BALANCE OF A ROBOTIC LEG ON UNEVEN TERRAIN

A Project

Presented to the

Faculty of

California State Polytechnic University, Pomona

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

In

Electrical Engineering

By

A Micah McClain

2020

SIGNATURE PAGE

PROJECT: BALANCE OF A ROBOTIC LEG ON UNEVEN TERRAIN

AUTHOR: A Micah McClain

DATE SUBMITTED: Spring 2020

Electrical and Computer Engineering Department

Dr. Salomon Oldak
Project Committee Chair _____

Dr. James Kang
Project Committee Member _____

Dr. Thomas Ketseoglou
Project Committee Member _____

ABSTRACT

Bipedal balancing is a complex subject as the process of balancing is difficult to define. As robot designs are becoming increasingly humanoid and human-robot interactions increasingly common, algorithms must be developed to replicate natural movement in non-ideal circumstances. This project implements such an algorithm on a robotic leg to attempt to balance on uneven surfaces and terrain. A pseudoinverse Jacobian was implemented to handle positional movement, and a linear quadratic regulator was used to shape smooth motion to desired balance points. They were then developed into an Arduino code which controls the motion of the robot in attempts at balance. Due to the limited stall torque of the chosen servos the robot could not achieve balance in practice. And although the position and control codes work well with the chosen microcontroller, the balance equations could be more versatile and accurate, so future work should focus on these aspects of the robot design process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

WITHIN the field of robotics there are many ways to incorporate movement. A popular technique is bipedal balancing as it mimics human movement. The ability to replicate human-like movement would allow robots to easily avoid obstacles, traverse terrain, and operate in areas heavily populated by humans. Not much is known about exactly how humans maintain bipedal balance [1], so replicating it in robotics is challenging. The ways that a robot can walk and maintain balance can be easily determined for ideal conditions using a periodic gait, but there is difficulty in understanding how to implement aperiodic gaits in non-ideal situations because of the difficulty in understanding what factors are necessary for stability [2].

Algorithms must be developed for balancing robots to be able to simulate the natural balance of humans. These algorithms should allow robots to maintain balance regardless of environment and external conditions. The algorithms must be able to maintain stability within human-shaped robotics. The purpose of this project is to adapt general control and robotics principles to a wholly original robot design to explore techniques that may be applied to other non-standard robot designs in the future.

II. THE CONSTRUCTION

The first components considered in the design of the robotic leg were the servos. Due to familiarity and ease of use the servos chosen were the Dynamixel AX-12A Robot Actuators from Robotis. The servos also have many available frames and brackets which could be modified with CAD software to fit a customized robot design. These servos would make up the majority of moving parts, however there is also an L16-R linear servo attached at the robot's foot, with the intent to provide additional foot movement as necessary.

TABLE 1.
Parts Price List

Item	Amount	Each
Dynamixel AX-12A Servos	4	$ 44.90
Arbotix-M Robocontroller	1	$ 39.95
FTDI USB Cable	1	$ 12.98
MPU-6050 3 Axis Gyroscope	1	$ 6.77
Actuonix L16-R Linear Servo	1	$ 70.00
11.1V LiPo Battery Pack	1	$ 13.99
DC 12V to 6V 3A Module	1	$ 6.40
2 ft. PVC Pipe	1	$ 2.34
3D Printing (Various Parts)		$ 199.73
Total		**$ 531.76**

Trying to keep costs as low as possible, PVC pipes were used as the links between joints. These links, L_1 and L_2, comprise of the length of the PVC pipe cut in half plus the distance from the end of the pipe to the axes of rotation of the two connected joint servos. The measurements for L_1 and L_2 came out to 375.92 mm and 396.24 mm respectively. To connect the servos to these links, several brackets were designed using SolidWorks. These designs include the foot, heel, servo-pipe connectors, servo-servo connectors, and the circuit housing. The circuit housing was 3D printed from PLA plastic, while most other parts were printed from Nylon 12 plastic for increased structural strength.

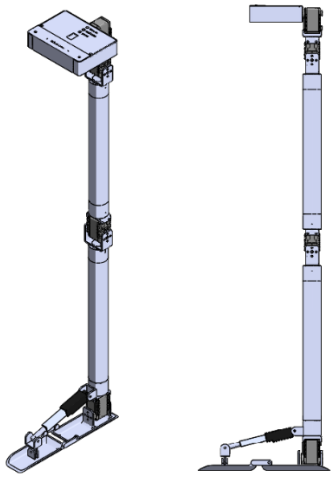

FIGURE 1. 3D Rendering of the Robotic Leg – Multiple Angles.

