Active Reflex-Based Balancing for Small Humanoid Robots

A thesis presented

by

Sara McGrath

to

The Department of Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Computer Science

The University of Manitoba

Winnipeg, Manitoba

March 2007

Thesis advisor

Jacky Baltes

Author

Sara McGrath

Active Reflex-Based Balancing for Small Humanoid Robots

Abstract

For any practical purpose, humanoid robots must be able to maneuver over a variety of terrains, at different speeds and with varying gaits. Sensors must be used to make sense of the surrounding environment, enabling robots to balance as they move. Though motion and pressure sensors exist, their input is just beginning to be used to dynamically balance gaits, and they are often used in conjunction with other sensors, limiting the knowledge obtained about each sensor. In this thesis, I implement three simple balancing algorithms on a robot equipped solely with an accelerometer to study the utility of simple algorithms and a single sensor in balancing. The basic threshold algorithm proves the most effective overall. The algorithms are able to balance for simple tasks, but as the balancing required becomes more complex (i.e., controlling multiple joints over uneven terrain), the need for more sophisticated algorithms becomes apparent.

Contents

List of Figures

Acknowledgments

Thank you to my family and friends, for all your prayers and support.

Thanks to Schawn Schaerer and Dominic Leung for help in building and modifying the robot, and Kevin Lambrecht for helping with hours and hours of low level testing. I know it wasn't that exciting, but it was a great help to me.

Thanks also to my advisor, Dr. Jacky Baltes in particular, and the Autonomous Agents lab in general for your expertise, help, and the coffee breaks. I couldn't have done it without you guys!

To Jonathan, for restoring and refreshing me, thanks always. I love you!

Soli Deo Gloria.

For my parents.

Thank you.

C The coronal plane of movement, found by bisecting the body from ear to ear. Movement out to the left or right side (eg, swinging the leg out from the body directly to one's right) is in the coronal plane.

COM The Center Of Mass, a weighted average of the mass of an object.

COP The Center Of Pressure, a weighted average of the pressure upon of an object.

CPG Cyclic Pattern Generator — a function that creates a series of control points that make up a gait that can be repeated through.

DOF Degree of Freedom: a joint that can rotate in only one plane.

H A hybrid correction method based on a minimal correction until a threshold point, and a percentage based correction thereafter. This method is a hybrid of the Threshold and PID methods.

LAC Left Ankle Coronal Joint

LAS Left Ankle Sagittal Joint

LKS Left Knee Sagittal Joint

P A standard control algorithm based on the PID algorithm, but only correcting by a Proportion of the error.

PD A standard control algorithm based on the PID algorithm, but only correcting by a Proportion of the error, and a percentage of the Derivative error.

PID A standard control algorithm based on correcting by a Proportion of the error, a percentage of the Integrated error, and a percentage of the Derivative error.

RAC Right Ankle Coronal Joint

RAS Right Ankle Sagittal Joint

RKS Right Knee Sagittal Joint

S The sagittal plane of movement, found by bisecting the body from nose through to the back of the head. Swinging the leg forwards and backwards is movement in the sagittal plane.

SAE Sum of the Absolute Error, calculated by subtracting the baseline or threshold from the actual error, and summing the absolute values obtained.

Static Balancing Balancing that maintains the COM within the support polygon.

T A control algorithm that makes a minimal correction every time the sensor error crosses a threshold.

XAS The LAS/RAS joints which control the ankles in the X plane.

XKS The LKS/RKS joints which control the knees in the X plane.

YAC The LAC/RAC joints, which control the ankles in the Y plane.

ZMP Zero Moment Point, the point at which all moment acting upon an object sums to zero.

Chapter 1

Introduction

For humanoid robots to move from fantasies such as Terminator or *The Jetson*'s Rosie to reality, they must be able to move over a variety of terrain with different speeds and gaits. Useful humanoid robots must be able to move about freely, to go where they need, to recognize items, and to understand commands. My focus is on the first of these challenges: humanoid robot mobility. Currently, many humanoid robots do not react to their environment when moving through it, severely limiting their movement. My research focuses on allowing feedback from the environment to be added back into the gait control, focusing on humanoid robots. In this thesis, the term 'robots' will be used to indicate humanoid robots in particular. Developing such control systems for the walking and balancing of humanoid robots is a hard problem. It becomes an even more difficult one when implemented on physical robots, but such control systems have enormous potential benefits.

In today's humanoids, changing between a hardwood and carpeted surface can call for re-calibration of the gait to account for the extra absorbency of the surface. To be

able to compensate for changing surfaces, humanoid robots require sensor feedback, such as that from gyroscopes, to be integrated into their walk pattern. This will apply to service robots working in human environments [Hirai et al., 1998], as well as humanoids on more challenging terrain, such as disaster areas, and military scouting. These control systems also may help in developing human prosthetics.

Few humanoids today use sensor feedback for active balancing in their control systems. Senchans from Osaka University, Japan is one of the few robots to demonstrate this control in open competition [Rawichote et al., 2004]. Other participants in the same competition, such as Robo-Erectus [Shou et al., 2004] and Foot-Prints [Okamoto and Okamoto, 2004], used no feedback. Honda's ASIMO [HONDA, 2003] and NE's ARNE [Iakovlev and Kritchoun, 2004] robots also incorporate sensory feedback in their controls, but with robots so expensive that very few researchers anywhere can follow their lead. Most sensory feedback systems exist in simulation. Fewer have been transferred successfully to actual robots, especially cheap robots. Though most humanoid robots have multiple sensors such as gyroscopes, accelerometers and pressure sensors, they are rarely used in implementing a walking gait [Wyeth and Kee, 2004]. Developing a basic working walk is difficult enough.

If it is so difficult to make a robot walk, why bother? Wheeled robots appear good enough, at first glance. Certainly we have all-terrain vehicles that can traverse most ground. However, try to imagine such a vehicle climbing stairs. Such specific problems can be overcome, but the general issue remains: we live in environments designed for human bodies. While we could construct wheeled robots capable of climbing stairs, or dealing with most tasks in a house, it would become very complex very quickly as we

had to add specific solutions for specific problems. General solutions are preferable, and humanoid robots are an excellent general solution for human domains. Further, while we know the human body is not the best one for any particular task (cheetahs can outrun us, monkeys out-climb us, dolphins out-swim us), humans are reasonably good at all of these tasks. The human form is quite versatile — it makes sense to utilize this knowledge rather than designing another form and so, perhaps, reinventing the wheel.

Walking is foundational to achieving any movement based goal: a robot cannot fetch a cup of water without being able to move to where the water and the cups are, and then returning to its origin. Complex tasks such as an obstacle run (avoiding multiple spots) or a treasure hunt (walking to a desired spot) depend on the basic movements of walking, turning and stopping. Further, the researcher should assume that the robot will walk on differing surfaces: human dwellings are full of stairs, ramps, and even bumps that must be avoided, or compensated for. Creating a walk capable of dealing with all these factors is no easy task.

To a human, this is something learned in early childhood, and too simple to bear thinking of. Indeed, humans may not be able to recognize all the learning accomplished to walk, due to the many distributed connections made while learning. For a robot, the difficulties are endless. The first problem is recognition: how do you recognize stairs or ramps or surfaces? Visually? By the "feel" - using pressure sensors? At what point should compensation begin, or end? How quickly should the corrections begin? How often must the corrections be applied? Is the robot walking upstairs, or down? Is the surface carpet or concrete? Is the robot's foot caught, or is

it an obstacle to step over? Can the same types of corrections be used for all these changing surfaces, or are different ones needed for each challenge? If so, when is each one used, and when is it time to switch to a difference one? Can you correct for more than one thing at the same time, or must you make adjustments to the corrections to allow multiple corrections? Are the algorithms and sensors in use capable of handling the balancing? Is the robot?

1.1 Motivation

Humanoid robots at present do little in the way of active balancing. The technical challenge at RoboCup 2005 involved a section of uneven terrain, requiring robots to navigate a field made up of hexagon blocks, with at most 3 mm differences between each block in the field. The only robot who managed to cross this field was Team Osaka's Vision robot, and it completed the task without reference to any balancing sensors, but simply a very fine tuned walk and foot extensions to lessen the slope [Yamato et al., 2006]. Later competitions, such as FIRA's HuroSot Lift and Carry, require robots to cross a similar playing field while carrying greater and greater weights. Humans crossing uneven terrain, with varying weights, would simply adjust their gait, walking slower, adjusting ankle and knee bending to compensate for the terrain. This means that humans find little difficulty in going for hikes in wilderness terrain, while a robot (especially a humanoid one) would find this a nearly impossible task. If robots are to be generally useful, they must be able to cope with terrain that varies throughout their environment. One would hardly expect a factory or schoolroom floor to be kept spotless so robots could traverse it, or all floors to be built

perfectly level in order that robots could navigate them. A simple ramp should not present a formidable obstacle to a humanoid robot, and presently it does for most of them.

Integrating dynamic balancing into robots will allow them to not only deal with these changing surfaces, but also allow them to compensate for sudden changes in their equilibrium. If a robot has no balancing compensation when pushed or shoved, or perhaps just hit by a particularly strong gust of wind, only the torque on the motors keeps the robot upright. Once that torque is overcome, the robot will fall. While I am not suggesting that robots should be expected to deal with pushes and shoves on a regular basis, it does happen, and adapting to it should not be a major concern. Robotic parts may wear out, causing motors to fail, and throwing the robot off-balance, and the robot should be able to adapt for this. Further, compensation for a suddenly changed center of mass could also result simply from adding new hardware to the robot, or having the robot carry a payload. The equivalent of adding a full backpack onto a robot could cause it to fall, and would strain the motors such that it would be significantly more likely to fall from normally withstandable forces.

Adding balancing to humanoid robots is then essential. What algorithms should be used to add this balancing? Standard algorithms involve calculating the center of gravity of the robot at the very least, suggesting that they will be less useful when objects of unknown weight are added to the robot. Is there a more general solution to balancing, that will involve less exact calculations of what weight is located where on the robot? Is there an algorithm that won't require as many calculations specific to the robot it is implemented on? How far can a simple algorithm go toward useful,

active balancing? This thesis will attempt to answer these questions by implementing simple algorithms tightly coupled to sensor loops to investigate these questions.

Finally, humans themselves use multiple sensors to balance: vision, position/force feedback (muscle feedback) and tilt/acceleration sensors (inner ear organs). Force feedback and motion-based sensors have been used by many researchers in combination, but very rarely have motion-based sensors been investigated on their own. Yet for humans, these are crucial. Humans can walk on stilts (removing most of the muscle feedback), or in ski boots; they can walk in the dark; but it is very difficult to walk without motion sensors. To test this, spin in a tight circle for 30 seconds, and try to walk! Investigating motion sensors on robots may help us in understanding how to use them to help humans as well. Further, most robots use multiple sensors to balance without having investigated each one separately. It is very possible that effects of the various sensors are being masked or ignored because of the other sensors.

The logical solution is to look at each sensor individually to determine the boundaries of its usefulness. These sensors add an extra level of complexity to the problems of walking and balancing. By throwing many sensors at a problem, one both complicates the problem and clouds the contribution of each sensor. By adding one sensor at a time, we can better understand the full ramifications of the data that we can glean from that sensor. Thus, part of my goal in this research is to use only one sensor at a time, until the limits of the data that sensor provides have been reached.

