8-2011

Dynamics and Control of the Shoot-the-Moon Tabletop Game

Peng Xu
Clemson University, pxu@clemson.edu

Dynamics and Control of the Shoot-the-Moon Tabletop Game

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Peng Xu
August 2011

Accepted by:
Dr. Richard E. Groff, Committee Chair
Dr. Timothy Burg, Co-Chair
Dr. Darren Dawson

Abstract

The classic table-top game Shoot-the-Moon has interesting dynamics despite its simple structure, consisting of a steel ball rolling on two cylindrical rods. The two sloped rods are hinged at the lower ends and allowed to freely slide in a slot at the higher end. The ball can amazingly roll upward along the rods under carefully manipulation of the rods. There is also an interaction between ball rotation and translation that cause the ball to "shoot" (quickly accelerate).

In this thesis, the kinematics are developed for Shoot-the-Moon and then equations of motion are derived using both Lagrangian and Newtonian approaches. The modeling work yields an examine the underactuated, nonlinear, nonholonomic dynamic model for Shoot-the-Moon.

Two controllers are designed based on the dynamic model. The Linearized Position Regulator is developed using a local linearization at an equilibrium point of the dynamics. The Position Tracking Controller takes nonlinearities into account by inverting the significant nonlinear terms in the dynamics so that the system appears linear at the input and can be controlled using a PD controller. Simulations of both controllers are performed, showing that the ball converges to the setpoint for the linearized controller and continuous signals can be tracked by the nonlinear controller.

An experimental platform, an automated Shoot-the-Moon game controlled using the Position Tracking Controller, is built to facilitate understanding of the

dynamics, explore the nonholonomic property of the system and demonstrate efficacy of the proposed controllers.

Experiment results are presented showing the effectiveness of the controller on the physical system. The results are compared with simulations under same conditions in order to highlight the fidelity of the dynamic model. The effect of the nonholonomic constraint relating the ball's linear and angular position is also demonstrated.

Shoot-the-Moon is a familiar system with rich dynamics. Moreover, it is one of the simplest system that shows nonholonomic properties and has substantial nonlinearity in dynamics. As such, it can provide an appealing challenge problem for control design techniques and serve as a new educational tool.

Dedication

This thesis is dedicated to my parents, who provide me with their best, give me maximum reasonable freedom through my growth, and encourage me during my hard times.

Acknowledgments

First, I would like to express my sincere gratitude to my co-advisor Dr. Richard Groff and Dr. Timothy Burg for providing me with the opportunity of participating in this research and their motivation and patience throughout this project. Their always available guidance helped me in all the time of research and writing of this thesis.

Besides my advisors, I would like to thank my committee member Dr. Darren Dawson, for his encouragement and insightful comments.

I would also like to give my thanks my labmates, Ninad Pradhan, Appoorva Kapadia, Darshana Salvi, Pallavi Srikanth for the stimulating discussions and encouragement through their interests in this project.

Lastly, and most importantly, I wish to thank my parents, Jianxin Xu and Yongqing Guo, for their support and love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Shoot-the-Moon game (see Fig. 1.1) may give people the illusion of breaking the laws of physics by rolling a ball "uphill" on two rods. The secret is that the interaction between the shape of the ball and the angle between the two rods forms an extra slope, which, combined with the apparent, visible slope of the rods, create a virtual slope for the ball. The ball moves down the virtual slope, which, depending on ball location and rod separation, may be in the opposite direction from the visible slope. Moreover, the virtual slope can be reshaped by manipulating the rods, which is equivalent to changing the force acting on the ball. This can be used to control ball position and velocity.

This game is similar to the "ball-and-beam", another famous control and dynamics plant widely used in education [1], yet is more akin to "rolling disk" or "unicycle" problem [2] in that they all have rolling constraint which leads to nonholonomic system dynamics.

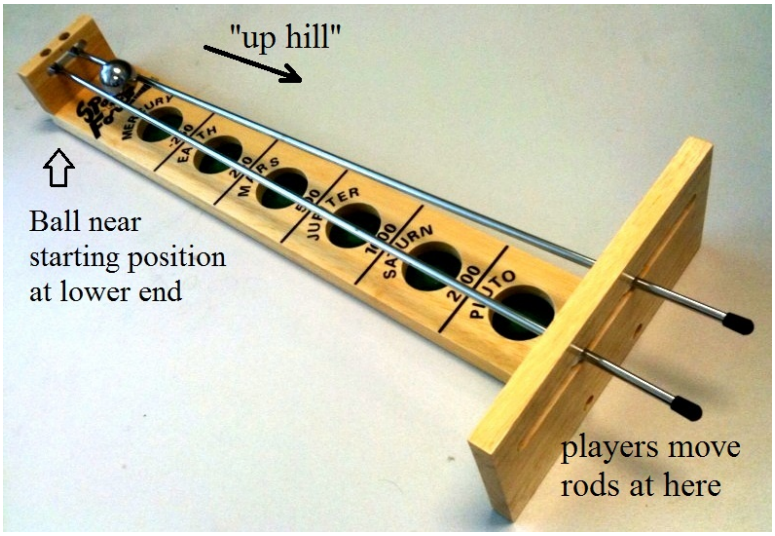

Figure 1.1: Shoot-the-Moon Game Board

1.1 The Shoot-the-Moon Game

Shoot-the-Moon game is a classic wooden board game addictive to both adults and children. It belongs to a category of skill games, for which performance is primarily related to the mental or physical skill of the player instead of only chance. In this particular case, a player's fine motor skill and familiarity with the dynamic behavior of the game are critical for achieving higher scores. The exact history of its creation cannot be found, however multiple sources suggest that is has been popular since the 1940s.

The design of the Shoot-the-Moon game board varies from manufacturer to manufacturer; however, the basic structure is the same. A model from Wood Expressions Inc is shown in Fig. 1.1. Two round and smooth rods are mounted above, but not parallel to, a rectangular game board base. The lower ends of the rods are hinged vertically by screws or other mechanisms. The high ends can slide freely in a horizontal slot or support. The space between the two rods at the pinned ends is smaller than the diameter of the polished steel game ball. The ball could be placed

at any point on these two rods if they are held parallel or slightly separated.

At the beginning of play, the ball is placed at pinned ends of the rods. Players adjust the width between two rods at the free-moving ends in order to move the ball up the slope while avoiding prematurely dropping the ball. Performance is gauged by dropping the ball in one of a series of holes, with farther holes assigned higher scores.

Although the theory of play is simple, few people manage to get high scores without practice. It is partially because human hands lack the ability to actuate objects precisely and quickly and the ball will drop as a result, especially when the ball is closer to the higher end, where the margin for error is very narrow. Another source of difficulty comes from the nonlinear dynamics of the ball, which may confound the expectations of an untrained player.

A trick of playing this game is a dynamic strategy: initially open the rods wide to let the ball accelerate fast; then quickly close the rods so that energy stored in the ball rotation is transfered to linear movement (the ball will suddenly obtain a high velocity); keep the rod close until the ball reaches the higher end; and then drop at the desired location. In this way, kinetic energy is imparted to the ball in the early stage, where the margin for error is still wide, and the user can "play it safe" afterwards.

1.2 Previous Work

Shoot-the-Moon, despite its popularity and essence of a game involving dynamics, is seldom mentioned in the control and modeling literature. Neither dynamics nor control algorithms for the Shoot-the-Moon game appear to have been published. In [3], the Shoot-the-Moon game is mentioned for its potential in dynamics education, yet no specific dynamic model or controller design is presented. In our previous

conference paper [4], the Shoot-the-Moon dynamic model is derived and some simulation results of controllers are shown. This thesis will describbe in more detail both simulation and physical experiment results as well as the experimental platform.

The Lagrangian and Newtonian approaches used for deriving rigid body dynamic model are standard and well described in textbooks focusing on classical dynamics [5, 6, 7]. The Newtonian method is natural and focuses on effects of forces on objects with inertia but does not easily scale up for systems with many rigid bodies interacting with each other. The Lagrangian approach avoids internal force analysis by only considering total energy of the whole system and thus is suitable for deriving dynamics of systems with interactions of multiple objects. Moreover, the Lagrangian approach is also more systematic and is thus more amenable to using computer symbolic algebra software to perform the dynamics derivation. However, Shoot-the-Moon is a simple system and both methods can be used. Notice the system has a nonholonomic rolling constraint and thus a Lagrangian multiplier should be introduced in the Lagrangian approach to represent the force that relate the linear and angular movement of the ball. System linearization and state-space linear system analysis techniques used in the controller development chapter are well illustrated in linear control books such as [8, 9].

Methods in computer vision are utilized in the experimetal platform for ball linear and angular position sensor development. Camera calibration algorithms [11, 12, 13] greatly simplify the measurement of the intrinsic and extrinsic parameters of the camera. Extrinsic parameters are the 3D transformation between world frame and a frame attached to the camera and the intrinsic parameters describe how to project 3D points in the camera frame onto the image plane. These parameters are necessary for establishing the conversion from 2D image coordinates to ball linear position. Principle component analysis (PCA) is a data dimension reducing tool based

on statistics [14, 15]. PCA can be used to extract samples distribution information in the state space. In ball angle sensing, pixels representing a slender color marker attached on the ball are treated as samples. The pixel locations are fit into 2D gaussian and parameters of the gaussian is extracted with PCA to determine the angle of the marker.

1.3 This Work

In this work, the dynamic model of the Shoot-the-Moon game is derived using both Lagrangian and Newtonian method under realistic assumptions about the physical property of the game board components in Chapter 2.

Two controllers are designed based on the dynamic model in Chapter 3. The Linearized Position Regulator is developed using a local linearization at an equilibrium point of the dynamics. The Position Tracking Controller takes nonlinearities into account by inverting the significant nonlinear terms in the dynamics so that the system appears linear at the input and can be controlled using a PD controller. Simulations of both controllers are performed, showing that the ball converges to the setpoint for the linearized controller and continuous signals can be tracked by the nonlinear controller.

Chapter 4 shows that an experimental platform, an automated Shoot-the-Moon game controlled using the Position Tracking Controller, is built to facilitate understanding of the dynamics, explore the nonholonomic property of the system and demonstrate efficacy of the proposed controllers. Experiment results are presented showing the effectiveness of the controller on the physical system. The results are compared with the simulations under same conditions to highlight the fidelity of the dynamic model. The effect of the nonholonomic constraint relating the ball's linear

and angular position is also demonstrated via a sinusoidal reference trajectory.

The final conclusions are found in Chapter 5.

1.4 Impact of this Research

Shoot-the-Moon is a tabletop game that appeals to a broad audience. This thesis contains the Shoot-the-Moon nonholonomic dynamic model derivation and controller design which could greatly benefit control system education and nonholonomic control research.

The experimental platform is simple and can be constructed at low cost. This makes it an ideal example in classroom presentation that stimulate students' interest about dynamic systems. It can also be used in a laboratory for students to experiment with various control techniques and compare their difference.

The dynamic model of the game is also useful as a standard plant for nonholonomic control research and can play a similar role to the rolling disk, unicycle, or car problems. Since a system requires at least two-degree-of freedom to have a nonholonomic constraint, Shoot-the-Moon is one of the simplest nonholonomic systems possible. Moreover, unlike other example nonholonomic systems currently used, the dynamics of Shoot-the-Moon model is essential for nonholonomic controller design since it is an underactuated system with one input and two generalized coordinates. The nonholonomic controller must account for system dynamics in order to control linear and angular position of the ball.

Chapter 2

Model Development

The Shoot-the-Moon dynamic model reveals the source of its counterintuitive behavior and sets the basis for development of a controller. In this chapter, notation and a few modeling assumptions are presented, the kinematics of the game is investigated, and then the dynamic model is derived with both Lagrangian and Newtonian approaches. The results from both methods are identical.

2.1 Preliminaries and Notation

The kinematics of the Shoot-the-Moon game is more complicated than a general robotics system in that the position of the ball is determined by the contact points formed by the ball and the rods, instead of linear or revolution joints. The existing solutions for resolving robotics kinematics, such as the Denavit-Hartenberg method, are not very useful in this specific case. It is necessary to use fundamental geometric relations to describe the kinematics.

A rod coordinate system, noted as Ω, is aligned with the game board to simplify the notation (see Fig. 2.1). Its origin, O, is located at the center of the pinned ends of

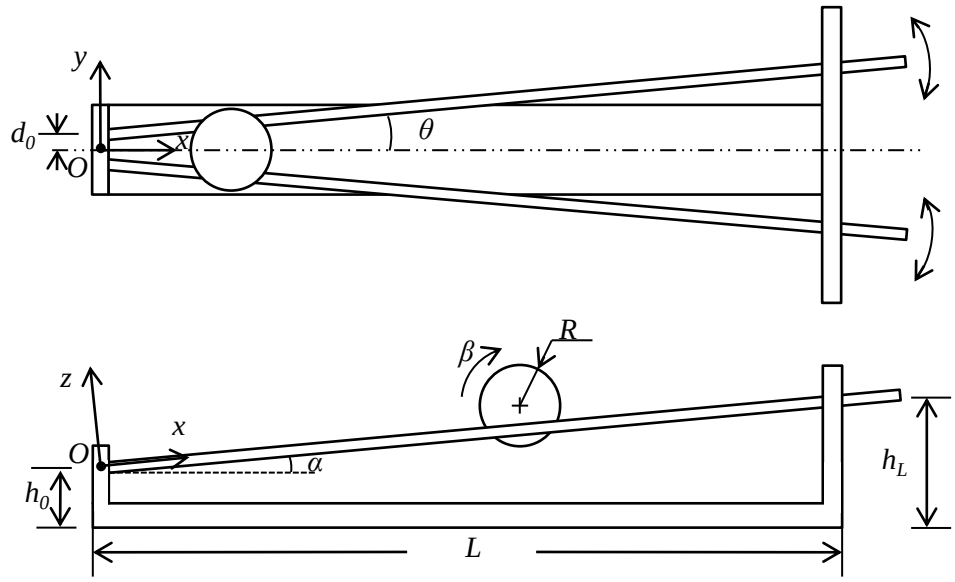

Figure 2.1: Top view and front view of Shoot-the-moon game with rod coordinate system Ω. In the front view, the illustration of the ball needing to roll "uphill" to reach the scoring zone is apparent.

the two rods; the x-axis is parallel to the neutral position of the rods ($\theta = 0$); the x- and y-axis form an inclined plane that contains the motion of the rods; and the z-axis completes the right-handed coordinate system. Since the game board is stationary during play, the coordinate system Ω is used as an inertial frame in the derivation of the dynamic model. All notation is listed in Table. 2.1.