The robot is run using an Arbotix-M Robocontroller, which is specifically designed for use with Dynamixel servos. The Arbotix-M has a ATMEGA644p chip, allowing for the servos to be programmed using the Arduino IDE ver. 1.6.13 with the use of an FTDI cable. Connected to the Arbotix-M is an MPU-6050 3 axis gyroscope/accelerometer. The Arbotix-M is powered by an 11.1 V 2200 mAh LiPo battery pack. As the dynamixel servos require 12V and the linear servo requires 6V, it was necessary to add a step-down module. Fortunately, the Arbotix-M has an input for an auxiliary power supply. This allowed all servos to be plugged into, and controlled from, the same microcontroller. Due to the use of PVC piping as the joint links, the wiring was able to be routed through the inside of the pipes which both protected the wires and resulted in a cleaner look.

III. THE CURRENT POSITION

To get the robot to balance, it is necessary to know its position in space. When the robot is not balanced it can move to a position of known balance via inverse kinematics equations. These positions are the Height position (distance from the floor), mediolateral position (side-to-side distance from center), and anteroposterior position (front-to-back distance from center).

To know the position of the system's balancing point the frames must be defined. These frame assignments can be seen in ***FIGURE 2***. With the origin being at frame 0, coincident with frame 1, the balance of the system can be determined by the differences in position of frame 3.

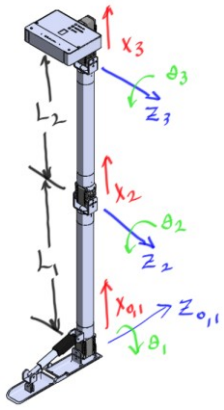

FIGURE 2. Frame Assignment for the Robotic Leg.

To determine the actual position of frame 3, the Denavit-Hartenberg parameters, seen in ***TABLE 2***, are used to find the transformation matrices for each joint. By finding the matrices that define the transformation of the system from frame 0 to frame 3, the position of frame 3 can be expressed as a function of θ_1 and θ_2, as seen in (1).

TABLE 2.
Denavit-Hartenberg Parameters for the System.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$-90°$	L_1	0	θ_2
3	0	L_2	0	θ_3

$$\begin{bmatrix} \text{Height} \\ \text{Mediolateral} \\ \text{Anteroposterior} \end{bmatrix} = \begin{bmatrix} \cos\theta_1\,(L_1 + L_2\cos\theta_2) \\ \sin\theta_1\,(L_1 + L_2\cos\theta_2) \\ -L_2\sin\theta_2 \end{bmatrix} \quad (1)$$

With this the position of frame 3 in space can be determined based on angle feedback from the servos. The transformation matrices found here are also used in further calculations.

IV. THE DESIRED ANGLES

To determine the desired position for system balance it is important to know how it is the system balances. Due to the complexity of the problem this is no easy task. The approach to this solution was to find equations that describe the balance of the system as a function of the offset of frame 3 in terms of θ_1 and θ_2 by experimentation.

The first step was to find angles that allowed the system to balance on a flat surface. Locking the servos to these angles, data could be recorded that define the balance conditions. This data included servo angle position, balance point position, and gyroscope offsets for pitch and roll. After finding the mode average the data was recorded and the experiment repeated for a total of 10 trials.

Mediolateral and anteroposterior data was similarly recorded, with a total of 20 angles, 5 in each direction. For mediolateral data, the servo at θ_1 was unlocked and θ_2 was locked at the vertical balance angle, allowing the robot to be moved into a position of balance. For anteroposterior data, θ_2 was unlocked and θ_1 was locked at the vertical balance angle. In total, 210 separate trials were recorded.

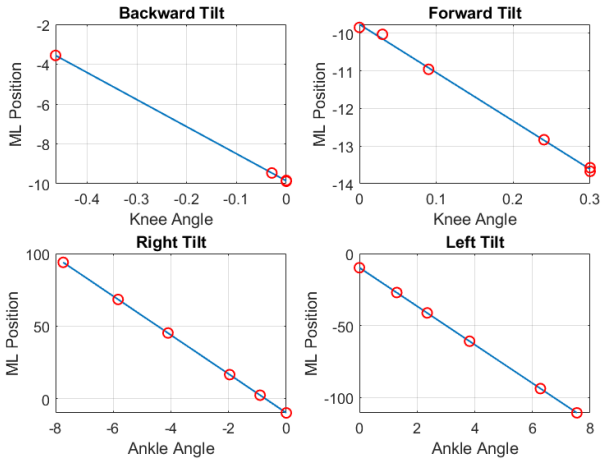

FIGURE 3. Mediolateral Position vs. Servo Angles.