This sensory data should be integrated into tightly actuated control systems, in much the same that humans use reflexes to control many of their own movements. Human balancing is not something consciously done by most people, especially for

areas such as ramps, or mildly uneven terrain such as gravel. A robot therefore should similarly be able to compensate for at least basic disturbances without specialized motion planning.

Finally, a cheap humanoid robot should be used as the test platform. While kit robots, or expensive well-built robots are easy to work with, and provide a useful testbed, they also encourage dependence on the robotic hardware. With a kit robot, you can depend on the robot not bending or changing, the motors staying in the position you expect, and a generally excellent performance. This means that there will be little variation in the gaits or the use of the robot. While this is definitely easier on the researcher, it does not encourage robust algorithm development, as there will be few difficulties in using the robot. By comparison, using a cheap robot not only keeps the price down to where hobbyists can join in on the research [Mueller, 2006], but also encourages development of robust, general algorithms. This is not to imply that a poorly constructed robot will be created, merely that it will be less precise or stable than a kit robot would be. As the robot is less precise, and thus more prone to errors creeping in physically, the algorithms must then be able to deal with the imprecision, naturally leading to more robust algorithms.

1.2 Terminology (or Background)

Before I go any further into any means of dealing with these problems, it will be useful to review some background knowledge.

Robots are mechanized constructs that can be programmed to move, or more generally to perform a specific task. Humanoid robots, intuitively, are those robots

that look like humans (or miniature ones). There are three main types of human-like robots: those with torsos (generally including arms and head as well), those with two legs (bipedal) and those with both legs and heads. Humanoid torso robots tend to focus on vision, cognition, and speech, while bipedal robots are more engaged in locomotion, and movement generation. Fully humanoid robots, with legs and torsos, can focus on any and all of these problems, as well as the problems that arise once torso and legs have been integrated, such as path planning and obstacle avoidance. A common benchmark problem for humanoids is walking.

Humanoid walking gaits are usually generated by central pattern generators (CPGs) [Ijspeert, 1998]. Each joint moves along a cyclic control curve with a certain period. These curves describe the positioning of each joint at a particular point in time along the movement pattern. The larger the change in the curve, the greater the speed of movement. Figure 1.1 shows such a set of control curves — a pattern — used to make Tao-Pie-Pie, a minimal humanoid [Baltes et al., 2006], walk. Each line represents the angle held by the joint at that moment in the pattern.[1] For instance, the bottommost line in Figure 1.1 shows the positions held by the left knee, over a period of 100 time units.

The control curves dictate the movement the robot makes for only a short period of time. Looking at someone walking, we can notice that they take a step with one foot, then with their other foot, and repeat. In the same manner, CPGs endlessly cycle through the core components of a gait by continually repeating a smaller pattern. A gait may be limited to preprogrammed movements described by a pattern, or may

[1]Figure 1.1 was created using linear interpolation, though other methods such as cubic splines have also been found to be effective [Huang et al., 1999].

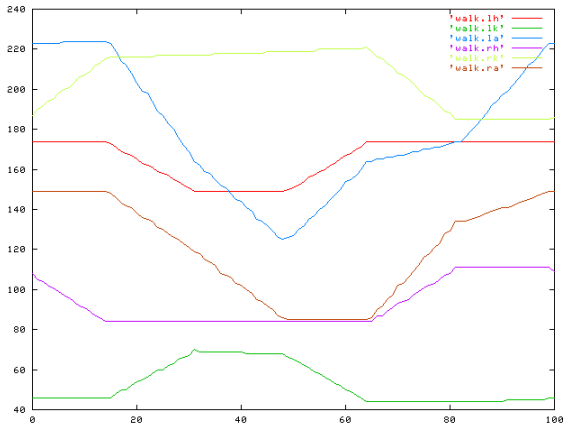


Figure 1.1: Set of control curves for Tao-Pie-Pie, a humanoid robot with six DOF in his legs

incorporate other control measures as well.

Because each human joint may move in up to three different planes (x, y, z), robots are classified by DOF (Degrees of Freedom) or number of robotic joints (each robotic joint moves along one axis only) [Craig, 2005]. Each DOF represents movement in one plane: sagittal, coronal or transversal (see Figure 1.2). A robot with an ankle that can only bend forwards and backwards (frontal plane) has one DOF in the ankle, while a robot with an ankle that can also bend from side to side (lateral plane) has two DOF. More accurately, then, control curves describe the position that a DOF has — it may take two (eg., ankle frontal and lateral) or three (eg., hip frontal, lateral, and transversal) curves to accurately describe the movement of one human joint. Each DOF is usually controlled by a single actuator. Thus, each control curve is the position that a given motor must hold at a given moment in time. On Lillian, the thesis platform detailed in Chapter 3, each joint is uniquely identified by a three letter acronym consisting of the leg the joint is on (L or R), the joint being controlled (A: ankle, K: Knee, H: Hip), and the plane in which the joint is moving (C: Coronal,

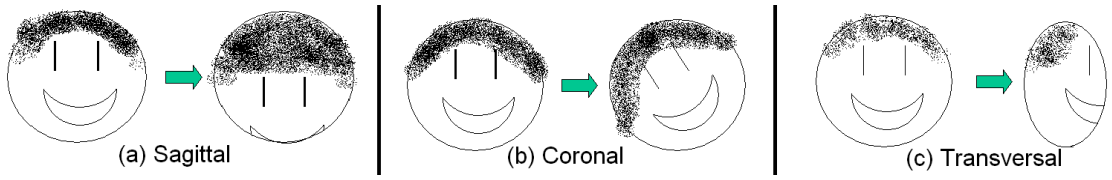


Figure 1.2: Three axis of rotation

S: Sagittal, T: Transversal). The three most important joint sets in this thesis are often referred to simply as YAC: the LAC/RAC joints, which control the Y plane; XAS: the LAS/RAS joints in the X plane, and XKS: LKS/RKS which controls the knees in the X plane.

By increasing or decreasing the setting of a DOF, the corresponding servo will extend or retract the joint. Thus, in the left knee servo curve from Figure 1.1, the control curve tells the knee to remain stable, then extend, stabilize, retract and remain stable again.

Each curve is made up of a number of control points, with interpolations between them to create a prescribed pattern of movement for the robot. Adjusting control points allows the problem to be simplified to a manageable level, while still being accurately solvable. Generally, static sets of control curves are used to create control points for moving through the robot's gait pattern, allowing the patterns to be greatly compressed, yet easily retrievable. This is especially important, as robots are often severely limited in the amount of processing power and memory at their disposal.

Though some robots use an offboard processing on a remote computer for some or all of their processing needs, most competition-based robots use only the processor on board the robot. All humanoid robots at the University of Manitoba rely solely

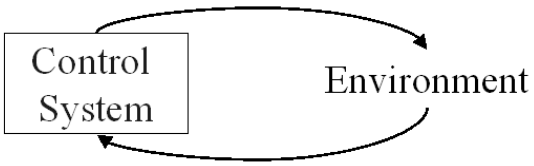


Figure 1.3: Closed-loop control

on onboard processing power.

In addition to the DOFs inherent in the robot's structure, most humanoids have several sensors. Force feedback sensors, inclinometers, accelerometers and gyroscopes are among the most common. Feedback sensors give a reading of either the force at a particular position, or the torque (generally at a specific joint). Inclinometers read the angle the robot is currently leaning at, while accelerometers measure the robot's current acceleration. Integrating the accelerometer results calculates estimates of the current angle and position. Gyroscopes read the angular velocity at which at robot is rotating. The velocities are often then transformed into acceleration readings, or inclination angle.

Once we have reached the point of adding sensory feedback, we can classify humanoid walking gaits into two categories: open-loop and closed-loop (see Figure 1.3). Open-loop control is a static control of the robot, with no input from the outside environment. A robot may be walking up a slope, but will never know, as no feedback from the environment will ever enter the control loop. To add this feedback to the control is to close the loop — creating a closed-loop control system.

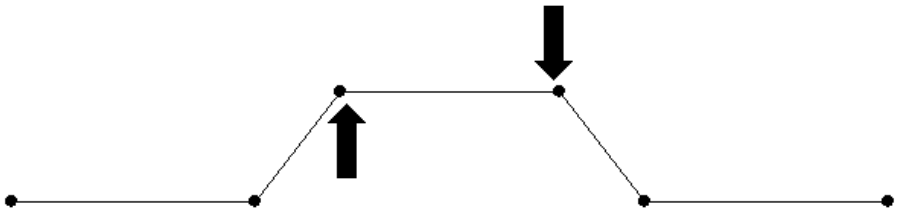


Figure 1.4: Up-and-down (target position) manipulation of a control curve

1.3 Methodology

The means of closing the loop in general means that instead of using a completely static set of control curves, the control points may be moved to change the target position for a joint during the walk pattern. (Fig 1.4) This allows the robot to dynamically control the amount each joint must move by increasing or decreasing the size of the control curve. This can be used, for example, to adjust the ankle tilt during the walk, as the environment of the humanoid demands. (Fig 1.5) Laboratory tests at the University of Manitoba have shown that this dynamic control is better able than the static one to handle changes in the weight distribution of the robot (i.e., the robot picks up an object). At FIRA 2003 [FIRA, 2003], a basic threshold controller using only gyroscope data was able to adjust to poor flooring better than the static walk.

This approach is different from the standard approach to balancing. Usually, a model-based approach is used. The researcher creates a mathematical model of the robot to work on, and a control algorithm is then implemented and tested on the model (often in simulation). Finally, the controller is moved to the physical robot. Often, some readjustments are required at this point to transfer the controller.

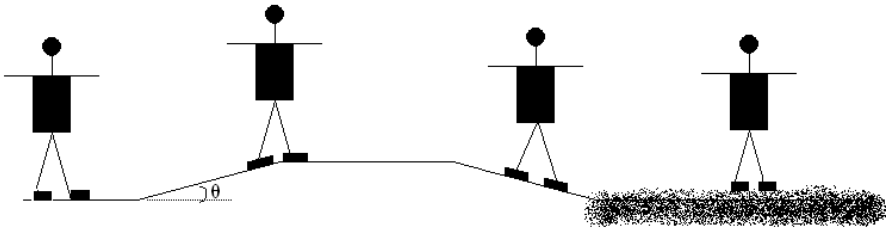


Figure 1.5: Closed loop control allowing robot to maneuver over inclines and rough terrain.

