DOUBLE ARM JUGGLING SYSTEM
Progress Report for ECSE-4962 Control Systems Design

Team 1
Trinell Ball
John Kua
Linda Rivera

March 29, 2006
Rensselaer Polytechnic Institute
Executive Summary

Our goal for this project is to design and develop a juggling system utilizing a pivoted, double-ended arm that is able to handle two ping pong balls at the same time, maintaining one in the air at all times. The system will systematically juggle the two balls using a pan and tilt mechanism to move the arms. Feedback on the arm position will be provided by shaft encoders and feedback on the ball position will be provided by an overhead camera. With the camera input, the system should be capable of learning from toss/catch errors. The control challenges in this project include high positioning accuracy with fast response and recovery, as well as accurate tracking and high acceleration in the tilt axis in order to launch a ball on the correct trajectory.

In pursuit of this goal, we have completed the bulk of the design stage and are moving into the testing stage. Initial parameter specification and selection have been completed, with a final design having emerged from that process. The physical system has been built, including a number of support structures such as a system mount and a camera mount. The system itself has been modeled, with the various parameters measured or calculated for use in a Simulink model (linear and nonlinear). We are currently in the process of verifying and refining this model to return accurate results. For the control system, we have developed controllers for both axes using root locus methods and hand-tuned PID controllers and have achieved, in simulation, results that meet nearly all of our specifications with the PID controllers. While we hit a small roadblock in the implementation of our simulated PID controllers in the physical system involving a parameter mismatch between MATLAB and LabVIEW, we have tested a hand-tuned version on the system and achieved good results in terms of speed of response, power, and position accuracy. However, open loop launch tests (1-D and 2-D) have shown that our net system is causing issues with repeatability, and so we are in the process of reworking them. Finally, our vision system is on track, with the processing scripts returning X-Y and Z data.

Compared to our original proposal, we are fairly close in terms of scheduling, costs, and plan of action. In terms of the schedule, we have completed most of the items to date – we are only behind on a few items and those are well on their way to completion. We are about 10% over budget and 20% below our initial estimated costs as we a major purchase we expected to make was not necessary, but the construction of support items have taken those funds. Our plan of action has been followed faithfully to this point, and we do not foresee deviating from it.

With the bulk of the design and build process behind us, we do not currently foresee any significant additional issues that were not predicted in the initial proposal and are very confident that we will be able to achieve our goal of building a two-ball juggling system.
Table of Contents

Introduction ................................................................................................................... 1
Preliminary Results......................................................................................................... 3
Physical Design................................................................................................................ 3
  Overall System Design & Inertia .................................................................................. 3
  Other Physical Modifications ....................................................................................... 4
  Overall System Mounting ............................................................................................ 4
  Camera Mounting ......................................................................................................... 5
  Challenges .................................................................................................................. 5
Model Development........................................................................................................ 7
  Parameter Identification .............................................................................................. 8
  Friction Identification .................................................................................................. 10
  Inertia/Mass Parameters ............................................................................................ 12
  Spring Constant Determination of the Carbon Fiber Shaft ........................................ 13
Model Linearization........................................................................................................ 13
Model Verification.......................................................................................................... 14
Velocity Estimation........................................................................................................ 14
Trajectory Calculation.................................................................................................... 15
Control System Development ....................................................................................... 18
  Overview .................................................................................................................... 18
  Root Locus (Pole Placement Method) .......................................................................... 18
  PID Controller Design ............................................................................................... 20
  Controller Comparisons with Specifications (Table 1) ................................................ 22
  Controller Implementation and Goals ......................................................................... 23
Camera Development .................................................................................................... 25
  Vision Module Familiarization .................................................................................... 25
  Image Processing & Performance ................................................................................. 26
  Processing Challenges ............................................................................................... 28
  Data Verification ......................................................................................................... 29
  Next Steps ................................................................................................................ 30
Summary of Progress .................................................................................................... 31
List of Figures

Figure 1 - Complete Juggling Mechanism Assembly ................................................................. 3
Figure 2 - Inertia & Mass Property Data.................................................................................... 3
Figure 3 - Mounting Mechanism for Shaft ................................................................................ 4
Figure 4 - System Mounting ..................................................................................................... 4
Figure 5 - Initial Camera Mounting Design .............................................................................. 5
Figure 6 - Nonlinear Model of a Single Joint ........................................................................... 7
Figure 7 - End to End Model of a Single Joint .......................................................................... 8
Figure 8 - Voltage to Current and Digital Value to Current Transfer Curves – Tilt Axis ....... 9
Figure 9 - Voltage to Current and Digital Value to Current Transfer Curves – Pan Axis ......... 9
Figure 10 - Velocity Curves for a Series of Constant Torques - Tilt ........................................ 10
Figure 11 - Velocity Curves for a Series of Constant Torques - Pan ........................................ 11
Figure 12 - Digital Value to SSV and Torque to SSV Transfer Curves – Tilt ......................... 11
Figure 13 - Digital Value to SSV and Torque to SSV Transfer Curves – Pan ......................... 12
Figure 14 - Friction Parameters ............................................................................................. 12
Figure 15 - Inertia Parameters ............................................................................................... 13
Figure 16 - Simulated vs. Measure Responses to Constant Torque ....................................... 14
Figure 17 - Sample Iterative Trajectory Calculation .............................................................. 16
Figure 18 - Simulink Diagram of Trajectory Calculation ....................................................... 16
Figure 19 - Comparison Between Iterative and Simulink Trajectory Calculations ............... 17
Figure 20 - Root Locus of the Discretized System (Tilt Axis) .................................................. 19
Figure 21 - Step Response of the Root Locus Derived Controller on a Linear System (Tilt Axis) .... 19
Figure 22 - Step Response of the Root Locus Derived Controller on a Nonlinear System (Tilt Axis) ..... 20
Figure 23 - Step Response of the PID Controlled System (Tilt Axis) ................................. 21
Figure 24 - Tracking Response of the PID Controlled System (Tilt Axis) ............................ 21
Figure 25 - Step Response of the PID Controlled System (Pan Axis) ..................................... 22
Figure 26 - Preliminary Vision Data ........................................................................................ 25
Figure 27 - Image Processing Results – Test 1 Image 16/30 .................................................... 27
Figure 28 - Circle Detected Data (Pixels) ............................................................................. 27
Figure 29 - Image Processing Results – Test 2 .................................................................... 27
Figure 30 - Blurred Image .................................................................................................... 28
Figure 31 - Circles Detected Data (pixels) ............................................................................ 28
Figure 32 - Background Threshold Problem .......................................................................... 29
Introduction

