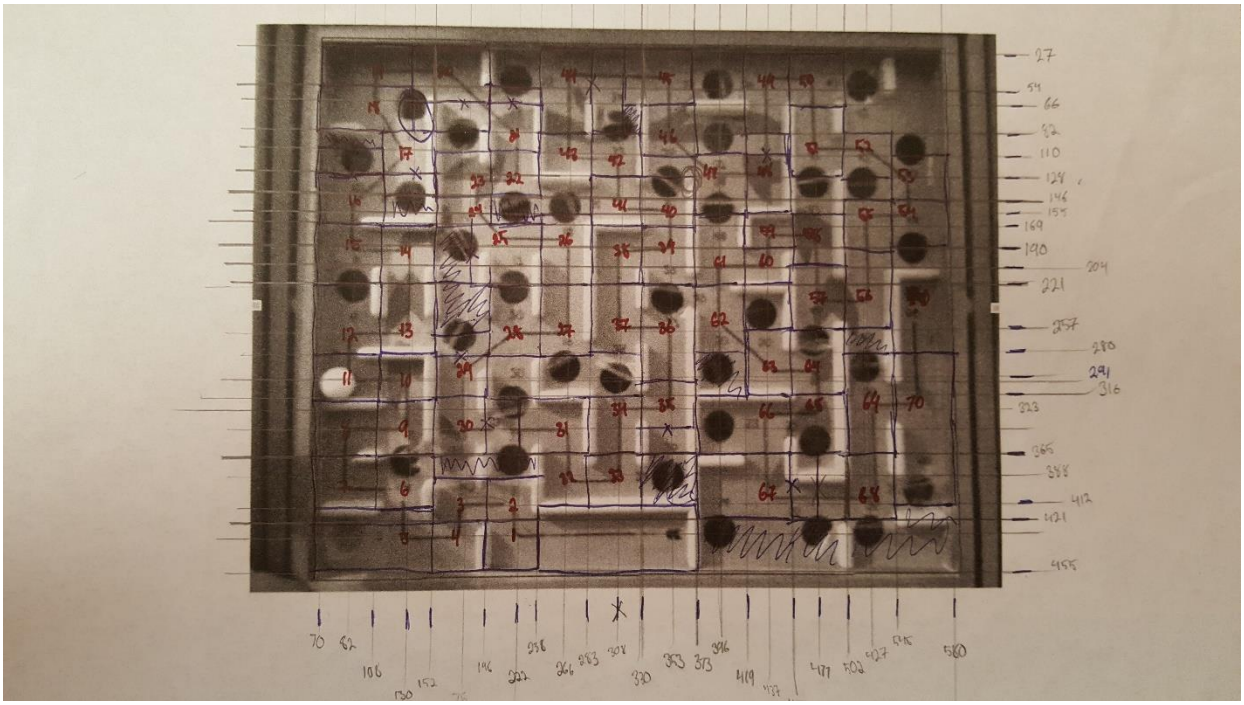

Figure 8: Navigation coordinate system

The pixel based point to point navigation system is implemented because it is the quickest algorithm to execute and there is already a delay in the system. The other option was to search around the balls location to find the line marking the desired path but this would require another object tracking algorithm and create another delay in the system.

3.5: PID

The backbone of the Auto-labyrinth control system is a PID controller. The design of a PID controller is described in Chapter 1.3. LabVIEW has a toolbox that contains a few algorithms that help implement a PID controller. The PID controller is duplicated in the main algorithm so the algorithm for each axis can execute simultaneously. This allows the algorithm to execute quicker and does not cross any signals by calling the same function for each axis at the same time. Both PID controllers execute the same way. The compensated location is compared to the desired next target point and the PID determines the output based on the difference as well as the previous output of the PID controller. The limits of the PID controller are set to the limits on the duty cycle of the motor control signal, 0.55ms and 2.5ms respectively. This allowed the PID controller to control the duty cycle of the motors directly. This also means the PID gains need to reflect the large range of outputs. This increased the proportional gain to 75 and kept the integral and derivative at 0.01 and 0 respectively. This makes the PID controller a PI controller but this combination worked best for the system.

