Self-Correcting Projectile Launcher

by
Ryan Kindle
Diana Mirabello
Yena Park
Josh Schuster

ECSE-4460 Control Systems Design
Final Project Report
May 3, 2005

Rensselaer Polytechnic Institute
The design of a Self-Correcting Projectile Launcher is motivated by the desire to improve the accuracy of existing launchers without the use of extensive sensor networks, and to demonstrate the viability of an iterative learning algorithm for disturbance rejection. The goal is to fire a disc from a commercially available toy gun at a target, and have its accuracy improve with each subsequent shot until the target is struck precisely. The gun is manually oriented so that the disc will strike a touchscreen sensor at a point other than the predetermined target. As the disc strikes the touchscreen, the error in the trajectory is calculated based on the difference between the target and impact site, and the orientation of the launcher adjusted to compensate using a pan and tilt mechanism. Subsequent disc impacts further refine the orientation of the launcher, until it successfully strikes the target. The system successfully strikes the target in three tries or less in all demonstrations.
# TABLE OF CONTENTS

1. Introduction ......................................................................................................................... 1

2. Professional and Societal Consideration ............................................................................... 3

3. Design Procedure .................................................................................................................. 4

4. Design Details ....................................................................................................................... 6
   4.1. Model Development .......................................................................................................... 6
   4.2. Control Development ...................................................................................................... 15
   4.3. Physical Design ............................................................................................................. 17
   4.4. Sensor Development ..................................................................................................... 20
   4.5. Algorithm Development ............................................................................................... 21

5. Design Verification ............................................................................................................... 24
   5.1. Model Verification .......................................................................................................... 24
   5.2. Control Verification ....................................................................................................... 26
   5.3. Physical Verification ..................................................................................................... 29
   5.4. Sensor Verification ...................................................................................................... 37
   5.5. Algorithm Verification ................................................................................................. 38

6. Costs ................................................................................................................................... 41

7. Conclusions ......................................................................................................................... 43

8. References ........................................................................................................................... 45
   Appendix A: Contribution of Team Members ....................................................................... 46
   Appendix B: Datasheets ......................................................................................................... 47
   Appendix C: MATLAB Scripts ............................................................................................. 50
   Appendix D: Simulink Models ............................................................................................... 65
   Appendix E: Team Member Resumes .................................................................................. 67
LIST OF FIGURES

Figure 4.1.1: Friction Identification Simulink Model ................................................................. 6
Figure 4.1.2: Velocity Diagram without Filter ............................................................................. 7
Figure 4.1.3: Velocity Diagram after Implementation of Second Filter ....................................... 8
Figure 4.1.4: Velocity Diagram After Implementation of Third Filter for Tilt Axis .................... 8
Figure 4.1.5: Velocity Diagram After Implementation of Third Filter for Pan Axis ..................... 9
Figure 4.1.6: Plot of Torque vs. Steady State Velocity of Tilt Axis ............................................. 10
Figure 4.1.7: Plot of Torque vs. Steady State Velocity for Pan Axis ............................................ 10
Figure 4.1.8: Parameter Equation Verifying Simulink Model ...................................................... 11
Figure 4.1.9: Experimental Data on Tilt Axis with Chirp Input ................................................... 12
Figure 4.1.10: Experimental Data on Pan Axis with Chirp Input ............................................... 13
Figure 4.1.11: Experimental Data on Tilt Axis with Sine Input ................................................... 14
Figure 4.1.12: Experimental Data on Pan Axis with Sine Input ................................................... 14
Figure 4.2.1: Closed Loop Feedback Model .................................................................................. 15
Figure 4.2.2: Experimental Data on Tilt Axis with Step Input ..................................................... 16
Figure 4.2.3: Experimental Data on Pan Axis with Step Input ..................................................... 16
Figure 4.3.1: Aluminum Mounting Base ....................................................................................... 17
Figure 4.3.2: Disc Launcher ......................................................................................................... 18
Figure 4.3.3: Automated Firing .................................................................................................... 18
Figure 4.3.4: Mounting of System ............................................................................................... 19
Figure 5.1.1: Simulation vs. Experimental Response to Chirp Input on Tilt Axis ...................... 24
Figure 5.1.2: Simulation vs. Experimental Response to Chirp Input on Pan Axis ..................... 25
Figure 5.1.3: Simulation vs. Experimental Response to Sine Input on Tilt Axis ....................... 25
Figure 5.1.4: Simulation vs. Experimental Response to Sine Input on Pan Axis ..................... 26
Figure 5.2.1: Transient Response Due to Step Input on Tilt Axis ............................................... 27
LIST OF TABLES

Table 1.1: System Specifications ................................................................................................................2
Table 4.1.1: Tilt & Pan Friction Constants ...............................................................................................10
Table 4.1.2: Parameters for Pan and Tilt Axes .........................................................................................12
Table 4.2.1: PID Gains on Both Axes .......................................................................................................15
Table 4.4.1: Touchscreen & Controller Pinouts .......................................................................................21
Table 4.5.1: Vertical Displacement Equations .........................................................................................22
Table 5.2.1: Response Characteristics of Closed Loop System .................................................................27
Table 5.2.2: Gain and Phase Margins of Closed Loop System .................................................................29
Table 5.3.1: Disc Shooter Accuracy .........................................................................................................30
Table 5.3.2: Preliminary Trajectory Results .............................................................................................31
Table 5.3.3: Short Range Trajectory Results ............................................................................................31
Table 5.3.4: Average Short Range Trajectory ..........................................................................................32
Table 5.3.5: Average Vertical Displacement at Set Angle and Distance ..................................................34
Table 5.3.6: Variables a & b in Y = aX + b Best Fit Line Equations .........................................................35
Table 6.1: Pan and Tilt Mechanism Cost ..................................................................................................41
Table 6.2: Cost of Additional Parts Needed for System ...........................................................................41
Table 6.3: Labor Costs ..............................................................................................................................42
Table 6.4: Lab Usage Costs .......................................................................................................................42
Table 6.5: Total System Costs ..................................................................................................................42
Table 6.6: Actual Cost of System ..............................................................................................................42
The world is an unpredictable place, and when it comes to projectile motion the unknowns one must take into account are numerous and complex. The purpose of the project proposed in this paper is to create a prototype Self-Correcting Projectile Launcher. Instead of rejecting disturbances in advance to improve the chance of striking a target in a single shot, the system fires repeatedly at a target and learns from its errors to strike more accurately with subsequent shots. The motivation for the project was to create a launcher that was capable of high accuracy but without the extensive sensor setup required to predict disturbances. The creation of such a system would be an economical solution for many applications, as well as a significantly more portable system.

A trajectory may be altered by the varying currents, density, and stresses found in any given medium. While some of these may be measured and accounted for in advance, the variable nature of the environment dictates that no disturbance will be completely rejected. The measurements and computations to achieve such an incomplete error reduction would also require a significant investment in equipment. If an iterative learning approach is utilized instead, all that is required to reject an error is the ability to sense to actual impact’s location relative to the target and projectiles for multiple shots.

The design of projectile launchers is hardly a new endeavor; everything from batting cages to artillery has attempted to hit a target accurately. While small consumer systems such as a ping-pong machine or batting cage lack sensory capabilities and rely instead on calibration, military applications are usually far more sophisticated. However, while guided artillery batteries are capable of rejecting disturbances through sensors and computers, smaller launchers such as mortars are not. The same desire for compactness and economy is what drives commercial applications to rely simply on calibration to ensure accuracy. Both commercial enterprises that market projectile launchers requiring accuracy as well as the military would benefit from a small cost-effective trajectory correction system.

Since the purpose of the prototype is simply to demonstrate the viability of such a system, the most significant functional requirement is accuracy. The firing system must be above all be capable of
consistently striking the same location when locked into a position; otherwise, the iterative learning algorithm will be unable to calculate corrections. Also, the pan and tilt mechanism used to aim the launcher must be able to accurately adjust and detect its orientation, and the controller must ensure that steady-state position error is near zero and that noise tolerance is high. In order to reject disturbances that affect the launcher directly, the rise time and settling time of the control system must be low so that it returns to the desired orientation quickly; the percent overshoot should be small but is not a critical factor. The range of motion for the launcher has to be sufficient to strike any target within range. The payload for the pan and tilt mechanism, specifically the launcher and its mounting apparatus, will be made of plastic and aluminum and hence lightweight. The cost of the system will be low enough to allow for its widespread adaptation to the various low-cost applications discussed previously. Quantitative specifications are described in Table 1.1 below.

<table>
<thead>
<tr>
<th>TABLE 1.1: SYSTEM SPECIFICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range of Motion (Pan)</strong></td>
</tr>
<tr>
<td><strong>Range of Motion (Tilt)</strong></td>
</tr>
<tr>
<td><strong>Rotational Accuracy</strong></td>
</tr>
<tr>
<td><strong>Steady State Error</strong></td>
</tr>
<tr>
<td><strong>Noise Tolerance</strong></td>
</tr>
<tr>
<td><strong>Rise Time</strong></td>
</tr>
<tr>
<td><strong>Settling Time</strong></td>
</tr>
<tr>
<td><strong>Percent Overshoot</strong></td>
</tr>
<tr>
<td><strong>Payload</strong></td>
</tr>
</tbody>
</table>

The scope of effort required to realize the project proposed here is significant and diverse. As such, it has been broken down into several subsystems. A model of the pan and tilt mechanism is developed, with friction identification and parameter estimation being the major factors. The mechanical design of the launcher and its attachment to the pan and tilt mechanism is another project that requires refinement of the model. Once that model accurately reflects the completed launching system, the design of the PID controller becomes possible. The other two major components were the development of the touchscreen sensor and learning algorithm. The final step is to integrate all the separate projects so that they interface through a personal computer.
2. PROFESSIONAL AND SOCIETAL CONSIDERATION

The prototype model will not suffer from any constraints due to economics, environment, sustainability, manufacturing, ethics, health and safety, or sociopolitical impact. Although some of the potential applications of the concept, specifically those involving weaponry, may generate concerns based on ethical, environmental, and sociopolitical impacts, it is believed that those concerns are due to existing technology and will not be significantly amplified by the project. Improved accuracy of small artillery may in fact improve a number of factors such as safety and economics due to the more contained nature of the destruction. A possible manufacturing concern will be that each application will likely need to modify the project to utilize a different sensor depending on how it is to be used: a glass touchscreen is not quite up to the task of determining the error in an 80 mph fastball’s trajectory. Beyond that however, the project is an exercise in theory that by itself, carries no inherent moral quandaries.
3. DESIGN PROCEDURE

The first step in the system’s development is to determine the Coulomb friction and viscous friction. Data sets are collected in MATLAB for various voltages containing angular velocities, and by using the equation relating the torque and voltage, the torque is calculated. Analysis of the torque versus angular velocity graph allows both the positive and negative Coulomb friction and positive and negative viscous friction to be determined.