Our goal for this project is to design and develop a juggling system utilizing a pivoted, double-ended arm that is able to handle two ping pong balls at the same time, maintaining one in the air at all times. The motivation for this scheme arose primarily from a previous project developed by Team 7 from the Control Systems Design course in 2004. Their system consisted of a single arm that would toss a ball, which followed a projectile motion trajectory. The arm then would pan approximately 180 degrees to catch it. Our purpose is to improve this mechanism by including a second arm that allows handling of two or more balls in a continuous juggling motion. We also intend to extend their open loop control system to a closed loop feedback control.

The system will systematically juggle the two balls using a pan and tilt mechanism to move the arms. Pan and tilt motions will be achieved by the use of two distinct motors, controlled by an embedded control system. Feedback on the arm position will be provided by shaft encoders and feedback on the ball position will be provided by an overhead camera. A second camera, facing one side of the mechanism, may be added later on in order to more accurately predict the ball's location. We expect that our arm will be able to execute toss and catch operations quickly, relative to the flight time of the ball, with a minimum of overshoot and steady state error. With the camera input, the system should be capable of learning from toss/catch errors. In addition, disturbances in the flight path of the ball, perhaps caused by wind, will be compensated for in the catching mechanism.

The tilt axis of the arm, which handles throwing and catching operations, must be able to traverse $\pm 30^\circ$ and achieve a maximum velocity of 15 rad/s at an acceleration of 200 rad/s so as to be able to launch a ball on a 1s flight trajectory and recover quickly. High positioning accuracy (less than 5% overshoot, $\pm 1^\circ$ steady state error, and less than 0.1s settling time) is required to position the net in the proper place to catch the ball. Good tracking accuracy (able to track a 2Hz sinusoidal input) is required to launch the ball on the correct flight profile.

The pan axis only has to assist with catching operations and allow for angular separation between each ball's trajectory. As such, it is only required to traverse $\pm 10^\circ$ and achieve a maximum acceleration of 35 rad/s. However, positioning accuracy is still critical, which call for tight specifications in this area (less than 5% overshoot, $\pm 1^\circ$ steady state error, and less than 0.1s settling time).

In pursuit of this goal, we have completed the bulk of the design stage and are moving into the testing stage. Initial parameter specification and selection have been completed, with a final design having emerged from that process that we have determined to be achievable as far as our calculations can
determine. The physical system has been built, including a number of support structures such as a system mount and a camera mount. The system itself has been modeled, with the various parameters measured or calculated for use in a Simulink model (linear and nonlinear) that is being used to develop and test our control systems. We are currently in the process of verifying and refining this model to return accurate results. For the control system, we have developed controllers for both axes using root locus methods and hand-tuned PID controllers and have achieved, in simulation, results that meet nearly all of our specifications with the PID controllers. While we hit a small roadblock in the implementation of our simulated PID controllers in the physical system involving a parameter mismatch between MATLAB and LabVIEW, we have tested a hand-tuned version on the system and achieved good results in terms of speed of response, power, and position accuracy. However, open loop launch tests (1-D and 2-D) have shown that our net system is causing issues with repeatability, and so we are in the process of reworking them. Finally, our vision system is on track, with the processing scripts returning X-Y and Z data.

In terms of our schedule, we are on track, having met most of our scheduled deadlines. Model verification is behind schedule, but we are working to finish than soon. The vision system is behind schedule according to our original plan, however there is room to complete that before it is needed to be integrated with the system.