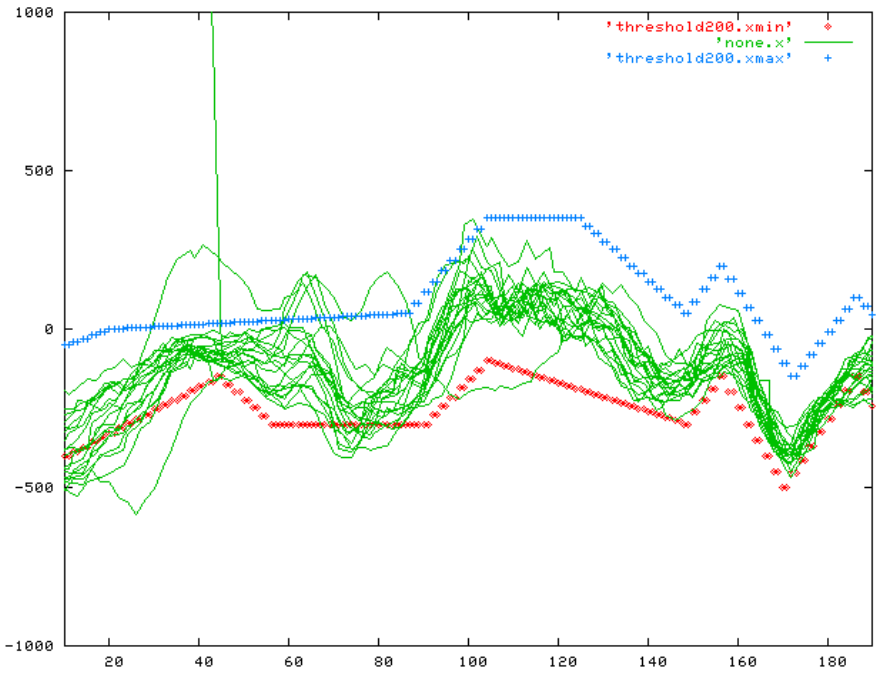


Figure 1.6: Gyroscope readings with boundary lines shown from Tao-Pie-Pie.

This approach is physically-based. Instead of creating a model of the robot, the robot itself is used as a test platform. This removes the necessity of adjusting the control strategy for the robot, as the testbed is the most accurate one possible. Further, testing on the robot means that results will be extremely accurate, as there are no factors unimplemented due to programming or simulation difficulties. The drawback is in maintaining the robot and the length of time required for each test.

The methodology used for this research is to begin with a pre-existing gait, and then modify that gait to improve it. Thus, our control algorithms are added to the gait already in place, allowing the results of each strategy to be directly compared with each other and the uncorrected walk on the real robot.

1.3.1 Algorithm Development

Dynamic balancing begins by recording stable and unstable sensor readings to create boundary lines between the two. For the cheap sensors the robotics lab generally uses, bad (outlying) data is a constant problem. A running average is used to soften the effects of outliers while still presenting the data reasonably accurately, and a zero point is calculated to adjust readings to be comparable throughout multiple trials. As is noticeable in Figure 1.6, the mass of gyroscope readings band together, and it is easy to create boundaries containing most of the readings. Error can then be easily determined as the amount by which the reading oversteps the boundary. In a dynamically balanced walk, sensor readings outside these boundaries will then trigger corrective actions in the robot. Figure 1.6 shows an actual reading of Tao-Pie-Pie's gyroscopes for about 20 steps. The thick lines on the chart are the boundary lines;

Figure 1.7: Side-to-side (time) manipulation of a control curve.

the many thin lines are the gyroscope readings, using a running average to reduce outliers. Note that most readings are within the boundary lines; only a few (in this graph) require corrections. Correction measures may differ based on the boundary zone broken, and the point in the pattern it is broken. For example, if a robot's gyroscope showed an unstable reading such as too fast to the right, the corrective measure triggered would be to tilt the right ankle toward the centre to slow the velocity with which the robot moves toward the right. The relative time within the cycle at which this correction is made is important. For instance, a tilt in the right ankle would only slow the movement if that foot is on the ground.

The methods used to calculate the correction measures can vary greatly. In this research, a simple threshold method previously developed is compared against the standard PID control (see Section 2.3 for further details), and a hybrid version. Threshold (T) corrections are minimal, triggered by a broken boundary, while PID corrections are relative to the amount of error. Hybrid (H) corrections are minimal up to a threshold, at which point they increase relative to the error. Each of these methods must be tuned to find the best thresholds and correction values.

This correction tuning is further complicated by adding in multiple planes to

correct. Making corrections based on sensor information in one plane at once is relatively easy, hence why most studies begin testing on only a single plane. Adding in a second or third plane includes extra noise, created by any overcorrections or missed corrections in the first plane, or simply from the interactions between the two planes. This can lead to corrections rebounding on each other in multiple planes, quickly causing oscillation. Thus, corrections are implemented and tested first in one plane before moving on to multiple planes of simultaneous correction. Further, not all tasks are equal in balancing: standing upright on a tilting surface is easier than balancing during walking, when shifts in the robot's COM are required to enable the robot to move. Thus, tuning must begin with simple tasks in each plane before moving on to more difficult ones.

A system that could correctly and dynamically balance in multiple planes at once is of course the end goal of this research. But to implement it with only accelerometer feedback may well be impossible. Certainly, it would be much more human and therefore, make for a more appropriate humanoid, to add in other sensors as well. However, this should only be done once the initial control system for the sensor has matured. Then other sensors should be added incrementally, to ensure that as much data as possible will be gained from each sensor. This will leave behind a working physical robot that has integrated multiple sensors to provide a closed loop control, as well as the knowledge of how to implement such techniques on an actual robot.

1.4 Problem Description

I am interested in implementing a closed loop control system on small inexpensive humanoids, in small incremental stages to reduce the complexity of dealing with additional sensors and actuators. Due to the difficulties of actually integrating feedback data into a dynamic walk gait, most researchers end up ignoring multiple sensors on their robots while integrating only one, or incorporating only simplest of feedback from the sensor. They thus waste money and processing power on equipment that is never used. Therefore, a more iterative approach — dealing with small changes at a time — is appropriate for this research. Simple control methods are used to incorporate the sensory data into the walk gait with tightly coupled feedback loops, creating balancing reflexes.

1.5 Research Questions

The problem description above leads to the following research questions:

1. Can a balancing reflex that is a tightly coupled feedback loop implemented on a single motion sensor dynamically balance a small humanoid robot in real-time? If so, can the robot measurably improve its gait with this?

2. What balancing reflex should be used to dynamically balance a robot? What are the strengths and weaknesses of the various algorithms? How does the algorithm previously developed for the gyroscope compare to a standard control algorithm?

While examining these two main research foci, several other secondary research questions can also be investigated.

1. How can a researcher evaluate the effects of a particular algorithm on a humanoid robot's balancing?

2. How far can a single motion based sensor be taken in balancing?

3. Can a robot actively balance using only an accelerometer?

1.6 Summary

Humanoid robotics is a challenging and interesting field that requires more research into balancing to improve the robots' utility and applicability in multiple tasks. This research should be done in smaller steps when integrating sensors, with each one fully explored before another is added. Thus, this research will concentrate on accelerometers. Several simple algorithms will be implemented to test their usefulness as a control mechanism. These algorithms all control the manipulation of set points in the robot's gait, investigating the effectiveness of the accelerometer as well as the algorithms in dynamically balancing a cheap, minimal humanoid robot.

This chapter has covered the motivation as well as the background necessary to understand the research. The minimalist philosophy that drives the methodology used to answer the research questions stated here was given, as well as a basic overview of the methodology itself. These research questions spring naturally from the problem description, and give a basis for the rest of this thesis.

1.7 Thesis organization

The remainder of this document is organized as follows:

1.7.1 Chapter 2: Related Literature

Chapter 2 shows the related research in the field. The different sensors, their most common uses, and the control algorithms used on them are examined. These are explored in further detail by looking at a number of robots and the balancing techniques implemented on them.

1.7.2 Chapter 3: Building Lillian

Chapter 3 details the steps involved in building the robot used as a platform for this research, mounting the sensors and the reasoning behind the design choices. The robots that inspired Lillian are discussed, as well as the robotic progression that led to her creation. Any modifications found necessary during the building and testing of the robot are also discussed and documented.

1.7.3 Chapter 4: Implementation

Chapter 4 discusses the basic control of the robot and the sensor. The low level control of the actual servos, the mid level interpolation between set positions and the high level planning and correction of Lillian's position are all detailed. Further, the accelerometer is similarly discussed: the low level sensor readings, the mid level data processing and manipulations, and the high level method for making movement

corrections based on that data are all explained.

1.7.4 Chapter 5: Methodology

Chapter 5 details the various correction methods used in this work: T (threshold based), PID, and H (hybrid). The means of tuning each of these algorithms are discussed, and then methods for tuning. The tuning is the most involved portion of the process, and a method for incorporating the corrective method into various gaits and planes is given. This method starts with the simplest gait (a stand) in a single plane, and moves up to a walking gait in both planes, with intermediary steps along the way.

1.7.5 Chapter 6: Evaluation

Chapter 6 describes my methods of evaluating the balancing algorithms. The various experiments to measure Lillian's balancing are described in detail, along with the parameters of each experiment. The various experiments use differing walking gaits and uneven terrain to measure the dynamic balancing ability.

1.7.6 Chapter 7: Conclusions and Future Work

Chapter 7 explores the conclusions reached, and suggests future work that may be undertaken. The conclusions examine the benefits and drawbacks to each corrective method implemented, and the feasibility of balancing using only an accelerometer. The limitations of this research, and ways to overcome those limitations in future, are also explored.

Chapter 2

Related Literature

Balancing a robot is becoming a much more significant problem as interest in humanoid robotics increases. Unfortunately, there are relatively few papers discussing dynamic balancing on real robots in print ([Kim and Oh, 2004; Ogino et al., 2004]) (though many on simulation), or on the websites of various labs ([University of Western Australia; Changjiu Zhou; HONDA, 2003; Osaka University]). Knowledge of different approaches implemented is most easily gained by talking to fellow competitors at competitions, such as RoboCup and Fira. [FIRA, 2005; RoboCup, 2005]

As with many other control systems, a common approach to suggesting balance mechanisms is in simulation. While in many domains this is adequate, it can produce interesting questions in applying these systems to robots. One such control system recently developed by Mu and Wu [2004] (and, in simulation, very good), does not account for friction. In real robots, friction is important enough that Kajita et al. [2004] have researched the question completely on its own. Thus, I will focus on approaches implemented on robots.

Before I discuss any particular robots, I will examine the general sensors and algorithms in use. I will then look at some robots being used in balancing research in more depth, discussing what sensors and algorithms are used in their balancing techniques.

2.1 Sensors

As with humans, robots have three main kinds of balancing aids [Dickman]. A human has muscle feedback, vision, and motion sensors in the inner ear. A robot also generally has the choice of force sensors (feedback, pressure, or actuator), vision, and angular sensors (inclinometers, accelerometers, gyroscopes). A combination of feedback and motion sensors is most commonly used: the classic combination is actuator feedback, pressure sensors and accelerometers. Vision sensors are much more rarely used to aid in balancing and walking (e.g., with an artificial horizon) especially as these techniques are quite new [Ogino et al., 2004]. While any of these sensors can theoretically be used alone, feedback sensors are generally the only ones so used.

2.1.1 Angular Sensors

Angular sensors fall into two main categories: velocity and tilt based sensors. The most common of these are tilt-based inclinometers and position feedback sensors, and accelerometers and velocity-based gyroscopes. Inclinometers measure the current incline; position sensors measure the immediate position; accelerometers measure the current acceleration; and gyroscopes measure velocity. They are most often used to

determine the orientation of a robot (and often, just its trunk).