3.6: Motion control

The output of the PID controller directly corresponds to the desired duty cycle of the PWM control signal for the servo on each axis. In order to create a PWM signal based on the duty cycle some hardware is needed as discussed in chapter 3.1. Using the Arduino board and the MakerHub toolkit the characteristics of the PWM signal could be set in LabVIEW which include; amplitude and frequency and varying the duty cycle. The computer interfaces with the Arduino as a serial device so at the initiation of the program it takes some time to open a

communication channel with the Arduino but once it is open it can be updated every 1ms. This interface removed any delay encountered between the duty cycle being calculated by the PID controller and the motors executing it. The algorithm uses two independent PID controllers to determine the duty cycle for each axis. The Arduino is programmed to change the respective channels on its output immediately and hold that output until a new duty cycle is calculated.

Chapter 4

Results

The Auto-Labyrinth project is a great exploration into control system theory as well as interfacing hardware with software through LabVIEW. The project was a success with the ball navigating the maze autonomously with a few minor modifications.

4.1: Discussion of results

The goal of the Auto-labyrinth was to autonomously navigate the ball through the maze with a control system that uses a PID controller. The Auto-labyrinth successfully navigates the maze but the board required some slight modification. Without an automatic method to reset the ball onto the game-board when it fell through the holes a modification had to be made to keep the ball from falling through the holes. This problem was addressed by covering the holes in the board. This does not affect the ability of the Auto-labyrinth to test a control algorithm and remains in line with the goals of this project. The design of the algorithms worked well and with some extensive PID controller tuning it completes the entire maze every time. The system can still measure the effectiveness of a control algorithm by reducing the number of times the ball would have fallen through a hole in the maze if it was not for the tape.

The Auto-labyrinth is limited in a few ways: the delay in acquiring the image from the camera and having it ready for processing in LabVIEW and the delay in writing the PWM signal to the motors. A solution to reduce the delay in image acquisition with LabVIEW could not be reached. This is a limitation of LabVIEW and how it interfaces with a USB2.0 camera. Exploring the use of a different programming language could possibly alleviate this problem but would not stay aligned with the goal of programming in LabVIEW. The other option that could be explored to alleviate the problem with the current hardware is running the program on a MyRIO board. The delay in the motor control system was able to be

overcome. Initially the Auto-labyrinth used a NI MyDAQ to bridge the gap between LabVIEW and a physical control signal. The MyDAQ is a software timed data acquisition device designed for student use. For the Auto-labyrinth it became very limited since it goes above the intended purpose of the device. After swapping the MyDAQ out for an Arduino Uno board the delay was minimized. The Arduino is hardware timed and could be configured to update more frequently than the MyDAQ, this was discussed in detail in Chapter 2.4.

4.2: Evaluation of future work

The Auto-labyrinth was successful in its goal of providing a testbed for control theories and exhibiting its effectiveness with a PID controller. Despite the success there are some areas that could be expanded on. The project was started with the intent to program in LabVIEW to show the robust tools it offers for control systems. LabVIEW's interface with the camera proved to have the greatest delay in the entire system and programming in a different language could reduce that delay. The most appealing choice for the image acquisition and processing would be in C and use the Open CV toolkit that is open source. The entire control system could be rewritten in C code since the Arduino is programmed with C natively.

The delay between an image being seen by the camera and LabVIEW having it available for processing is the largest delay. This delay is not caused by the LabVIEW environment itself but by the USB camera interface with Windows. USB 2.0 cameras such as the one implemented communicate with Windows and therefore LabVIEW through a DirectShow driver provided by the manufacturer. In this configuration, which is standard on all USB 2.0 cameras, the camera captures an image and instead of the image being imported directly to LabVIEW the camera drivers on the computer compress the image. The compression occurs to allow the video files to be transferred over the USB 2.0 protocol. Other video interface standards do not have this extra compression step and would be much faster. To remove the

image capture delay, a camera based on a different standard could be employed like an IP or USB3.0 camera. Unfortunately, all of these options are far beyond the scope of the budget available for this project. As a proof of concept the USB2.0 camera will suffice for this project despite its limitations.