The physical mounting of the system consists mainly of an aluminum base plate with a cylindrical mounting bracket, secured by screws. The disc launcher itself is then mounted to that plate using adjustable screws, to allow for the later installation of an automated firing system. The touchscreen sensor is mounted to a rack with several available mounting positions using a back plate and C-clamps. The rack is unsecured so that several ranges can be used in the verification of the system. Alternative means of mounting either system are virtually limitless, and these were chosen for their simplicity and adjustability.

Since the inertia of the completed mechanical system cannot be precisely calculated, the identification of the final model has to be done differently than the identification of friction. First, verifying the validation of the method is done. Second, with the method validated, parameters are identified. Lastly, the accuracy of the parameters is observed by comparison to the actual system.

That model is integral in the development of the PID controller. The stability limits of the system are observed in the step response, and then calibrated to values that achieve stability while maintaining an adequate transient response. The controller is then tuned to eliminate steady-state error while maintaining a rise time and settling time that met specifications. Once the response of the model meets that of our desired system, it is applied to the completed pan and tilt launcher and compared to the response the model expected. Other controller options include phase-lead, phase-lag, and full-state feedback pole placement; PID was chosen for its familiarity, ease of tuning, and its ability to alter the transient response for further refinement later on.
The touchscreen interacts with a commercially available SC800 controller, which then interfaces with a personal computer as a Human Interface Device (HID). That allows for processing of the sensor input similar to that of calculating the position of a mouse. Some calibration is required in order to adjust the coordinates the computer records into those usable by the learning algorithm. In addition, some modification of the SC800 controller to interact with the touchscreen and computer is required, which could be avoided with an SC4 controller. Other options for sensing the location of impact include image processing, an LED sensor array, knock sensors on a plate, and several other alternatives. The touchscreen is used due to its affordability, sensitivity, and the ease with which it can be integrated with the other components of the system.

In order to allow the system eventually to converge on the target, an iterative learning algorithm has to be developed; it is implemented in MATLAB and Simulink. By taking the touchpad data regarding the impact position and calculating new pan and tilt angles to reduce the error, it will guarantee convergence on the intended target given enough iterations. There were various different possibilities for learning algorithms that could be used, however it was decided that the Newton Raphson root finding method approach would be the best. The Newton-Raphson algorithm is robust in that it will still converge on the target even if the trajectory equation is not perfect, or if the firing mechanism is not completely repeatable. Though the Newton-Raphson algorithm is robust enough to handle some degree of non-repeatability, there will be problems if launches that go too far off of their planned trajectory are considered valid. Thus, impacts more than 6 cm from their expected location are disregarded.
4. DESIGN DETAILS

4.1 Model Development

The purpose of the Simulink model in Figure 4.1.1 is to obtain necessary data for estimating the Coulomb frictions and viscous frictions for the tilt axis. For the pan axis, the two inputs to the PCIM-DAS1602 16 block have to be swapped. PCI-QUAD04 has one output port that needs to be multiplied by $\frac{2\pi}{(2048*4)}$ to acquire theta, which is the angular position of the motor. Theta1 and thetadot1 correspond to the angular position and speed of the motor. Since the amount of torque that is required to start the system moving is greater than the torque required to keep the system in motion, an impulse was input 0.3 seconds after the simulation started with a pulse width of 0.01 seconds to break the static friction. Input voltage was incremented automatically with the ‘for loop’ from the scripts C1 and C2 found in Appendix C.

![Figure 4.1.1: Friction Identification Simulink Model](image)

Some filtering was necessary because the velocity collected from a single simulation fluctuates. Figure 4.1.2 shows the raw data before the filter is applied. Filtering is implemented in three steps. The first step is taking average from five runs for each voltage step. During second step, the average of the...
twenty-five adjacent data points is taken from the data set calculated in the first step. Finally, the impulse spikes are eliminated. The data plot in Figure 4.1.3 is the result after the second filtering step. For both pan and tilt axes, all velocities with input voltage greater than -0.5 volts and less than 0.5 volts saturated to zero. Velocities with an input voltage greater than 1 Volts or less than -1 volts all saturated to 9.3143 rad/s and -9.0014 rad/s for the tilt axis, and 9.4629 rad/s and -9.1584 rad/s for the pan axis. The final velocity plots for the tilt and pan axes can be found in Figures 4.1.4 and 4.1.5 respectively.
Figure 4.1.3: Velocity Diagram after Implementation of Second Filter

Figure 4.1.4: Velocity Diagram After Implementation of Third Filter for Tilt Axis
Torque was calculated using following equation:

\[
\text{Torque} = \text{Voltage} \times N \times Nm \times Kt \times Ka \tag{4.1.1}
\]

\[
\text{Voltage} = \text{Input Voltage (-1~1 Volts)}
\]

\[
N = \text{External Gear Ratio (2.47)}
\]

\[
Nm = \text{Internal Gear Ratio (19.5)}
\]

\[
Kt = \text{Motor Torque Constant (4.36e-2Nm)}
\]

\[
Ka = \text{Amplifier Gain Constant (.1A)}
\]

From Figure 4.1.6 and Figure 4.1.7, the Coulomb and viscous frictions are obtained. The Coulomb frictions are the two end points of the line where steady state velocity is zero, and the viscous frictions are the velocity of the line where it connects the maximum velocity to zero. From this fact, for the tilt axis, the negative Coulomb friction came out to be -0.1365Nm, and the positive Coulomb friction came out to be 0.115Nm. The negative viscous friction and positive viscous friction are 0.0082NmS/rad and 0.0150NmS/rad respectively. These values are calculated using scripts C4 and C5 attached in Appendix C. Also for pan axis, negative Coulomb friction and positive Coulomb friction came out to be -0.0945Nm, and negative Coulomb friction came out to be 0.0735Nm. The negative viscous friction and positive viscous friction are 0.0218NmS/rad and 0.0582NmS/rad.
### TABLE 4.1.1: TILT & PAN FRICTION CONSTANTS

<table>
<thead>
<tr>
<th>Axis</th>
<th>Coulomb Friction (Nm)</th>
<th>Viscous Friction (NmS/rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>negative</td>
<td>positive</td>
</tr>
<tr>
<td>Tilt Axis</td>
<td>-0.1365</td>
<td>0.115</td>
</tr>
<tr>
<td>Pan Axis</td>
<td>-0.0945</td>
<td>0.0735</td>
</tr>
</tbody>
</table>

**Figure 4.1.6:** Plot of Torque vs. Steady State Velocity of Tilt Axis

**Figure 4.1.7:** Plot of Torque vs. Steady State Velocity for Pan Axis
Equation (4.1.2) is provided in class lecture notes and the Simulink model in Figure 4.1.8 is derived from it. Through integration and simplification, matrices are developed in terms of $\dot{\theta}, \theta, V,$ sample time, and parameter $a_1, a_2, a_3,$ and $a_4.$

$$\dot{\theta} + a_1\dot{\theta} + a_2 \text{sgn}(\dot{\theta}) = a_3 V + a_4 \sin \theta$$  \hfill (4.1.2)

In order to verify the validity of this method, the parameters, $a_1, a_2, a_3,$ and $a_4$ were set to be random numbers, 1, 2, 3, and 4 respectively. Data was collected with the chirp input, and the data was
again used to calculate parameters using above matrices. $a_1$ was calculated to be 1.0053, $a_2$ came out to be 1.7730, $a_3$ came out to be 2.8580, and $a_4$ came out to be 4.0053. All four parameters are close to their original values, proving that the method is valid.

A chirp with frequencies varying from 1 to .1Hz and amplitude of 3 was used as the input voltage to collect angular displacements and angular velocities of the pan and tilt mechanism. The resulting output is shown in Figures 4.1.9 and 4.1.10 for the tilt and pan axes respectively. With this data, the parameters, $a_1$ through $a_4$ were calculated for both pan and tilt axes as shown in Table 4.1.2.

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>tilt</td>
<td>70.0680</td>
<td>376.8135</td>
<td>783.4932</td>
<td>325</td>
</tr>
<tr>
<td>pan</td>
<td>1.7758</td>
<td>4.5490</td>
<td>9.3423</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4.1.9: Experimental Data on Tilt Axis with Chirp Input
Now that the parameters are calculated, they have to be checked if they are reasonable on different types of input voltages. Hence, a sine wave was chosen as the input voltage, and the data collected from the pan and tilt mechanism. The resulting output for the tilt and pan axes can be seen in Figures 4.1.11 and 4.1.12.
Figure 4.1.11: Experimental Data on Tilt Axis with Sine Input

Figure 4.1.12: Experimental Data on Pan Axis with Sine Input
4.2 Control Development

Three gains, derivative, integral, and proportional, are tuned to meet the required transient response. The proportional gain is tuned for a fast rise time but increases the overshoot. The overshoot is controlled by derivative gain but also decreases the response time. Finally, the integral gain is tuned to reduce the steady state error. The objective of the control design was to make the steady state error as small as possible while maintaining a rapid rise time, with overshoot the least important characteristic. The gains shown in Table 4.2.1 were found to produce the best response for the model of the system. These gains are simulated using closed loop feedback as shown in Figure 4.2.1, and the step response for the tilt and pan axes for the model is that shown in Figures 4.2.2 and 4.2.3 respectively.

<table>
<thead>
<tr>
<th>TABLE 4.2.1: PID GAINS ON BOTH AXES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tilt Axis</td>
</tr>
<tr>
<td>$k_p$ = 50, $k_i = 0.005$, $k_d = 0.01$</td>
</tr>
<tr>
<td>Pan Axis</td>
</tr>
<tr>
<td>$k_p$ = 50, $k_i = 4$, $k_d = 10$</td>
</tr>
</tbody>
</table>

Figure 4.2.1: Closed Loop Feedback Model
The plots above show that there is some overshoot in the tilt axis, and the response was not too fast in the pan axis. This response closely matched that of the desired system, with virtually no steady-state error, relatively small rise times, and a minimal ripple effect.
4.3 Physical Design

To mount the launcher to the pan and tilt mechanism and allow for automated firing requires substantial modification to the system. First, a way to mount the launcher to a flat surface that could pan and tilt is necessary. The design in Figure 4.3.1 illustrates the solution to this problem.