2.2 Assumptions

The derivation of the kinematics and dynamic model is based on the following assumptions:

1. The rods and ball are rigid.

2. The ball is always on the rods, i.e. $z > 0$ and the normal force $F_n \geq 0$. Note that for $z \leq 0$, the rods do not support the ball and for $F_n < 0$ the motion of

Table 2.1: List of notation.

Notation	Description
x, y, z	The position of the ball's center of gravity, C, in rod frame Ω.
β	The rotation angle of the ball about the y-axis. Since the ball is symmetric, it is defined that $\beta = \beta_0$ at $t = 0$, where $\beta_0 \in \mathbb{R}$ can be any value.
h	The height of the ball's center of gravity with O as reference point.
θ	The angle between the rods and the x-axis.
d_0	The distance between the fixed point of the rods, D, and the origin, O.
L	The length of the game board from the lower end to the higher end, measured in the horizontal plane.
h_0	The height of point O from the base of the game board.
h_L	The height measured from the base of the game board to the slot in which the rods slide.
α	The constant angle between the x-axis and the horizontal plane. With geometry, $\alpha = \arctan \frac{h_L - h_0}{L}$.
R	The radius of the ball.
r	The radius of the cylindrical rods.
m	The mass of the ball.
J	The moment of inertia of the ball about a rotation axis through its center.

the ball follows a ballistic trajectory that leaves to the rods. Thus, these two conditions are left out of discussion.

3. The friction between the ball and the rods is sufficient to prevent the ball from sliding along the rods, but the rods are also able to separate and close freely under the external input force.

4. The game is played by moving the rods symmetrically with respect to the vertical plane formed by the x-axis and z-axis, so the ball will always have its center of gravity C in the x-z plane ($y \equiv 0$). A single variable θ is used to describe the angle of each rod relative to the neutral position where the rods are parallel, and the direction that widens the gap between the two is defined positive.

5. Rod angle θ is treated as the system input, *i.e.* forces are applied to the rods to track any $\theta(t) \in \mathcal{C}^2$.

These realistic assumptions rule out unnecessary minutiae from the Shoot-the-Moon system model without changing the essence of the dynamics.

2.3 Kinematics

This section will show the derivation on the kinematic relationship between the angle of the rods θ and the position and attitude of ball. With assumption 1, the z coordinate of the ball is constrained by the x coordinate and θ and will be derived using the geometric relationships illustrated in Fig. 2.1 and Fig. 2.2.

Let C be the center of the ball and let B be the point on the center line l of the rod such that $\overline{CB} \perp l$. Let C_z be the projection of C onto the $x-y$ plane, and let B_2 be the point on l such that $\overline{C_z B_2}$ is parallel to the y axis. Note that point T, the

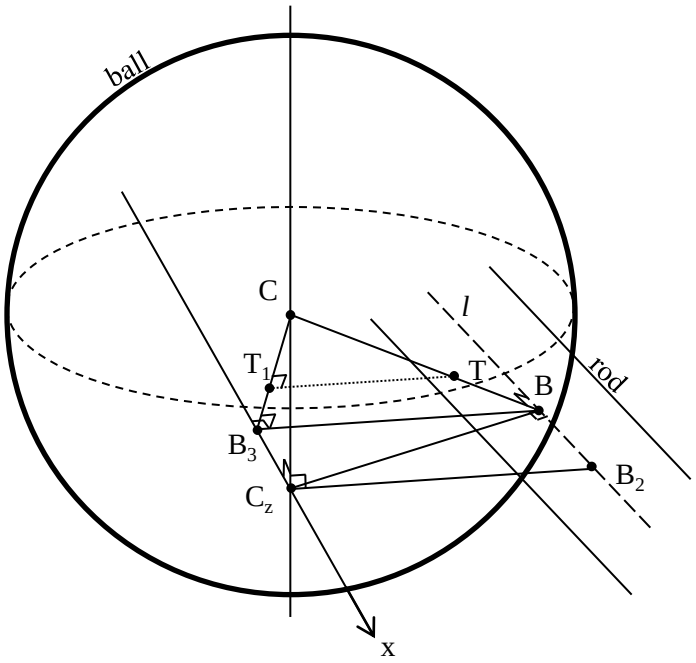

Figure 2.2: A 3D view of the ball and rod. Note that T is the point of contact of the ball with the rod.

tangent point between the ball and the rod, lies on $\overline{CB}$. Let B_3 be the point on the x-axis such that BB_3 is parallel to the y-axis, and let T_1 be the projection of T onto CB_3.

The z coordinate of the ball is calculated from the right triangles $\triangle CC_zB$ and $\triangle B_2C_zB$ (see Fig. 2.3)

$$\begin{aligned} z &= \left|\overline{CC_z}\right| = \sqrt{\left|\overline{CB}\right|^2 - \left|\overline{C_zB}\right|^2} \\ &= \sqrt{(R+r)^2 - (\cos\theta \cdot \left|\overline{C_zB_2}\right|)^2} \\ &= \sqrt{(R+r)^2 - \cos^2\theta \cdot (d_0 + x \cdot \tan\theta)^2} \\ &= f_z\left(x, \theta\right). \end{aligned} \tag{2.1}$$

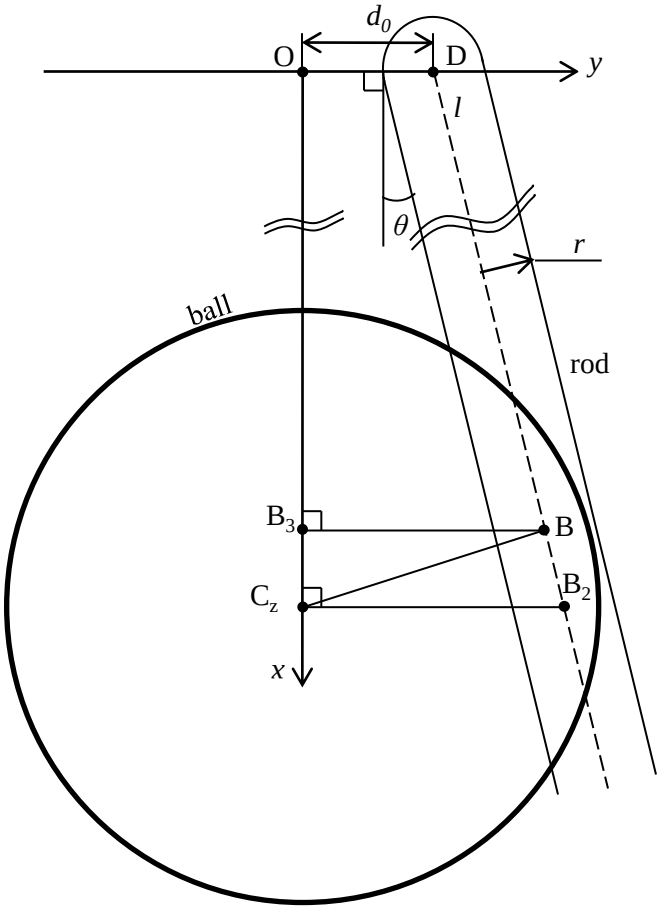

Figure 2.3: A top view of the ball and rod.

The height of the center of the ball with respect to the origin point of Ω is

$$h = \cos\alpha \cdot z + \sin\alpha \cdot x.$$

The rolling constraint of the ball determines the ratio of linear to angular velocity, $\dot{x}$ and $\dot{\beta}$ respectively, based on the geometry. This ratio is defined as the effective radius,

$$R_{\mathit{eff}} \triangleq \frac{\dot{x}}{\dot{\beta}}. \tag{2.2}$$

R_{eff} is solved by considering an infinitesimal rotation, $\Delta\beta$, of the ball (see Fig. 2.4). The change in x is Δx. The center of the ball moves from C to C' along an arc with a length of Δd. The radius of the arc is called the equivalent radius $R_{eq} = \left|\overline{CT_1}\right|$, which is the projection of $\overline{CT}$ on the x-z plane (see Fig. 2.2). Thus, from the geometry we have

$$\Delta\beta \cdot \left|\overline{CT_1}\right| = \Delta d, \tag{2.3}$$

and from the definition of R_{eff}, we have

$$\Delta\beta \cdot R_{\mathit{eff}} = \Delta x. \tag{2.4}$$

By the similarity between $\triangle CB_3B$ and $\triangle CT_1T$,

$$\frac{R_{eq}}{\left|\overline{CB_3}\right|} = \frac{\left|\overline{CT}\right|}{\left|\overline{CB}\right|} = \frac{R}{R+r}.$$

And from the similarity between $\triangle CB_3C_z$ and $\triangle C'CC_2$,

$$\frac{\Delta x}{\Delta d} = \frac{\left|\overline{CC_z}\right|}{\left|\overline{CB_3}\right|}.$$

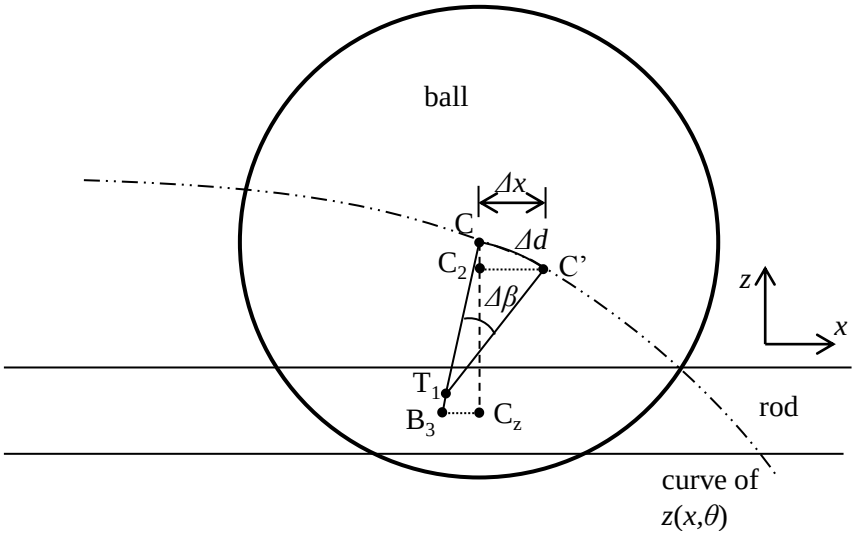

Figure 2.4: An infinitesimal movement of the ball.

Thus,

$$R_{\mathit{eff}} = \left|\overline{CB_3}\right| \cdot \frac{R}{R+r} \cdot \frac{\left|\overline{CC_z}\right|}{\left|\overline{CB_3}\right|} = \frac{R \cdot z}{R+r}. \tag{2.5}$$

Note that R_{eff} is not constant, but varies with z, and hence x and θ. From Eqn. (2.2), we have, the expression for $\dot{\beta}$,

$$\dot{\beta} = \frac{\dot{x}}{R_{\mathit{eff}}}; \tag{2.6}$$

$\ddot{\beta}$ is found by taking the time derivative of $\dot{\beta}$,

$$\ddot{\beta} = \frac{d}{dt}\left(\frac{\dot{x}}{R_{\mathit{eff}}}\right) = \frac{\ddot{x} \cdot R_{\mathit{eff}} - \dot{x}\frac{R}{R+r}\dot{z}}{R_{\mathit{eff}}^2}. \tag{2.7}$$

Further information can be derived from the kinematics stated above. Since $\frac{\partial h}{\partial x} = \cos\alpha \cdot \frac{\partial z}{\partial x} + \sin\alpha$, $\frac{\partial z}{\partial x}$ is the "extra" slope created by the two rods, calculated by

taking the partial derivative of (2.1),

$$\frac{\partial z}{\partial x} = -\frac{\cos\theta \cdot \sin\theta \cdot (d_0 + x \cdot \tan\theta)}{z}. \tag{2.8}$$

Similarly, $\dot{z}$ and $\ddot{z}$ are defined as,

$$\dot{z} = \dot{x}\frac{\partial z}{\partial x} + \dot{\theta}\frac{\partial z}{\partial \theta}, \tag{2.9}$$

$$\ddot{z} = \ddot{\theta}\frac{\partial z}{\partial \theta} + \ddot{x}\frac{\partial z}{\partial x} + \dot{x}^2\frac{\partial^2 z}{\partial x^2} + 2\dot{\theta}\dot{x}\frac{\partial^2 z}{\partial x \partial \theta} + \dot{\theta}^2\frac{\partial^2 z}{\partial \theta^2}. \tag{2.10}$$

2.4 Dynamic Model by the Lagrangian Approach

The kinematics analysis provides several constraints on movement of the ball.

- Position Constraints: There are two positions constraints on the ball movement. Equation (2.1) imposes a constraint on z, and from assumption 4, $y = 0$.

- Rolling Constraints: There are two contraints on the rolling motion. First, the game is symmetric about the x-z plane, thus the ball will only rotate about the y-axis, the angle is denoted by β. Second, Eqn. (2.6) is a constraint relating $\dot{x}$ and $\dot{\beta}$. This constraint cannot be reduced to a constraint on positions alone, *i.e.* it cannot be rewritten in the form $f(x, \beta, t) = 0$, and thus it is a nonholonomic constraint.