We have made significant progress, with the system built and beginning to operate as our initial calculations predicted. The difficulties we have run into so far all have been solved or we have found what we believe will be good solutions for them. This project is well on its way to completion, and we are confident that we will be able to achieve our goals.
Preliminary Results

Physical Design

**Overall System Design & Inertia**

Below is the updated version of our SolidWorks model:

![Figure 1 - Complete Juggling Mechanism Assembly](image)

Inertia and Mass Property Data were obtained from SolidWorks model and they are summarized in the table below:

<table>
<thead>
<tr>
<th></th>
<th>Density (Kg/m²)</th>
<th>Mass (Kg)</th>
<th>Volume (m³)</th>
<th>Surface Area (m²)</th>
<th>Moment of Inertia (kgm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pan Axis</strong></td>
<td>1931.19118581</td>
<td>0.82370203</td>
<td>0.00042653</td>
<td>0.20179167</td>
<td>Iyy = 0.00624529</td>
</tr>
<tr>
<td><strong>Tilt Axis</strong></td>
<td>1389.80833613</td>
<td>0.37840067</td>
<td>0.00027227</td>
<td>0.15592179</td>
<td>Iyy = 0.00553462</td>
</tr>
<tr>
<td><strong>Motor &amp; Yoke Pinion Sprocket</strong></td>
<td>2773.32976738</td>
<td>0.23962383</td>
<td>0.00008640</td>
<td>0.01307505</td>
<td>Iyy = 0.001000312</td>
</tr>
</tbody>
</table>

![Figure 2 - Inertia & Mass Property Data](image)
Other Physical Modifications

Other physical modifications that have been made to the system include building a mounting mechanism for the shaft. This mounting consists of an aluminum plate, two brackets, and screws. A SolidWorks model for this is shown below. The image on the left shows the components for the mounting separately; the image on the right shows how the shaft is actually mounted.

![Figure 3 - Mounting Mechanism for Shaft](image)

Another modification has to do with the cables used in the mechanism. In order to move the system to an adequate location, it was necessary to lengthen the cables of the system. Connectors were added inline to allow for easy disconnection.

Overall System Mounting

The location where the system is at this moment does not give us enough space to perform our tests. Therefore, it was necessary to build a mount for the system itself that we could clamp onto a stool and move to an appropriate location. This mounting was built out of 1” poplar stock.

![Figure 4 - System Mounting](image)
Camera Mounting

As we had planned, a camera will be utilized in our system to provide position feedback for the ball. Since we needed to mount the camera right above the juggling system, building a mounting structure for the camera was necessary. The preliminary design is shown below:

![Initial Camera Mounting Design](image)

This camera mounting system was built from lightweight wood that allow us to easily move it to different locations when performing testing of the system. We approximated that the required height above the shaft and nets should be 1m in order for the camera to view the complete length of the arm and nets; however, we have added some flexibility by being able to adjust the middle shaft up or down so that the camera can be at a number of heights in case any of the components of the juggling mechanism is modified.

Challenges

The main challenge that has come up in terms of the physical design has to do with the nets. The material we used to build them is very light, however it is not stiff enough, which causes continuous variations in the shape of the cup. These variations have caused poor repeatability of system response from launch to launch. Building nets that are very symmetrical and sturdy has not been possible. Due to these issues, we will proceed as follows: First, we will attempt to continue to use the nets we have, but we will wrap them in aluminum foil. We think that this
may be a way of achieving sturdy, uniform shaped nets. If this fails, we have purchased a set of alternate pre-manufactured nets to replace the old ones. The challenge we would be facing with these is that they are made out of a heavier material that will affect the inertia of our system. Another challenge that could come up has to do with figuring out a way of properly mounting the new nets.
Model Development

The basis for the model of the system is the Lagrange-Euler model:

\[ \tau = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + F(\dot{\theta}) + G(\theta) \]

By treating the two joints as being decoupled and because the gravity has no effect on either axis (the pan axis is unaffected and the tilt axis is balanced), it is possible to eliminate the Coriolis/centrifugal and the gravity terms. Now we have two independent equations of the form:

\[ \tau = M(\theta)\ddot{\theta} + F(\dot{\theta}) \]

Expanding the mass/inertia term for separate load/actuator figures and the friction term for effective viscous and Coulomb figures:

\[ \tau_m = \left( \frac{J_a + J_m}{n} + nJ_L \right)\ddot{\theta}_s + F_v\dot{\theta}_s + F_c\text{sign}(\dot{\theta}_s) \]

In a Simulink diagram, this looks like:

\[ J_a = \text{Actuator Inertia} \]
\[ J_m = \text{Gear Inertia} \]
\[ J_L = \text{Load Inertia} \]
\[ n = \text{Gear Ratio} \]
\[ F_v = \text{Viscous Friction} \]
\[ F_c = \text{Coulomb Friction} \]
\[ \tau_m = \text{Motor Torque} \]
\[ \theta_s, \dot{\theta}_s, \ddot{\theta}_s = \text{Output Shaft Position / Velocity / Accel} \]

In a Simulink diagram, this looks like:

Figure 6– Nonlinear Model of a Single Joint

Added to this nonlinear model of each joint is the motor speed saturation – our motor has a maximum output speed of 720 RPM or 75.4 rad/s, which translates to an output shaft speed of 18.85 rad/s, given the gear ratio of 4:1.

The entire open loop system, from the LabVIEW control outputs to the physical shaft position are shown in the following system:
The LabVIEW code outputs a digital value from -32768 to +32767 (modeled by the DAC output saturation block) to the D/A converters, which converts that value to an analog voltage, which is then passed to the motor controllers, which outputs a proportional amount of current (calibrated to approximately 0.1A/V) to the motor, which outputs a torque equal to the controller current output times a torque constant (0.0436N*m/A).

**Parameter Identification**

**DAC to Current**

In the above diagram, the DAC to Current block converts the digital input into the current output - this represents the system from the D/A converters to the motor controller outputs. While each D/A converter is fairly close to its specifications, there is still a certain amount of offset and deviation from the ideal slope. In addition, the motor controllers which convert that voltage to current also each have slightly different gain constants (adjusted by pots) and offsets.