![Aluminum Mounting Base](image)

Figure 4.3.1: Aluminum Mounting Base

A proper sized piece of aluminum stock can be cut and drilled to specifications. The hole in the bottom is where the shaft of the tilt axis will slide through and be set-screwed securely on. Once this is in place, a flat surface perpendicular to the tilt shaft is now available to mount the launcher.

Next, the launcher is reduced to only the components necessary for firing discs. An illustration of this mechanism can be seen in Figure 4.3.2. By simply connecting the disc spinning motor to a power source, the launcher can now fire discs as before with less weight, drag, and a now flat and easily mountable surface.
This reduction of the launcher results in removal of the trigger, but this actually aided in the mounting process. Now, by simply connecting a DC motor to the gear on the bottom of our launcher, the motor will turn a set distance, turning the gear that pushes the disc into the launching chamber. The sideways motion at the end of the movement causes the disc to press against the spinning wheel, propelling it form the launcher. The process is illustrated in Figure 4.3.3. By simply reversing the voltage on the DC motor for the same set time, the gun can reload as well.
In order to fit the motor between the base and launcher simple screws, nuts, and washers are all that is needed. The motor screws into position on the base with the launcher placed above it; the gear on the motor aligns with the gear on the launcher arm. The launcher is screwed into the base, using the screws to suspend the launcher in the air while holding it firmly in place; the long screw lengths act as spacers for the motor. Figure 4.3.4 illustrates the mounting of the launching system.

Figure 4.3.4: Mounting of System

Unfortunately, the automated firing of the disc launcher was not completed as planned. A series of events occurring within the last few days of the project prevented this idea from becoming a reality. First, there was a delay in receiving the cable to control the motor via MATLAB and Simulink. Next, it was found that the motor chosen for the task did not have enough torque to move the launching arm. A new servomotor was quickly located for replacement, but this motor was found to be too large for the
already designed and implemented mounting system. With less than a week before the final deadline, there was not enough time to rebuild the mounting system to accommodate the new motor and then re-perform all the necessary tests. It was decided to abandon the automatic firing and reloading idea and simply to launch the discs by manually turning the gear to move the launching arm back and forth. The original mounting did prove useful for this, allowing room between the base and the launcher to operate the gear and propel the launching arm. In addition, the mounting was sturdy enough to prevent any accidental movement in the position or orientation of the launcher when there was human interaction.

4.4 Sensor Development

The impact location sensor being used in the system is a Dynapro 95645 8-wire 13.8” resistive touchscreen (Appendix B). The screen consists of two sheets of circuit layers separated by a spacer, with a pliable glass surface on one side and stiff backing on the other. When the glass surface is touched, there is a voltage change due to the contact of the two circuit layers, which is then sent to a controller to be decoded into Cartesian coordinates.

The sensor used in the system is a 3M SC801U USB Resistive Controller (Figure 4.4.2). The controller interfaces directly with the touchscreen and acts as an interpreter between it and a personal computer. When attached to a personal computer, it behaves as a Human Interface Device, similar to a mouse [8]. After an impact is registered on the touchscreen, the controller automatically moves the mouse to the relative position on the computer’s desktop, and sends the command equivalent of a mouse button click.

Unfortunately, the SC801U is not designed to work with the touchscreen being used, so certain modifications are required to make it function. The main difficulty lies in the connectors between the two, specifically that the touchscreen has an XXYY style, non-latched connector pinout, while the controller expects an XYXY latched connector pinout. X and Y refer to the two sheets in the touchscreen itself. If the connectors are not mated properly, then the touchscreen acts erratically and the
The computer receives erroneous data. The pinouts of the controller and touchscreen can be seen in Table 4.4.1.

**TABLE 4.4.1: TOUCHSCREEN & CONTROLLER PINOUTS**

<table>
<thead>
<tr>
<th>Controller</th>
<th>YE-</th>
<th>YE+</th>
<th>XS-</th>
<th>XS+</th>
<th>YE-</th>
<th>YE+</th>
<th>XS-</th>
<th>XS+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Touchscreen</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

Once a crossover interface is created to correct for the discrepancies between the two connectors, the controller receives the correct data and is able to interpret it onto the screen. The unavailability of a USB cable with a Molex 51004-0500 connector to attach to the controller makes the creation of a custom cable a requirement.

After all of components are successfully integrated, the touchscreen successfully acts as an HID allowing the input from it to be determined using MATLAB. By superimposing a figure that occupies the entire screen, it is possible to find the position of the pointer after an impact is detected without having to worry about the input affecting the computer. After that, it is a simple matter of translating the pixel location on screen into its appropriate metric translation on the touchscreen, with (0, 0) representing the center. To accomplish that task, equations (4.4.1) and (4.4.2) are used, where \(a\) and \(b\) are the horizontal and vertical screen resolution respectively. The resulting location is the Cartesian coordinates relative to the touchscreen’s center in millimeters and is passed to the learning algorithm.

\[
x' = (y - b/2) * 206 / b \tag{4.4.1}
\]

\[
y' = (x - a/2) * 279.6 / a \tag{4.4.2}
\]

### 4.5 Algorithm Development

The Newton-Raphson root finding method is an iterative process that takes, as input, the current pan and tilt joint angles, as well as the desired \(x\) and \(y\) positions and the current \(x\) and \(y\) positions, and outputs a new set of desired pan and tilt joint angles. The goal is to drive \(x\) to \(x_{\text{desired}}\) and \(y\) to \(y_{\text{desired}}\). The method can be used if the pan and tilt angles are coupled, however the amount of coupling present...
due the edges of the touchscreen being farther away from the launcher than the center is minimal and can be ignored.

The Newton-Raphson equations for the pan angle are simple since the trajectory equation for pan is the same regardless of the angle of launch and therefore does not have to be scaled to the angle of firing. The following equations are the calculations for the pan angle and the x directions. Simulation shows that there is no need to change the alpha constant for the x direction.

\[
X = \text{distance} \times \tan(Q_{\text{pan}})
\]

\[
Q_{\text{pan-new}} = Q_{\text{pan}} - a \times (X - X_{\text{desired}})/(dX/dQ_{\text{pan}})
\]

\[
Q_{\text{pan-new}} = Q_{\text{pan}} - a \times (\text{dist} \times \tan(Q_{\text{pan}}) - X_{\text{desired}}) \times (\cos^2(Q_{\text{pan}})) / \text{dist}
\] (4.5.1)

For the tilt angle, the equations are a bit more complicated because the trajectory equations have to be scaled to the angle since the trajectory is differs based on the launcher’s orientation. The difference shows up in the distance from the y = 0 point on the touchpad that the disc hits. When the disc is fired downward, it drops more than when it is fired upward due to the aerodynamics. It does not take as large of a downward angle to yield a given amount of difference in the y direction as it would take pointing upward to yield the same distance. The equations to get the distance are part of the advanced trajectory model found in Chapter 5.3. For the learning algorithm, they are incorporated into the trajectory equations. Table 4.5.1 shows the trajectory equations obtained at various tilt angles. For the tilt angle, the simulations work well with the alpha constant equal to one so there is no need to change it.

**TABLE 4.5.1: VERTICAL DISPLACEMENT EQUATIONS**

<table>
<thead>
<tr>
<th>Tilt Angle (degrees)</th>
<th>Trajectory Equation Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>(Y=(\text{dist}^2 + (21.17\times\text{dist} + 19.2)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
<tr>
<td>10</td>
<td>(Y=(\text{dist}^2 + (13.91\times\text{dist} + 13.02)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
<tr>
<td>5</td>
<td>(Y=(\text{dist}^2 + (2.8\times\text{dist} + 2.4)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
<tr>
<td>0</td>
<td>(Y=(\text{dist}^2 + (-.53\times\text{dist} + .9)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
<tr>
<td>-5</td>
<td>(Y=(\text{dist}^2 + (-14.24\times\text{dist} - 8.94)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
<tr>
<td>-10</td>
<td>(Y=(\text{dist}^2 + (-21.88\times\text{dist} - 13.5)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
<tr>
<td>-15</td>
<td>(Y=(\text{dist}^2 + (-37.37\times\text{dist} - 25.02)^2)^{1/2} \times (\tan(Q_{\text{tilt}})))</td>
</tr>
</tbody>
</table>
Since the Newton-Raphson root finding method is robust enough to converge even if the trajectory equation is not perfect, each equation in the table above is used to satisfy more than one angle. For example, the 0 degree tilt angle equation is used for any angle between -2.5 degrees and 2.5 degrees; the 5 degree tilt angle equation is used for any angle from 2.5 degrees to 7.5 degrees, and so forth. In simulation using MATLAB, this method works well and the proper y position is still found in three tries so there is no need to obtain more advanced and accurate equations.

The following equations are the Newton-Raphson equations using the trajectory equation for when the tilt angle is zero.

\[
\begin{align*}
Q_{\text{tilt-new}} &= Q_{\text{tilt}} - a \frac{(Y - Y_{\text{desired}}) \cdot (dY/dQ_{\text{tilt}})}{dY/dQ_{\text{tilt}}} \\
Q_{\text{tilt-new}} &= Q_{\text{tilt}} - a \frac{(\text{dist} \cdot \tan(Q_{\text{tilt}}) - Y_{\text{desired}}) \cdot (\cos^2(Q_{\text{tilt}}))}{(\text{dist}^2 + (-.53 \cdot \text{dist} + .9)^2)^{1/2}} \quad (4.5.2)
\end{align*}
\]

With the trajectory equations, it can be predicted where the disk theoretically should hit the touchpad assuming it follows the equations exactly. In order to filter out unusual launches so that they are not taken into account by the learning algorithm, the projected point of impact is calculated when the disk is launched from a given pan and tilt joint angle configuration. The actual point of impact is compared to the projected point of impact and if there is a difference of more than six centimeters, the launch is considered inaccurate and is thrown out and another disk is launched. Six centimeters is thought to be a reasonable value because of the fact that, in the initial testing, most of the disks hit the touchpad within 4 cm of each other when fired at the same spot. With the launcher being, in general, accurate to 4 cm, a 6 cm deviation from the projected point of impact is enough to call that specific launch an unusual case and throw it out.
5. DESIGN VERIFICATION

5.1 Model Verification

Figures 5.1.1 and 5.1.2 show the experimental and simulated angular velocities generated from a chirp signal input. The experimental angular velocities are filtered to aid in comparison. For both axes, the system shows both frequencies and amplitudes closely mirroring that of the simulation. Even though the comparison shows that the parameters are reasonable, another test had to be done. The parameters are calculated from the experimental data, so it is in a way obvious that the simulated data should look similar. Therefore, comparing these responses is not enough to determine that the parameters are accurate. To test the accuracy of the model further, a sine wave is used as an input instead of a chirp signal, and the resulting responses for the tilt and pan axes can be seen in Figures 5.1.3 and 5.1.4 respectively. Though the results are differ in the first oscillations, which can be attributed to differences in the starting orientation, the steady state signal are virtually identical.