An unconstrained ball has 6 degrees-of-freedom from translational and rotational movement. Subtracting the 2 constraints on linear position and 2 constraints on angular position, the ball has 2 remaining degree-of-freedom. Thus, two generalized coordinates, x and β, are required in the Lagrangian dynamics.

The Lagrangian L is given by

$$L = T - V, \tag{2.11}$$

where

$$T = \frac{1}{2}m\left(\dot{x}^2 + \dot{z}^2\right) + \frac{1}{2}J\dot{\beta}^2 \tag{2.12}$$

is the kinetic energy and

$$V = mgh = mg\left(\cos\alpha \cdot z + \sin\alpha \cdot x\right) \tag{2.13}$$

is the potential energy.

Because of the nonholonomic rolling constraint in Eqn. (2.2), a Lagrange multiplier λ is introduced to represent the unknown forces ensuring the constraint. From Eqn. (2.6), the coefficients of λ in the Lagrangian equation are

$$a_\beta = R_{\mathit{eff}}$$
$$a_x = -1.$$

With the nonholonomic constraint taken into account, the Lagrangian equations are:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} - \quad 1 \cdot \lambda = 0, \tag{2.14}$$
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\beta}}\right) - \frac{\partial L}{\partial \beta} + R_{\mathit{eff}} \cdot \lambda = 0. \tag{2.15}$$

Solving for $\ddot{x}$ from these two equations, eliminating λ in the result, and then substituting in $\dot{\beta}$ and $\ddot{\beta}$ from Eqn. (2.6) and (2.7) yields the dynamics of x. Together with

dynamics of β stated in (2.6), the complete dynamics of the ball are

$$\boxed{\begin{aligned}\ddot{x} &= \frac{-mg\sin\alpha + P - m\frac{\partial z}{\partial x}(g\cos\alpha + Q)}{I} \\ &= f_1\left(x, \dot{x}, \theta, \dot{\theta}, \ddot{\theta}\right), \\ \dot{\beta} &= \frac{\dot{x}R_r}{z},\end{aligned}} \tag{2.16}$$

where

$$\begin{aligned}R_r &= \frac{R+r}{R}, \\ z &= \sqrt{(R+r)^2 - \cos^2\theta\cdot(d_0 + x\cdot\tan\theta)^2}, \\ P &= JR_r^2\dot{x}\dot{z}z^{-3}, \\ Q &= \ddot{\theta}\frac{\partial z}{\partial\theta} + \dot{\theta}^2\frac{\partial^2 z}{\partial\theta^2} + 2\dot{x}\dot{\theta}\frac{\partial^2 z}{\partial x\partial\theta} + \dot{x}^2\frac{\partial^2 z}{\partial x^2} = \ddot{z} - \ddot{x}\frac{\partial z}{\partial x}, \\ I &= JR_r^2 z^{-2} + m\left[1 + \left(\frac{\partial z}{\partial x}\right)^2\right].\end{aligned}$$

2.5 Dynamic Model by the Newtonian Approach

This section shows the Newtonian solution for the dynamics of the Shoot-the-Moon. While the Lagrangian approach provides the simplicity of not explicitly dealing with the force interaction between rods and ball at the contact point, the Newtonian method offers an opportunity to find out the physical explanation of each term in the dynamics equation. Deriving identical dynamics using two distinct approaches also helps validate the derivations.

Force analysis of the ball is shown in Fig. 2.5, where F_n denotes the supporting force normal to ball linear velocity v, F_f is the sum of friction force on both rods

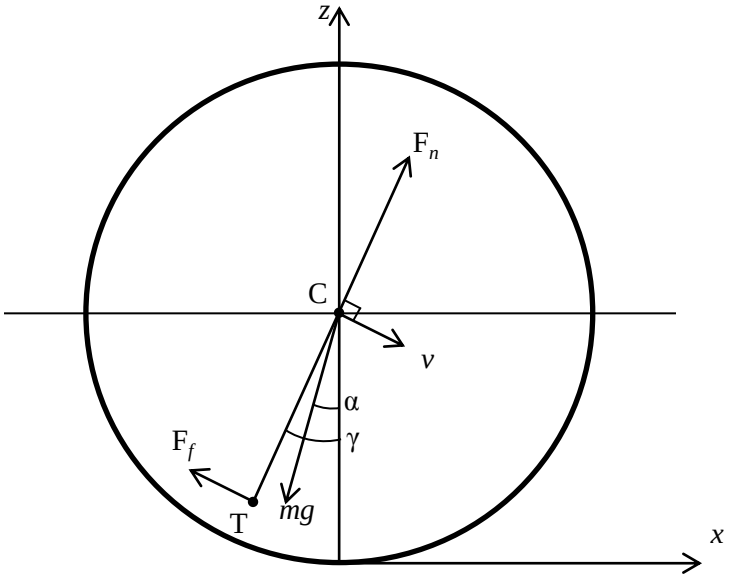

Figure 2.5: Free body diagram of the ball.

projecting onto the x-z plane, mg is the gravitational force, and γ is the angle from the F_n vector to the $+z$ vector.

Applying Newton's law on the ball along the x- and z-axes yields

$$-\sin\alpha \cdot mg + \sin(-\gamma) F_n - \cos\gamma F_f = m\ddot{x}, \tag{2.17}$$

$$-\cos\alpha \cdot mg + \cos\gamma \cdot F_n + \sin(-\gamma) F_f = m\ddot{z}. \tag{2.18}$$

Summing up the moments at the center of gravity of the ball, there is

$$J\ddot{\beta} = R_{eq} F_f, \tag{2.19}$$

where $R_{eq} = \frac{R_{eff}}{\cos\gamma}$. Thus,

$$F_f = \frac{J\ddot{\beta}}{R_{eq}} = \frac{J\cos\gamma\ddot{\beta}}{R_{eff}}. \tag{2.20}$$

Substitute $\ddot{\beta}$ from (2.7),

$$F_f = \frac{J\cos\gamma\,(R_{eff}\ddot{x} - R_r^{-1}\dot{x}\dot{z})}{R_{eff}^3}. \tag{2.21}$$

From the geometry shown in Fig. 2.5 and Fig. 2.4, it is clear that

$$\tan\gamma = \frac{\left|\overline{C_zB_3}\right|}{\left|\overline{CC_z}\right|} = -\frac{\cos\theta\cdot\sin\theta\,(d_0 + x\tan\theta)}{z}. \tag{2.22}$$

Comparing to (2.8), there is

$$\tan\gamma = \frac{\partial z}{\partial x}. \tag{2.23}$$

Eliminating F_n by calculating (2.17)+ $\tan\gamma\cdot$(2.18), then substituting (2.21) into the result and rearranging terms yields

$$m\ddot{x} = -mg\sin\alpha - \frac{J\,(\ddot{x}R_{eff} - R_r^{-1}\dot{x}\dot{z})}{R_{eff}^3} - m\tan\gamma(\ddot{z} + mg\cos\alpha). \tag{2.24}$$

Substituting (2.23) and (2.10) into (2.24), and then collecting all terms with $\ddot{x}$ to the left side of the equation produces

$$I\ddot{x} = -mg\sin\alpha + JR_r^2 z^{-3}\dot{x}\dot{z} - m\frac{\partial z}{\partial x}\left(g\cos\alpha + Q\right), \tag{2.25}$$

which is identical to the solution from the Lagrangian approach in (2.16).

Chapter 3

Controller Design

Two controllers for the ball's linear position x are proposed in this chapter. The Linearized Position Regulator is designed to regulate the position of the ball locally based on system linearization of the system at an equilibrium point. The Position Tracking Controller inverts the nonlinearities in the dynamics and achieves tracking control of the position of the ball.

3.1 Linearized Position Regulator

Rewriting the dynamic equation (2.16) in state space form, with state $W = [w_1, w_2]^\mathsf{T} = [x, \dot{x}]^\mathsf{T}$ and input θ, produces

$$\dot{W} = \begin{bmatrix} w_2 \\ f_1\left(x, \dot{x}, \theta, \dot{\theta}, \ddot{\theta}\right) \end{bmatrix}, \tag{3.1}$$

where $f_1\left(x, \dot{x}, \theta, \dot{\theta}, \ddot{\theta}\right)$ is given in Eqn. (2.16). Linearizing the system at an equilibrium point

$$W = [x_e, 0]^\mathsf{T} = W_e \text{ and } \theta = \theta_e \tag{3.2}$$

where, x_e is any possible x coordinate of the ball and θ_e satisfies

$$f_1\left(x_e, 0, \theta_e, 0, 0\right) = 0.$$

After substituting the equilibrium point into f_1, the equation simplifies to

$$\sin\alpha + \cos\alpha \cdot \frac{\partial z}{\partial x}\left(x_e, \theta_e\right) = 0, \tag{3.3}$$

which is equivalent to a horizontal virtual slope at x_e, $\frac{\partial h}{\partial x}\left(x_e, \theta_e\right) = 0$. Given x_e, it is possible to find θ_e by solving Eqn. (3.3).

The linearized system is defined with state $\delta W = W - W_e$ and input $\delta\theta = \theta - \theta_e$, with state equation,

$$\delta\dot{W} = A \cdot \delta W + B \cdot \delta\theta,$$

where,

$$A = \begin{bmatrix} 0 & 1 \\ \frac{\partial f_1}{\partial w_1} & \frac{\partial f_1}{\partial w_2} \end{bmatrix}, B = \begin{bmatrix} 0 \\ \frac{\partial f_1}{\partial \theta} \end{bmatrix}.$$

Evaluating A and B at the equilibrium point stated in (3.2), we have,

$$A = \begin{bmatrix} 0 & 1 \\ a_{21} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ b_2 \end{bmatrix}, \tag{3.4}$$

where,

$$a_{21} = -\frac{mg\cos\alpha \cdot \frac{\partial^2 z}{\partial x^2}\left(x_e, \theta_e\right)}{I}, \tag{3.5}$$

$$b_2 = -\frac{mg\cos\alpha \cdot \frac{\partial^2 z}{\partial x \partial\theta}\left(x_e, \theta_e\right)}{I}. \tag{3.6}$$

The controllability matrix $M = \begin{bmatrix} B & AB \end{bmatrix}$ indicates the system is controllable because $\text{rank}(M) = 2$. Common linear controller design techniques can be applied directly on the linearized system. Pole placement is used to design a full state feedback control for the linearized system. The poles are placed at p_1 and p_2 for δw_1 and δw_2 respectively. Therefore, the control law will be $\delta\theta = -K \cdot \delta W$. Solving equation $\Delta_{A-BK}(s) = (s-p_1)(s-p_2)$ for the control gain $K = \begin{bmatrix} k_1 & k_2 \end{bmatrix}$, we have,

$$K = \begin{bmatrix} \dfrac{p_1 p_2 + a_{21}}{b_2} & -\dfrac{p_1 + p_2}{b_2} \end{bmatrix}. \tag{3.7}$$

Since the system is linearized at a certain pair of (x_e, θ_e), it only works well when $x \approx x_e$ and $\theta \approx \theta_e$. In the other words, when x is too far from x_e for the linearization, the control gain obtained from the pole placement calculation no longer yields the expected close-loop system response. Naturally, gain scheduling can be implemented to compensate by actively re-linearizing the system in runtime and evaluating the θ_e, a_{21}, and b_2 for the x at that time.

3.2 Augmented Linearized System

If the states of the system are augmented by adding in the ball rotation angle β. The states becomes $W = [w_1, w_2, w_3]^\intercal = [x, \dot{x}, \beta,]^\intercal$, and the system state equation is

$$\dot{W} = \begin{bmatrix} w_2 \\ f_1\left(w_1, w_2, \theta, \dot{\theta}, \ddot{\theta}\right) \\ \frac{w_2 \cdot R_r}{z} \end{bmatrix}.$$

Linearizing this system with respect to the augmented equilibrium point from (3.2) gives

$$A = \begin{bmatrix} 0 & 1 & 0 \\ a_{21} & 0 & 0 \\ 0 & a_{32} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ b_2 \\ 0 \end{bmatrix},$$

where, a_{21} and b_2 are the same expressions as in (3.5) and (3.6), and $a_{32} = \frac{R_r}{z}$.

This system is uncontrollable since the rank of its controllability matrix $M = \begin{bmatrix} B & AB & A^2B \end{bmatrix}$ is 2, less than the number of the state variables, which is 3. The range space of M is span $([1, 0, a_{32}]^\intercal, [0, 1, 0]^\intercal)$. In other words, only one of two states, x and β, can be independently controlled locally, because linearization hides the nonholonomic nature of the constraint.

3.3 Nonlinear Controller

The Position Tracking Controller causes the ball position x to track a reference trajectory x_r. The controller achieves this by selecting a desired acceleration $\ddot{x}_d$ for the ball based on the errors in current ball position and velocity, *i.e.*

$$\ddot{x}_d = k_p (x_r - x) + k_d (\dot{x}_r - \dot{x}).$$

This is the form of a standard PD controller. The controller then computes the rod angle θ to generate that acceleration. The structure of the controller is shown in Fig. 3.5.