Inclinometers generally measure one axis. They produce a tilt reading which can be used to determine at what angle a robot (or robotic component) is leaning.

Accelerometers can measure up to three axes at once. Their reading is the instantaneous acceleration: whether the sensor is speeding up or slowing down, and how quickly it is doing that. Accelerometers can be used to determine if a robot is remaining relatively motionless or not. The integrated readings can be used as a inclinometer, but with a loss of precision. More commonly, the acceleration with regard to gravity is used to provide the incline of the accelerometer.

Gyroscopes can measure up to three axis of rotation at once. They measure the instantaneous angular velocity of the sensor: the speed at which it is rotating. The gyroscope's readings can be integrated to determine the position drift error, but once again with a loss of precision.

2.1.2 Force Based Sensors

Force based sensors generally measure force, torque or pressure. They often are used to provide feedback on whether a joint has reached a desired position, or if the robot has come into contact with a surface (e.g., the floor).

Pressure sensors measure the pressure being exerted upon the sensor. They are generally mounted on the soles of a robot's feet and used to calculate the Center of Mass (COM) of the robot.

Force sensors measure the Newtonian force exerted upon the sensor. Similar to a pressure sensor, they are typically mounted on the robots foot to calculate the COM

or related data.

Torque sensors measure the torque, or rotational twist, present. They are generally used in joints or joint actuators to measure the torque present at the joint.

2.2 Physical Models

The basic physical models used to analyze bipedal robots were first given by Pratt [2000]. Pratt shows how a bipedal robot can be modeled as a simple inverted pendulum. He then derives more complicated models for bipedal robots, and gives a flywheel and an acrobat model that can approximate bipedal robots. I will describe these models before discussing the algorithms that use them.

The COM and Center of Pressure (COP) are the two most important concepts to begin with. The COM of the human body can be calculated as the point equidistant from the sum of the mass when weighted by distance, using the following equation:

$$\overrightarrow{P}_{COM} = \frac{\sum_i \overrightarrow{P}_{m_i} m_i}{\sum_i m_i} \tag{2.1}$$

The COP of the foot can be similarly calculated with:

$$\overrightarrow{P}_{COP} = \frac{\sum_i \overrightarrow{P}_{p_i} p_i}{\sum_i p_i} \tag{2.2}$$

Observing the COM (of the body) and COP (on the foot) when a human walks leads to the conclusion that they approximate a simple inverted pendulum. Therefore, insight into the movement of the pendulum will provide insight into walking movements.

As Pratt explains, an inverted pendulum's angular momentum is due to the torque produced by gravity. This can be expressed as:

$$H_0 = ml^2\dot{\Theta}_1 \tag{2.3}$$

By taking the derivative with regard to time, the equation gives the changes made by gravity to the angular momentum:

$$\dot{H}_0 = mgl\sin\Theta_1 \tag{2.4}$$

If we differentiate H_0 in Equation 2.3, we get

$$\dot{H}_0 = mgl^2\ddot{\Theta}_1 \tag{2.5}$$

Substituting this into Equation 2.4 gives

$$mgl^2\ddot{\Theta}_1 = mgl\sin\Theta_1 \tag{2.6}$$

which reduces to

$$\ddot{\Theta}_1 = \frac{g}{l}\sin\Theta_1 \tag{2.7}$$

an equation describing the pendulum's motion. Now, a biped's leg length may vary while in motion, so this model could be made more accurate by the addition of an actuator. As Pratt reasons, a linear actuator has the same dynamics as multiple joints, given all the joints are not straight. A virtual pendulum may be calculated with the point mass in the same location as in the multi-jointed case that will have the same dynamics as the multi-jointed pendulum.

The dynamics of the linearly actuated inverted pendulum now must account for the changes potentially produced by the linear actuator, as show in equations 2.8 and 2.9.

$$ml^2\ddot{\Theta}_1 = mgl\sin\Theta_1 - 2ml\dot{l}\dot{\Theta}_1 \tag{2.8}$$

$$m\ddot{l} = F - mg\cos\Theta_1 + ml\dot{\Theta}_1{}^2 \tag{2.9}$$

The equations describing the dynamics of a multi-joint system are as follows:

$$ml^2\ddot{\Theta}_1 = mgl\sin\Theta_1 - 2ml\dot{l}\dot{\Theta}_1 \tag{2.10}$$

$$m\ddot{l} = f(\tau_2, \tau_3, \Theta_1, \Theta_2, \Theta_3) - mg\cos\Theta_1 + ml\dot{\Theta}_1{}^2 \tag{2.11}$$

Equation 2.10 is identical to Equation 2.8, and the only difference between Equation 2.11 and Equation 2.9 is that F is replaced by $f(\tau_2, \tau_3, \Theta_1, \Theta_2, \Theta_3)$: that is, the force acting on the point mass is no longer direct, but must be calculated from the pivot to the point mass, taking into account the various joint torques.

Note that adding a foot to the pendulum does not affect the dynamics of the system. Instead, the pendulum no longer rotates about the actual COP, the ground contact point of the pendulum, but instead the COP of the foot, a virtual pivot point.

The main insight Pratt draws from these models is that like a pendulum, if the main mass is traveling away from the pivot point, it accelerates due to gravity. If it is traveling towards the pivot point, it will slow down. This is inherent in the dynamics

of the system, and so inherent in the dynamics of a walking robot. Any controller attempting to balance a robot must take this into consideration.

2.3 Algorithms

There are several common algorithms used to adjust the control of a robot's motions. The COM based algorithms keep the robot's center of mass within the supporting polygon of the robot's feet. The Zero Moment Point (ZMP) algorithm calculates the point in the horizontal plane at which all the moments are zero, and keeps this point within the supporting polygon. A PID (Proportional Integral Derivative) controller is a basic control strategy used to adjust the error of a feedback output to a desired reading, calculating corrections based on a proportion of the error, and the integral and derivative components of the error. The threshold based T control algorithm was developed in previous research by McGrath et al. [2004b]. Other more advanced control strategies have been used as well, with learning based algorithms becoming of greater interest.

Algorithms also control for static or dynamic balancing. Static refers to keeping the COM within the support polygon of the feet. Static balancing, therefore, is balancing in order for the robot to remain statically stable. If the robot is statically balancing, it can be stopped at any point and it will be stable. A dynamically balanced robot may not be able to remain stable if abruptly stopped at any point. Dynamic balancing keeps the ZMP of the robot within the support polygon of the feet. Thus, the moment of the robot is controllable and stable, with the current dynamics of the situation.

The two main methods of control are model-based algorithms that involve balancing to calculate or modify gait creation (be it on- or off-line), and reflexive algorithms that are based on small, tightly-coupled control loops between sensors and actuators. A main difference here is that while the model-based algorithms can be fine-tuned with more complicated techniques (i.e., reinforcement learning, genetic algorithms), they often apply to a specific robot, and are more computationally intensive to create, with potential need for pre-processing. Reflexive algorithms use less complicated methods, and are simple, general functions that make immediate corrections based on sensor input. As simpler methods, they are a better choice for a minimalist controller, and ideally transfer more effectively than do the highly-tuned algorithms. Reflexive methods also more directly ape the human cerebellum, with balancing occurring in the lower levels of the brain, as instinctive behaviours, not highly complicated ones using the entire brain.

Exploration into new sensors and technologies tends to begin with simpler, classic control methods and later expand into more complicated control algorithms if needed. While the most common algorithms for balancing are ZMP based, PID and PID-like controllers have been successfully used for robots less dependent on force based sensors. Numerous robots have been built and used to explore force based sensors and ZMP-based algorithms; less have looked at purely angular sensors or PID-type control mechanisms. The University of Manitoba's robots are the only ones to use only a single angular sensor for balancing input. While algorithms exist that can balance for any robot, they tend to be very slow. The simpler, or more robot specific algorithms, tend to work on a quicker (realtime) basis.

2.3.1 COM

COM calculations are a very basic algorithm for controlling the robot [Pratt, 2000]. The robot's COM is calculated, and kept over one or both of the robot's feet. These calculations are useful for static positions, but less so for dynamic ones, as a robot may have the COM over a foot and still be unstable, due to the moments affecting it. This is easiest noticed by creating a walk composed of statically stable positions, that falls when the robot moves through the positions at a constant speed.

2.3.2 ZMP

ZMP algorithms concentrate on determining the ZMP and controlling it [Kim and Oh, 2004]. The ZMP is the point at which the sum of all moments acting on the robot are zero. This point is viewed horizontally, so that it can be compared to the support polygon of the robot. The support polygon (see Figure 2.1) is the smallest possible polygon created by the robot's feet and the area between. If an elastic band were placed around both of the robot's feet, the area inside the elastic would be the support polygon of the robot. By keeping the ZMP within the support polygon, the dynamic moment of the robot is also accounted for, not just the static calculations. This currently is the basis of almost all balancing algorithms in the field.

ZMP depends on knowing many (if not all) of the exact parameters of your robot. The length and weight of all joints and links must be known, as well as the dynamic moment affecting the robot. These are calculated using force-based sensors, like pressure sensors mounted in the foot. While the COM has been estimated using accelerometers and gyroscopes, it is uncertain if they provide enough information for

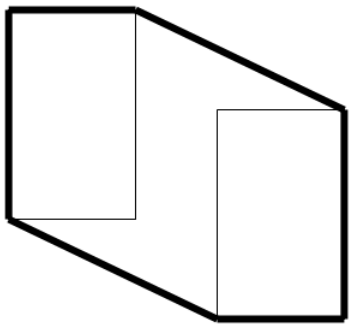


Figure 2.1: Support polygon created by the robot's two feet.

the ZMP to be accurately estimated.

Kagami et al. [2000]'s AutoBalancer is a ZMP-based system, implemented in such a way as to allow it to be easily transferred to other robots. A function, unique to each robot, is used to calculate the Center Of Gravity (COG) from the joint angles of the robot. In concert with the ZMP, changes in the COG are used to balance the robot's desired position. Thus, a robot may use this algorithm to balance and have the code transfer from robot to robot by changing the COG-calculation function.

2.3.3 PID

A PID controller aims to correct the sensory feedback it is given to a designated setpoint. This correction is implemented as a proportion of the error, and fine tunes that correction with further minor corrections from the integral and derivative. The proportional term corrects for the immediate error. The derivative term is used to correct for predicted future error. The integral term is used to correct for accumulated small errors over time. This creates a general controller used in many applications

from speed control and power steering to electric power generation plants [de Vegte, 1994].

PID controllers have a gain, consisting of a numerator and denominator, that is separately tuned for each component. P is usually tuned first, as it does the majority of the control work, with D tuned next, and then I tuned last (if necessary). The timing of each component (or just of the correction as a whole) may also be tuned. That is, a controller may not be allowed to make a correction within a given time step of its last correction made.