Figure 5.1.1: Simulation vs. Experimental Response to Chirp Input on Tilt Axis
Figure 5.1.2: Simulation vs. Experimental Response to Chirp Input on Pan Axis

Figure 5.1.3: Simulation vs. Experimental Response to Sine Input on Tilt Axis
5.2 Control Verification

In order to verify the transient response meets the specifications, the experimental data is compared to the angular displacement input to the system. The resulting responses for the tilt and pan axes are shown in Figure 5.2.1 and 5.2.2, with angular displacements in radians. In the demonstration and the video clip, it might seem as if there is more overshoot than it is shown in the graph. It is due to the mounting of the disc launcher, which uses only two supports on the back instead of the four originally designed. Table 5.2.1 shows the notable aspects of the responses charted. It is to be noticed that these do not meet the specifications for the pan axis’s rise time or settling time, and the tilt axis’s settling time or overshoot. Due to the inertia of the system, the tilt axis requires too much proportional control to expect as low of an overshoot. The pan axis had difficulty meeting the settling time for a displacement of that magnitude. The specifications were likely overzealous though, as in experimental work the system responds adequately for the project’s needs. The most important requirement, the steady-state error, was exceeded by both axes.

Figure 5.1.4: Simulation vs. Experimental Response to Sine Input on Pan Axis
**TABLE 5.2.1: RESPONSE CHARACTERISTICS OF CLOSED LOOP SYSTEM**

<table>
<thead>
<tr>
<th></th>
<th>Rise Time</th>
<th>Steady State Error</th>
<th>Settling Time</th>
<th>Percent Overshoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tilt Axis</td>
<td>0.0230 s</td>
<td>0.8435%</td>
<td>1.21 s</td>
<td>75.1%</td>
</tr>
<tr>
<td>Pan Axis</td>
<td>0.450 s</td>
<td>0.215%</td>
<td>1.91 s</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
In order to check the robustness of the system, the Bode plots of both axes are taken to examine the gain and phase margins. Figure 5.2.3 shows the tilt axis Bode diagram, while Figure 5.2.4 shows the pan axis Bode diagram. Table 5.2.2 shows that both axes have an infinite gain margin, as is expected in the second order system. As such, it does not provide much insight into the stability of the system. The phase margin of the tilt axis is very large at 170 degrees, so it will not easily be made unstable by a phase change disturbance. The pan axis is not as stable, with only a 25.7 degree phase margin; however, that could be remedied in further design refinement by implementing an extra phase-lag compensator on the pan axis. Doing such should effectively shift the bump in the magnitude high enough so that it only crosses the 0dB axis once, resulting in a greater phase margin. Regardless, for the design of a prototype system, a 25.7 degree phase margin is deemed sufficient.

Figure 5.2.3: Tilt Axis Closed Loop Bode Diagram
5.3 Physical Verification

To verify that the toy disc shooter obtained is consistent and accurate enough to be viable for use in the system a several tests are run. The first test is a simple test to determine consistency. Holding the gun on a level surface and aiming it at a pint glass on the same level surface, the gun was fired 5 times at a set distance. A diagram of this testing procedure can be seen in Figure 5.3.1.
This sequence is repeated for three trials for each distance. The number of times the disc hit the glass is recorded and the percent of that number out of 5 is shown in Table 5.3.1. The Total Accuracy of the disc shooter at that set distance is averaged from the three trials as well.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Try 1</th>
<th>Try 2</th>
<th>Try 3</th>
<th>Total Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3ft</td>
<td>0.9m</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>5ft</td>
<td>1.5m</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>7ft</td>
<td>2.1m</td>
<td>80%</td>
<td>80%</td>
<td>80%</td>
</tr>
<tr>
<td>9ft</td>
<td>2.75m</td>
<td>80%</td>
<td>60%</td>
<td>80%</td>
</tr>
<tr>
<td>11ft</td>
<td>3.35m</td>
<td>80%</td>
<td>100%</td>
<td>80%</td>
</tr>
<tr>
<td>13ft</td>
<td>4m</td>
<td>60%</td>
<td>60%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Finding the results of this first experiment satisfactory, it is decided that the disc shooter can be used. To obtain a more accurate understanding of how the disc is fired and how it travels a second test was designed to determine the trajectory of the disc after being launched from the toy gun. This experiment is set up similarly to the previous one. The gun is held on a level surface and now aimed at a piece of paper with a graph drawn on it. This graph has marks for each centimeter vertically and horizontally away from the origin and is taped to the wall so that the origin is on the same level surface as that of the barrel of the toy gun. This experiment is illustrated in figure 5.3.2.

Figure 5.3.2: 2\textsuperscript{nd} Launcher Testing Procedure
The gun is fired 10 times at a set distance and the x and y coordinates where the disc hit are recorded for each shot. The gun is then moved to another set distance and the experiment repeated. The results of this experiment are listed in Table 5.3.2.

**TABLE 5.3.2: PRELIMINARY TRAJECTORY RESULTS**

<table>
<thead>
<tr>
<th>Shot #</th>
<th>0.5 m</th>
<th>1.0 m</th>
<th>1.5 m</th>
<th>2.0 m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x position (cm)</td>
<td>y position (cm)</td>
<td>x position (cm)</td>
<td>y position (cm)</td>
</tr>
<tr>
<td>1</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>-6</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>-5</td>
</tr>
<tr>
<td>4</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-5</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-2</td>
<td>1</td>
<td>-5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>-1</td>
<td>-2</td>
<td>-5</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>0</td>
<td>-3</td>
<td>-5</td>
</tr>
<tr>
<td>8</td>
<td>-2</td>
<td>0</td>
<td>2</td>
<td>-4</td>
</tr>
<tr>
<td>9</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-6</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-5</td>
</tr>
</tbody>
</table>

- X means the disc hit lower than -14cm and could not be recorded

The conclusion drawn from this experiment is that the trajectory should be determined from data collected between 0.5 and 1.5 meters. Anything before 0.5 meters will be fired at a negative angle and anything after 1.5 meters will be fired at a positive angle relative to the horizon. This led to the next experiment that is performed in the exact same manner as above. The result of the trajectory experiment with data from 0.5 to 1.5 meters is shown in Table 5.3.3. The experiment is graphed in Figure 5.3.3.

**TABLE 5.3.3: SHORT RANGE TRAJECTORY RESULTS**

<table>
<thead>
<tr>
<th>Shot #</th>
<th>0.5 m</th>
<th>0.7 m</th>
<th>0.9 m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x position (cm)</td>
<td>y position (cm)</td>
<td>x position (cm)</td>
</tr>
<tr>
<td>1</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-2</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>8</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>9</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
A summary of this experiment can be seen in the Table 5.3.4, which shows the average x and y displacement at each distance in centimeters.

**TABLE 5.3.4: AVERAGE SHORT RANGE TRAJECTORY**

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>x-position (cm)</th>
<th>y-position (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 m</td>
<td>-1.1</td>
<td>-0.8</td>
</tr>
<tr>
<td>0.7 m</td>
<td>0.6</td>
<td>-1.5</td>
</tr>
<tr>
<td>0.9 m</td>
<td>0.1</td>
<td>-3.5</td>
</tr>
<tr>
<td>1.0 m</td>
<td>-0.6</td>
<td>-4.9</td>
</tr>
<tr>
<td>1.1 m</td>
<td>-0.6</td>
<td>-6.4</td>
</tr>
<tr>
<td>1.3 m</td>
<td>-1.1</td>
<td>-7.2</td>
</tr>
<tr>
<td>1.5 m</td>
<td>0.3</td>
<td>-10.1</td>
</tr>
</tbody>
</table>

Figure 5.3.3: Plot of Short Range Trajectory Results
From this data, it can be concluded that the horizontal displacement of the disc stays relatively close to zero with a maximum of 1 cm average deviation from the origin. It can also be seen that as the distance from the target increases, the vertical position that the disc strikes the target also decreases. A plot of distance from target (in meters) vs. vertical position relative to the horizontal (in centimeters) is shown in Figure 5.3.4.

Fitting a best-fit line to this data, the equation is found to be $Y = -9.5X + 4.59$, where $X$ is the distance from the gun barrel to the target and $Y$ would be the vertical displacement where the disc would hit the target. This equation is a good estimation of the trajectory of the disc knowing the distance to the target between 0.5 and 1.5 meters. This equation can be used to estimate the vertical position at which the disc will strike the target if the distance from the launcher barrel to the target is known. This equation can also be used to estimate the distance the launcher is from the target given the vertical displacement of the striking area of the disc. A final use for this equation comes into the learning algorithm where knowing both the distance from the launcher to the target and the vertical displacement
of the disc at the target, the algorithm can check the accuracy of the shot by comparing it with this model.

In an effort to create the most accurate trajectory model for the disc launched from the toy gun a third and final test is established. This experiment is again set up similarly to the previous two trajectory tests, with the launcher being held securely to the mount attached to the pan and tilt mechanism for more stability. The gun was once again aimed at a graph with one centimeter scaling for the horizontal and vertical deviations from the origin that was lined up with the mouth of the launcher. Again, see Figure 5.3.2.

This test continued in similar fashion as before with ten shots being fired at each set distance from the launcher to the graph, and at each of seven angles at that set distance. The set distances were 0.5, 0.75, 1.0, 1.25, and 1.5 meters and the set angles were −15°, −10°, −5°, 0°, 5°, 10°, and 15° from the horizon. These are chosen based on previously defined requirements and experimentation.

Since it was determined from the previous experiment (and continually shown throughout this one) that the x direction had little deviation, only the y coordinate where the disc struck the graph was recorded. Once again, it was attempted to get best fit lines from this data to determine the trajectory of the disc given the distance from the launcher to the target. The average vertical displacement of the 10 shots at each set angle at each set distance is shown in Table 5.3.5. The X’s in the table mean that the disc hit the ground before the target.