It remains to find the rod angle θ for a given acceleration $\ddot{x}_d$. The complexity of the dynamics model make it impossible to find the θ that satisfies the desired

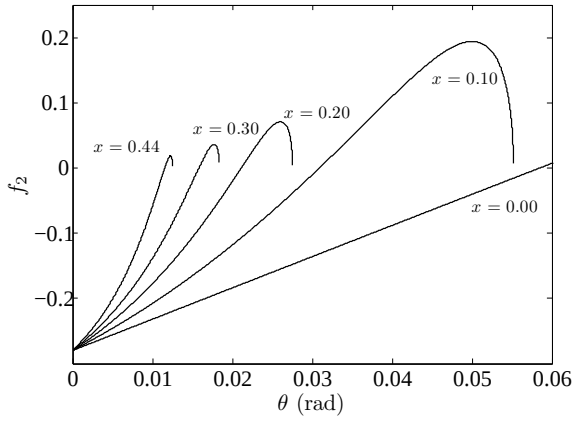

Figure 3.1: Plot of $f_2(x, \theta)$ with different x. Variable θ is on the horizontal axis. The ball drops at the rightmost point of each curve with $x \geq 0.10$. Parameters for this plot are listed in Sec. 3.4.

trajectory by directly solving the original dynamics equation in (2.16). However, some of the nonlinear terms play a trivial role in the dynamics, and ignoring them during controller development helps to simplify this process without sacrificing performance of the resulting closed-loop system.

We approximate the system by assuming x and θ are slowly varying signals, so that in (3.1), the P and Q terms that constitute $f_1(\cdot)$ in (2.16), which contain first and second derivatives of x and θ, are dominated by the relatively long lasting terms $mg\sin\alpha$ and $g\cos\alpha$. Thus, P and Q are dropped and the approximated system dynamics becomes

$$\begin{bmatrix} \dot{w}_1 \\ \dot{w}_2 \end{bmatrix} = \begin{bmatrix} w_2 \\ f_2(x, \theta) \end{bmatrix}, \tag{3.8}$$

where,

$$f_2(x, \theta) = \frac{-mg\sin\alpha - mg\cos\alpha \cdot \frac{\partial z}{\partial x}(x, \theta)}{I(x, \theta)}. \tag{3.9}$$

Therefore, the control problem is reduced into a task of finding the θ that

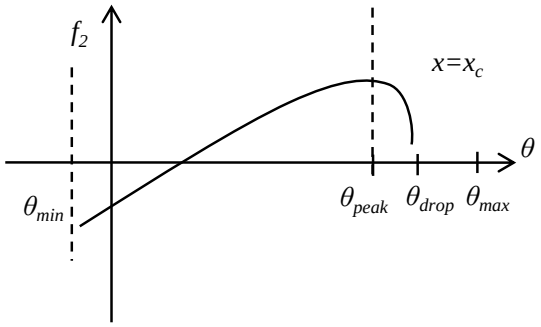

Figure 3.2: Definition of some important coordinates on the f_2 curve with $x = x_c$.

satisfies $f_2(x, \theta) = \ddot{x}_d$. Considering f_2 is not monotonic on θ for constant x (see Fig. 3.1), it is impossible to find a unique f_2^{-1}. However, considering the nature of this problem, an exact inverse is not necessary for the controller development. f_2 is monotonic on θ in a subrange. In Fig. 3.2, θ_{max} and θ_{min} are the constant absolute maximum and minimum values for θ dictated by the construction of the game board; $\theta_{drop}(x)$ is the angle that the ball at x_c will drop between the rods; and $\theta_{peak}(x)$ is the θ that maximizes f_2 at x_c. Clearly, f_2 is monotonic on $\theta \in [\theta_{min}, \theta_{peak}]$, where all possible f_2 values in $[f_{2min}, f_{2max}]$ are covered. Beyond this region, θ is either not attainable on the physical game board or does not improve the performance. A function $g_2(x, \theta)$, which is the same as f_2, but with a smaller domain $\{(x, \theta) | \theta \in [\theta_{min}, \theta_{peak}]\}$, is defined for later use due to its monotonicity for constant x. Let $T_{x_c}(\theta) = g_2(x_c, \theta)$, the appropriate θ can be easily found by calculating its inverse,

$$\theta = T_{x_c}^{-1}(\ddot{x}_d). \tag{3.10}$$

The range of $T_{x_c}(\theta)$ is the same as $g_2(x, \theta)$ under the condition $x = x_c$, which is $[g_2(x_c, \theta_{min}), g_2(x_c, \theta_{peak})]$. The desired acceleration $\ddot{x}_d$ may go beyond this range and pose a problem in finding the appropriate θ. Thus, a saturation function $\eta_{x_c}(\ddot{x}_d)$ is introduced to make the control law well defined. Recall that θ is a variable defining

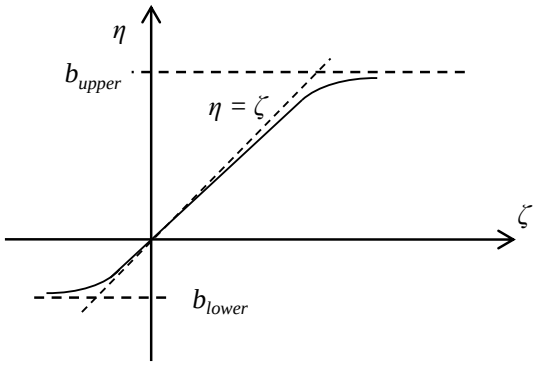

Figure 3.3: The desired saturation function η_{x_c}.

the angle between two rods. Since θ is a control input, it should be at least continuous to its second derivative, which means the saturation function $\eta_{x_c}(\ddot{x}_d)$ must be $\mathcal{C}^2$ as well. Moreover, as a saturation function, the output should be close enough to the input if the input is far away from the clipped region, i.e. $|\zeta - \eta_{x_c}(\zeta)| < f_{error}(\epsilon)$, if $\frac{b_{upper}-b_{lower}}{2} - \left|\zeta - \frac{b_{upper}+b_{lower}}{2}\right| > \epsilon$, where b_{upper} and b_{lower} are upper and lower bounds for the saturation function. It is also desirable to have $\eta_{x_c}(0) = 0$. With these requirements on the saturation function, it should appear similar to Fig. 3.3.

The saturation function η is constructed using the hyperbolic tangent function $S(t) = \tanh(t)$, which saturates at $t = 1$ and has a slope of 1 near $t = 0$. Since η is not symmetric about the origin as normal saturation functions, scale factors are used to shape the curve: η_{x_c} with saturation point $b_{lower} = T_{x_c}(\theta_{min}) < 0$ and $b_{upper}(x_c) = T_{x_c}(\theta_{peak}(x_c)) > 0$ is easily defined with S as

$$\eta_{x_c}(\zeta) = \begin{cases} b_{upper} \cdot S\left(\frac{1}{b_{upper}} \cdot \zeta\right), & \zeta < 0, \\ -b_{lower} \cdot S\left(-\frac{1}{b_{lower}} \cdot \zeta\right), & \zeta \geq 0. \end{cases} \tag{3.11}$$

Plots of the η_{x_c} function are shown in Fig. 3.4 for several shape parameters. Clearly, η_{x_c} and its first derivative are continuous. The second derivative of η_{x_c} is also con-

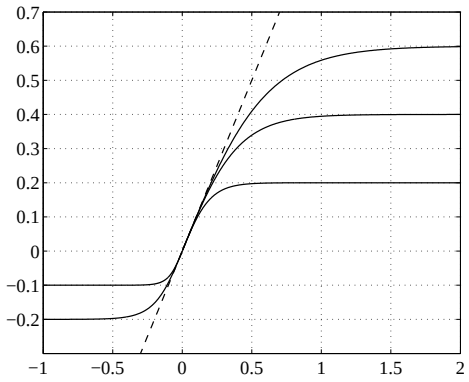

Figure 3.4: A family of curves generated from the saturation function S using different saturation limit.

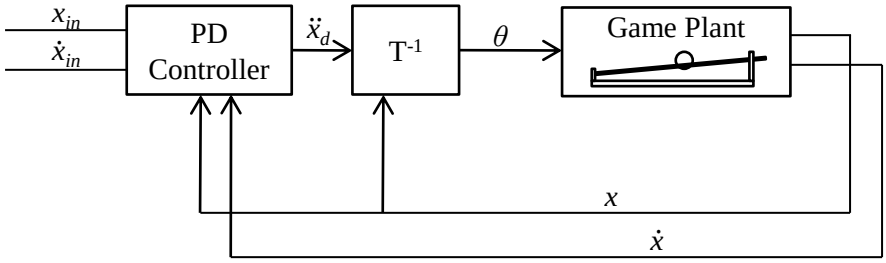

Figure 3.5: Diagram of Position Tracking Controller. In the diagram, T^{-1} is the inverse function with x, θ_p, $\ddot{x}_d$ as inputs, and θ as output.

tinuous because $\frac{d^2S}{dt^2} = 0$ at $t = 0$. With the saturation function in place, the control law becomes,

$$\theta = T_{x_c}^{-1}\left(\eta_{x_c}\left(\ddot{x}_d\right)\right). \tag{3.12}$$

3.4 Simulation

Numerical simulations were conducted using MATLAB/Simulink to demonstrate the viability of both controllers proposed. The Linearized Position Regulator was tested with various initial ball positions x_0 and the Position Tracking Controller was assessed with ramp and sinusoid trajectories. An extra simulation was done to

show that independent x and β can be achieved by oscillating x, which reveals the effect of the nonholonomic constraint.

The following system parameters for the simulation were estimated from the physical Shoot-the-Moon game board used in Chap. 4:

$$m = 20.8\,[g] \qquad R = 12.6\,[mm]$$
$$r = 3.1\,[mm] \qquad L = 440\,[mm]$$
$$\theta_{min} = -0.01\,[rad] \qquad \theta_{max} = 0.15\,[rad]$$
$$d_0 = 9.4\,[mm] \qquad \alpha = 0.035\,[rad]$$
$$J = 1.32 \times 10^{-6}\left[kg \cdot m^2\right]$$

3.4.1 Simulation of Linearized Position Regulator

The controller was implemented for the system linearized at the equilibrium point $x_e = 150$ mm. Solving Eqn. (3.3) numerically, yields $\theta_e = 0.02385$ rad and Eqn. (3.5) and (3.6) generates, $a_{21} = 0.8782$, $b_2 = 11.8645$. The control gain to place the poles at $p_1 = -2$ and $p_2 = -4$ was calculated from Eqn. (3.7) to be $K = \begin{bmatrix} 0.478 & 0.506 \end{bmatrix}$.

The ball was simulated at different initial locations around the equilibrium point $x_e = 150$ mm, and the target ball position was 150 mm as well. The results in Fig. 3.6 showed that the ball position converged to the target position in about 3 s from the initial positions near the equilibrium point.

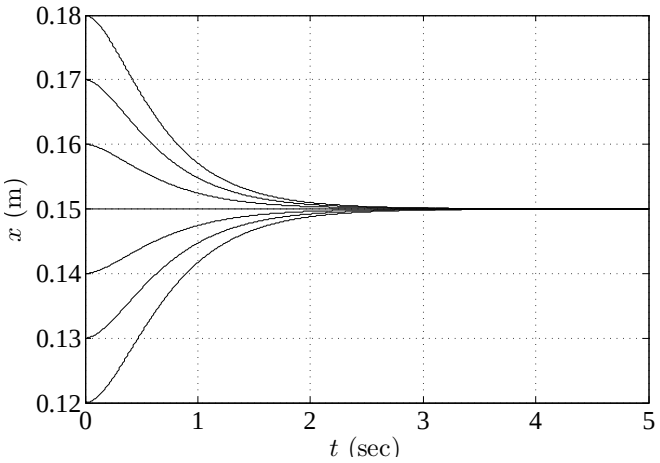

Figure 3.6: Simulation plot of the Linearized Position Regulator.

3.4.2 Simulation of Position Tracking Controller

The Position Tracking Controller was simulated with ramp and sine wave signals as the reference trajectory. In the simulation, control gains $k_p = 2$, $k_d = 3$ were used.

Figure 3.7 shows the controller tracking performance for a reference ramp signal with slope 25 mm/s. The initial position of the ball was the start of the rods, and the initial velocity was zero, (*i.e.* $x(0) = 0$ mm and $\dot{x}(0) = 0$ mm/s). The reference trajectory covered the entire range of the rods from 0 mm to 440 mm. The tracking error reached its peak at the first second while the ball was catching up with the ramp from rest and then quickly converged to zero. After three seconds, the error is less than 2 mm and thus it is hard to differentiate the reference and actual trajectory of the ball.

Figure 3.8 shows the controller tracking performance for a sine wave signal with frequency 0.5 rad/s, amplitude 40 mm and bias 150 mm. The initial condition of the ball was $x = 150$ mm and $\dot{x} = 0$ mm/s. The tracking error is largest at the peaks and troughs of the input sine wave but always less than 5 mm. This error is

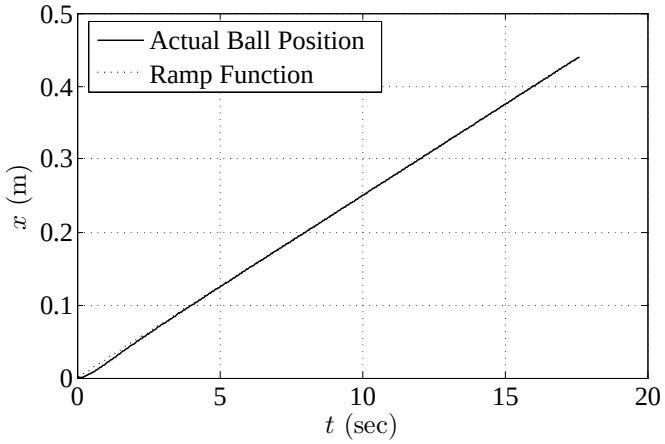

Figure 3.7: Simulation plot of Position Tracking Controller with a ramp input.