In order to alleviate the image capture delay and remain in the LabVIEW programming language the use of a MyRIO device could be explored. The MyRIO would allow the entire program to execute on the device without the use of a computer. This may reduce the delay in interfacing with the webcam to acquire an image. This option is very intriguing since it would make the entire system stand alone and explore the use of the MyRIO which is more common in industry than the MyDAQ. The My RIO may not be plagued by the DirectShow delay associated with connecting a USB webcam to the computer. This would allow us to keep all of the existing hardware and algorithms the same and is an even more intriguing option for future research when paired with a more modern camera protocol like USB 3.0.

To expand upon the algorithms already in place adding a Kalman filter could help to remove the delay from the image acquisition. The Kalman filter requires an input parameter and in the case of the Auto-Labyrinth the duty cycle of the motor control PWM signal could be used. From the duty cycle the angle of the board could be calculated as well as how it would affect the motion of the ball. The calculation of the board tilt based on the control signal duty cycle would be an approximation since the tilt mechanism includes hysteresis. This algorithm would work very well with a faster camera and should make the Auto-Labyrinth able to navigate the maze without the holes being taped over. The addition of the Kalman filter would be limited in the current set up since the camera is not fast enough. If the Kalman filter could be modified to provide an estimated location of the ball more frequently than the camera alone can then it could be a great improvement. Historically speaking, implementing the Kalman filter based on the PWM signal would be best since it limits the amount of hardware required. With cell phones bringing the price of gyroscope sensors down dramatically adding one to the project would be within the budget. The added sensor could serve as the input to the Kalman filter and provide very accurate measurements of the board angle when paired with a faster camera. Provided the gyroscopic sensor could be read from

more frequently than the current camera it would make estimating the balls location between camera images even more precise. The next evolution of this project should include a faster camera and a Kalman filter with or without the gyroscope sensor.