<table>
<thead>
<tr>
<th>Angle</th>
<th>Distance</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>1.25</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td>18.8</td>
<td>20.2</td>
<td>25.2</td>
<td>25.5</td>
<td>26</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>10.3</td>
<td>15.7</td>
<td>17.5</td>
<td>15.8</td>
<td>14.7</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>2.6</td>
<td>2.5</td>
<td>3.3</td>
<td>3.5</td>
<td>4.1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>-0.3</td>
<td>-1</td>
<td>-2.4</td>
<td>-6.1</td>
</tr>
<tr>
<td>-5</td>
<td></td>
<td>-9.1</td>
<td>-13.4</td>
<td>-20.5</td>
<td>-24.8</td>
<td>-29.9</td>
</tr>
<tr>
<td>-10</td>
<td></td>
<td>-15</td>
<td>-19.3</td>
<td>-30.8</td>
<td>-39.3</td>
<td>-46.9</td>
</tr>
<tr>
<td>-15</td>
<td></td>
<td>-25.1</td>
<td>-37.2</td>
<td>-49.8</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

From this data, a line was graphed for each angle so that a best fit line could be determined in the form of $Y = aX + b$ where X is a known distance from the launcher to the target and Y is the vertical
displacement, or the position the disc should hit the target based on the angle that the launcher is currently firing at. A graph of these lines can be seen in Figure 5.3.5.

![Graph of Y Position vs. X Distance from Target at Each Set Angle](image)

Figure 5.3.5: Graph of Y Position vs. X Distance from Target at Each Set Angle

This leads to the advanced trajectory model used in the learning algorithm. The equations for the best fit lines at each angle are displayed in Table 5.3.6.

<table>
<thead>
<tr>
<th>Angle</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>15°</td>
<td>19.2</td>
<td>21.17</td>
</tr>
<tr>
<td>10°</td>
<td>13.02</td>
<td>13.91</td>
</tr>
<tr>
<td>5°</td>
<td>2.4</td>
<td>2.8</td>
</tr>
<tr>
<td>0°</td>
<td>0.9</td>
<td>-0.53</td>
</tr>
<tr>
<td>-5°</td>
<td>-8.94</td>
<td>-14.24</td>
</tr>
<tr>
<td>-10°</td>
<td>-13.5</td>
<td>-21.88</td>
</tr>
<tr>
<td>-15°</td>
<td>-25.0167</td>
<td>-37.3667</td>
</tr>
</tbody>
</table>

These equations are fed into the learning algorithm, which uses them to compute the new position and orientation of the launcher based on the distance from the launcher to the target, and the current angle of the firing mechanism.

The size of the target will dictate the required accuracy for the system. The initial expectation is that the firing system be able to hit consistently within a 7.5 cm diameter circle at a maximum range of 1.5m, when at a zero degree tilt. Through simple trigonometry, it can be deduced that ensuring an
impact within 3.25 cm of center at 1.5 m requires that the rotational position be within ±1.24 degrees. Taking into account the standard deviation of the horizontal displacements shown in Table 5.3.3 and Figure 5.3.4, the largest standard deviation was 2.60 cm, with the deviation at 1.5m being 2.21 cm. By subtracting the standard deviation from the allowable error, a circle of approximately 1 cm radius is left, resulting in a maximum position error of ±0.382 degrees. The specified error of ±0.1 degrees falls well within the tolerable error, and should allow the desired target to be hit at an even greater range with consistency. The mechanism itself should have no difficulty achieving this low error, as the encoder allows for accuracy within 0.0439 degrees since it has a 2048-bit quadrature reading on the output shaft.

Figure 5.3.6 shows the offset resulting from the possible position errors discussed above. Since the current trajectory of vertical motion is a best-fit line, it can be assumed that the same values will hold true for the tilt axis. These are likely to differ once a more precise trajectory can be calculated experimentally; however, it appears that a ±0.1 degree error will supply the necessary accuracy and feasibility for the project.

![Figure 5.3.6: Target Cones Resulting from Different Position Errors](image)

5.4 Sensor Verification
Since the most important part of the system is accuracy, it is critical that the sensor used to detect impacts be as precise as possible. As shown in Figure 5.4.1, the touchscreen itself has a dot pitch of 160 mil, which is more than accurate enough to give the location of the size disc being used. The limiting factor then becomes the controller itself, as it has only the ability to transmit 10-bit coordinate data [8]. That allows for a vertical resolution of 0.27 mm and horizontal resolution of 0.20 mm. Even with the lower resolution, it is more than capable of accurately sensing the location of impact.

![Figure 5.4.1: Dot Pitch of Touchscreen Sensor](image)

Once the accuracy of the touchscreen itself is established, the consistency between the touchscreen input and output on the computer must be verified. In testing each corner of the screen, it can be seen that the pointer reaches the bottom corners of the screen perfectly, while falling a few pixels short of the top corners. That discrepancy is likely due to the high resolution of the monitor used relative to the controllers, combined with the small dead zones at the corners of the touchscreen. Fortunately, the dead zone was found to be only six pixels wide, or approximately 0.57% of the screen. Considering the likelihood of the disc striking precisely the corner of the touchscreen without activating one of the nearby sensors, it can be safely assumed that the dead zone will not significantly alter the results of the system. Random test points throughout the monitor confirmed that the calibration of the touchscreen is accurate enough to ensure proper reading of the results.
The problem with the touchscreen comes when discs are fired at it. Although able to respond to the slightly prick of a fingertip, the disc is unable to activate the sensor. Several factors thought to play a part in the sensor’s failure can be tested. The mass of the disc is minute, and even though it is fired at high velocities, it is possible that the momentum is insufficient to bend the screen enough to connect the two circuit sheets. The problem is not as simplistic as using a heavier mass, as attempts to activate the sensor using a mouse ball thrown at high speeds fail as well. The possibility that organic contact is required to activate the sensor can be dismissed as pressing any object against the screen succeeding in activating it. The conclusion that contact with the screen must be prolonged for a longer period of time than an elastic collision permits in order to register an impact is the only remaining conclusion. Having no means to accomplish this task or time to pursue other sensory options when this difficulty was encountered, the location of the impact is now manually entered on the touchscreen.

In experimentation, the touchscreen did provide accurate feedback to the learning algorithm when the impact location was manually input. The launcher converged on the target in three shots, and thus from that aspect the sensor can be considered a success. It is likely in further refinement that it will be replaced by one more capable of detecting rapid and faint impacts however.

5.5 Algorithm Verification

In order to verify that the Newton-Raphson root finding method is a suitable learning algorithm, it has to be simulated on MATLAB. A script is written for this purpose. The following figure displays the simulation for the pan angle. The desired x position is at $x = 20$ and the starting x position is at zero, with an angle of zero degrees. It is assumed that the first launch hits the touchpad at zero. It can be seen from the figure that it converges at $x = 20$ within three tries.
Newton-Raphson Simulation for Pan Angle: \( X_{\text{desired}} = 20 \) cm

Figure 5.5.1: Simulated Newton-Raphson Pan Angle Algorithm Results

In reality, it also only took three tries to hit the target on most runs. However, they do not converge quite as neatly in the actual running of the system. Although it hits the target in three tries, it does not hit the center of the target or get as close to the target on the second try as when simulated. The cause for the discrepancy is the launcher’s lack of perfect repeatability. Fortunately, the learning algorithm is robust enough that it can still converge on the target even if the trajectory is a little off.

For the tilt trajectory, the results are similar. The ideal simulation converges on the target within three tries. The numbers in the idea simulation are slightly different in that the second try is farther away from \( y = 20 \) than it was for the \( x \) simulation but that is because the trajectory equations are different for the tilt angle. The fact that the tilt angle also converges in three tries proves that the algorithm will still converge on the desired location even if the trajectory is not perfect. The tilt axis trajectories were done such that each trajectory was used for a small range of angles so it is not as precise as the pan axis. The algorithm still converges, so the fact that the trajectory in the \( y \) direction is imperfect does not significantly affect the results. Figure 5.5.2 shows the simulation results when the learning algorithm is tested in MATLAB for the tilt angle.
In reality, when the algorithm is tested with the actual pan and tilt device, the tilt angle is not quite as accurate as the pan angle. This is probably because the launcher is more unpredictable in the y direction than it is in the x direction as was seen in the initial launcher test results. It was found that in the y direction sometimes the launcher did not move as much as it should have.

Another factor in the performance of the algorithm is the filtration of outlying impacts from being used in calculation. The performance of the learning algorithm in the tilt direction can probably be improved if the tolerance is lowered to four or five centimeters because then more shots would be thrown out as unusual cases. A deviation from the projected point of more then four centimeters is more common in the y direction then in the x direction.
6. COSTS

The cost of the assembled pan and tilt system used as the basis of this project has been itemized and calculated as follows in Table 6.1.

<table>
<thead>
<tr>
<th>Item</th>
<th>Part Number</th>
<th>Price ($)</th>
<th>Quantity</th>
<th>Total ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor</td>
<td>GM8724S017</td>
<td>192.86 [1]</td>
<td>2</td>
<td>385.72</td>
</tr>
<tr>
<td>Timing Belt</td>
<td>A6Z16-C01806</td>
<td>1.86 [1]</td>
<td>2</td>
<td>2.70</td>
</tr>
<tr>
<td>Encoder</td>
<td>S1 &amp; S2 Optical Shaft Encoders</td>
<td>49.00 [3]</td>
<td>2</td>
<td>98.00</td>
</tr>
<tr>
<td>Gear</td>
<td>A6A6-75NF01812</td>
<td>15.15 [1]</td>
<td>2</td>
<td>30.30</td>
</tr>
<tr>
<td></td>
<td>A6A6-25DF01806</td>
<td>7.76 [1]</td>
<td>2</td>
<td>15.52</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>532.24</strong></td>
</tr>
</tbody>
</table>

The costs of additional parts and programs needed for the complete system beyond those of the assembled pan and tilt mechanism are listed in Table 6.2.