In order to accurately characterize this for the system, a series of tests were performed where a series of digital values across the entire range were applied to the system and the current applied to the motor was measured with an ammeter. This was done after hand calibrating the motor controllers to approximately 0.1A/V. For large values where the motor reached its maximum velocity, it was necessary to apply a load to the motor in order for the correct current to be measured. With those values, a linear curve fit was performed. This test was performed for each controller channel/motor pair, with the following results:
Figure 8 - Voltage to Current and Digital Value to Current Transfer Curves - Tilt Axis

Figure 9 - Voltage to Current and Digital Value to Current Transfer Curves - Pan Axis

\[ I_{\text{tilt}} = -3.4197 \times 10^{-5} x + 0.0072 \]

\[ I_{\text{pan}} = -3.4696 \times 10^{-5} x + 0.0043 \]
As we can see from the graphs, there is a polarity change (negative slope) and the linear fit is quite good. From the fitted curve, we observe a slight offset in both axes.

**Friction Identification**

The idea behind friction identification is to apply a constant torque to the system and then measure the steady state velocity as the forces due to friction will eventually cancel out the acceleration due to torque. In order to perform this test, a LabVIEW code was written to perform a series of tests at various inputs and capture the resulting velocity data to a file. Each test consisted of hitting the system with a brief torque pulse (250ms/500ms tilt/pan) to break stiction and then applying a constant torque for 10 or 20 seconds (tilt/pan) to allow the system to reach steady state velocity.

That data was imported into MATLAB where it was processed to remove glitches (there were problems with the calculated velocity in LabVIEW, which was later determined to be an overflow problem). These curves were plotted as follows for the two axes:

![Figure 10 - Velocity Curves for a Series of Constant Torques - Tilt](image)
From these plots, we observe that the maximum velocity of the output shaft is approximately 20 rad/s. The last 100 velocity data points (equivalent to 1 second) were averaged and used as the steady state velocity. The smallest and largest data points were discarded and the rest used to fit a line. From this, the viscous and Coulomb friction could be determined. The following plots show the curve fit:
The curve fit is quite good, once the saturated values and the stiction values are discarded, and the following friction parameters are determined:

<table>
<thead>
<tr>
<th></th>
<th>Viscous Friction (N*s)</th>
<th>Coulomb Friction (N*m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tilt Axis (positive)</td>
<td>-7.8155e-005</td>
<td>-0.0024</td>
</tr>
<tr>
<td>Tilt Axis (negative)</td>
<td>-6.8796e-005</td>
<td>0.0026</td>
</tr>
<tr>
<td>Tilt Axis (average)</td>
<td>7.3476e-005</td>
<td>0.0025</td>
</tr>
<tr>
<td>Pan Axis (positive)</td>
<td>-5.0093e-005</td>
<td>-0.0025</td>
</tr>
<tr>
<td>Pan Axis (negative)</td>
<td>-5.1430e-005</td>
<td>0.0026</td>
</tr>
<tr>
<td>Pan Axis (average)</td>
<td>5.0762e-005</td>
<td>0.0025</td>
</tr>
</tbody>
</table>

**Figure 14 - Friction Parameters**

**Inertia/ Mass Parameters**

Our model of the effective load on each axis is as follows:

\[
J_{eff} = \frac{J_a + J_m}{n} + nJ_L
\]

- \(J_a = \text{Actuator Inertia}\)
- \(J_m = \text{Gear Inertia}\)
- \(J_L = \text{Load Inertia}\)
- \(n = \text{Gear Ratio}\)

To find the effective inertial loads in the system, we used a combination of calculation and measurement. The load of the basic pan and tilt system was provided in the SolidWorks model, and by adding our additional components with the proper weights/densities, we were able to
determine the load on each axis in SolidWorks. The gear inertia was measured separately in SolidWorks and the actuator inertia was determined from the motor datasheet.

<table>
<thead>
<tr>
<th></th>
<th>Tilt Axis</th>
<th>Pan Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actuator Inertia (kg*m²)</td>
<td>1.6E-6</td>
<td>1.6E-6</td>
</tr>
<tr>
<td>Gear Inertia (kg*m²)</td>
<td>4.4814918e-007</td>
<td>4.4814918e-007</td>
</tr>
<tr>
<td>Load Inertia (kg*m²)</td>
<td>0.00553462</td>
<td>0.00624529</td>
</tr>
<tr>
<td>Effective Inertia (kg*m²)</td>
<td>2.7124e-004</td>
<td>2.9944e-004</td>
</tr>
</tbody>
</table>

Figure 15 – Inertia Parameters

Spring Constant Determination of the Carbon Fiber Shaft

In the initial development of our design, we were concerned that any flexibility of the arm shaft would cause issues with oscillation. In order to attempt to model this, we developed a test to determine the spring constant of the shaft.

To determine the spring constant of the carbon fiber shaft, it was clamped down to a stable platform. Weights were hung of one end of the rod. We measured the displacement at both ends of the rod due to the applied force, and then we took the difference between them to determine the actual displacement. The formula for the spring constant is given by

\[ F = kx \]

First we applied a force of 11.121 Newtons and obtained a displacement to calculate a spring constant \( k_1 \), and then we applied a force of 22.241 Newtons to obtain a spring constant \( k_2 \). Then we took the average of the two constants to determine our actual spring constant. The \( k \) that we obtained was approximately 4633.63 Newtons/m. Such a high spring constant tells us that the rod is quite rigid and that the oscillation due to the rod is not an issue.