Appendix A

Algorithm Glossary

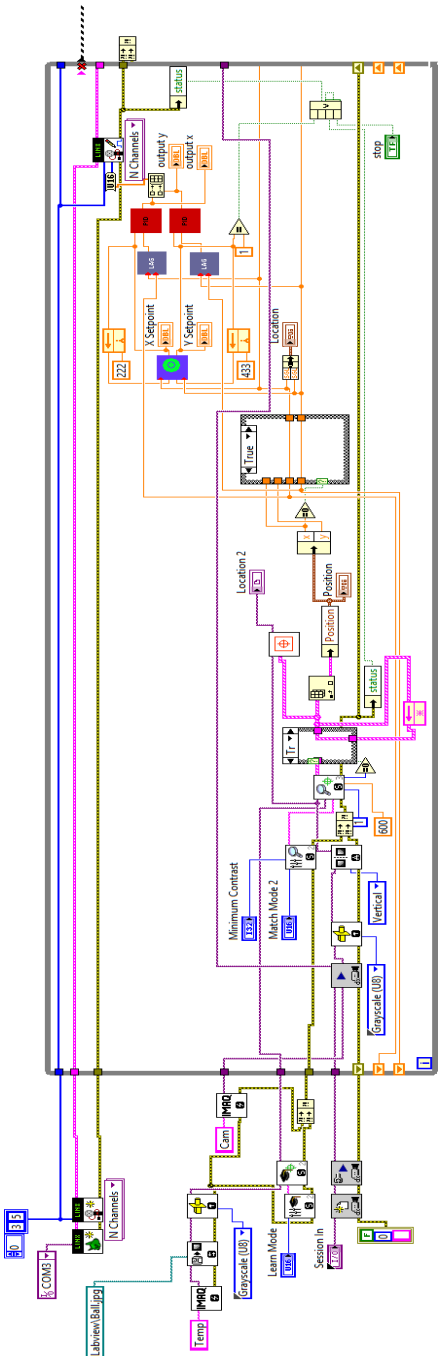

Figure 9: Main Control Algorithm

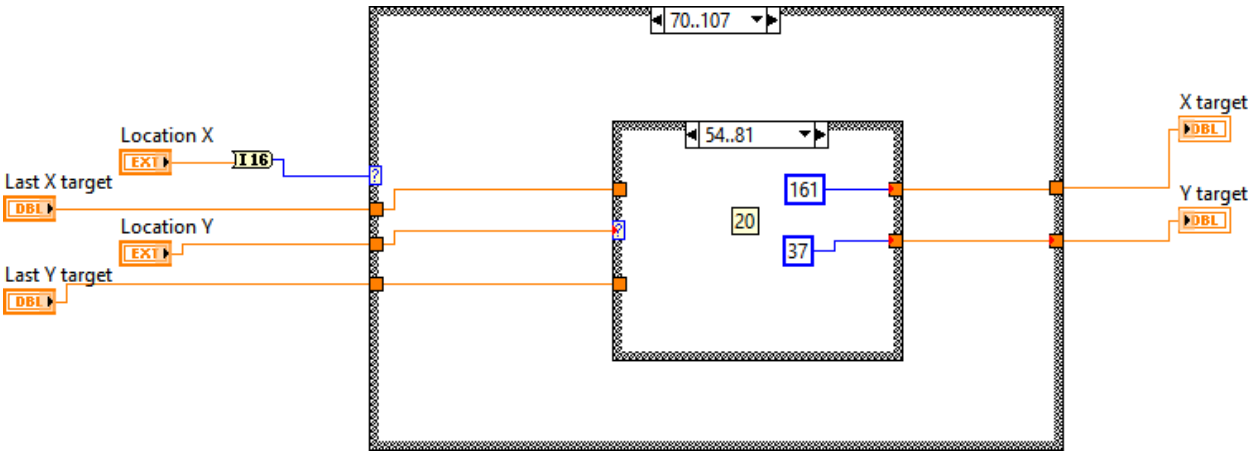

Figure 11: Set-point selecting algorithm

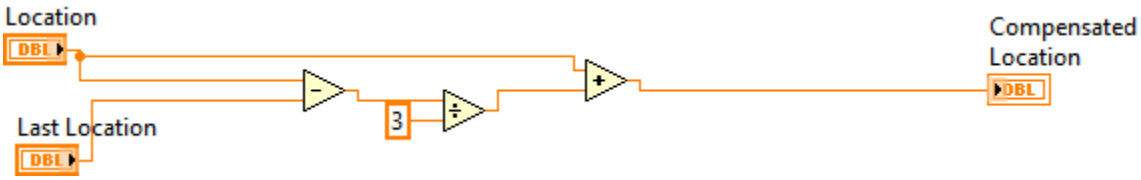

Figure 12: Location Compensation Algorithm

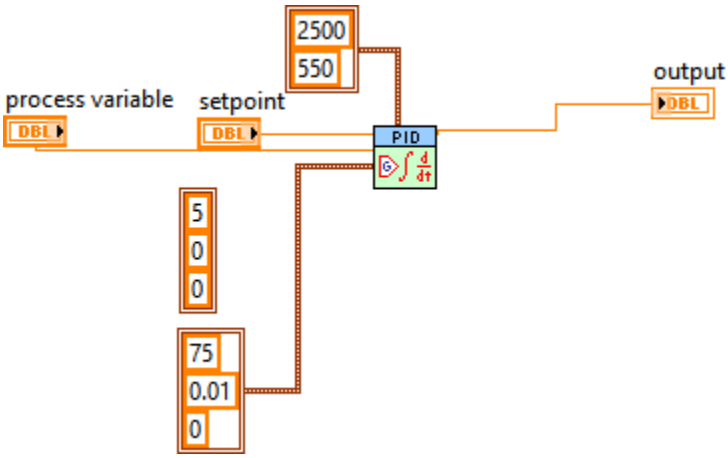

Figure 10: PID conditioning algorithm

BIBLIOGRAPHY

[1] Ni.com, 'PID Theory Explained - National Instruments', 2015. [Online]. Available: http://www.ni.com/white-paper/3782/en/. [Accessed: 30- Sep- 2015].

[2]G. Goodwin, S. Graebe and M. Salgado, *Control system design*. Upper Saddle River, N.J.: Prentice Hall, 2001.

[3]*IMAQ Vision for Labview User Manual*, 1st ed. Austin, Texas: National Instruments, 2004.

[4] Home.wlu.edu, 'The Extended Kalman Filter: An Interactive Tutorial', 2015. [Online]. Available: http://home.wlu.edu/~levys/kalman_tutorial/. [Accessed: 30- Jul- 2015].