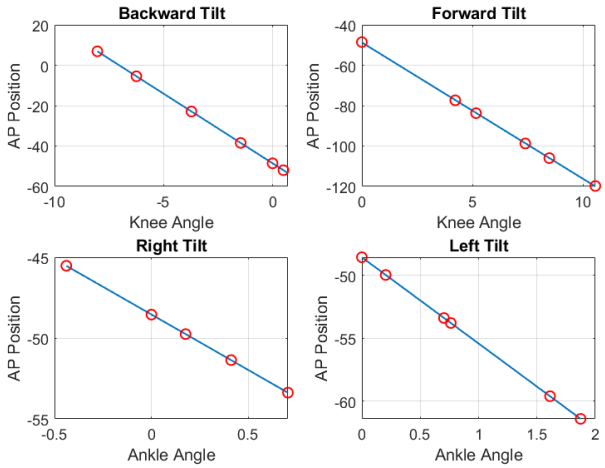

FIGURE 4. Anteroposterior Position vs. Servo Angles.

For each of the 21 positions of balance the mean average of the data was recorded. The functions for desired position were obtained by comparing the position values to the offsets of servo angles and finding best-fit lines. This was accomplished using MATLAB's *polyfit* and *polyval* functions, which allow you to find a polynomial given a set of data. These functions also allow you to specify the order of the resulting equations. With the order set to 3rd, it was clear that the results were very linear, as seen in ***FIGURE 3*** and ***FIGURE 4***, and so the order was set to 1st for simplicity and computation time.

$$\theta_2' = \operatorname{asin}\left(\frac{\mathrm{ML}_{\mathrm{Desired}}}{-L_1}\right) \tag{2}$$

$$\theta_1' = \operatorname{asin}\left(\frac{\mathrm{AP}_{\mathrm{Desired}}}{L_1 + L_2\cos\theta_2'}\right) \tag{3}$$

The equations that result are the desired positions from both the mediolateral and anteroposterior perspectives. That is, no equation satisfies both the mediolateral and the anteroposterior balance conditions. To attempt to satisfy both perspectives, the mean average of the two are taken. This can be problematic if one or more perspectives are far from the balance point, but for small offsets it should be near enough to a point of balance.

These equations allow for solutions to desired mediolateral and anteroposterior positions, but do not solve for a desired height position. To find this it is necessary to return to (1). Using the desired anteroposterior position to solve for a desired θ_2, seen in (2), then using the desired θ_2 to solve the mediolateral position for a desired θ_1, seen in (3), and substituting these into the height position equation gives the desired height.

$$\dot{\theta} = J_{\mathrm{V}}^{-1}\dot{V} \tag{4}$$

Equipped with the desired position it is now possible to find the desired servo angles. Equation (4) shows how a change in angle relates to a change in position via an inverse Jacobian matrix. Using the transformation matrices from earlier, the Jacobian can be found. However, the Jacobian is non-invertible and so cannot be used. This problem is avoided by utilizing the pseudoinverse Jacobian (see Appendix A), found using the *pinv* MATLAB function. The resulting values for $\dot{\theta}$ are then added to the current servo angles to reach the desired position.

V. THE LQR CONTROLLER

Dynamixel servos, such as those used in this project, are fast and accurate. However, their speed comes at the cost of a slight overshoot which may contribute to balance instability. To reduce the overshoot a controller needed to be implemented. A linear quadratic regulator (LQR) was chosen because it allows you to design a controller based on desired performance.

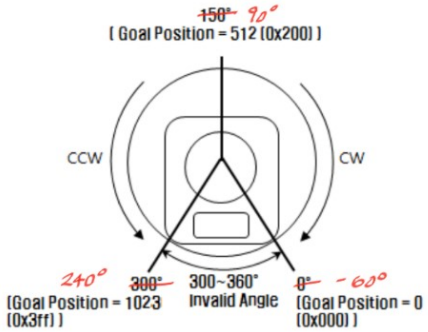

FIGURE 5. Servo Positions vs. Degrees Comparison Chart.

The first step of designing the LQR controller was identifying the system. A code was written to the microcontroller which set the servo to position 0. It then commanded the servo to move to position 512. As the servo moved to the new position it would output its current position as well as the current time every 10 milliseconds. Saving this data in a CSV file and importing it to MATLAB, the servo position data was then divided by 512. Plotting this against the time data resulted in a unit step response for the servo.