Model Linearization

In order to linearize this model, the Coulomb friction term was discarded and the following state space equations were created:

\[
\begin{align*}
x_1 &= \theta, & x_2 &= \dot{\theta} \\
\dot{x}_1 &= \dot{\theta}, & \dot{x}_2 &= \ddot{\theta} \\
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{F_r}{J_{eff}} x_2 + \frac{u}{J_{eff}} \\
y &= x_1
\end{align*}
\]
Which gives us the transfer function:

\[ \frac{Y(s)}{R(s)} = \frac{3687}{s^2 + 0.2709s} \]

**Model Verification**

To verify our model, the first test performed was to input a series of constant torques into the system and verify that the velocity curves and steady state velocity was comparable to that measured during the friction identification steps.

![Figure 16 - Simulated vs. Measure Responses to Constant Torque](image)

As we can see here in a comparison of the simulated response vs. the measured responses, the acceleration seems to be about 4 times slower to reach steady state. This may be due to inaccuracies in the estimation of the effective inertia. In addition, stiction is not modeled, so much lower steady state velocities are possible on the simulated model.

Further testing is necessary to continue to refine our model and achieve accurate simulation results as our control system design is done in simulation. We will continue to compare simulated controlled outputs with outputs from the actual system to determine which parameters need refinement. If necessary, we may use system identification methods such as with a “chirp” excitation and/or the LabVIEW System Identification Toolkit.

**Velocity Estimation**

Velocity estimation is currently done with a finite difference method at a rate of 10ms. This allows us to capture a minimum rotational velocity of 0.1534 rad/sec, which is about 1/140th of
our maximum speed (21 rad/sec) and approximately 1/95th of our design peak velocity (14.5 rad/sec).

\[
\omega = \frac{\theta_{\text{current}} - \theta_{\text{previous}}}{T}
\]

One major problem with the basic finite difference method was with overflow. When the register that held the current position overflowed, the velocity output would have a large positive or negative spike. Initially, a filter was used to remove glitches from the data, but in the end both the position register and velocity calculation was rewritten to handle the overflow properly. The position register was restricted to +/- 180 degrees and the velocity calculation added or subtracted one revolution as necessary when the calculated velocity was outside certain limits (+/- 153 rad/s).

**Trajectory Calculation**

In order to model the trajectory of the ball, we have to account for drag, which is somewhat significant for our selected projectile, a ping-pong ball. Drag force is described as:

\[
D = \frac{1}{2} \rho v^2 AC_d
\]

which is proportional to the square of the velocity, making our trajectory equation a differential equation:

\[
\ddot{\theta} = g - \frac{\text{sign}(\dot{\theta}) D(\dot{\theta})}{m}
\]

Initially, an iterative approach was used where by the next position and velocity was calculated using the current position, velocity, and acceleration due to gravity and drag. Values for range (distance traveled in x), max height, and flight time were determined from the data. The result seemed satisfactory and formed the basis of our preliminary analysis. A sample trajectory is shown below:
Later, the trajectory was implemented as a Simulink simulation to take advantage of its ODE solver:

This produced the same results for the no drag case, but calculated that drag had less of an effect than the original simulation determined. We believe this to be due to small inaccuracies in
the iterative process propagated and accumulated across the simulation. A comparison of the results from the two systems:

![Graph showing comparison between iterative and Simulink trajectory calculations.](image)

Figure 19 - Comparison Between Iterative and Simulink Trajectory Calculations
Control System Development

Overview
Presently we have developed and tested two types of controller for our system: PID and controller design in MATLAB by Pole placement. At the moment we have a non-linear system model, which resembles the actual system dynamics, and a linear model for design purposes. In simulation we are applying our controller design to the linear and non-linear model, observing how the system responses to various inputs. We are also in the process of testing our controller on the physical system.

Root Locus (Pole Placement Method)
The pan and tilt controller designed via the pole placement method required the use of the following steps:
1. Obtain linearized transfer function
2. Import transfer function to rltool
3. Convert continuous model to discrete model (sampling time 10ms)
4. Define design constraints, such as rise time and settling time
5. Place gain constant at the crossings of design constraints
6. Export controller to Simulink model of system
7. Run simulation to test

Tilt Controller (Pole Placement method)
The linearized model the transfer function obtained for the tilt system, as discussed in the modeling section, is:

\[
\frac{1.153}{s^2 + .2079s}
\]

The transfer function was discretized using 10ms sample time. Using the rltool command in MATLAB a root locus was obtained in the z-plane. The specifications were set according to our proposal, with the settling time at 0.1 seconds and overshoot less than 5%. The process led to the root locus plot shown in Figure 20.
The specifications placed constraints represented by the yellow boundaries, which represents the natural frequency and a damping factor. Our closed loop poles, represented by the little pink squares, were placed approximately at .7 damping and 62.8 rad/s natural frequency. The discretized controller obtained from this analysis was:

$$\frac{11089(z - .7171)}{z + .7408}$$

Figure 21 is the plot of the step response with the above controller implemented in the linear system. The overshoot here is significant, almost 44%, and the rise time is well below our desired specification. As of now, this controller clearly underperforms given our objectives. Data tips at the critical points of interest are inserted in the plot. Table 1 compares the values of specifications to obtained simulation data.
Figure 22 shows the step response of the nonlinear model implementing the above controller. The rise time in this case worsened compared to the linear model, but the overshoot improved; this is due to the coulomb friction damping present in the non-linear model. Once again data tips are inserted at the critical points of interest.