<table>
<thead>
<tr>
<th>Item</th>
<th>Price ($)</th>
<th>Quantity</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy Disc Shooter and Discs</td>
<td>4.97</td>
<td>2</td>
<td>9.94</td>
</tr>
<tr>
<td>Touchpad</td>
<td>24.95 [4]</td>
<td>1</td>
<td>24.95</td>
</tr>
<tr>
<td>Touchpad Controller</td>
<td>50.00</td>
<td></td>
<td>50.00</td>
</tr>
<tr>
<td>Wires (spool)</td>
<td>4.00</td>
<td>1</td>
<td>4.00</td>
</tr>
<tr>
<td>Programs:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SolidWorks</td>
<td>64.98 [7]</td>
<td>1</td>
<td>64.98</td>
</tr>
<tr>
<td>MATLAB 7</td>
<td>1900.00[2]</td>
<td>4</td>
<td>7,600.00</td>
</tr>
<tr>
<td>Simulink 6</td>
<td>2800.00[7]</td>
<td>4</td>
<td>11,200.00</td>
</tr>
<tr>
<td>Motor [6]</td>
<td>3.00</td>
<td>1</td>
<td>3.00</td>
</tr>
<tr>
<td>Gears [6]</td>
<td>2.25</td>
<td>1</td>
<td>2.25</td>
</tr>
<tr>
<td>Clamps (for Mounting)</td>
<td>8.95</td>
<td>1</td>
<td>8.95</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td><strong>18,968.07</strong></td>
</tr>
</tbody>
</table>
The labor cost of designing, building, testing and operating the system among the four members of the group are listed in Table 6.3.

<table>
<thead>
<tr>
<th>Laborer</th>
<th>Salary ($/hr)</th>
<th>Hours/Week</th>
<th>Weeks</th>
<th>Total ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ryan Kindle</td>
<td>25</td>
<td>10</td>
<td>10</td>
<td>2,500</td>
</tr>
<tr>
<td>Diana Mirabello</td>
<td>25</td>
<td>10</td>
<td>10</td>
<td>2,500</td>
</tr>
<tr>
<td>Yena Park</td>
<td>25</td>
<td>10</td>
<td>10</td>
<td>2,500</td>
</tr>
<tr>
<td>Josh Schuster</td>
<td>25</td>
<td>10</td>
<td>10</td>
<td>2,500</td>
</tr>
<tr>
<td><strong>Total for 4 Employees ($)</strong></td>
<td></td>
<td></td>
<td></td>
<td>10,000</td>
</tr>
</tbody>
</table>

The use of different labs to build and test the system has been assumed to cost as follows in Table 6.4 and the total system costs are listed in Table 6.5.

<table>
<thead>
<tr>
<th>Lab</th>
<th>Rate ($/day)</th>
<th>Use (days)</th>
<th>Total ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Systems Lab</td>
<td>50</td>
<td>30</td>
<td>3,000</td>
</tr>
<tr>
<td>Engineering Lab</td>
<td>50</td>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td>3,200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pan and Tilt Mechanism</th>
<th>Additional Parts Needed for System</th>
<th>Labor Costs</th>
<th>Lab Use</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$532.24</td>
<td>$18,968.07</td>
<td>$10,000</td>
<td>$3,200</td>
<td>$32,700.31</td>
</tr>
</tbody>
</table>

Seeing as many of the components, programs, labs and equipment are provided free of charge, along with the fact that there is no labor cost, the actual cost to the group can be calculated and is displayed in Table 6.6.

<table>
<thead>
<tr>
<th>Pan and Tilt Mechanism</th>
<th>Additional Parts Needed for System</th>
<th>Labor Costs</th>
<th>Lab Use</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0</td>
<td>$97.04</td>
<td>$0</td>
<td>$0</td>
<td>$99.09</td>
</tr>
</tbody>
</table>
7. CONCLUSIONS

Despite some setbacks, the system was an overall success and many of the goals put forth in the project proposal were achieved. In the end, the working system was able to fire discs at a target on a touchscreen and then correct for error using an iterative learning algorithm, usually converging on the target in three tries. The only major problems that prevented some of the original goals from being met were problems with equipment that prevented the system from having automated firing and feedback.

The controllers designed for both the pan and the tilt angles worked very well. They moved quickly to the proper pan and tilt joint angles with negligible steady-state error. There was a slight amount of overshoot, however that was not a concern for this project since the final position of the launcher was more important that its trajectory.

The learning algorithm worked very well in the pan direction and performed similarly to the initial MATLAB simulation. In the pan direction, it could converge on the target within three tries almost every time. In the tilt direction, the learning algorithm worked well after some tuning of the alpha constant since at first it converged significantly slower then it had in the initial simulation. In the end, the system was able to converge on the target in three tries almost every time, with both the pan and the tilt directions working correctly.

Unfortunately, automated firing of the launcher was not completed. This was due to the difficulty in getting the servomotor to work, and the DC motor’s insufficient torque. Given more time to either find a motor with more torque, or adjust the mounting to allow for the integration of the servomotor, automated firing could be accomplished. The servomotor provided did seem to have a slow reset time however, upwards of twenty seconds, so it would not be ideal for this application and further investigation into other methods might prove useful.

The system was supposed to get automatic feedback from the touchscreen when the disc hit it, and its failure was another major setback. Though the touchscreen was very accurate in detecting position, it could not detect the discs being fired at it because they did not make contact with the
touchscreen for a long enough amount of time to register. Instead, the impact location had to be manually entered on the touchscreen to provide the feedback required. Once that was done however, it was able to send the data to the computer with no problems whatsoever.

Though a completely automated system would have been ideal, the results of the prototype are quite promising overall since the control portions of the system all worked well. Given time for further refinement and development, automated firing could have been accomplished and alternate sensor options explored. Extensions to the system developed include the installation of a directional sensor so that the launcher can detect the approximate location of the target without manual aiming, and the application of the system to other kinds of projectile launchers.
8. REFERENCES


Appendix A: Contribution of Team Members

Ryan Kindle
- Introduction / Professional & Societal Considerations
- Sensor Development
- Sensor Verification
- Control Verification
- Editing & Compilation

Diana Mirabello
- Friction Identification
- Algorithm Development
- Algorithm Verification
- Conclusion

Yena Park
- Friction Identification
- Model Development
- Model Verification
- Control Development
- Control Verification

Josh Schuster
- Physical Development
- Physical Verification
- Control Verification
- Costs
Appendix B: Datasheets

13.8" Resistive Touch Screen
95645

Spacer adhesive
Optical adhesive
0.083 Chemically strengthened glass
PET stable circuit
PET ANR flex circuit
PET buffer
HC612 hard coat

Section A-A

Detail B (dot information)

Notes
- Workmanship standard per DTF document PS014, rev current
- Recognized to U.S. and Canadian requirements under the Component Recognition Program at Underwriters Laboratories Inc.
- Refer to:
  - Dynapro Touch Screen Integration Guide
  - Data sheet 1005 for hardware specifications
  - Data sheet 1001 for full PL type specifications
  - Stock touch screen drawing no. 95645-34 rev current

Call Toll-Free 1-888-222-9214
www.dynapro.com

Table 1

<table>
<thead>
<tr>
<th>PIN #</th>
<th>Assignment</th>
<th>Tolerances</th>
<th>All dimensions in inches (mm for reference only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Right excite</td>
<td>±0.05</td>
<td>Not to scale</td>
</tr>
<tr>
<td>2</td>
<td>Right sense</td>
<td>±0.05</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Left sense</td>
<td>±0.05</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Left excite</td>
<td>±0.05</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Top excite</td>
<td>±0.05</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Top sense</td>
<td>±0.05</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Bottom sense</td>
<td>±0.06</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Bottom excite</td>
<td>±0.06</td>
<td></td>
</tr>
</tbody>
</table>

Dynapro simplifies interaction between people and technology by designing and manufacturing world-class touch products, from touch screen components to touch computers, terminals and monitors.
# GM8724S017

Lo-Cog® DC Servo Gearmotor

<table>
<thead>
<tr>
<th>Assembly Data</th>
<th>Symbol</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Voltage</td>
<td>E</td>
<td>V</td>
<td>24</td>
</tr>
<tr>
<td>No-Load Speed</td>
<td>S&lt;sub&gt;N&lt;/sub&gt;</td>
<td>rpm (rad/s)</td>
<td>230 (24.1)</td>
</tr>
<tr>
<td>Continuous Torque (Max.)&lt;sup&gt;1&lt;/sup&gt;</td>
<td>T&lt;sub&gt;S&lt;/sub&gt;</td>
<td>oz-in (N-m)</td>
<td>42 (2.9E-01)</td>
</tr>
<tr>
<td>Peak Torque (Stall)&lt;sup&gt;2&lt;/sup&gt;</td>
<td>T&lt;sub&gt;P&lt;/sub&gt;</td>
<td>oz-in (N-m)</td>
<td>117 (8.3E-01)</td>
</tr>
<tr>
<td>Weight</td>
<td>W&lt;sub&gt;M&lt;/sub&gt;</td>
<td>oz (g)</td>
<td>11.3 (32.0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motor Data</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Torque Constant</td>
<td>K&lt;sub&gt;T&lt;/sub&gt;</td>
<td>oz-in/A (N-m/A)</td>
<td>6.18 (4.36E-02)</td>
</tr>
<tr>
<td>Back-EMF Constant</td>
<td>K&lt;sub&gt;E&lt;/sub&gt;</td>
<td>V/krpm (V/rev/d)</td>
<td>4.57 (4.36E-02)</td>
</tr>
<tr>
<td>Resistance</td>
<td>R&lt;sub&gt;T&lt;/sub&gt;</td>
<td>Ω</td>
<td>17.0</td>
</tr>
<tr>
<td>Inductance</td>
<td>L&lt;sub&gt;T&lt;/sub&gt;</td>
<td>mH</td>
<td>9.35</td>
</tr>
<tr>
<td>No-Load Current</td>
<td>I&lt;sub&gt;N&lt;/sub&gt;</td>
<td>A</td>
<td>0.09</td>
</tr>
<tr>
<td>Peak Current (Stall)&lt;sup&gt;2&lt;/sup&gt;</td>
<td>I&lt;sub&gt;P&lt;/sub&gt;</td>
<td>A</td>
<td>1.41</td>
</tr>
<tr>
<td>Motor Constant</td>
<td>K&lt;sub&gt;M&lt;/sub&gt;</td>
<td>oz-in/rev (N-m/rev)</td>
<td>1.49 (1.05E-02)</td>
</tr>
<tr>
<td>Friction Torque</td>
<td>T&lt;sub&gt;F&lt;/sub&gt;</td>
<td>oz-in (N-m)</td>
<td>0.35 (2.5E-03)</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>J&lt;sub&gt;R&lt;/sub&gt;</td>
<td>oz-in-s&lt;sup&gt;2&lt;/sup&gt; (kg-m&lt;sup&gt;2&lt;/sup&gt;)</td>
<td>2.3E-04 (1.6E-06)</td>
</tr>
<tr>
<td>Electrical Time Constant</td>
<td>τ&lt;sub&gt;E&lt;/sub&gt;</td>
<td>ms</td>
<td>0.54</td>
</tr>
<tr>
<td>Mechanical Time Constant</td>
<td>τ&lt;sub&gt;M&lt;/sub&gt;</td>
<td>ms</td>
<td>14.7</td>
</tr>
<tr>
<td>Viscous Damping</td>
<td>D</td>
<td>oz-in/krpm (N-m-s)</td>
<td>0.020 (1.4E-06)</td>
</tr>
<tr>
<td>Damping Constant</td>
<td>K&lt;sub&gt;D&lt;/sub&gt;</td>
<td>oz-in/krpm (N-m-s)</td>
<td>1.6 (1.1E-04)</td>
</tr>
<tr>
<td>Maximum Winding Temperature</td>
<td>θ&lt;sub&gt;MAX&lt;/sub&gt;</td>
<td>°F (°C)</td>
<td>311 (155)</td>
</tr>
<tr>
<td>Thermal Impedance</td>
<td>R&lt;sub&gt;TH&lt;/sub&gt;</td>
<td>°F/Watt (°C/Watt)</td>
<td>70.5 (21.4)</td>
</tr>
<tr>
<td>Thermal Time Constant</td>
<td>τ&lt;sub&gt;TH&lt;/sub&gt;</td>
<td>min</td>
<td>10.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gearbox Data</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction Ratio</td>
<td></td>
<td></td>
<td>19.5</td>
</tr>
<tr>
<td>Efficiency&lt;sup&gt;3&lt;/sup&gt;</td>
<td></td>
<td></td>
<td>0.87</td>
</tr>
<tr>
<td>Maximum Allowable Torque</td>
<td></td>
<td>oz-in (N-m)</td>
<td>175 (1.24)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encoder Data</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Resolution</td>
<td>CPR</td>
<td>500</td>
<td></td>
</tr>
</tbody>
</table>