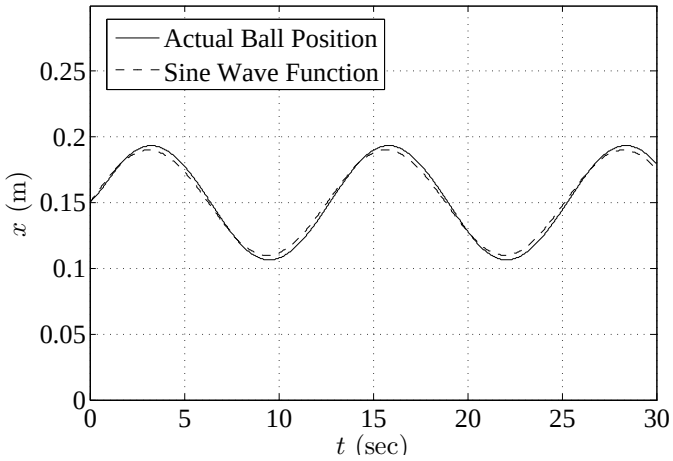

Figure 3.8: Simulation plot of Position Tracking Controller with a sine wave input.

higher than that observed for the ramp input. The increased tracking error could be attributed to the terms in (3.8) and (3.9) that were neglected during controller design under the assumption of a slow varying system. Following a sine wave, especially at the peaks and troughs, requires much higher acceleration than following a ramp function and so the approximation is not as accurate.

In addition, the controller was fed a sine wave signal with a frequency of 6 rad/s, an amplitude of 150 mm and a bias of 40 mm to show the nonholonomic property of the system. From Fig. 3.9, we can clearly see that while x oscillates in a

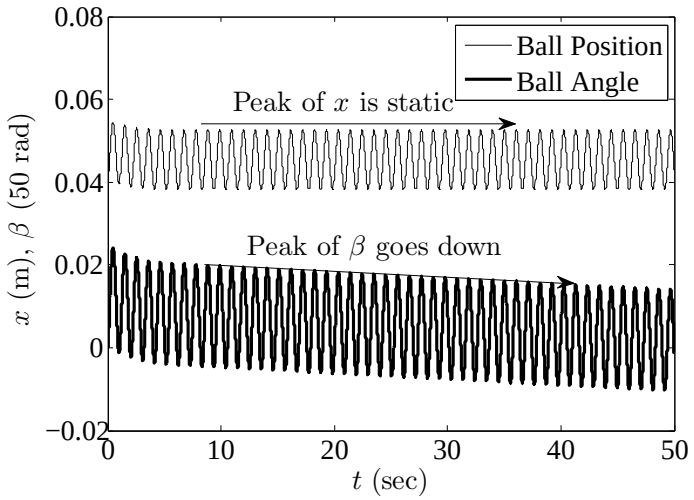

Figure 3.9: Simulation plot of Position Tracking Controller shows the nonholonomic property of the system.

small region, β slowly drifts away with oscillation. Therefore, it is probable to achieve x and β independently in a larger time scale.

3.5 Conclusion

Two controllers were proposed for the Shoot-the-Moon system based on the dynamic model derived in Chapter 2. The Linearized Position Regulator was designed to achieve setpoint regulation and the Position Tracking Controller was designed to force the ball to track a desired trajectory. Computer simulation of both controllers support their viability. A simulation of oscillation ball position shows the nonholonomic evolution of the ball angle.

Chapter 4

Experiment Testing

This chapter will be dedicated to experimentally verifying the mathematical dynamic model developed in Chapter 2 and the performance of the two controllers proposed in Chapter 3. A computer controlled manipulator was designed to automatically play an off-the-shelf Shoot-the-Moon game. A picture of the system is shown in Fig. 4.1. In this chapter, following a brief overview of the system architecture, each of three subsystems, Ball Sensing, Actuator, and Real-time Control is described individually.

4.1 Experiment System Architecture

In this section, the autonomous Shoot-the-Moon system is separated into three subsystems by functionality and described one after another: the ball sensing subsystem detects the linear and angular position of the ball on the rods; the actuation subsystem moves the input rods (*i.e.* plays the game); and the real-time control subsystem implements control algorithms. These three subsystems cooperate together (see Fig. 4.2) to implement a closed-loop control of on the real game board. This

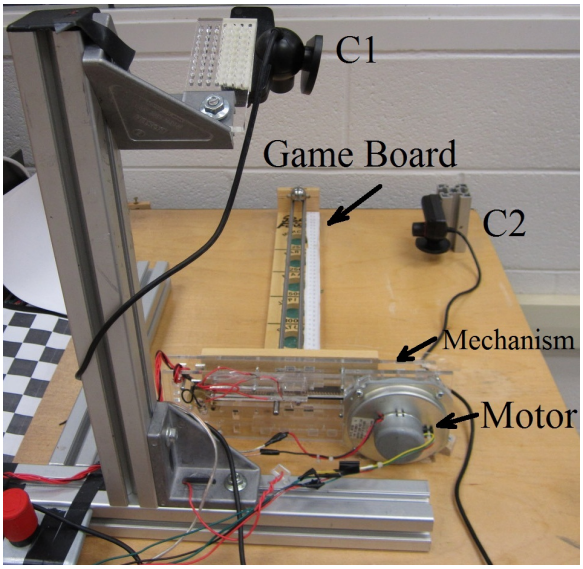

Figure 4.1: Picture of experiment setup. The PC running the real-time system and power amplifier are omitted.

setup can be viewed as a hardware-in-the-loop simulation as well, where the hardware components (game board, servos, sensors) are controlled from a computer implementation of the control algorithm.

4.1.1 Ball Sensing Subsystem

A sensor is a necessary component for a system to operate in a closed-loop manner. The variable being controlled in this experiment is the position of the ball along the x-axis of frame Ω. Since it is desired to compare simulation results based on modelled dynamics with data from a physical experiment, preserving the original dynamics of the ball is the primary concern. Moreover, the sensor system has to provide acceptable resolution, bandwidth, and accuracy within a reasonable cost. Given these constraints, an image sensor based system built with off-the-shelf webcams is proposed for this application.

Among hundreds of available webcams in the market, the Sony Playstation 3

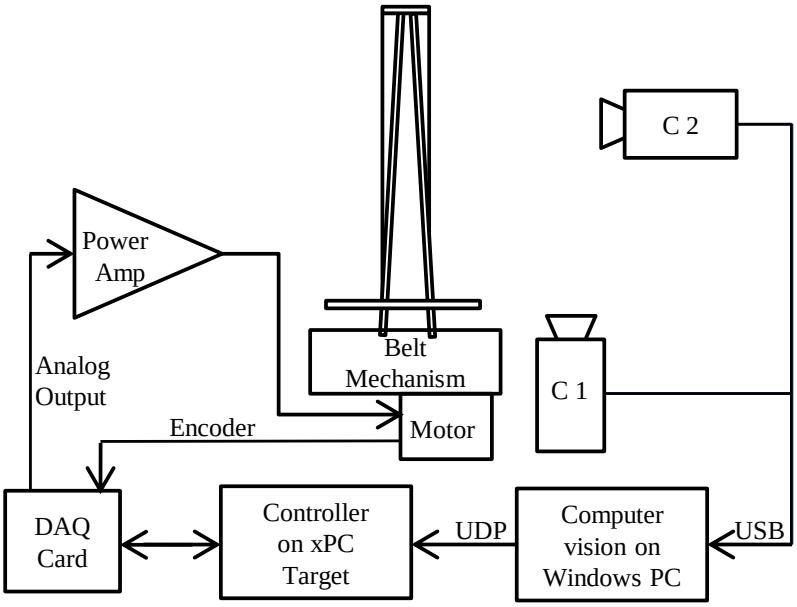

Figure 4.2: Diagram of the entire experiment system.

Eye was chosen for its relatively high performance, low cost, and availability. This camera, which was first manufactured in 2003 by Sony, was originally designed to work only with the Playstation 3 video game console. It is meant to extend the user interface of video games through vision-based object and face tracking, where sensor lag and image quality are both important. Thus it has some special characteristics. Most importantly, it has a USB 2.0 High-Speed mode connection which supports a faster frame rate than other webcams targeted for online video chatting. Its maximum frame rate is 125 FPS under QVGA (Quarter-VGA, equal to 320×240 pixels) resolution and 75 FPS under VGA (640×480 pixels) resolution [16]. In addition, it provides output in raw, uncompressed format (YUV 4:2:2), which saves the compression processing time at the camera side and decompression time at the receiver side. The uncompressed output format also guarentees higher image quality since image compression algorithms are usually lossy, the compression algorithm often cut off high-frequency information and blur the resulting image.

The drivers for the Sony Playstation 3 Eye (later referred as PS3 Eye) for PC platform are not supported by its manufacturer. On the Windows platforms, Code Laboratories provides a user-mode driver based on the WinUSB generic USB driver; on linux platforms, a community developed open source driver `gspca_ov534`, part of the `video4linux` API set, supports the PS3 Eye camera. The Code Laboratories driver are used in this project. This driver has a free version which is limited to two concurrent camera connections to one PC. For more cameras on a PC, licenses can be purchased from CodeLaboratory according to the license agreement document to support up to 16 cameras on one PC. In this Shoot-the-Moon project, one camera is needed for linear position sensing and another for angular position measurement. Thus, the free version is adequate.

The driver provides a set of APIs to control the camera and manipulate its various parameters, including contrast, exposure time, white balance, resolution, frame rate, and output format. It also includes features for performing common linear and nonlinear transformations on the captured image. A list of APIs is shown in Table 4.1

A process diagram (Fig. 4.3) shows the functionality and dataflow of both ball linear position and angle camera: Camera 1 is used to measure the ball linear position and Camera 2 is used to measure the ball angle. Although there is a desire to let the xPC Target run independently without the requirement of external computing power, camera acquisition and part of the vision processing code must run on the Host PC (a laptop) because USB video cameras are not supported in xPC Target.

4.1.1.1 Ball Position Sensing

Based on several considerations, Camera 1 is positioned at the upper end of the game, looking down the x-axis as shown in Fig. 4.1. The mounting position

Table 4.1: A list of APIs in the driver and brief introduction of their functions.

API Name	Function
CLEyeGetCameraCount	Get the number of PS3 Eye cameras on the system.
CLEyeGetCameraUUID	Get the Universal Unique Identification(UUID) for a camera specified, which will be used in creation of camera object.
CLEyeCreateCamera	Create camera object with a UUID.
CLEyeDestroyCamera	Destroy camera object and release resource.
CLEyeCameraStart	Start the camera capturing so it begins pipelining frames to PC.
CLEyeCameraStop	Stop the camera capturing.
CLEyeSetCameraParameter	Set camera parameters and linear or nonlinear transform parameters.
CLEyeGetCameraParameter	Get current settings of camera.
CLEyeCameraGetFrame	Get a latest frame from the camera and store it in provided buffer.

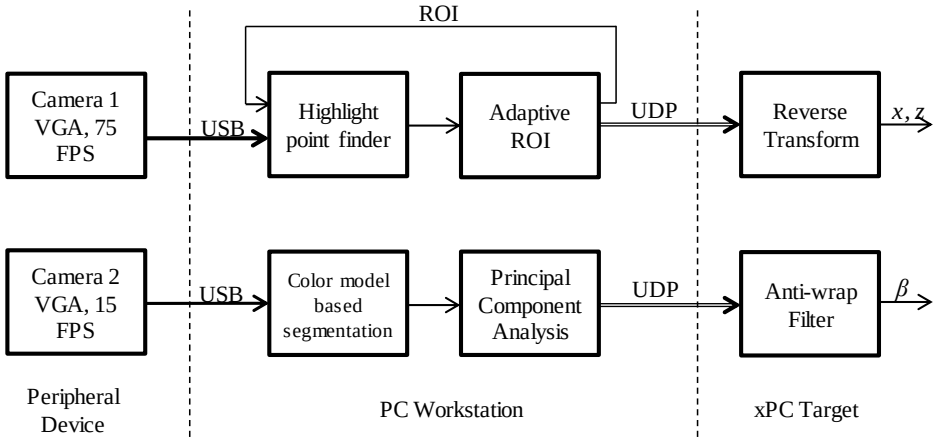

Figure 4.3: Block diagram of the ball state sensor system.

of the camera may not seem ideal at first look; an overhead location which gives an image plane parallel to the Shoot-the-Moon might seem more intuitive. However, this location was chosen to maintain a higher resolution on the region closer to the slot end of the game board because the ball dynamics in this region are more sensitive to changes in ball position. It also avoids the glare of overhead flurescent lights in the laboratory ceiling reflected from the rods. Camera 1 has its focal point located in the x-z plane and thus extra processing for view angle correction is eliminated. In addition, the long side of the image sensor is posed along the rods for maximum resolution along the x-axis. Although theoretically the diagonal of the image sensor is the longest on the sensor and provides the highest resolution, the programming and calibration become more complicated. The long side of the frame provides roughly 1 mm of resolution.

The ball has a polished metalic surface and acts likes a perfect convex mirror. A light sources forms a smaller virtual image behind the mirror surface and appears to be tiny a highlight dot on the ball due to specular reflection. This bright dot is easy to identify in a captured image using a simple thresholding approach, and the small size of the light dot enables an accurate measurement of the ball position.

The light source is positioned close to the camera. This carefully choosen location is convenient for calculating ball position x in frame Ω from the position of the specular reflection point in image coordinate system. As Figure 4.4 shows, the angle between the light source and camera seen from the ball's location is a_o. Angle of incidence and reflection are a_i and a_r respectively, where $a_o = a_i + a_r$, and a_e is the measurement error angle which should be minimized. From the geometry of the proposed light location and the law of reflection, angle $a_e \ll a_r$ and $a_i = a_r$, which leads to $a_e \ll \frac{1}{2}a_o$. Thus, given a small a_o, a_e is very small and can be neglected. Thus, the specular reflection point H and ball center C are projected to approximatly

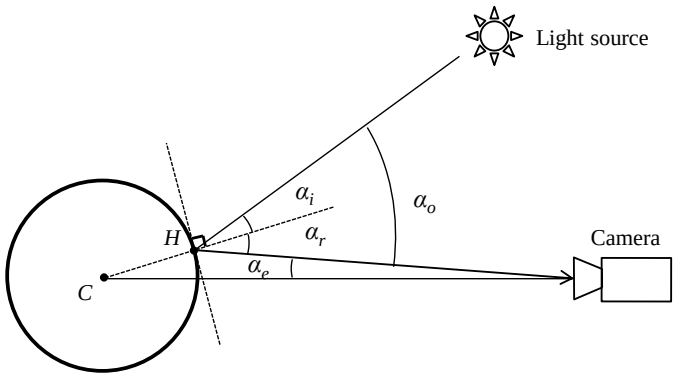

Figure 4.4: An illustration of reflection geometry. As α_i and α_r getting smaller, α_e reduces as well.

the same location in the image plane.