![Figure 22 - Step Response of the Root Locus Derived Controller on a Nonlinear System (Tilt Axis)](image)

**Pan Controller (Pole Placement Method)**

The Pan Controller pole placement method was in development. Currently we are experiencing difficulties between rltool and the Simulink model in terms of system stability. We are abandoning this approach because the PID controller is achieving good performance.

**PID Controller Design**

As far as meeting our system specifications, PID has proven to be the superior controller for both the pan and the tilt axes. PID controllers were designed using the MATLAB Simulink PID block. Proportional, derivative and integral gains were hand tuned until the system performance met our desired specifications or came very close to meeting them.

**Tilt PID Controller Design**

While designing the PID controller, the controller itself was implemented directly in the non-linear model, disregarding the linearized model since our interest lies primarily with the actual system. In Figure 26, the response of the non-linear tilt mechanism due to a step response can be observed. The controller is very close to meeting our specifications. It has almost zero steady state error and no overshoot (at least not visible in this plot). It is also very close to meeting the requirements of the specified rise time. A 1 mV, 60Hz sinusoidal noise source was added to the system in the sensor path and the output response was the same as the uncorrupted system. The system response with the corrupted signal is in Figure 23.
Figure 23 – Step Response of the PID Controlled System (Tilt Axis)

Figure 24 represents the system’s ability to track a sinusoid (amplitude: 1 rad, velocity: 1 rad/sec). The input and output overlap on top of one another in the figure, and we can see there has been no phase introduced in the response. The ability of the system to track this sinusoid is quite good, however we still have to run more demanding tests.

Figure 24 – Tracking Response of the PID Controlled System (Tilt Axis)

PID gain values used for the designing the tilt controller:

\[ K_p = 800, \ K_i = 20, \ K_d = 40 \]
Pan PID Controller Design

The PID pan controller used the same method of development as the tilt axis controller. The only difference is that the step input the controller has to track is smaller than that of the tilt because the range of motion is much smaller for the pan axis. So, the pan axis tracks a 0.1 rad step input in the simulation.

![Figure 25 – Step Response of the PID Controlled System (Pan Axis)](image)

In Figure 25, it can be seen that the overshoot and the rise time is close to our specifications, but we have a small amount of a steady state error, which is slightly above our specification. This problem can be eliminated by tracking an input that is slightly less than the desired specification, or by increasing the integral gain. The controller can also track a sinusoidal signal accurately. It is also robust when a low frequency noise is applied to the system in the sensor feedback path.

PID gain values used for the pan axis simulation:

\[ K_p = 710, \ K_i = 650, \ K_d = 23 \]

Controller Comparisons with Specifications (Table 1)

<table>
<thead>
<tr>
<th>Category</th>
<th>Specifications</th>
<th>Simulation (linear)</th>
<th>Simulation (non-linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot</td>
<td>&lt; 5%</td>
<td>44%</td>
<td>28.7%</td>
</tr>
<tr>
<td>Rise Time</td>
<td>0.2s</td>
<td>0.16s</td>
<td>0.18s</td>
</tr>
<tr>
<td>Settling Time</td>
<td>0.1s</td>
<td>0.38s</td>
<td>0.3s</td>
</tr>
<tr>
<td>Steady State Error</td>
<td>±1°</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Tilt Axis - PID

<table>
<thead>
<tr>
<th>Category</th>
<th>Specifications</th>
<th>Simulation (non-linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot</td>
<td>&lt; 5%</td>
<td>0%</td>
</tr>
<tr>
<td>Rise Time</td>
<td>0.2s</td>
<td>0.2s</td>
</tr>
<tr>
<td>Settling Time</td>
<td>0.1s</td>
<td>0.25s</td>
</tr>
<tr>
<td>Steady State Error</td>
<td>±1°</td>
<td>0</td>
</tr>
<tr>
<td>Can track a sinusoid</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Pan Axis - Root Locus Derived

<table>
<thead>
<tr>
<th>Category</th>
<th>Specifications</th>
<th>Simulation (linear)</th>
<th>Simulation (non-linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot</td>
<td>&lt; 5%</td>
<td>unstable</td>
<td>unstable</td>
</tr>
<tr>
<td>Rise Time</td>
<td>0.18s</td>
<td>unstable</td>
<td>unstable</td>
</tr>
<tr>
<td>Settling Time</td>
<td>0.1s</td>
<td>unstable</td>
<td>unstable</td>
</tr>
<tr>
<td>Steady State Error</td>
<td>±1°</td>
<td>unstable</td>
<td>unstable</td>
</tr>
</tbody>
</table>

### Pan Axis - PID

<table>
<thead>
<tr>
<th>Category</th>
<th>Specifications</th>
<th>Simulation (non-linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot</td>
<td>&lt; 5%</td>
<td>2%</td>
</tr>
<tr>
<td>Rise Time</td>
<td>0.2</td>
<td>0.1s</td>
</tr>
<tr>
<td>Settling Time</td>
<td>0.1s</td>
<td>0.2s</td>
</tr>
<tr>
<td>Steady State Error</td>
<td>±1°</td>
<td>+1.09°</td>
</tr>
<tr>
<td>Can track a sinusoid</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Controller Implementation and Goals