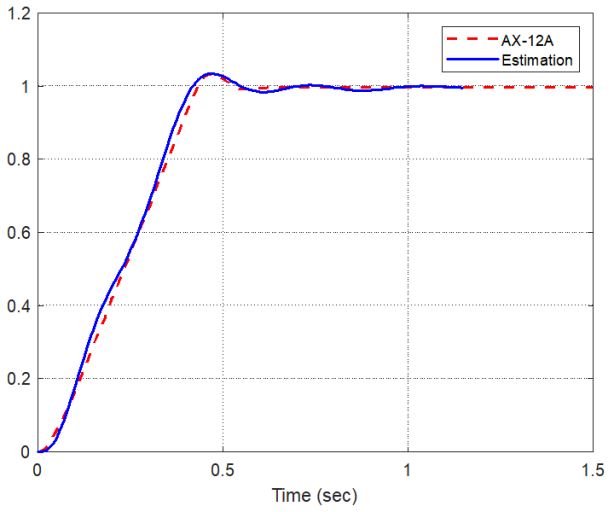

FIGURE 6. Servo Unit Step and System Estimation Comparison.

Using this data with MATLAB's *System Identification Toolbox*, an estimated 5th order transfer function was generated which was 97.91% accurate to the original unit

step function (see Appendix A). A comparison between the original unit step function and the estimated unit step function can be found in ***FIGURE 6***. Using MATLAB's *tf2ss* function, the state space representation was calculated as seen in ***TABLE 3***.

TABLE 3.
LQR Controller State Space Variables

Variable	Value
$\boldsymbol{A}$	$\begin{bmatrix} -31.54 & -952.9 & -15010 & -150900 & -697100 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$
$\boldsymbol{B}$	$[1 \quad 0 \quad 0 \quad 0 \quad 0]^T$
$\boldsymbol{C}$	$[0 \quad 0 \quad 2828 \quad -2665 \quad 694900]$
$\boldsymbol{D}$	0
$\boldsymbol{L}$	$\begin{bmatrix} 1.27 \times 10^{-13} \\ -1.03 \times 10^{-15} \\ -1.37 \times 10^{-16} \\ 3.68 \times 10^{-18} \\ -4.09 \times 10^{-19} \end{bmatrix}$
$\boldsymbol{Q}$	$\begin{bmatrix} 10 & & & & \\ & 10 & & & \\ & & 10 & & \\ & & & 10 & \\ & & & & 10 \end{bmatrix}$
$\boldsymbol{R}$	0.1
$\boldsymbol{LQR_{GAIN}}$	$[4.6137 \quad 106.16 \quad 1185.9 \quad 5382.4 \quad 0.0001]$

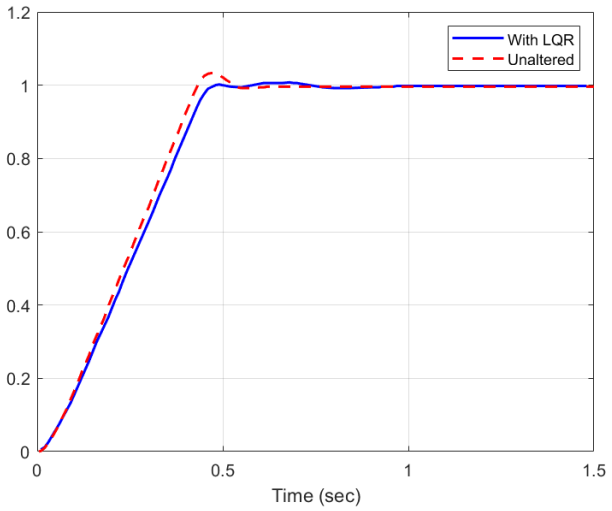

FIGURE 7. Servo Unit Step and LQR Controlled Output Comparison.

After checking the system for observability and controllability, an observer gain was calculated using uncontrolled system's eigenvalues due to a lack of any desired characteristic equation. Choosing values for the LQR controller variables Q and R began with setting $R = 1$ and Q equal to a 5-by-5 identity matrix. These numbers were then altered until the output curve satisfied the desired shape. The resultant curve has a lower overshoot at the cost of a slower settling time (***FIGURE 7***), which may be better for balancing.

VI. THE ALGORITHM

The algorithm closely follows the flow diagram in ***FIGURE 8***. It begins by saving the values of the pitch and roll from the gyroscope at the balancing point. The current position is then derived from the current servo angles reported by the servos. Ideally, these are offset by the same values as the pitch and roll.