[5]S. Challa, M. Morelande, D. Musicki and R. Evans, *Fundamentals of Object Tracking*. Cambridge: Cambridge University Press, 2011.

[6] Ni.com, 'Pulse Width Modulation (PWM) Using NI-DAQmx and LabVIEW - National Instruments', 2015. [Online]. Available: http://www.ni.com/tutorial/2991/en/. [Accessed: 30- Jul- 2015].

[7]F. Reed, 'How Servo Motors Work', *Jameco.com*, 2015. [Online]. Available: http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html. [Accessed: 30- Jul- 2015].

[8]J. Liu, P. Zhang and F. Wang, 'Real-Time DC Servo Motor Position Control by PID Controller Using Labview', *2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1, pp. 206-209, 2009.

[9] Hitec, *Announced Specification of HS-422 Standard Delux Servo*, 1st ed. Hitec, 2015.

[10]*NI-IMAQ for USB Cameras User Guide*, 1st ed. Austin, Texas: National Instruments, 2005.

[11]"Why Is There a Lag with My USB 2.0 Camera in My Frames with a Lot of Movement? - National Instruments", *Digital.ni.com*, 2010. [Online]. Available:

http://digital.ni.com/public.nsf/allkb/93CFD4E9068B9278862576C0007386CA. [Accessed: 20- Oct- 2015].

[12]"14 | Servo [LabVIEW MakerHub]", *Labviewmakerhub.com*, 2012. [Online]. Available: https://www.labviewmakerhub.com/doku.php?id=learn:tutorials:libraries:linx:sparkfun_inventors_kit:servo. [Accessed: 20- Oct- 2016].

[13]"Community: Proximity-Controlled Servo Motor - National Instruments", *Decibel.ni.com*, 2011. [Online]. Available: https://decibel.ni.com/content/docs/DOC-16989. [Accessed: 20- Oct- 2015].

[14] R. Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]," in *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128-132, Sept. 2012.doi: 10.1109/MSP.2012.2203621
[Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6279585&isnumber=6279563 [Accessed: 20- Oct- 2015].

Matthew Baranoski

267.221.7093
mib5230@psu.edu

39 Harvest Lane
Perkasie, PA 18944

Academic Vita

Education

The Pennsylvania State University **December 2016**
Bachelors of Science in Electrical Engineering
Honors: Schreyer Honors College, student

- **Related Coursework**: Circuits and Devices, Logic Design, Electronic Circuit Design, Electromagnetics, Nanoelectronics, Continuous-time Linear Systems, Communication Systems, Computer Organization and Design, UHF and Microwave Engineering, Remote Sensing Systems, Microprocessors and Embedded Systems
- **Honors Thesis:** Completed a control system theory exploration through a vision based motor control system programmed in LabVIEW
- **Extracurricular Activity:** President, Penn State University Lehigh Valley Cycling Team

Pennridge High School **2011**

- National Honors Society
- Top 5% of graduating class

Experience

US Olympic Team **2016**
Track Cyclist

- Train and compete at events across the globe
  Manage relations and contracts with sponsors
- Organize and carry out a successful training program
- Coach junior development camps for USA Cycling

Penn State University **2014- 2015**
Student Tutor: Calculus, Physics, Economics

- Reinforce concepts learned in the classroom
- Arrange appointments to meet with students and professors

Select Related Course Projects

Honors Thesis: Auto-Labyrinth **2016**

- Develop object tracking algorithm in LabVIEW
- Interface servo motors with LabVIEW using an Arduino board
- Create a real-time vision based control system

MIPS Processor design **2014**

- Design a MIPS based CPU
- Code processor in Verilog Hardware Descriptive Language (VHDL)
- Simulate and test processor
- Execute sample programs on processor

Proficiencies

LabVIEW Certified, C, C++, Assembly, Verilog, MS Office, ADS, Matlab

Achievements and Accomplishments

- 2016 US Olympic Team 2016
- Dean's List Penn State 2011-2014
- Chancellor's Award 2011-2013
- Schreyer Academic Excellence Award 2011-2016
- Madden Honor's Scholarship 2016