A forward geometry analysis is helpful for establishing the reverse tranform from image coordinates to coordinates in the Ω frame. As Fig. 4.5 shows, from the law of reflection we know the line $\overline{SH}$ formed by camera focal point S and the specular reflection H will be perpendicular to the tangential plane σ at the surface of the ball and we know that extension of $\overline{SH}$ will go through the ball's geometric center C as well.

Let the constant homogeneous transform from frame Ω to the camera frame κ be ${}^{\kappa}H^{\Omega}$, the projection transform by the camera is described by

$$ {}^{\Pi}p_H = T_{cam}\left({}^{\kappa}H^{\Omega} \cdot^{\Omega} p_H\right), \tag{4.1} $$

where the image plane is noted as Π, T_{cam} is the camera transformation, and all the coordinates in this expression are in homogeneous form. Note that, in the image plane Π, ${}^{\Pi}p_H$ is a point in two dimensional space. Although it is represented as a three dimensional vector, its third component only serves as a scaling factor; in the other words, ${}^{\Pi}p_H = [x, y, z]^{\mathsf{T}} = \left[\frac{x}{z}, \frac{y}{z}, 1\right]^{\mathsf{T}}$.

In many applications, the ideal pinhole camera model is used where the T_{cam}

is regarded as a linear transformation which can be represented by a 3×3 matrix. However, because the lens distortion of the camera being used is not negligible and the high accuracy is sought, the camera transformation is decomposed into a linear part, which captures the major characteristics of the camera, and a nonlinear part, which accounts for the fact that the lens is not a perfect pinhole camera lens. Following the convention adopted by the calibration method [17, 11, 13, 12], the decomposition can be written in methematically as,

$$T_{cam}(\cdot) = A_{cam} N_{cam}(\cdot), \tag{4.2}$$

where

$$A_{cam} = \begin{bmatrix} f_x & \alpha_c \cdot f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

is the linear part that describes an ideal pinhole camera and

$$N_{cam}(p) = \left(1 + k_{c1} r^2 + k_{c2} r^4 + k_{c5} r^6\right) p_n + d_p, \tag{4.4}$$

is the nonlinear part that accounts for the lens radial and tangential distortion [18, 19]. The term d_p is

$$d_p = \begin{bmatrix} 2k_{c3} x_n y_n + k_{c4}\left(r^2 + 2x_n^2\right) \\ 2k_{c4} x_n y_n + k_{c3}\left(r^2 + 2y_n^2\right) \end{bmatrix} \tag{4.5}$$

and $p = [x, y, z]^\mathsf{T}$; p_n, the normalized p, is $p_n = [x/z, y/z, 1]^\mathsf{T}$; and $r = x_n^2 + y_n^2$.

From the forward geometry described above and the constraints on the ball's position, the reverse transform, which accepts a pair of image coordinates and outputs the ball's position, can be determined. First, the inverse transform for T_{cam} can be

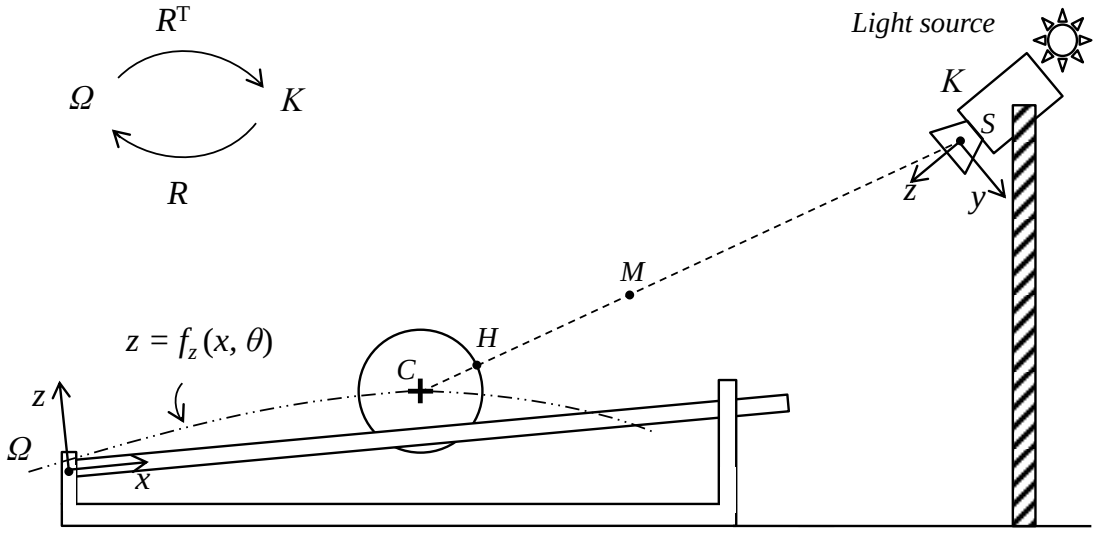

Figure 4.5: Illustration of detecting ball position x with Camera 1.

defined as

$$T_{cam}^{-1}\left({}^{\Pi}p\right) = N_{cam}^{-1}\left(A_{cam}^{-1} \cdot {}^{\Pi}p\right), \tag{4.6}$$

where A_{cam}^{-1} is simply the inverse of matrix A_{cam} and N_{cam}^{-1} is the inverse of the nonlinear transformation. Note that though it is difficult to find N_{cam}^{-1} analytically, it can be approximated with an iterative method [12].

The inverse of the constant homogeneous tranform from Ω frame to κ frame is simply the inverse of the matrix, *i.e.* ${}^{\Omega}H^{\kappa} = \left({}^{\Omega}H^{\kappa}\right)^{-1}$. With camera inverse tranformation T_{cam}^{-1} and ${}^{\Omega}H^{\kappa}$, it is possible to determine ${}^{\Omega}p_H$ from ${}^{\Pi}p_H$, which is the image coordinate of the specular reflection point H, by

$${}^{\Omega}p_H = {}^{\Omega}H^{\kappa} \cdot T_{cam}^{-1}\left({}^{\Pi}p_H\right). \tag{4.7}$$

Note that the image coordinate is scalable on its third component. Thus, instead of getting a single point of ${}^{\Omega}p_H$, a class of points satisfy this equation with various scale factor. These points form a line that connects point S and H. This fact

matches with the experience in the real world: points on the ray starting from the camera center are all projected onto the same point on the image plane; or a single point on the image plane represents a ray starting from the camera center.

Since the fact that the ball only moves in the x-z plane, which was assumed during the dynamic model derivation and dictated by the construction of the driving mechanism, solving for the ball position, originally a 3D problem, can be treated as a 2D rather than 3D problems (Fig. 4.5). With the known (through camera calibration) camera extrinsic parameters, the ${}^{\Omega}p_S$ is determined by ${}^{\Omega}H^{\kappa} \cdot [0,0,0,1]^{\mathsf{T}}$ and ${}^{\Omega}p_M$, which is a point along $\overline{CS}$ is calculated by ${}^{\Omega}H^{\kappa} \cdot [x_c, y_c, 1, 1]^{\mathsf{T}}$. The y components of ${}^{\Omega}p_S$ and ${}^{\Omega}p_M$ will be small and only due to system error, and are discarded when projecting the problem in $x-z$ plane. To simplify the expression, let (x_1, z_1) and (x_2, z_2) be the 2D coordinates of point S and M respectively. The equation of line $\overline{MS}$ is

$$\frac{x - x_1}{x - z_1} = \frac{x - x_2}{x - z_2}. \tag{4.8}$$

From the ball and rod kinemetics model presented in Sec. 2.3, the coordinate of ball's geometric center (or CoG) will satisfy (2.1). Since θ is actively actuated, its value is known during runtime and can be used in solving the ball position. Since C is at the intersection of the line and curve, coordinate of C satisfy both (4.8) and (2.1). With some algebra, coordinates of C, x and z, can be found as

$$x = \frac{-B + \sqrt{B^2 - 4AC}}{2A}. \tag{4.9}$$

where the $x = \frac{-B-\sqrt{B^2-4AC}}{2A}$ is an extraneous root.

$$
\begin{aligned}
A &= k_1 + k_2, \\
B &= 2\left(k_1 b_1 + k_2 b_2\right), \\
C &= b_1^2 + b_2^2 - R_r^2, \\
k_1 &= \frac{z_1 - z_2}{x_1 - x_2}, \\
b_1 &= \frac{x_1 z_2 - x_2 z_1}{x_1 - x_2}, \\
k_2 &= \sin\theta, \\
b_2 &= \cos\theta d_0.
\end{aligned}
$$

4.1.1.2 Ball Angle (β) Sensing

Camera 2 is located at the side of game board to actively monitor the angular position of the ball. This fixed-positioned camera covers a region from 0 to 200 mm in the x-direction on the rod. The span is sufficient for observing the nonholonomic property of the ball dynamics and provides acceptable resolution on the ball for angle calculation. For a broader sensing region, the camera could be mounted on a sliding mechanism that tracks the ball position or multiple cameras could be used to cover the entire region.

The ball is made of steel and is polished, thus few features on the ball can be detected by the camera for ball angle measurement. To solve this problem, a slender colored marker is attached to the ball to facilitate the tracking of the ball's rotation. The vibrant red color of the mark maintains a good contrast with other objects nearby and enhances the robustness of the color marker segmentation algorithm.

Segmentation to identify the color marker can be done using a variety of methods. The constraint is that the algorithm should be relatively fast in order to void too

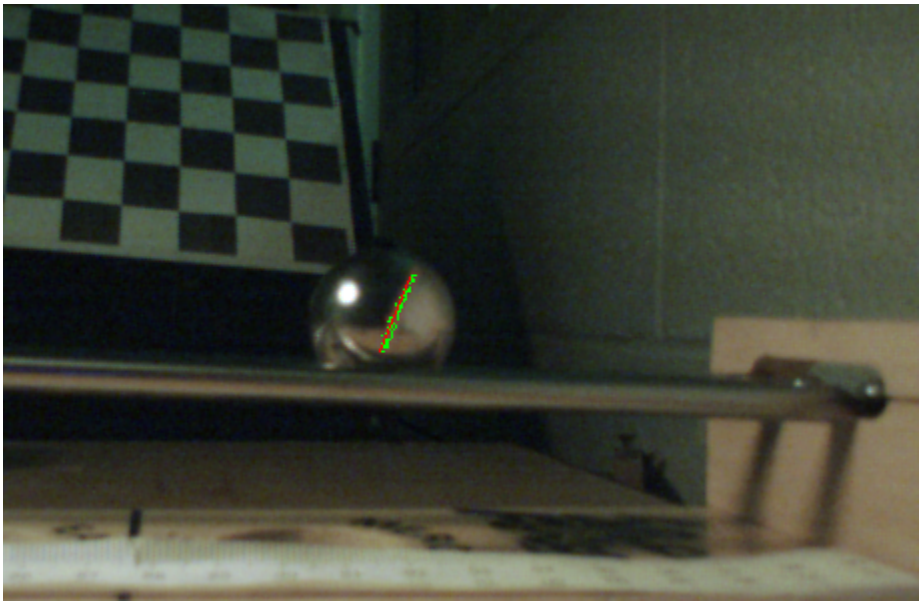

Figure 4.6: Picture showing the red marker on the ball and the detected marker in the computer vision program.

much delay in the feedback loop. The lighting condition at the ball is quite complex in the experiment platform. It is a blend of overhead lighting of the laboratory, the dedicated light source for ball position measurement and their reflections from nearby objects such as the game board, the steel rods, etc. Thus, working with the RGB color model directly is not straight forward since all three components are sensitive to lighting conditions. Color models such as HSL and HSV provide more perceptually relevant representation of color [20]. Hue component of both HSL and HSV is invariant to changes in intensity of lighting and thus a good choice for segmenting a marker with known color. However, the image from the PS3 Eye is in RGB, which is very common in computer image input and output devices, and converting RGB into HSL or HSV involves a nonlinear transformation. The conversion algorithms are usually implemeneted in floating-point arithmatic and are slower than algorithm that require only fixed-point calculations. An algorithm that is easy to understand and can be implemented in fixed-point operation with minimal effort is used. This

method is similar to the $I1, r, g$ or $I1', I2, I3$ method described in [21] and the method illustrated in [22].

It is assumed that the RGB components will vary linearly with the change of intensity of white lighting, *i.e.*

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = I \cdot \begin{bmatrix} 1 \\ G \\ B \end{bmatrix} + N, \tag{4.10}$$

where r, g, b represent the three components of the pixels in the acquired image, and G, B are components that pertain to a specific color of the color marker, I is a scalar that describes the lighting intensity, and the vector N is noise that has zero average. It is a simple linear model and can be used for color segmentation. The parameters in the linear model are obtained by least square fit of manually labeled pixels on the color rotation marker, where parameters G, B and the boundary of I, I_{min} and I_{max} are determined. For pixel-based segmentation, the criterion for a foreground pixel is $I_{min} \leq r \leq I_{max}$ and $|g - r \cdot G| + |g - r \cdot G| \leq T_h$, where variable T_h is a tunable threshold.

The thresholded image is then passed through the principle component analysis (PCA) to find a best fitting ellipse that covers the foreground blob (see Fig. 4.7). The vector parallel to the longer axis of the fitted ellipse is used to determine the angle of the marker, which is also the angle of the ball. This approach was illustrated in computer vision literatures in the work by Sonka el al. [14] and the work by Archarya et al. [15].