We are currently in the process of implementing our controllers on the actual system. We have been able to perform some open loop tosses and were successful in making a few catches. However, the PID controllers discussed above have not been implemented yet because the PID controller in LabVIEW requires a different set of gain parameters which are unfamiliar to us. We recently solved the problem and figured out the conversion factors between the three parameters used in LabVIEW to control the PID controller ($K_p$, $t_i$, $t_d$) and our PID gains ($K_p$, $K_i$, $K_d$):

- $K_p \rightarrow$ Proportional Gain
- $t_i = K_p/(60K_i) \rightarrow$ Proportional Gain / Integral gain
- $t_d = K_d/(60K_p) \rightarrow$ Derivative Gain / Proportional gain

With these conversion factors, we will be testing our designed PID controllers to the system in the near future.
We have, in the meantime, applied a PID controller by hand-tuning the LabVIEW parameters and conducted some 1-D and 2-D open loop tosses. Each test yielded different results. Currently the repeatability of the system is not very good, largely in part due to the difficulties in the net design. The ball tends to get shifted during launches to different angles, or get caught underneath the rim. Additionally, we have had problems with the mounting location of our system where the range of motion is limited by the mounting bars and nearby shelving. As of yesterday we built a mounting platform to solve this, which will enable us to dismount the system from its current location and achieve a wide range of motion without physical restrictions. These completed tasks will allow us to test the design of our controller more thoroughly.

Our future goal is to fine tune the controllers to achieve our design specifications and integrate vision feedback. We will also possibly investigate, if our current controllers do not provide us with satisfying results when implemented on the actual system, the possibility of using a Linear Quadratic Regulator based controller design. This will give us a better approach to pole placement and give us a balance between power and accuracy.
**Camera Development**

The camera development has been a parallel stage to Building the System, and Model Development, as laid out by our Plan of Action. The following are the steps and progress that have been achieved in this part of the project. The successful completion of the camera development will allow us to obtain feedback on the ball/s to continually calculate a trajectory prediction and determine the optimal position for the arm to be in to catch the ball, even if there is undershoot. At the same time, it will allow us to correct for environmental changes and errors in the control of the arm.

**Vision Module Familiarization**

To become familiar with LabVIEW Vision Development Module we have been using the NI Vision Assistant. This software allows us to capture images, 30 frames per second for our purpose, and create scripts for image processing that could then be translated into LabVIEW code. Preliminary data was acquired, which was later used to determine the ball position with respect to the image captured. This data is listed in the table below:

<table>
<thead>
<tr>
<th>Angle of vision</th>
<th>27.10°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height required for camera mounting (above system height)</td>
<td>97.99 cm</td>
</tr>
<tr>
<td>Image Resolution</td>
<td>640 X 480 pixels</td>
</tr>
<tr>
<td>Actual Image Length (AIL)</td>
<td>15.6 cm</td>
</tr>
<tr>
<td>Actual Image Height (AIH)</td>
<td>11.7 cm</td>
</tr>
<tr>
<td>AIL to pixel ratio</td>
<td>0.024375</td>
</tr>
<tr>
<td>AIH to pixel ratio</td>
<td>0.024375</td>
</tr>
</tbody>
</table>

**Figure 26 - Preliminary Vision Data**

A number of tests were carried out in order to establish the level of performance that we can achieve with the Vision Module. These experiments can be divided in two categories that will be called Test 1 and Test 2. **Test 1** consisted of projectile motion launches emulating the trajectory that the ball would follow given the actual juggling mechanism. These tests allowed us to, on one hand, observe the interaction of the ball with the camera and the clarity of the images, and on the other hand, it served as a trajectory for which an image processing script was written. **Test 2** consisted of launching the ball directly upward towards the overhead camera; with this experiment we obtained images of the ball at different heights in order to determine a pixel to ball size ratio. The way these tests were set up is described in more detail next.
Test 1:
- Hypothetical nets where set 1m apart
- Nets were 1.27m below the camera and .76m from the ground
- Ping pong ball was launched in a projectile motion from one net to the other capturing 30fps.

Test 2:
- Ping pong ball was launched vertically from 1.27m below the overhead camera
- 30fps were captured for the upward trajectory

**Image Processing & Performance**

After having performed the tests mentioned above, the frames captured were used to create image processing scripts that would give us information regarding the position of the ball in the X-Y and Z directions. The scripts developed deal primarily with image masking, thresholding, geometry recognition, and size measurements. They were created with the help of the Vision assistant which contains descriptions of the functionality of the different available blocks. Examples of the results obtained after running the scripts are shown below:

**Results from Test 1**
The images below show each one of the tasks performed by the script to ultimately identify the position in the X-Y coordinate of the ball.