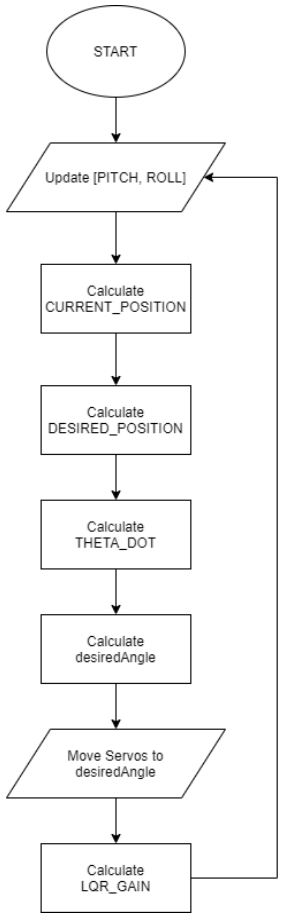

FIGURE 8. Flow Diagram for the Balance Algorithm.

The desired position is then calculated based on the pitch and roll readings. This desired position is then used to calculate $\dot{\theta}$, the change in angle required to reach the new position. Desired angle is then calculated as the current angle plus the change in angle, minus the LQR gain, as seen in (5).

$$\theta_{\text{new}} = \theta_{\text{current}} + \dot{\theta} - LQR_{\text{Gain}} \tag{5}$$

After the new angle is calculated, the corresponding servo is sent the command to move to this angle. After this, the LQR gain is calculated based on current servo angles. At this point the loop begins again. It should be noted, for each moving servo a separate version of these calculations is being handled concurrently. It should also be noted that on the first run of this algorithm the LQR gain is initialized to 0, and as such has no effect on the first new angle. LQR gain is calculated last specifically for iterative purposes, as the LQR gain of the current iteration is meant to be applied to the new angle of the next iteration as a means of system feedback.

VII. RESULTS AND CONCLUSION

The future of bipedal humanoid robotics depends on the ability of these robots to balance in non-ideal circumstances. To this end algorithms must be developed to achieve this. This project has demonstrated a way to handle position correction via pseudoinverse Jacobian. It has also demonstrated the effectiveness of an LQR controller for smoothing the transition between these positions to maintain balance. The algorithm itself works as intended and correctly attempts to correct the position of the robot, however the robot does not entirely work in practice due to a lack of foresight in choosing servos. Although the AX-12A servos are accurate and easy to use, this particular model has a relatively small stall torque. This causes the robot to fail in reaching the desired angle, often resulting in either lack of movement of wild over-correction.

Another problem often encountered in this project was the inconsistency of the gyroscope readings. The code accounts for the cases where the gyroscope reports a single extreme error, but it can not account for the small errors or situations where the same position would give different results. Future work might include a new design with more power servos. It could also be beneficial to explore more sophisticated mathematical models for calculating points of balance as well as a more thorough statistical analysis for the distribution of points of balance in space. These could include a real-time calculation of the sum of force vectors, for example.

REFERENCES

[1] S. M. Bruijn and J. H. van Dieen, "Control of human gait stability through foot placement," *Journal of the Royal Society Interface,* vol. 15, no. 143, Jun 2018.

[2] J. W. Grizzle, C. Chevallereau, R. W. Sinnet and A. D. Ames, "Models, feedback control, and open problems of 3D bipedal robotic walking," *Automatica,* vol. 50, no. 8, pp. 1955-1988, Aug 2014.

APPENDIX A – LARGE EQUATIONS

Pseudoinverse Jacobian:

$$J_V^{-1} = \begin{bmatrix} \dfrac{-\sin\theta_1\,(L_1 + L_2\cos\theta_2)}{L_2^2\cos^2\theta_2 + L_1^2 + 2L_1L_2\cos\theta_2 + 1} & \dfrac{\cos\theta_1\,(L_1 + L_2\cos\theta_2)}{L_2^2\cos^2\theta_2 + L_1^2 + 2L_1L_2\cos\theta_2 + 1} & 0 \\ \dfrac{-\cos\theta_1\sin\theta_2}{L_2} & \dfrac{-\sin\theta_1\sin\theta_2}{L_2} & \dfrac{-\cos\theta_2}{L_2} \\ \dfrac{-\sin\theta_1}{L_2^2\cos^2\theta_2 + L_1^2 + 2L_1L_2\cos\theta_2 + 1} & \dfrac{\cos\theta_1}{L_2^2\cos^2\theta_2 + L_1^2 + 2L_1L_2\cos\theta_2 + 1} & 0 \end{bmatrix}$$

Estimated AX-12A Transfer Function:

$$G(s) = \frac{2828s^2 - 2665s + 6.949 \times 10^5}{s^5 + 31.54s^4 + 952.9s^3 + 1.501 \times 10^4 s^2 + 1.509 \times 10^5\, s + 6.971 \times 10^5}$$

APPENDIX B – CIRCUIT SCHEMATIC