The physical models laid out in Section 2.2 demonstrate the strong effects of gravity on a biped robot. This is why a PID controller is not inherently the best choice for a humanoid robot in balancing. While a standard control technique, it does no compensate for the changing effects of gravity on a system. Instead, it assumes all error to be equal regardless of the robot's position and/or movement relative to the pivot. For some joints, this works well enough, but most joints are located in such a way that the effects of gravity will work either for or against the movement of the joint. This means that while the PID controller will register and correct for the same amount of error regardless of the placement of the joint, the effects of gravity added to the corrections made by the controller may unbalance the robot instead of balancing it.

2.3.4 Threshold (T) Control

The threshold based T control developed in McGrath et al. [2004b] is essentially a simplified variant of a PID controller. This controller is very similar to an on-off

deadband controller. Deadband refers to a band surrounding the desired setpoint (or thresholds above and below) that restrict when a controller may begin to act. Not until the current reading breaks a threshold (leaves the deadband) by a given amount may it correct, and then only by a very small, designated amount (on-off).

Unlike a PID algorithm, there is relatively little to tune in the T control mechanism. Corrections are very small (1 servo setting), and the only tuning is creating the thresholds, fixing the bounds around the thresholds, and determining the time step separating the corrections. The second bound around the threshold is a constant amount for all gaits, and is useful to allow the thresholds to be drawn as tightly and accurately around the sensor readings as possible. This boundary can also be used to relax constraints when desired, something not possible in a PID controller.

2.3.5 Advanced

Zhou and Meng [2003] have looked at using reinforcement learning to dynamically control balancing in biped robots, but this is one of the few attempts to do so. While reinforcement learning is generally too slow and risky (the robot may fall) for most humanoids to learn to walk with, Zhou used reinforcement learning to improve the robot's balance from randomly created parameters, based on feedback from pressure sensors on the robot's feet. This technique was then used to improve pre-existing walk gaits.

Fuzzy logic [Zhou and Meng, 2003; Zhou, 2000] has also been used, but more commonly in simulation due to the near certainty of the robot falling while learning. Imitation learning has also been touched on, but only implemented in one

plane. [Nakanishi et al., 2004]

2.4 Approaches to Balancing

Now that the basic sensors and algorithms have been discussed, I will review some of the robots and approaches to balancing being used upon them. I previously separated robotic balancing into reflex and model-based approaches. Reflex balancing seeks to alter the robot's movement online, in real-time, while model-based balancing refines the gait itself either on the robot or by pre-processing. Beijing University's Huang et al. [2004] approach was reflex-based, whereas Zhou and Meng [2003] uses a model-based approach. The model-based approach is a method of improving a given gait offline, but may not be applicable to general gaits, and will require more time and processing power to implement in real-time. Reflexes are more transferrable across robots, as they are simple real-time additions to a gait.

An improved algorithm is not always the first or only step in balancing; Waseda University's Yamaguchi [Yamaguchi et al., 1994] also developed new feet to improve the balancing on one of their earlier robots. The original robot, WL-12RV, used the ZMP to balance, but found that physical modifications were helpful in improving the balancing. New feet were created that reduced the difficulty in balancing by aping human ones in their ability to absorb and reduce the shock of landing. They also providing extra information for the gait generator on the landing itself. Linear potentiometers in the feet allowed the relative position of the top of the foot from the bottom of the foot (on the ground) to be calculated. This information was used to maintain the ZMP of the robot within desired bounds. The balancing was applied

to the pre-existing walk, for one step, and then the robot would return to the preset walk gait. Yamaguchi et al found that the robot (1.8m in height) was able to cope with a slight step (1.2 cm, or $1/150^{th}$ (0.6%) of the robot's height).

Waseda University's WL-12RIII [Lim and Takanishi, 2005] extended a ZMP controller that used a virtual plane, one created by the uneven terrain the robot walked on and the robot's feet. The ZMP was then manipulated to stay within this plane by adjusting the angle of the torso. Feedback for the controller was provided by the rotational angle, angular velocity and torque being measured in each of the robot's 8 DOF, using a total of 4 sensors per actuator. In addition, two switches in each foot were used to detect contact at the heel and the toe, enabling the robot to determine the position of the feet, and thus construct a virtual plane either between the feet, or under the hind foot. The algorithm iteratively determined a torso angle for the robot until the ZMP was within desired bounds, and then implemented the motion. This allowed Lim and Takanishi [2005] to walk over a slight ramp (10%), as well as a staircase 10 cm high. However, as with many of the model-based control strategies, this one made some assumptions that would not hold up in the real world, such as the feet never sliding, and contact between the foot and the ground being unaffected by force or moment, but remaining steady.

Zhou and Meng [2003]'s balancing is based on integrating numerical data and linguistic rules. The approach allows Zhou to combine intuitive human knowledge, biomechanical studies, and data from previous experiments on the bipeds, thus integrating linguistic, numerical and linguistic, and strictly numerical data in this case. A basic walking gait is created by minimizing the ZMP's deviation from a desired

setpoint, with a given set of gait parameters (i.e., step interval, step height). Balancing is performed by moving the trunk to offset any errors remaining in the ZMP. The gait created was then further improved by using a reinforcement learning algorithm to smooth the walking gait. The reinforcement learner was a modified GARIC model that used fuzzy feedback to evaluate the gait, and was adapted for use on robotic gaits. The fuzzy feedback improved the learning process on the gait, and provided a gait with much less deviation from the desired ZMP. Initially, the gait was developed and improved in simulation, but was later transferred to the Robo-Erectus robot for use in competition at Robocup.

Honda's high tech humanoid robot [Hirai et al., 1998] uses a 6-axis force sensor to determine the robot's balance, as well as several inclinometers to measure the stance and lean. The balance is corrected by a combination of factors that adjust the robot's stance and stride. The robot's trunk may be adjusted, the stride lengthened, the foot tipped forwards or backwards, or a combination of these. The stride lengthening corrections are integrated into the pattern generated by the CPG, while the other corrections are implemented in addition to the gait pattern. Their balancing focus appears to be on smooth gait generation, though unfortunately, not much is stated in their papers.

Kagami et al. [2000] implemented their AutoBalancer approach on the 16 DOF, 32 cm high remote-brained 'Akira', and the 30 DOF, 1.3m 'H5'. Akira has 4 force sensors in each sole to measure ZMP; H5 has 12 in each. As described in Subsection 2.3.2, AutoBalancer uses a matrix of information about the specifics of the robot to calculate the ZMP of the robot, and other data used to balance the robot. This

method worked well enough that it could be transferred from the 32 cm Akira to the 1.3m H5 and still work. However, the complexity of the balancing algorithm caused delays in processing the motions on the test robots (one of which had a PIII-333 MHz controller), placing the processing power required far beyond that available on the University of Manitoba's Autonomous Agents Lab's robots, or most onboard controllers on small sized robots.

Kuffner et al. [2002] expanded on AutoBalancer by strategically planning motions beforehand. Their earlier work [Kuffner et al., 2001] consisted of planning dynamically stable paths of motion, around obstacles or to accomplish tasks such as sitting down. This was then expanded to include dynamic balancing constraints by using the AutoBalancer program above to balance planned motions. This allowed them to program their robot to crouch and retrieve an object, as well as balance on one leg while placing the other leg above an obstacle. They still were unable to accomplish this in realtime.

Another line of approach is to use a standard control algorithm, such as PID, to control the robot, based on sensory information or ZMP. These approaches focus on using smaller, simple control mechanisms to balance with. Due to the simplicity of the controller, multiple controllers may be used simultaneously to control the balancing. As before, these approaches can be either reflex or model-based as well.

Beijing University's Huang et al. [2004] used a simpler testbed to investigate the influence of their balancing algorithm, confining their work to the sagittal plane only. Similar to the work done on Waseda's WL-12RV [Yamaguchi et al., 1994], they looked at adjusting the previously calculated walks with simpler corrective functions. They

investigated using sensory reflexes on BHR-01, incorporating a ZMP reflex, a landing phase reflex, and a posture reflex into the dynamic walking pattern. The ZMP reflex was measured by a foot force sensor, and the ankles compensated to keep the ZMP in the desired area. The landing phase reflex was used to determine if the foot landed sooner or later than expected, and raised or lowered the foot to compensate. The posture reflex measured the acceleration and angular rate, correcting these values by adjusting the hips, and therefore the posture of the trunk. These reflexes were triggered by sensory information, and when active, would compensate for any imbalances in the walk. The corrections were use to adjust the pre-calculated walk pattern, which had been been calculated offline. This allowed the balancing to occur in realtime. These reflexes, added to the walking pattern, proved effective in walking over uneven terrain.

GuRoo Kee et al. [2003] from the University of Queensland is another robot to explore using PID control on its sensors to balance Low [2003]. GuRoo is outfitted with 3 Inertial Measurement Units (IMUs) for sensors, and a gyroscope. The IMUs were located in the robot's head to most accurately imitate the human sensory systems, and simply measured the angle due to the noise in the sensors. A ZMP based PID controller for this data was developed in simulation, and then transferred to the robot. Balancing disturbances were divided into three categories based on the angle of the robot after the disturbance, and the controller was able to deal with small and medium disturbances (less than 10°). This yielded improvements in the robot's crouching (it no longer required a human assistant), but not to the walk. The speed of the walk, and sway resulting from quick movements were different than in the sim-

ulator, and the balancing did not transfer to the real walk with any improvements. They attempted to compensate for this by increasing the response time, but did not find it helped. Balancing was limited to compensating with only the ankles and torsos. This was due, at least in part, to the model used for simulation which modeled the legs as a single unactuated link.

Kim and Oh [2004] improve upon the standard PD controller in their Force/Torque sensor by adding a damping controller, compensating for the inherent oscillation in biped constructions, most especially for that created at the ankle joints. This improves on previous ZMP-based controllers that assume any unwanted deviations in the ZMP are based on poor control algorithms, not physically caused by the robot's construction. The PD controller they implemented is based on the ZMP and allowed dynamic walking to be implemented on their robot. The controller was derived algorithmically, based on the single mass inverted pendulum model. It was then simulated, and finally implemented on a real robot, at the beginning of a stride. Like Honda [Hirai et al., 1998], they used a stride lengthening mechanism at the end of their strides, as well as a separate controller to regulate the orientation of the landing foot.

Pickel [2003] used inclinometers to help balance his minimal robot Rock Steady. The robot has only one actuator per leg, and uses a passive ankle to allow this. A moving mass was attached to the back of the robot, and actuated in the coronal plane to allow the robot to manipulate its COM. As the COM is intensive to calculate online, the actual COM was precalculated offline, and those calculations used to approximate the actual COM online. Two inclinometers are used as sensors, as well as encoders on

the joint motors to calibrate them at the beginning of use. Balancing was explicitly added to this controller using logical rules based on human intuition. Depending on the angle registered on the inclinometer, the mass was moved to compensate for the angle. Pickel only managed to implement static balancing on the robot, and suggested that dynamic balancing could be implemented with the aid of an accelerometer in future.

Kulvanit et al. [2006]'s Team KMUTT has looked using velocity based control to dynamically balance their robot. This balancing mechanism is part of a specific walk, chosen by the robot if its sensors indicate conditions are appropriate, not a standard part of the robot's behaviour. The robot has two walks: a slower static walk that uses the force sensors on the robot for information, and a faster dynamic walk that uses the accelerometers and gyroscopes on the robot to balance. Both walks use PD controllers for balancing. In the static walk, the PD controller manipulates the height at the robot's hip based on the force sensors in the foot. The dynamic walk controls the velocity at the hip with its PD controller. Team KMUTT competed at RoboCup 2006 using this code.