1. Specified at max. winding temperature at 25°C ambient without heat sink
2. Theoretical values supplied for reference only
3. Effective gear box efficiency for this unit improved by use of ball bearings.

### Included Features
- 2-Pole Stator
- Ceramic Magnets
- Heavy-Gauge Steel Housing
- 7-Slot Armature
- Silicon Steel Laminations
- Stainless Steel Shaft
- Copper-Graphite Brushes
- Diamond Turned Commutator
- Motor Ball Bearings
- Output Ball Bearing
- Wide Face Gears

### Customization Options
- Alternate Winding
- Sleeve or Ball Bearings
- Modified Output Shaft
- Custom Cable Assembly
- Special Brushes
- EMI/RFI Suppression
- Alternate Gear Material
- Special Lubricant
- Optional Encoder
- Fail-Safe Brake

---

All values are nominal. Specifications subject to change without notice. Graphs are shown for reference only. © 2001 Pittman.
1. SHAFT ROTATION IS DETERMINED WITH POSITIVE VOLTAGE (+) ON #1 TERMINAL, WHILE LOOKING AT MOUNTING END.
2. MOTOR IS PRELOADED BALL BEARINGS PER P-107 .020 MAX. ON OUTPUT SHAFT.
3. MAX GEARBOX TORQUE RATING IS 100 oz.in. STANDARD GEARBOX, 160 oz.in. FOR CUT STEEL.
4. TERMINALS ARE TIN PLATED FOR SOLDERING.
5. ENCLOSED IS A HEAT-SENSITIVE OPTICAL ENCODER.
6. OPTIONAL REAR SHAFT EXTENSIONS AVAILABLE.
7. ENCODER LEAD CONNECTIONS TO BE DONE PER INDIVIDUAL LEAD WIRE DRAWING.
Appendix C: MATLAB Scripts

Script C1:

```matlab
%friction_param_save_tilt.m
%02/26/05
%run friction_param.mdl in a for loop
%collect the data and plot it
voltage_range = [-1:.05:1];
times = [0:0.001:5];

for k = 1:length(voltage_range)
    voltage_range(k)
    for i = 1:5
        setparam(tg, 28, voltage_range(k));
        i
        if voltage_range(k)==0
            else
                if voltage_range(k)>0
                    setparam(tg, 27, 10);
                    setparam(tg, 24, 10);
                else
                    setparam(tg, 27, -10);
                    setparam(tg, 24, -10);
                end
                start(tg);
                pause(5);
                stop(tg);
                outputlog = tg.OutputLog;
                theta_1(i, :) = outputlog(:, 1);
                theta_dot_1(i, :) = outputlog(:, 2);
                voltage_out(i, :) = outputlog(:, 5);
                pause(.5);
            end
        end
    end
    %first filtering
    thetadot_ave_temp(k, :) = (theta_dot_1(1, :) + theta_dot_1(2, :) + theta_dot_1(3, :) + theta_dot_1(4, :) + theta_dot_1(5, :))/5;
    %second filtering
    for j = 1: length(thetadot_ave_temp(k, :)) - 25
        sum = 0;
        for ii = 0 : 24
            sum = sum + thetadot_ave_temp(k, j+ii);
        end
        thetadot_ave(k,j) = sum/25;
```
% third filtering
for i = 1 : 41
    for j = 1 : 4975
        if abs(thetadot_ave(i, j) - thetadot_ave(i, j+1)) > 1
            thetadot_ave(i, j +1) = (thetadot_ave(i, j) + thetadot_ave(i, j + 2))/2;
        end
    end
end

for i = 1 : 25
    plot(times(1:4976), thetadot_ave(i, :));
    hold on;
end

ylabel('thetadot');
xlabel('second');
title('Tilt Axis');
save tilt_axis.mat thetadot_ave_temp thetadot_ave
% friction_param_save_pan.m
% run friction_param.mdl in a for loop
% collect the data and plot it

% voltage_range = [-1, -.95, -.9, -.85, -.75, -.7, -.65, -.6, -.55, -.5, .55, .6, .65, .7, .75, .8, .85, .9, .95, 1];
% voltage_range = [-.45:.05:.45];
% [.1:.1:.1];

for k = 1:length(voltage_range)
    voltage_range(k)
    for i = 1 : 5
        times = [0:0.001:5];
        setparam(tg, 29, voltage_range(k));
        if voltage_range(k)==0
            else
                if voltage_range(k)>0
                    setparam(tg, 28, 10);
                    setparam(tg, 25, 10);
                else
                    setparam(tg, 28, -10);
                    setparam(tg, 25, -10);
                end
            start(tg);
            pause(5);
            stop(tg);
            outputlog = tg.OutputLog;
            theta_1(i, :) = outputlog(:, 3);
            theta_dot_1(i, :) = outputlog(:, 4);
            voltage_out(i, :) = outputlog(:, 5);
            pause(.5);
        end
    end
end

% [num, den] = butter(10, 0.005);
% butter_filter = tf(num, den, 0.001);
% thetadot1(i,:) = lsim(butter_filter, theta_dot_1(i,:), times);

% thetadot1_butter(i,:) = theta_dot_1(i,:).*butter_filter;
thetadot1_ave1(k,:) = (theta_dot_1(1,:) + theta_dot_1(2,:) + theta_dot_1(3,:) + theta_dot_1(4,:)
+ theta_dot_1(5,:))/5;

for j = 1 : length(thetadot1_ave1(k,:)) - 25
    sum = 0;
    for ii = 0 : 24
sum = sum + thetadot1_ave1(k, j+ii);
end
thetadot1_ave2(k,j) = sum/25;
end
end

for i = 1 : length(thetadot1_ave2)
    plot(times(1:4976), thetadot1_ave2(i, :));
    hold on;
end
Script C3:

```matlab
%02/26/05
%set the initial values for friction_param.mdl
%put the logs on the signals
%subsystem under friction_param.mdl needs to be selected before running
%this file

ts = 0.001;          %sample time
const_voltage = 666;
peak_voltage = 777;

handles = get_param(gcb, 'porthandles');
outputports = handles.Outport;
for i = 1 : length(outputports)
    set_param(outputports(i), 'DataLogging', 'on');
    set_param(outputports(i), 'TestPoint', 'on');
end

set_param(outputports(1), 'Name', 'theta1_log');
set_param(outputports(2), 'Name', 'thetadot1_log');
set_param(outputports(3), 'Name', 'theta2_log');
set_param(outputports(4), 'Name', 'thetadot2_log');
set_param(outputports(5), 'Name', 'voltage_log');
```
clear save_torque
close all
clear all
% voltage = [-1:.05:1];
load datalog.mat
voltage = [-1, -.95, -.9, -.85, -.8, -.75, -.7, -.65, -.6, -.55, -.5, .5, .55, .6, .65, .7, .75, .8, .85, .9, .95, 1];

%calculate Torque
N1 = 2.47;     %External Gear Ratio
Nm1 = 19.5;    %Internal Gear Ratio
Kt = 4.36e-2;  %motor torque constant
Ka = .1;       %amplifier gain constant
torque = voltage .* N1 * Nm1 * Kt * Ka;

k = 1;
positive = 0;
viscous_friction_negative_sum = 0;
viscous_friction_positive_sum = 0;
negative_count = 0;
positive_count = 0;
for i = 1 : 22
    thetadot1_ss(i) = 0;
    for j = 4976-1000:4976
        thetadot1_ss(i) = thetadot1_ss(i) + thetadot1_ave2(i, j)/1000;
    end
    if abs(thetadot1_ss(i)) < 0.0001
        save_torque(k) = torque(i);
        k = k + 1;
        positive = 1;
    elseif i ~= 1 & abs(abs(thetadot1_ss(i)) - thetadot1_ss(i-1))>0.05
        if positive ==0
            viscous_friction_negative_sum = viscous_friction_negative_sum + (torque(i)-torque(i-1))
            /(thetadot1_ss(i)-thetadot1_ss(i-1));
            negative_count = negative_count + 1;
        else
            viscous_friction_positive_sum = viscous_friction_positive_sum + (torque(i)-torque(i-1))
            /(thetadot1_ss(i)-thetadot1_ss(i-1));
            positive_count = positive_count + 1;
        end
    end
end

coulomb_friction_negative_tilt = save_torque(1)
coulomb_friction_positive_tilt = save_torque(k-1)
viscous_friction_negative_tilt = viscous_friction_negative_sum/negative_count
viscous_friction_positive_tilt = viscous_friction_positive_sum/positive_count

plot(thetadot1_ss, torque, thetadot1_ss, torque, 'x')
xlabel('steady state velocity')
ylabel('torque');
title('Tilt Axis');
voltage = [-1:.05:1];