Let the set of n foreground pixels be $P = \{p_1, p_2, ..., p_n\}$, where $p_i = [p_{ix}, p_{iy}]^\mathsf{T}$ is pixel i, and p_{ix}, p_{iy} are its x and y coordinates in the image frame. A frame parallel

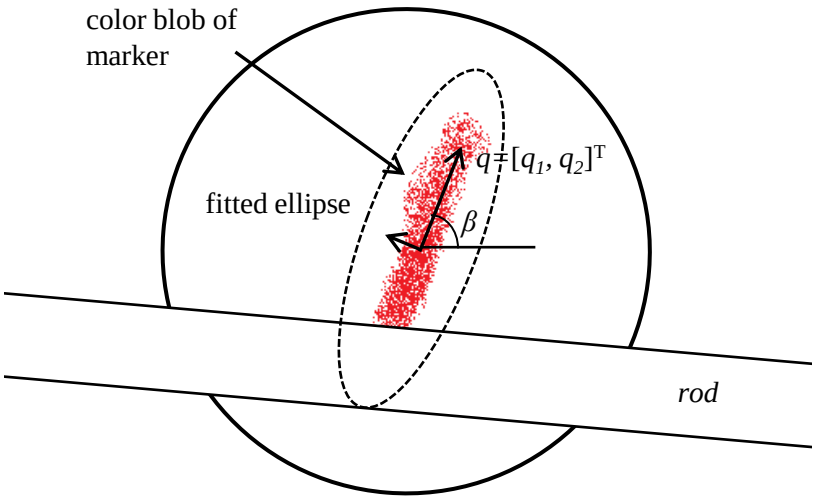

Figure 4.7: Illustration of the Principle Component Analysis to find the anglular position of the ball.

to the original image frame with its origin at centroid of the foreground pixels, which is point $\bar{p} = \frac{1}{n}\sum_{i=0}^{n} p_i$, is defined. Pixel coordinates in this newly defined frame are noted as $\tilde{p}_i$. Thus, the set P in the original image frame is equivalent to $\tilde{P}$ in the new frame. Covariance matrix C of $\tilde{P}$

$$C = \frac{1}{n}\sum_{i=0}^{n} \tilde{p}_i \cdot \tilde{p}_i^{\mathsf{T}} \tag{4.11}$$

summarize its two dimensional distribution. Eigendecomposition of C can separates the orientation and broadness information as $C = Q\Lambda Q^{-1}$, where Q is the matrix of eigenvectors and Λ is the diagonal matrix containing eigenvalues. Each eigenvector represents a distribution axis and its associated eigenvalue is the variance projected on that axis. The eigenvector $q = [q_1, q_2]^{\mathsf{T}}$ associated with highest eigenvalue λ_{max} is the orientation of the long axis of the ellipse. The angle of the long axis can be found by $\arctan(q_2/q_1)$.

The domain of arctan is $(-\pi/2, \pi/2)$. If the ball rotates beyond this limit from

either boundary, it will wrap back to the opposite boundary. The camera measurement is in discrete time. The output of PCA is denoted as $\tilde{\beta}[n] = \varrho(\beta(nT))$, where T is the sample period and $\varrho(\beta) = \beta + m\pi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, with $m \in \mathbb{Z}$, is the wrapping function. Assuming the sample period is short enough so that $\left|\dot{\beta}T\right| < \Delta\beta_{th} < \frac{\pi}{2}$ is always satisfied, a differentiation and summation based anti-wrap algorithm can be devised as

$$\hat{\beta}[n] = \tilde{\beta}[0] + \sum_{0<i\leq n} \varsigma\left(\tilde{\beta}[i] - \tilde{\beta}[i-1]\right), \tag{4.12}$$

where $\varsigma(\cdot)$ is the anti-wrap function defined as,

$$\varsigma(\zeta) = \begin{cases} \zeta, & |\zeta| < \Delta\beta_{th}, \\ \zeta + \operatorname{sgn}(\zeta)\cdot\pi, & |\zeta| \geq \Delta\beta_{th}. \end{cases} \tag{4.13}$$

4.1.2 Actuator Subsystem

The actuator subsystem consists a series of mechanisms that couple the movement of the actuator, a DC motor in this project, to the angle between the two rods of the Shoot-the-Moon game.

During the controller development in Chapter 3, the angle θ is assumed to be an input which only depends on the states of the ball and control reference. The interfacing system, as any mechanical system, will have inertia, which can introduce extra dynamics into the system and affect system behavior. In an extreme case, where the dynamics of the interfacing system has a higher time constant than the Shoot-the-Moon system, the system responce will be dominated by the interfacing system rather than the original system. In contrast, if the interfacing system has a much faster response than the Shoot-the-Moon dynamics itself, the effect of the additional dynamics will be negligible.

Table 4.2: Motor Specification

Shaft Diameter	6 mm
Rated Voltage	24V DC
Rated Current	180 mA (no load at 4600 RPM), 2.3 A (12 oz.-in. load at 4000 RPM).
Stall Torque	70 oz.-in.
Stall Current	10 A
Encoder	Two channel, 400 line/rev, TTL output

The actuator choosen is a Tohoko Ricoh disk shaped DC motor. The specification of this motor could be found in Table 4.2. The high torque of this motor enables a direct drive design, where no torque converting gearbox is used, thus eliminated the disadvantages of a geared motor. For example, the backlash and parasitic friction in a gearbox is avoided, which provides a more linear characteristics and the response time is greatly improved since few moving parts are involved. Also, since the encoder for position sensing is mounted directly on the output shaft, the actuation can be more precise. The power for the motor is provided by a Techron 5530 power amplifier, which converts the low-power signal output from the analog I/O board into a high power output capable of driving the DC motor.

For safety consideration, several limit switches and an emergency stop button are included in the system to prevent the mechanism and motor from being damaged by a faulty control signal from the controller during development.

As illustrated in Fig. 4.8, the interfacing system is based on a timing-belt driven mechanism. The timing-belt is built with high-tensile strengh fiber substrate and rubber surface. The teeth on the belt and drive wheel gurantees no slip will happen during runtime, which is required since the position of motor shaft is measured and will be used to calculate the angle between the two rods, which is the value being servoed. Timing belt is also very efficient way to transfer power due to its

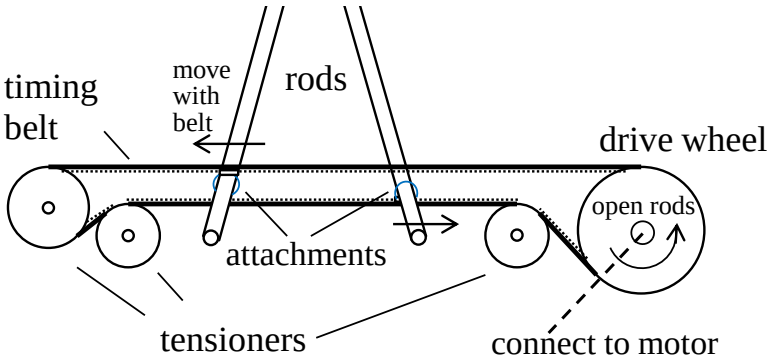

Figure 4.8: Illustration of the timing belt driven actuation mechanism. As th left is attached to the top loop of the belt and the right rod is attched to the bottom loop, the rods move in opposite direction as the motor turns.

low friction. Two small wheels are used to tension the timing belt and keep a small distance between top and bottom loops of the belt. The thin attachment wires secure the two rods to the belt.

The base structure of the mechanism and all passive wheels are built with laser cut acrylic boards. The driving wheel and shafts are purchased from hobby stores. A complete CAD drawing for the mechanism is shown in the Appendix A.

4.1.3 Real-time Control Subsystem

A real-time system is a system that enforces certain "real-time constraint" [23, 24] on the execution of software. The delay due to controller computation is usually not modeled during controller design and thus the real-time execution is implied in controller implementation. The computation power delivered by modern mainstream PCs are usually more than enough for controllers that require limited computation in the control law, such as the two controllers proposed here. In other words, delay caused by pure computation is not a serious problem. On the contrary, operating system scheduling is the major contributor for delay in controller program. Modern computer modular software and hardware components forces information from sensors

to travel among buffers and process boundaries before reaching the controller program that really process it, where each process switch can take up to 10ms. Similar situations happen in the outgoing branch from the controller as well. Even worse, common operating systems such as Windows, Unix and Linux are optimized for throughput and efficiency instead of real-time processing, *i.e.* their process scheduling time cannot be exactly specified and varies from time slice to time slice.

A real-time operating system is designed to fit the specific needs for real-time processing. They tends to have a small and fast running process scheduler and minimized interrupt handler in device drivers. All heavy computation is offload to processes, of which priorities can be specified to requirement of the application. There are many real-time operating systems available, open- or closed-sourced, free or proprientary, for workstations and embedded systems [25]. Some widely used real-time OSs includes VxWorks, QNX, RTLinux, eCOS, uC/OS, Symbian and Windows CE.

The real-time system being used in this project is the xPC Target from Mathworks Inc. The xPC Target is a complete solution for real-time system implementation of Mathworks that is deeply integrated into MATLAB and Simulink from the same company. With this integration, it offers many desirable features such as quick prototyping and implementation, high portability of software, development, and hardware-in-the-loop (HIL) simulation. From a technical perspective, xPC Target is based on RTOS-32, a real-time operating system for x86 platform, from On Time GmbH in Germany. Above RTOS-32, a layer of runtime libraries are provided by Mathworks to construct a software environment very similar to that on PC, where normal Simulink simulation is running.

The development of the controller is primarily accomplished with Simulink. To build binary code for the controlller, the Real-time Workshop Toolbox (part of

MATLAB product) first translates the Simulink model, which is a networks of blocks, into C or C++ code and pass the code to a compiler targeted to the Windows platform to generate binary file. The binary file is then modified to adapt for the xPC Target and compressed for a smaller file size. The compressed binary is finally downloaded and loaded to the xPC Target PC automatically via ethernet and ready to be executed.

The controller for the experiment is developed in Simulink and is identical to the one used in simulation. The only difference to the simulation model is the simulated system dynamics block is replaced by sensor input blocks and motor output blocks.

The analog input and output for the system is fulfilled with Quanser Q4 multifunctional data acquisition card. The driver for Q4 is included in xPC Target package and using ports of the card is as simple as dragging in Simulink blocks. Q4 card has four -10V to 10V range 14-bit analog input ports, four programmable range (-10 to 10V, 0 to 10V or -5 to 5V) 14-bit analog output ports, sixteen digital I/O channel, four quadrature encoder input ports and two PWM output ports. Additional information about Q4 card can be found in [26].

4.2 Experiment Results

The Linearized Position Controller and the Position Tracking Controller are compared for performance. Test trajectories include a rate limited set points signal and a sinewave signal. A specially designed sinewave signal is also shown to be able to control the ball angle in open loop. Results of physical system are also compared with simulation and the validness of the modeling is revealed.

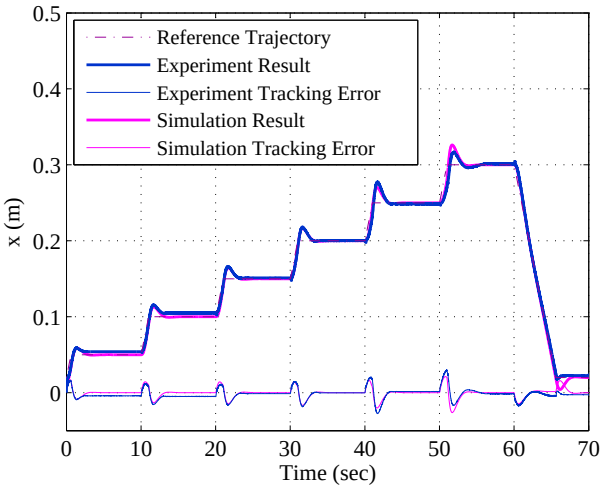

Figure 4.9: Results of Nonlinear Tracking Controller following a series of set points from simulation and experiment.

4.2.1 Set Points Trajectory

The set points trajectory starts from the lowest end of the game board and goes up through a series of set points, frow low to high at 50 mm, 100 mm, 150 mm, 200 mm, 300 mm. The reference signal is rate limited to 50mm/s to prevent discontinuities in the reference signal. The gains of the Position Tracking Controller are set to $k_p = 3.3$ and $k_d = 2.4$. Results of experiment and simulation are compared in Fig. 4.9, where behavior of both systems matches closely. The steady errors of the physical system at all set points are below 5 mm, but they do not converge to zero, potentially due to interference from static friction when the velocity of the ball is almost zero.

4.2.2 Sinusoidal Trajectory

The sinusoidal trajectory began from the lower end of the game board, ramped up to 200 mm at a rate of 25 mm/s and stayed there for a few second while the ball

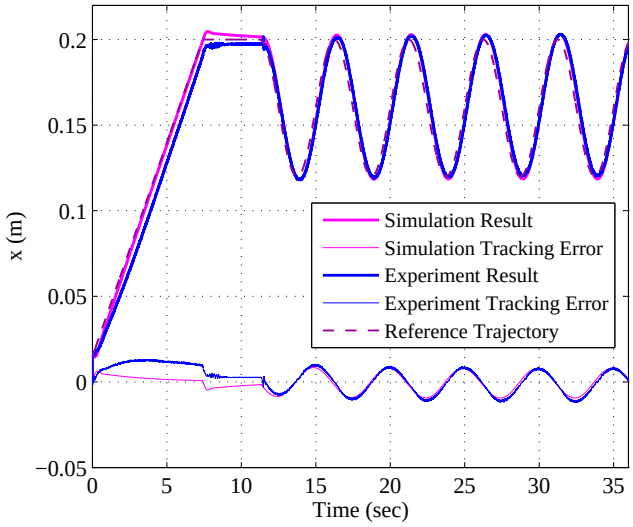

Figure 4.10: Results of Position Tracking Controller following sine wave trajectory from simulation and experiment.

settled, then oscillated in a sinusoid around point 160 mm with 5s period and 40 mm amplitude. The oscillation started at $\pi/2$ initial phase, and the whole reference curve was smooth. The PD gains of the Position Tracking Controller were manually tuned to $k_p = 1.5$ and $k_d = 6.0$ for better performance, measured in terms of the integrated squared error over the sinusoidal part. Fig. 4.10 shows the output of the physical system compared to simulation with the same controller. The error was always small and the physical system acted very similarly to its simulated counterpart. During the ramp, the maximum error was 12.5 mm. During the sinusoid, the error was bounded by ± 10 mm. This error was mostly due to phase shift from the reference trajectory. The amplitude of the sinusoid was close to the reference in both the experiment and simulation.