The University of Manitoba's Tao-Pie-Pie is the only robot to use only gyroscope readings for correcting balance [Baltes et al., 2003; McGrath et al., 2004b]. The readings are processed and run through a T controller, compensating for perturbations in the gait. The T controller simply applies a minimal correction when sensor readings break a predefined boundary. Balance is explicitly added on as corrections made to the pre-calculated walk gait. These corrections were used in competition during FIRA '03 to compensate for the poor surface of the competition flooring, and were found

to be better than the previous gait. Further, the T method is extremely simple to implement and tune, though it is dependent on being accurately tuned to the walk it is used on.

Kanehiro Kanehiro et al. [2003] uses balancing for a novel action: lying down and standing up again. HRP-2P uses mostly static motions to transition between lying on the floor and standing, but requires some balancing moving between squatting and kneeling. Kanehiro et al. use feedback control to move the robot, and further estimate the orientation of the robot's torso with gyroscopes and accelerometers. This data is used to find the position of the COM with relation to the support polygon.

2.5 Summary

This chapter has provided an overview of approaches taken to balancing humanoid robots. While interesting work has been done in simulation, I confined myself to physical robots, and the work implemented on them due to the difficulty of transferring simulated research onto a physical robot. Physical robots used for balancing research generally have force based sensors and/or angular sensors, but I look at a robot with only an angular sensor — an accelerometer. The control methods for these robots can be divided up into simpler, more general controllers like the PID or T often used in creating balancing reflexes, and more complicated ZMP-based controllers such as AutoBalancer which affect higher levels of the gait creation. Many robots are using ZMP-based controllers with multiple sensors, but very few look at a single sensor, or using a simple balancing reflex. None of the robots use purely an accelerometer for balancing, but instead use the accelerometer to supplement data obtained from other

sensors.

The next chapter explains the robot built for this research, the design decisions made to create it, and the basic gait code running on it.

Chapter 3

Building Lillian

This research was carried out on robots from the University of Manitoba's Autonomous Agents Lab. For my thesis, a specialized robot was required, as none of the other robots had the sensors required already equipped, or the specific control needed (in particular, the low level servo control necessary). A new robot was designed, based on previous robots created in the lab, with the sensor needed and a processor capable of interfacing with that sensor. This robot, Lillian, has 8 DOF, an accelerometer, an Eyebot controller board, and originally carried a camera with a pan/tilt assembly for vision. The main design principles in building the robot were simplicity and cheapness. The robot was tested to ensure the sensors worked, and gaits were designed for it. These gaits were created based on an initial position, with position changes made from fixed adjustments, interpolated over specified time periods. This allowed gaits to be easily adjusted for new robots, modified hardware, or corrections based on sensor input.

3.1 Previous Robots

The Autonomous Agents Lab at the University of Manitoba has a history of building simple, low cost humanoid robots [Baltes and McGrath, 2004; Baltes and Anderson, 2006], Figure 3.1(b), Figure 3.1(c)) and modifying existing budget platforms to create humanoid robots [Baltes and Anderson, 2006]. Minimalism in a robot allows researchers to learn the value of each and every part added to a robot as it is added. In contrast, many of the first humanoids were built with more servos, joints and sensors than their handlers could use (especially at first), resulting in parts of the robot being ignored. Not only does a minimal robot avoid this excessive complexity, but it also keeps costs down (even without the added focus on low cost) by simply adding what can be used immediately. In a competitive world filled with robot costing thousands, tens of thousands, hundreds of thousands of dollars and more, focusing on low costs and minimalism has inspired even amateur hobbyists to join the search for robotic knowledge [Mueller, 2006]. Building a cheap robot has further benefits to the quality of algorithms produced by the researcher. Using less expensive parts generally reduces the overall precision of the robot, leading to more robust, general algorithms that are better equipped to deal with error and imprecision. While easier to work with, an expensive robot will allow the researcher to create less robust algorithms, as the probability of error is much lower. Thus, a low cost minimal humanoid robot was used for this research.

While humanoid robots already existed in the lab, none were suitable for this research, and so a new robot was built. This robot, Lillian, was heavily dependent on previous generations of robots created in our lab. She is the fourth generation robot

in the series of robots beginning with Tao-Pie-Pie, and continuing through Hiro and Lillith.

3.1.1 Tao-Pie-Pie

Tao-Pie-Pie [Baltes and McGrath, 2004] (see Figure 3.1(a)) is a minimal 8 DOF humanoid with 6 DOF in his legs, a dual-axis gyroscope and a camera with 2 DOF (pan/tilt). The gyroscopic feedback was integrated into his control system to make it a closed-loop control system as is further described in Chapter 5 [Baltes et al., 2003; McGrath et al., 2004b]. This integration was used to perform some initial balancing tests while on a tilting platform, as well as during walking gaits. Tao's processor was an Eyebot controller capable of controlling 12 servos, a camera, and had a serial port interface. The 32 bit controller ran at 25 MHz, with 1 MB of RAM, and 512 KB of ROM. Tao was one of the few robots in the world to be able to walk with only 6 DOF, and won a Technical Merit Award for Autonomous Operation at his first competition [FIRA, 2002]. He competed at numerous competitions thereafter: RoboCup 2002, 2003 and 2004, and Fira 2002 and 2003. I worked on Tao-Pie-Pie from the summer of 2003 to the fall of 2004, after which he was retired with honours into the Heinz Nixdorf Museum in Germany.

3.1.2 Hiro

Hiro[1] [Baltes and Anderson, 2006] (see Figure 3.1(b)) was a 27 DOF robot constructed as a successor to Tao-Pie-Pie, and the first robot built in the lab to have a

[1]named after Neal Davidson's Hiro Protagonist [Stephenson, 1992]

fully functional upper torso as well as lower legs. Hiro had 12 DOF in his legs (3 in each hip, 1 at the knee, and 2 in each ankle), 3 DOF in his torso, 10 DOF in his arms (3 at the shoulder, 1 at the elbow, and a gripper or hand), and 2 DOF in his neck for a pan/tilt camera assembly. Hiro also was constructed with an Eyebot controller, and a MUX board to control additional servos beyond the 12 allowed by the eyebot controller. Unfortunately, the metal[2] used to construct Hiro was too flexible for the weight upon it, leading to excessive bending, and the servos were unable to move the robot far enough to develop a walking gait. Hiro was therefore passed over as a research platform, and at this time, has been stripped of almost all his servos. The design used to create Hiro, in particular his legs, was reused (as were some of the pieces) to created Lillith, his successor.

3.1.3 Lillith

Lillith (see Figure 3.1(c)) was built on simpler lines than Hiro, while remaining more complex than Tao-Pie-Pie. Lillith's leg assembly had 8 DOF, 2 more than Tao-Pie-Pie, and 4 less than Hiro. This was created by removing 2 DOF from each hip on Hiro, leaving 1 DOF at the hip, 1 DOF at the knee and 2 DOF (sagittal and coronal) at the ankle for each leg. Lillith's torso assembly harkened back to Tao-Pie-Pie's minimal torso, with no movement at all. Lillith was constructed as a prototype in the lab, with readily available materials. She was able to walk quite easily, and several walking gaits were developed on her.

Lillith's 8 DOF are each powered by a servo motor. The motors are of varying

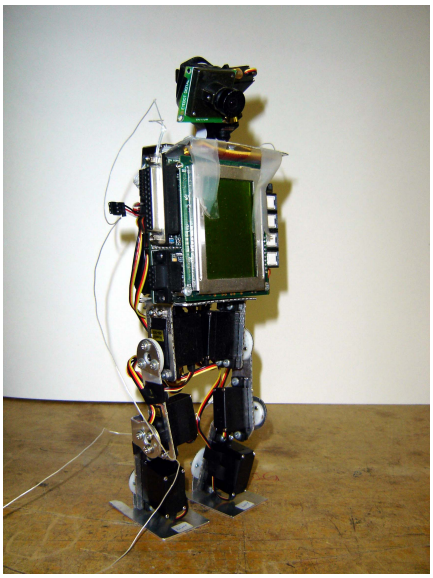
[2]Scrap metal from the Physics Department

sizes; the hips and knees are equipped with a $CS-35$ servo, and the ankles are equipped with more powerful $CS-59$ servos. The ankles were given more powerful servos as they are prone to greater amounts of torque, being farthest from the main weight (the Eyebot controller). Each servo motor has 256 set points [0..255], and a range of approximately 135°. The servos have no feedback.

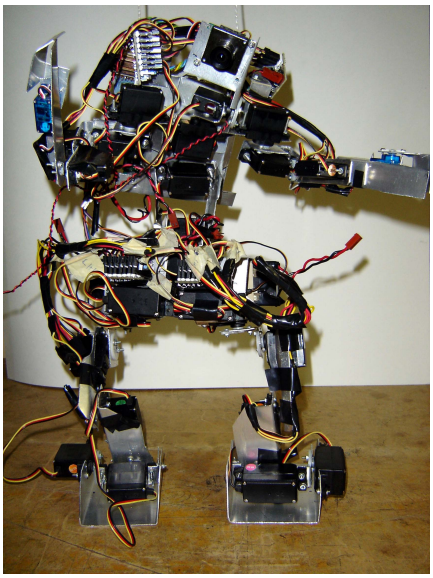
Lillith was the first humanoid in the Autonomous Agent lab to use a KAIST controller instead of an Eyebot. This led to some interesting control mechanisms: while the KAIST had a number of add-on sensors, it was only able to control 4 servo motors. The same MUX board designed for Hiro was used to compensate for this difficulty. The KAIST controller was also reprogrammed to allow for multi-threading capabilities, which can be very useful from a robotic point of view. However, access to the controller was only through a specialized cable that was significantly slower in transmitting than the Eyebot's serial port. Also, when an accelerometer was added to Lillian, it was necessary to convert the output of the accelerometer to analog from digital by a circuit, and then back to digital from analog in the controller itself to avoid overwhelming the controller's circuits with constant readings. Finally, there was no way of integrating a camera onto the KAIST kit at the time (or foreseen in the near future), an essential element of a robot if it was to compete.