calculate Torque
N1 = 2.47; %External Gear Ratio
Nm1 = 19.5; %Internal Gear Ratio
Kt = 4.36e-2; %motor torque constant
Ka = .1; %amplifier gain constant
torque = voltage .* N1 * Nm1 * Kt * Ka;

k = 1;
positive = 0;
viscous_friction_negative_sum = 0;
viscous_friction_positive_sum = 0;
negative_count = 0;
positive_count = 0;
for i = 1 : 41
thetadot2_ss(i) = 0;
for j = 4976-1000:4976
    thetadot2_ss(i) = thetadot2_ss(i) + thetadot2(i, j)/1000;
end
if abs(thetadot2_ss(i)) < 0.0001
    save_torque(k) = torque(i);
    k = k + 1;
    positive = 1;
elseif i ~= 1 && abs(thetadot2_ss(i) - thetadot2_ss(i-1))>0.05
    if positive ==0
        viscous_friction_negative_sum = viscous_friction_negative_sum + (torque(i)-torque(i-1))
        / (thetadot2_ss(i)-thetadot2_ss(i-1));
        negative_count = negative_count + 1;
    else
        viscous_friction_positive_sum = viscous_friction_positive_sum + (torque(i)-torque(i-1))
        / (thetadot2_ss(i)-thetadot2_ss(i-1));
        positive_count = positive_count + 1;
    end
end


coulomb_friction_negative_pan = save_torque(1)
coulomb_friction_positive_pan = save_torque(k-1)
viscous_friction_negative_pan = viscous_friction_negative_sum/negative_count
viscous_friction_positive_pan = viscous_friction_positive_sum/positive_count
plot(thetadot2_ss, torque, thetadot2_ss, torque, 'x')
xlabel('steady state velocity')
ylabel('torque');
title('Pan Axis');
clear A b col1 col2 col3 col4
ts = 0.001;

for i = 1 : length(theta1_ave)-1
  col1 = -(thetadot1_ave(i)^2 + thetadot1_ave(i+1)^2)*ts;
  col2 = -(abs(thetadot1_ave(i)) + abs(thetadot1_ave(i+1)))*ts;
  col3 = 2 * voltage_in(i) *(theta1_ave(i+1) - theta1_ave(i));
  col4 = -2*(cos(theta1_ave(i+1)) - cos(theta1_ave(i)));
  A(i, :) = [col1 col2 col3 col4];
  b(i) = thetadot1_ave(i+1)^2 - thetadot1_ave(i)^2;
end

a_tilt = pinv(A)*b'

clear A b col1 col2 col3 col4
ts = 0.001;

for i = 1 : length(thetadot_ave)-1
  col1 = -(thetadot_ave(i)^2 + thetadot_ave(i+1)^2)*ts;
  col2 = -(abs(thetadot_ave(i)) + abs(thetadot_ave(i+1)))*ts;
  col3 = 2 * voltage_in(i) *(theta2_ave(i+1) - theta2_ave(i));
  col4 = -2*(cos(theta2_ave(i+1)) - cos(theta2_ave(i)));
  A(i, :) = [col1 col2 col3 col4];
  b(i) = thetadot_ave(i+1)^2 - thetadot_ave(i)^2;
end

a_pan = pinv(A)*b'
clear theta2 thetadot2 voltage_in voltage_in_sum theta2_sum thetadot2_sum voltage_in_ave theta2_ave thetadot2_ave

for k = 1:1
    start(tg);
    pause(14);
    stop(tg);
    outputlog = tg.OutputLog;
    theta2(k,:) = outputlog(:, 3);
    thetadot2(k,:) = outputlog(:, 4);
    voltage_in(k,:) = outputlog(:, 5);
    pause(0.5);
end

% voltage_in_sum(1:14285) = 0;
% theta2_sum(1:14285) = 0;
% thetadot2_sum(1:14285) = 0;
%
% for i = 1 : 10
%    voltage_in_sum = voltage_in_sum + voltage_in(i, :);
%    theta2_sum = theta2_sum + theta2(i,:);
%    thetadot2_sum = thetadot2_sum + thetadot2(i,:);
% end

voltage_in_ave = voltage_in;
theta2_ave = theta2;
thetadot2_ave = thetadot2;

j = 1;
for i = 1 : 14001
    if abs(thetadot2_ave(i)) > 10
        else
        thetadot_ave(j) = thetadot2_ave(i);
        j = j + 1;
    end
end
clear theta1 thetadot1 voltage_in voltage_in_sum theta1_sum thetadot1_sum voltage_in_ave theta1_ave thetadot1_ave

times = [0:0.001:14];

for k = 1:1
    start(tg);
    pause(14);
    stop(tg);
    outputlog = tg.OutputLog;
    theta1(k,:) = outputlog(:, 1);
    thetadot1(k,:) = outputlog(:, 2);
    voltage_in(k,:) = outputlog(:, 5);
    pause(0.5);
end

voltage_in_ave = voltage_in(1,:) ;
theta1_ave = theta1(1,:);
thetadot1_ave = thetadot1(1,:);
Script C10:

%Model of learning algorithm using Newton_Raphson
%root finding method
%Diana Mirabello

clear all
k=0;
alpha=1;
xdes = 20;
x = 0;
dist = 100;
disp('x');
disp(x);
plot(k,x,'r+');
hold on
pan(1)=0;
for k=1:10;
    pan(k+1) = pan(k) - alpha*((x - xdes)*(cos(pan(k)))^2)/dist;
    x = dist*tan(pan(k+1));
    disp(x);
    plot(k,x,'r+');
end

title('Newton-Raphson Simulation for Pan Angle: Xdesired = 20 cm');
ylabel('X position');
xlabel('iteration');
hold off

%tilt
k=0;
alpha=1;
ydes = 20;
y = 0;
dist = 100;
disp('y');
disp(y);
figure(2), hold on;
plot(k,y,'r+');
hold on

for k=1:10;
    if tilt(k) >= -43.63 & tilt(k) <= 43.63
        z = (dist^2 + (-.53*dist + .9)^2)^(1/2);
    end
    if tilt(k) > 43.63 & tilt(k) <= 130.9
        z = (dist^2 + (2.8*dist + 2.4)*2)^(1/2);
    end
    if tilt(k) > 130.9 & tilt(k) <= 218.2
        z = (dist^2 + (13.91*dist + 13.02)^2)^(1/2);
    end
end
if tilt(k) > 218.2 & tilt(k) <= 305.4
    z = (dist^2 + (21.17*dist + 19.2)^2)^(1/2);
end
if tilt(k) < -43.63 & tilt(k) >= -130.9
    z = (dist^2 + (-14.24*dist - 8.94)^2)^(1/2);
end
if tilt(k) < -130.9 & tilt(k) >= -218.2
    z = (dist^2 + (-21.88*dist - 13.5)^2)^(1/2);
end
if tilt(k) < -218.2 & tilt(k) >= -305.4
    z = (dist^2 + (-37.3667*dist - 25.0167)^2)^(1/2);
end

tilt(k+1) = tilt(k) - alpha*((y - ydes)*(cos(tilt(k)))^2)/z;
y = z*tan(tilt(k+1));
disp(y);
plot(k,y,'r+');
end

title('Newton-Raphson Simulation for Tilt Angle: Ydesired = 20 cm')
xlabel('Iteration')
ylabel('Y Position')
hold off
% Viable Targets: [-103:103, -139.8:139.8]

% Initialize
dist = 500;
x_des = 62.3385;
y_des = -100.4303;
alp = 1;
tilt_des = 0;
pan_des = 0;
x = 1000;
y = 1000;
kp1 = 50;
ki1 = .0019;
kd1 = 6.5;
kp2 = 50;
ki2 = .09;
k = 2;
p = 20;

% MAIN LOOP BEGIN

% pan_actual & tilt_actual Simulink

while(abs(x - x_des) > 37.5 || abs(y - y_des) > 37.5)
    tilt_des = tilt_des
    pan_des = pan_des
    setparam(tg, 25, tilt_des);
    setparam(tg, 32, pan_des);
    start(tg);
    pause(1);
    stop(tg);
    % theta = tg.OutputLog;
    % tilt_actual = theta(length(theta), 1);
    % pan_actual = theta(length(theta), 2);
    tilt_actual = 0;
    pan_actual = 0;
    % Fire Shot

k = waitforbuttonpress;
if k == 0
    loc = get(0, 'PointerLocation');
    x = 103 - (loc(2) - 1) * 206 / 1034;
    y = -139.8 + (loc(1) - 1) * 279.6 / 1399;
    [x y]
else
  \%manual correction
end
\%hide figure

if abs(x - x_des) <= 37.5 & abs(y - y_des) <= 37.5
  disp('the target has been hit')
  break
end
\%Learning Algorithm

if tilt_actual >= -43.63 & tilt_actual <= 43.63
  dtilt = (dist^2 + (-5.3*dist + 9)^2)^(1/2);
end
if tilt_actual > 43.63 & tilt_actual <= 130.9
  dtilt = (dist^2 + (28*dist + 24)*2)^(1/2);
end
if tilt_actual > 130.9 & tilt_actual <= 218.2
  dtilt = (dist^2 + (139.1*dist + 130.2)^2)^(1/2);
end
if tilt_actual > 218.2 & tilt_actual <= 305.4
  dtilt = (dist^2 + (211.7*dist + 192)^2)^(1/2);
end
if tilt_actual < -43.63 & tilt_actual >= -130.9
  dtilt = (dist^2 + (-142.4*dist - 89.4)^2)^(1/2);
end
if tilt_actual < -130.9 & tilt_actual >= -218.2
  dtilt = (dist^2 + (-218.8*dist - 135)^2)^(1/2);
end
if tilt_actual < -218.2 & tilt_actual >= -305.4
  dtilt = (dist^2 + (-373.667*dist - 250.167)^2)^(1/2);
end
pan_des = pan_actual - alpha*((x - x_des)*(cos(pan_actual))^2)/dist;
pan_des = -pan_des + pan_actual;
tilt_des = tilt_actual - alpha*((y - y_des)*(cos(tilt_actual))^2)/dtilt;
tilt_des = 3.0*(tilt_des - tilt_actual);

end
\%move pantilt device to new angles

\%MAIN LOOP END
Appendix D: Simulink Models

Figure D1: Friction Identification Model

Figure D2: Parameter Verification Model
Figure D3: PID Controller Model

Figure D4: Final Control Model