1. Original Image
2. Image Masking
3. Color Threshold 1

4. Circle Detection

Figure 27 - Image Processing Results - Test 1 Image 16/30

Looking at image number 3 of Figure 27, we observe that the threshold is primarily on the white ping-pong ball. Also, there are a few spots other than the ball that are detected by this threshold, but they are not large enough to corrupt the data; the next part of the script, Circle Detection, is set up to ignore sizes of circles that we will not see given our radius specifications. This image shows detection of 1 circle and the position data in pixels is given below:

<table>
<thead>
<tr>
<th>Circle #</th>
<th>Center X</th>
<th>Center Y</th>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>534</td>
<td>306</td>
<td>34</td>
</tr>
</tbody>
</table>

Figure 28 - Circle Detected Data (Pixels)

Results from Test 2

The two images below show the significant change in the size of the ball as it approaches the camera from an upward toss. Since the change is major, we can predict how high the ball will be at different points in the trajectory. To do this we take the lowest or highest position of the ball data as a reference, then the height could be found as:

$$\text{ActualHeight} = \frac{\text{ReferenceHeight}}{\text{ReferenceBallRadius (pixels)}} \times \text{ActualBallRadius (pixels)}$$
Processing Challenges

Challenges that we have encountered involve primarily problems with properly thresholding certain images due to two factors: blurred images of the ball, and certain colored backgrounds. The first factor causes a problem because as the ball travels in the X-Y direction, there will be images where the ball looks as if it has a tail. When thresholding takes place in this situation, a large area is detected, and when the circle detection occurs right after, several circles are found in such area. An example of this case is shown below. In this example 16 circles were detected; a table with each circle data is also shown.

![Blurred Image](image)

![Circles Detected Data](image)

<table>
<thead>
<tr>
<th>Circle #</th>
<th>Center X</th>
<th>Center Y</th>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>546</td>
<td>162</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>531</td>
<td>167</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>518</td>
<td>177</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>527</td>
<td>198</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>528</td>
<td>200</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>530</td>
<td>204</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>515</td>
<td>218</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>487</td>
<td>222</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>505</td>
<td>227</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>334</td>
<td>339</td>
<td>37</td>
</tr>
<tr>
<td>11</td>
<td>314</td>
<td>341</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>299</td>
<td>344</td>
<td>37</td>
</tr>
<tr>
<td>13</td>
<td>287</td>
<td>347</td>
<td>38</td>
</tr>
<tr>
<td>14</td>
<td>276</td>
<td>349</td>
<td>39</td>
</tr>
<tr>
<td>15</td>
<td>272</td>
<td>350</td>
<td>39</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>353</td>
<td>40</td>
</tr>
</tbody>
</table>
To deal with this problem we are considering taking the average of the data of all circles and using that as our best approximation of the ball location.

The second factor that causes trouble with thresholds happens when there is something in the background that is a similar color as the ball. Here, it becomes difficult to prevent this extra area of color from being detected by the threshold in the script. An example of this case is shown below:

In this figure, the area circled in red in the threshold image on the right belongs to the stool on the right hand side of the image on the left. To deal with this issue we will create a uniform dark background for our juggling system. This will allow us to more accurately detect the position of the ball in the X-Y plane.

**Data Verification**

Although the data we have obtained from the vision scripts would allow us to predict the ball’s position in the X-Y and Z directions at any frame in a given trajectory, we still need to verify whether these results are valid, that is, whether or not the position predicted by the script is the actual position of the ball in the system’s workspace. In order to verify this, an additional test will be performed; in this experiment we will be using a heavier ball in order to decrease the effects of drag; the ball will travel downwards due to a free fall motion that will initiate right below the camera lens. We will have a stopwatch somewhere inside the field of vision of the camera to keep track of time. Knowing that in this situation the trajectory will be governed by:

\[ Y = \frac{1}{2}at^2, \]

we will be able to calculate the height of the ball at different times, and then compare these results with those obtained from the Vision script. If the result of this test shows that using the overhead camera we can obtain accurate data for the height of the ball, then we will base our
position feedback on this camera only. However, if the data appears to be unreliable, we will then consider utilizing a second camera that would be placed on one side of the system, and it would be exclusively used to obtain position of the ball in the Z direction.

**Next Steps**

After the validity of position data is resolved, the next step of the process according to our plan of action and schedule is to achieve a Running Trajectory Prediction, as well as integration of the Vision development with the Control System of the Juggling Mechanism. Along with this we will continue to validate data acquired from the different processes and experiments.
Summary of Progress

Compared to our original proposal, we are fairly close in terms of scheduling, costs, and plan of action. In terms of the schedule, we have completed most of the items to date – we are only behind on a few items and those are well on their way to completion. We are about 10% over budget and 20% below our initial estimated costs – we did not need to purchase the motor we originally thought would be necessary, but the construction of support items (system mount, camera mount, cable extensions) have taken those funds. Our plan of action has been followed faithfully to this point, and we do not foresee deviating from it much, if at all.

There have been a few unexpected challenges that have cropped up so far. The nets on the arm have been a source of inconsistency in our system in ways we did not predict. We are working to modify them to solve this problem and we are sure we can overcome this problem. Much of what we had to build to support the system and the camera were not originally planned, but were required due to the existing workspace. However, our solutions to these have been very workable and have not affected our schedule or budget overly much.

With the bulk of the design and build process behind us, we do not currently foresee any significant additional issues that were not predicted in the initial proposal and are very confident that we will be able to achieve our goal of building a two-ball juggling system.
Statement of Contribution

This statement affirms that all three members of this team participated actively in putting together this progress report. Each member contributed individually to information included here. John Kua wrote the Introduction, Model Development and Summary of Progress sections, Linda Rivera was responsible for the Physical Design and the Camera Development sections, and Trinell Ball authored the Control Design Development section.

_________________________________
John Kua

__________________________________
Trinell Ball

__________________________________
Linda Rivera