3.2 Design Schematics

Lillian (see Figure 3.1(d)) was constructed along the same lines as Lillith: 8 DOF, with two more possible for a pan/tilt camera assembly. The 8 DOF consist of 1 DOF at the hip (XH), 1 DOF at the knee (XKS) and 2 DOF (sagittal and coronal,

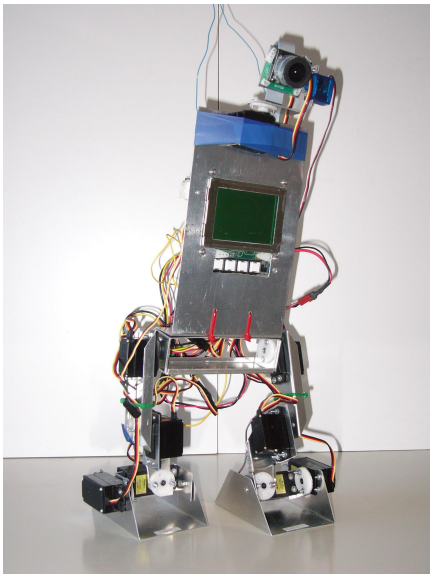


(a) Tao-Pie-Pie



(b) Hiro

(c) Lillith



(d) Lillian

Figure 3.1: Tao-Pie-Pie (a), the first minimalist humanoid used to explore the limits of balancing with gyroscopes. Hiro (b), the second generation 27 DOF robot built in the Autonomous Agents Lab. Lillith (c), the third generation and first humanoid with a KAIST controller. Lillian (d), the fourth generation and thesis platform.

or XAS and XAC) at the ankle for each leg. The metal pieces for her construction were machined in the Engineering Faculty's machine shop with the aid of Dominic Leung. This allowed for greater precision in the cutting and bending of the pieces of any lab robot since Tao-Pie-Pie. The metal used was initially 1/32" aluminum. Both shins had to be re-cut in 1/16" aluminum with extra bends for bracing, as too much compression occurred here in the prototypes. A hip brace was also necessary, constructed of 1/16" aluminum, to reduce bending in the hips. These modifications were not necessary for the robot to walk, but for Lillian's feet to rise up off the ground enough that she was able to clear uneven terrain, rather than constantly shuffling her feet along the ground and constantly stubbing her toes.

Lillian was created using an Eyebot controller once more, for ease (and speed) of data transfer to and from the robot, ease and simplicity of accelerometer installation, and potential to add a camera in future. A further benefit of the Eyebot was the library of software created for Tao-Pie-Pie that could, with little modification, be used on Lillian.

Power is provided by an onboard 7.4V lithium ion rechargeable battery (formerly a camera battery). This powers both the controller and the servos, and the lithium ion allows for multiple rechargings without wearing out the battery, or unduly creating a 'memory' on the battery. The battery is mounted just above the hips, and is securely fastened to the robot.

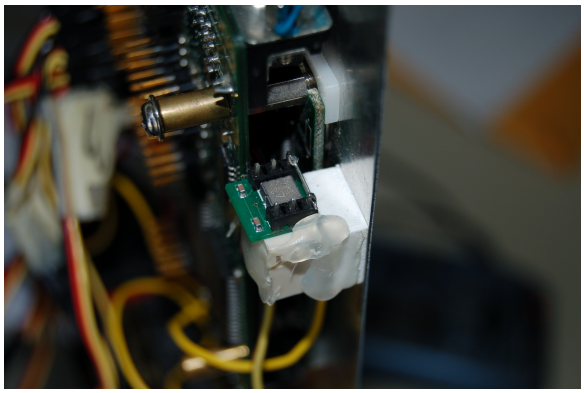


Figure 3.2: Memsic 2125 Accelerometer, as mounted on Lillian. The extra wire connects the two grounds, removing the need for an extra cable. The accelerometer can be easily removed from the mounting, if necessary, and then replaced.

3.3 Sensor Placement

Lillian was fitted with a single dual-axis accelerometer. This accelerometer is a Memsic 2125 Accelerometer Inc, measuring approximately 60° in each axis. This was deemed sufficient, as a tilt of over 30° is almost guaranteed to result in a fall. The accelerometer requires power and ground, and supplies two outputs (X and Y readings). The accelerometer was mounted high on the robot's torso, causing the accelerometer to be placed in the location with the earliest and greatest amount of movement. This positioning makes the accelerometer as accurate and indicative of early warning as possible. The accelerometer was placed in a platform securely mounted to the robot (screwed in as well as hot glued), such that the sensor is still removable if required (see Figure 3.2). Further, the two grounds on the accelerometer were soldered together to reduce the numbers of wires needed.

Lillian was also outfitted with an Eyebot camera, capable of capturing 4 frames per second. A pan and tilt camera mount was created, with the help of Dominic

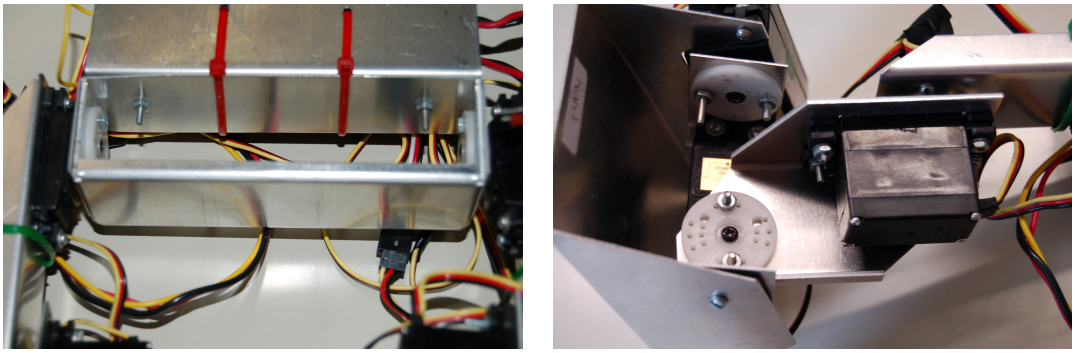
Leung, for a greater range of vision. The camera and mount were placed at the top of the robot's torso for two main reasons. First, it added greater verisimilitude to humanness of the robot. Second, it added extra sway to heighten the instability of the robot, and thus add extra difficulty to the balancing. This instability proved to be too much for the algorithms to overcome, so the camera was removed for the present research.

3.4 Modifications

As with every robot, and especially those developed on a prototypical basis, actual use of Lillian revealed that some modifications had to be made to the initial design of the robot. Compression and sway proved to be extremely difficult obstacles, and necessitated a hip brace, the replacement of the shins with thicker metal, and the removal of the camera.

Testing determined that Lillian was unable to lift one leg off the ground more than a millimeter. This was far less than what would be needed to be able to traverse uneven terrain. Close inspection revealed that the hip plate was bending and absorbing almost all of the height generated when lifting the opposing leg off of the ground. A new hip brace was created, with the aid of Shawn Schaerer, to reduce the compression in the hips and allow more lift. (See Figure 3.3(a)) This brace went under the hips, and was constructed of 1/16" aluminum, double the thickness of the previously constructed pieces. Once in place, it removed much of the compression problem.

However, Lillian still was not able to raise her leg as high as she should have been.



(a) Hip Brace (b) Modified Shin

Figure 3.3: New hip brace (a) constructed to minimize bending in Lillian's hip assembly, and modified shins (b) constructed to reduce compression and bending. The shins are double the thickness of the other metal used in Lillian's construction.

When lifted up to determine the position of the servos without gravity, the difference in the feet was significantly more than when placed back on a level surface. This time, inspection revealed significant compression in the shins. Two bends were added to each shin to reduce compression, and new ones were created out of 1/16" aluminum again. (See Figure 3.3(b)) The new shins allowed the robot to maintain a foot lifted approximately 1/4" in the air.

The final modification was due to the extra sway added by the camera. The total weight of the camera and pan/tilt assembly added so much sway to Lillian's movements that it was extremely difficult to compensate for them . Minute corrections were needed very often when the robot was walking correctly, but the slightest overcompensation (or neglect of correction) could cause the robot to move the upper body to a position where the servos could not move due to the high torque. Once the robot moved into a slightly better position, the servo motors would overcorrect to the given position and cause sway. Thus, the camera was removed to make the

balancing problem slightly simpler. Future work will hopefully allow the camera to be added back on.

3.5 Summary

This chapter described the steps taken to create a new humanoid robot, capable of walking, with accelerometers to implement this research on. This robot, Lillian, was the fourth generation robot built in the lab, and depending heavily on the previous robots — Tao-Pie-Pie, Hiro, and Lillith — for design plans and decisions. Sensors were placed on the robot to make them as sensitive to change as possible, and to ensure the robot had instabilities to contend with in its initial testing, though not too many to overcome. Walking gaits were developed for Lillian, extending methods developed for Tao-Pie-Pie. Use of the robot revealed instabilities in the design that required replacement shins and a hip brace to overcome. Excessive sway necessitated the removal of the camera assembly. The next chapter addressed the means of controlling Lillian in software.

Chapter 4

Control Implementation

Once a robot has been built, the work has only begun. This chapter deals with the implementation of basic control routines and data processing on Lillian. The actuator control is implemented using a state machine that applies changes to the robot's position. These positions are made up of sets of adjustments made to the robot's current position, linearly interpolated to move over a specified amount of time. Sensors are read and integrated into this process by being regularly polled, and producing adjustments for the robot's current position.

4.1 Actuator Control

The control functions for the robot are written at multiple levels, as shown in Figure 4.1.

At the lowest level, servos are simply moved to a given position, with no consideration to current position or timing.

```
turnTo(servo, setting);
```

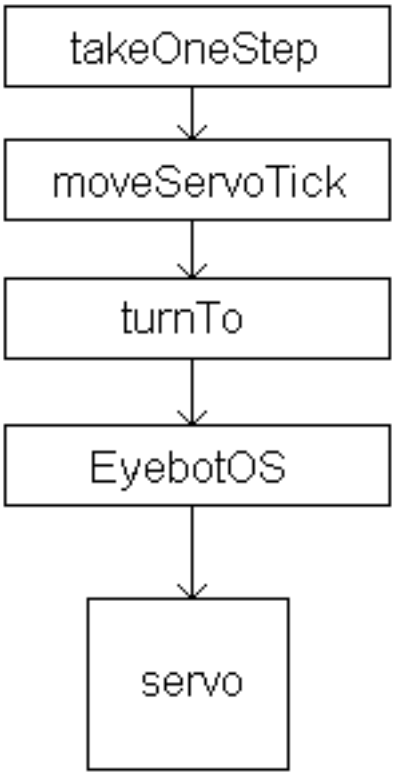


Figure 4.1: Hierarchy of servo control.

One level up, a servo starts recording information concerning the length of time specified to move from its current position to the desired one. Each servo has a current position (setPoint), a target position (targetPoint), an estimated completion time (arrivalTime), a remaining duration (deltaX), and a remaining movement (deltaY). Using these, a servo's position can easily be updated by linear interpolation:

```
remaining = setPoint - targetPoint;
adjustment = targetPoint - (deltaY * remaining ) / deltaX;
```

Servo positions are checked more or less frequently, depending on the load on the processor; generally, servos are checked for movement every operating system tick (about 1/100th of a second). Other sensors (eg, accelerometers) are generally checked every time the servos are, enabling their readings to be quickly applied to the servos.

Adjustments can be made to the phase's posture by updating the target position and duration of the movement (in ms), and then updating the rest of the values.

```
adjustTarget( struct ServoInfo * servo,
        int adjustment, int duration ) {
.
.
.
  if ( duration > 0 ) {
    servo->targetPoint = servo->targetPoint + adjustment;
    servo->arrivalTime = OSGetCount() + duration/10;
    servo->deltaY = servo->targetPoint - servo->setPoint;
    servo->deltaX = duration/10;
  }
.
.
.
}
```

4.2 Gait Development

Gaits are developed based on techniques developed for Tao-Pie-Pie [McGrath et al., 2004b; Baltes and Lam, 2004b]. Walking is divided into either 6 phases (left leg stand, right leg swing, stand, right leg stand, left leg swing, stand) or 4 phases (left leg stand, right leg swing, right leg stand, left leg swing). Each phase moves from one statically stable position to another statically stable position. Transition positions may not be statically stable, but are generally dynamically stable (thus avoiding falls).