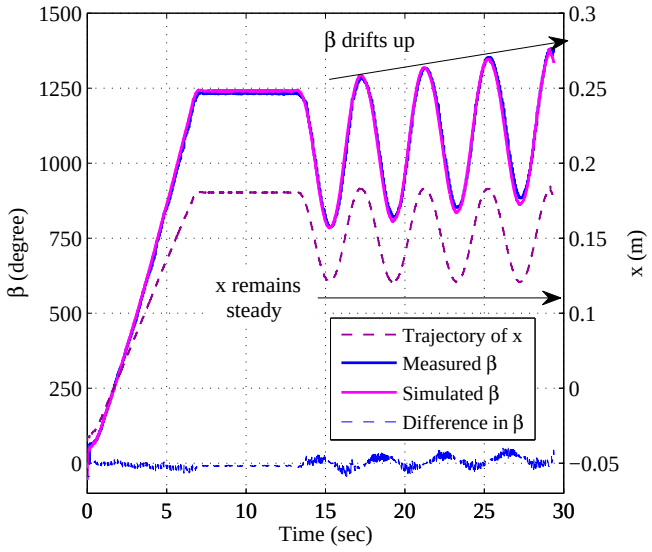

Figure 4.11: Demonstration of the nonholonomic property of the system in simulation and experiment.

4.2.3 Demonstration of Nonholonomic Behavior

Shoot-the-Moon is a simple two degree-of-freedom system with a nonholonomic constraint. The nonholonomic behavior was demonstrated (Fig. 4.11), by oscillating the ball position x while observing ball angle β. The system was commanded to follow a sinusoidal trajectory similar to the previous experiment. The sinusoid, centered at 150 mm, had amplitude 30 mm and period 4 s.

Note that the peaks of the ball angle β trended upward while the position x repeated the same pattern, demonstrating a characteristic feature of a nonholonomic system [2]. At the end of the fourth cycle, β has increased by 150° even though x has returned to the same value. This is analogous to how a car, another typical nonholonomic system, can parallel park by cycling the control inputs appropriately. β was also simulated at the same time using the system model and measurement of x, $\dot{x}$ and θ. The simulated value was very close to the measurement. The difference

between measured and simulated β was within $\pm 20^\circ$ during the initial ramp, $\pm 10^\circ$ during the plateau, and $\pm 45^\circ$ during the sinusoid. The increased difference during sinusoid tracking is almost entirely due to a small phase difference between the curves, corresponding to measured β lagging simulated β by approximately 0.05s. This lag may be largely attributed to the slower update frequency of the camera used to measure β. Measured and simulated β match very well, especially considering that β is integrated from other signals and slight model mismatches could cause measured β to diverge significantly from simulated β.

Results of this test exhibited the accuracy of the model, especially in that it successfully captured the nonholonomic constraint of the Shoot-the-Moon system. Using the model, methods from nonholonomic control theory could be applied to design trajectories that achieve an arbitrary combination of x and β and the desired results should be reproducible on the physical system.

Chapter 5

Conclusions

In this thesis, the dynamics of Shoot-the-Moon, which had not previously appeared in the literature by others, were derived following classical Lagrangian and Newtonian approaches. The dynamics are nonlinear, underactuated, and nonholonomic. In fact, since a system requires at least two degree-of-freedom to have a nonholonomic constraint, Shoot-the-Moon is one of the simplest nonholonomic systems possible.

Two ball linear position controllers were designed based on the dynamics model. The Linearized Position Regulator was developed using a local linearization at equilibrium points of the dynamics. The Position Tracking Controller took nonlinearities into account by inverting the significant nonlinear terms in the dynamics so that the system appears linear at the input and can be controlled using a PD controller. Simulations of these two controllers showed satisfying results, the Linearized Position Regulator was able to drive the ball to setpoint from nearby locations and the Position Tracking Controller could track ramp and sinusoid desired trajectories.

An experimental platform was developed to validate the Position Tracking Controller performance on physical Shoot-the-Moon game board. Design of the plat-

form was described in sufficient detail for later reconstruction or replication of the system. The experimental results match very closely with simulation, validating the modeling assumptions made in deriving the dynamics and designing the controllers. The nonholonomic constraint between the linear and angular position of the ball was observed in simulation and experiment. Nonholonomic control techniques could be applied to design trajectories that would guide the ball to a desired arbitrary combination of linear and angular positions.

Shoot-the-Moon is a tabletop game that appeals to a broad audience. The dynamics could serve as a useful educational example of a nonholonomic system. The experimental platform is simple and can be constructed at low-cost. The system would be suitable as a challenge problem for nonholonomic controls techniques or as a classroom example or laboratory exercise in applied controls techniques.

Appendices

Appendix

Appendix A CAD Drawing of the Mechanism

Shown in Fig. A.1 is the CAD drawing of the timing-belt rod driving mechanism. The drawing is consist of three parts. The bigger two pieces are front and back plates of the mechanism, with round holes for shafts (marked by blue cicles) and screws to go through. The small piece, which are duplicated 6 times during production, is sandwiched between the front and back plates in order to maintain parallelity and a proper distance between the two plates. The teeth-like structure on the small piece are going into pre-cutted rectangle slots on the plates to increase rigidity of the entire mechanism. Driving and passive wheels are placed in between two plates, and is supported by the shafts. A timing-belt was routed around these wheels and was attached to the rods of the game with metal hoops made from paper clips. All pieces are cut with 6 mm clear acrylic board.

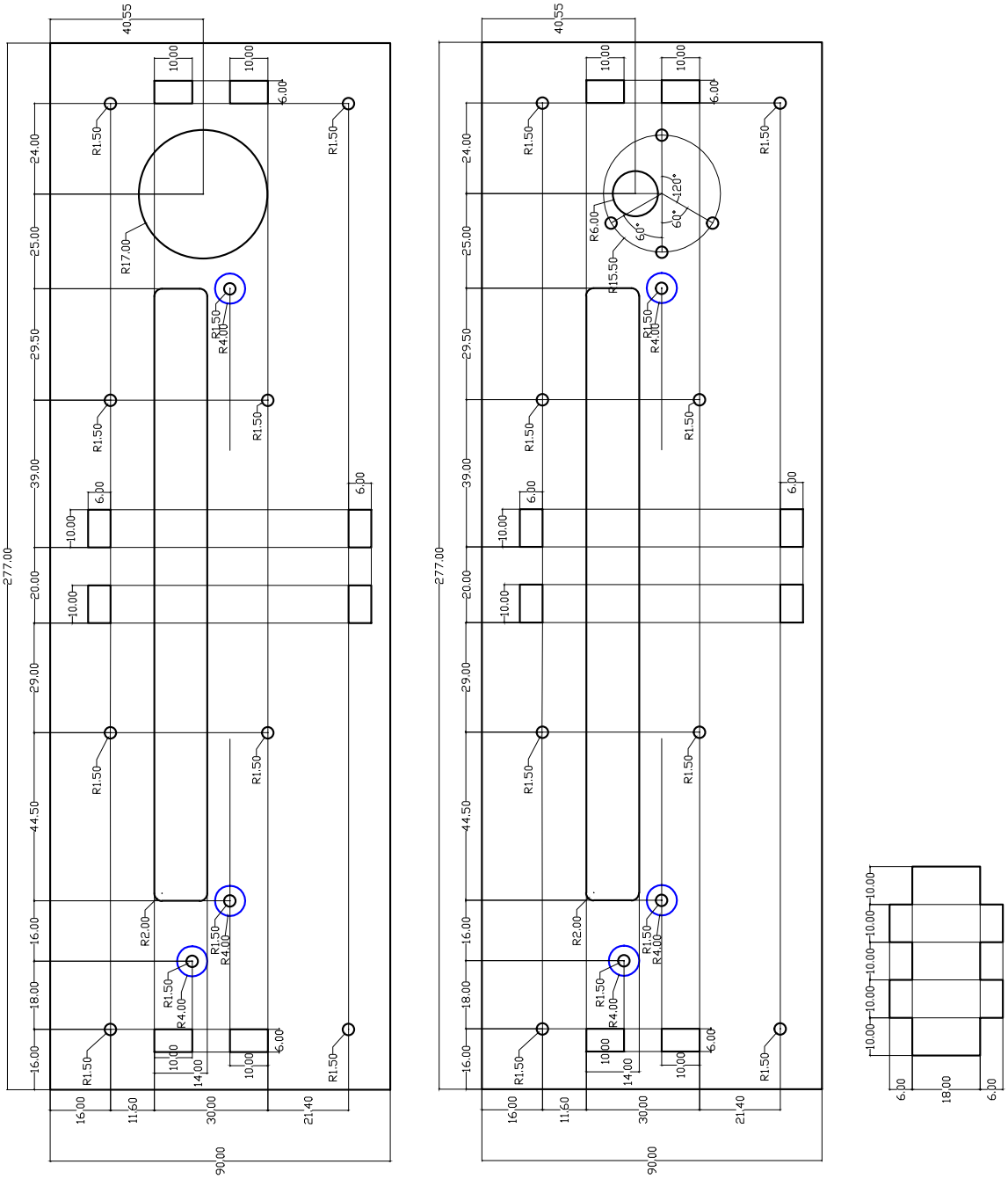

Figure A.1: Timing-belt rod driving mechanism CAD drawing.

Bibliography

[1] S. Sastry, *The ball and beam example*, ser. Interdisciplinary applied mathematics: Systems and control. Springer, 1999, ch. 10.

[2] A. Bloch, *Nonholonomic mechanics and control*, ser. Interdisciplinary applied mathematics: Systems and control. Springer, 2003.

[3] H. Aref, "Toys and games in mechanics education," in *XXI International Congress on Theoretical and Applied Mechanics*, Warsaw, Poland, 2004, pp. 15–21.

[4] P. Xu, R. E. Groff, and T. Burg, "The rigid body dynamics of shoot-the-moon game and model-based controller design," in *American Control Conference (ACC)*, 6 2010, pp. 396–401.

[5] D. T. Greenwood, *Hamilton's Equation.* Courier Dover Publications, 1997, pp. 147–164.

[6] H. Goldstein, C. P. Poole, and J. L. Safko, *Variational Principles and Lagrange's Equations*, 3rd ed. Addison-Wesley, 2001.

[7] R. A. Layton, *Lagrangian DAEs of Motion.* Springer, 1998.

[8] W. L. Brogan, *An Introduction to Nonlinear Control Systems*, 3rd ed. Prentice Hall, 1990, pp. 565–570.

[9] M. Gopal, *Physical Systems and State Assignment.* New Age International, 1993, pp. 116–118.

[10] H. K. Khalil, *The Invariance Principle.* Prentice Hall, 2002, ch. 4.

[11] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 323–344, 1987.

[12] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Computer Vision and Pattern Recognition, 1997.*

Proceedings., 1997 IEEE Computer Society Conference on. IEEE, 1997, pp. 1106–1112.

[13] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *iccv*. Published by the IEEE Computer Society, 1999, p. 666.

[14] M. Sonka, V. Hlavac, and R. Boyle, *Region-based shape representation and description*. Thompson Learning, 2008, pp. 351–370.

[15] T. Acharya and A. K. Ray, *Karhaunen-Loeve Transfrom or Principal component analysis*. Wiley-Interscience, 2005, pp. 73–78.

[16] P. Kirn. (2009, Aug.) Trick out your ps3 eye webcam, best cam for vision, augmented reality. [Online]. Available: http://createdigitalmotion.com/2009/08/trick-out-your-ps3-eye-webcam-best-cam-for-vision-augmented-reality/

[17] J. Bouguet. (2011, Jul.) Complete camera calibration toolbox for matlab. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

[18] J. Fryer and D. Brown, "Lens distortion for close-range photogrammetry," *Photogrammetric Engineering and Remote Sensing*, vol. 52, no. 1, pp. 51–58, 1986.

[19] D. Brown, "Close-range camera calibration," *Photogrammetric engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[20] A. Godse, *Colour Models*. Technical Publications, 2009.

[21] Y. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," *Computer graphics and image processing*, vol. 13, no. 3, pp. 222–241, 1980.

[22] I. Omer, "Image specific color representation: Line segments in the rgb histogram," Master's thesis, School of computer science and engineering, The Hebrew University of Jerusalem, Israel, 2003.

[23] I. Gupta, *Embedded Realtime Systems Programming*. McGraw-Hill, 2003.

[24] R. Mall, *Real-Time Systems: Theory and Practice*. Prentice Hall, 2009.

[25] Wikipedia. (2011) List of real-time operating systems. [Online; accessed 2-July-2011]. [Online]. Available: http://en.wikipedia.org/wiki/List_of_real-time_operating_systems

[26] Q. C. Inc. (2003) Q4 data acquisition system user's guide. [Online]. Available: http://www.clemson.edu/ces/crb/ece495/References/manuals/quanser_q4_manual.pdf