At least one state machine is used to move between positions in a gait. Multiple state machines may be used in a single gait for flexibility in the gait. For instance, in the longer walk gait, a state machine is used to move the robot from the initial stance into the position for beginning the long walk. A second state machine controls the desired number of steps to take. Finally, a third state machine moves the robot from the walking stance back into the initial stance. This use of state machines, and multiple state machines, allows gaits to have different numbers of phases from each other, and to repeat parts of a walk, or linger over specific positions.

Gaits are based on an initial position – usually a simple stand. Each phase is reached by a set of adjustments (one for each servo) that are applied to the position the robot is currently holding. This adjustment method allows for gaits to be easily corrected (i.e., because of hardware adjustments) or transferred between robots. It also minimizes the danger of failing servos being compensated for in adjustments to the gait, as the initial position can simply be modified without changing the adjustments that create the walk. Care must be taken to ensure the total adjustments sum

to 0, as otherwise the robot will slowly drift off the desired walk pattern.

For this research, two separate walking gaits were developed. The first simply repeats one step over and over again, using the 6 phase ideology. The second takes two adjustment phases to move from the initial stand into one of the 4 phases, and then cycles through the 4 phases for a specified number of steps. The walk finishes with two more adjustment phases to return the initial stand.

4.3 Accelerometer Overview

Lillian is outfitted with a dual axis accelerometer. The sensor driver was written in assembly language, and reads the data in from two separate Pulse Width Modulation (PWM, or servo) ports on the Eyebot controller. The controller automatically processes the PWM data, and converts it into a digital, number, format.

There are three basic levels to using feedback from a sensor, as displayed in Figure 4.2. First, the low level processing of data from the sensor. Second, the mid level application of the processed data to create corrections for the gait or stance. Finally, the high level application of the corrections to the gait or stance which is relatively trivial.

The low level processing consists of calibration of the sensors to a zero point. Then the data is filtered and used in a running average. The zero point is determined by taking a number of readings from the sensor while it is still, and averaging them to find the mean value. The running average is continuously updated every accelerometer reading. Readings are currently filtered to only read once every system tick.

The mid level corrections are created by either a threshold control algorithm, a

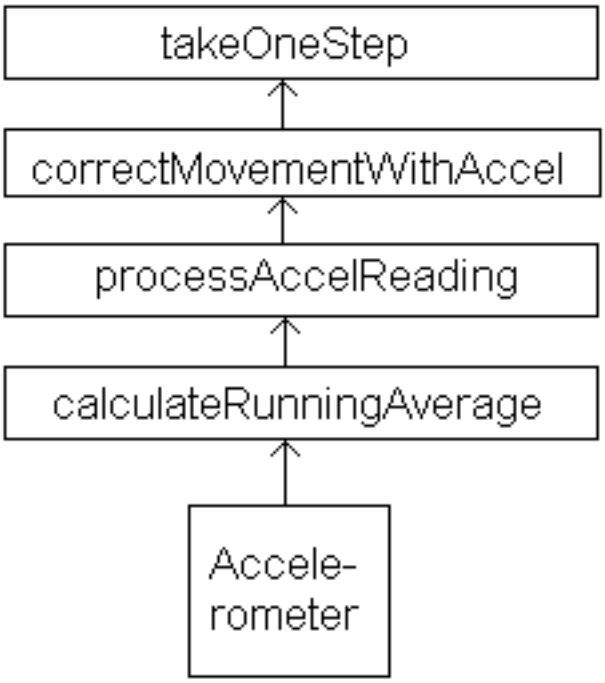


Figure 4.2: Hierarchy of accelerometer data.

PID control algorithm, or the hybrid algorithm. The threshold algorithm is very similar to the threshold algorithm developed for Tao-Pie-Pie's gyroscopes, mildly penalizing the robot every time the sensor readings go out of specified bounds. The PID control algorithm is a standard control strategy that seeks to adjust the robot to maintain preset ideal sensor readings. The hybrid algorithm combines elements of both of these, mildly correcting in general, and switching to the more exacting PID control for larger corrections.

The high level correction application adds the corrections returned by the mid level correction methods to the current servo positions. The corrections are thus instantly applied, allowing for more accurate feedback to the sensor. An overview of the entire control structure is shown in Figure 4.3.

4.4 Data Processing

The low level processing of the sensor data has three separate components:

1. Calibration of the zero point
2. Data filtering
3. Running average calculation

The code for these components is in the `accel_Processing.c` and `accel_Processing.h` files.

Data is processed every time the servos are moved, in a routine called just before the servo tick routine. The accel tick routine reads in the data from the accelerometer,

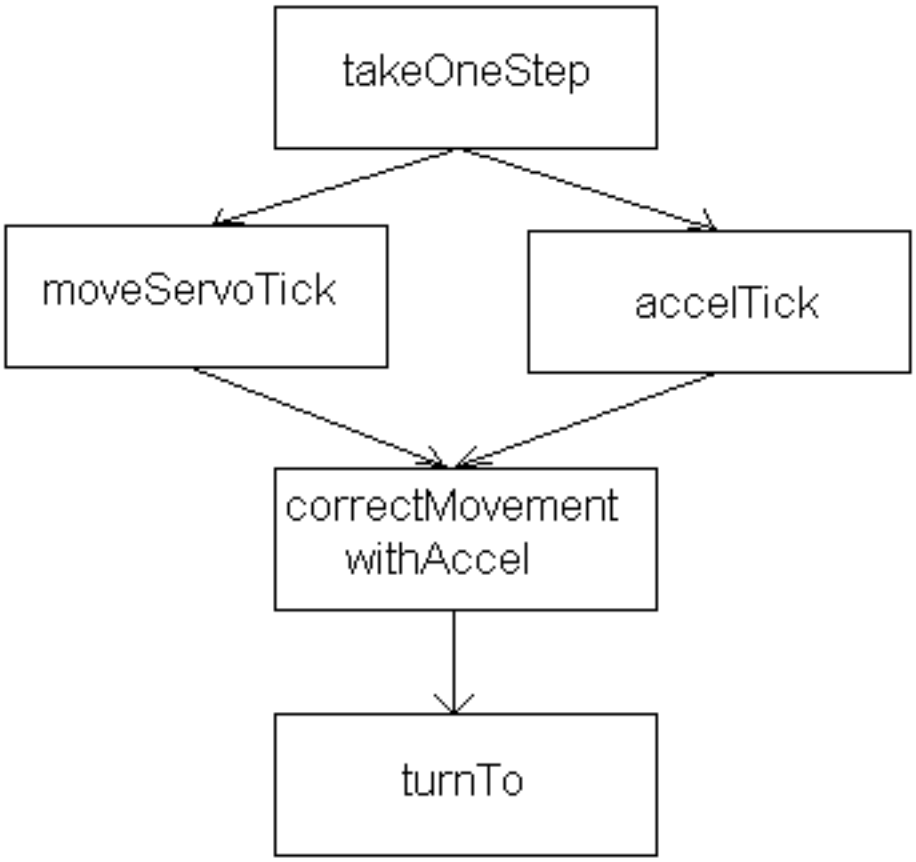


Figure 4.3: Hierarchy of the implementation of the accelerometer corrections.

then processes the readings. The running average, integral and derivative measurements are updated. The new reading from the average is added to the previous ones in the integral, and the difference between this and the last reading is calculated for the derivative.

4.4.1 Calibration of the Zero Point

Initial testing recorded accelerometer readings of approximately 20,000 for a standing position. This lead to the need to calculate a zero point — that is, to calculate readings taken with the robot standing straight up to be zero. Calculating this offset allowed for a more intuitive measurement system and a more reliable means of transferring data relative to a particular position or initial reading to other trials on the robot.

The zero point is calculated by averaging readings to get the mean value. The exact number of readings is specified by `NUM_TO_ZERO_POINT`, currently 200 readings. As each reading is separated by 3 system ticks, this takes approximately 600 ticks (about 6 seconds), sufficient to obtain a reasonable approximation of the zero points.

As there are only slight differences between the calculated zero points, I considered removing the calculations altogether, and replacing this with a fixed zero point. However, this proved not to be a good idea. As mentioned in the documentation (Inc), the accelerometer heats up over time. This means that over a period of time, the zero point gradually drifts to a significantly higher reading. Figure 4.4 shows 10 zero point readings in each plane, taken back to back on Lillian. The controller was turned off for 10 seconds after each reading. After the controller was turned on again, a new

	X Reading	Y Reading
1	18615	18260
2	18774	18354
3	18787	18382
4	18703	18249
5	18890	18493
6	18911	18460
7	18960	18534
8	18693	18290
9	18969	18590
10	18860	18433

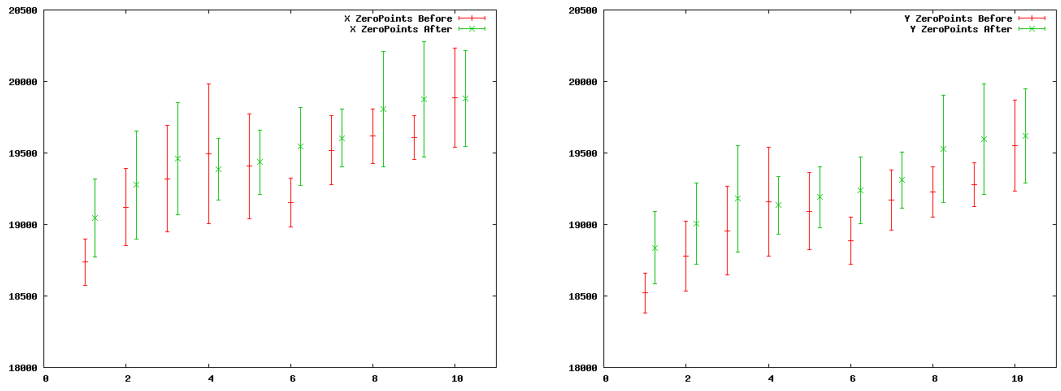
	Avg	Std Dev
X	18816.2	114.67
Y	18404.5	111.41

Figure 4.4: 10 Zero Point readings in the X and Y planes, taken with the controller's power cycled in between each reading. The average and standard deviation are shown.

accelerometer reading was taken. As is shown, the readings gradually drift upwards. The maximum difference between the minimum and maximum readings for X is 354 $(18969 - 18615)$, and 341 for Y $(18590 - 18249)$.

While not significant over a period of several minutes (eg, over a trial), on a larger time scale of an hour or more, this drift is enough to alter the accuracy of all readings, and may interfere with the use of accelerometers for balancing. Furthermore, if these readings progress linearly, they could be used to compensate for drift while using the robot, and the readings would be much more accurate. Testing this theory necessitates a zero point calculation on a much more regular basis. Figure 4.5 shows ten Zero Point readings, taken before and after Lillian walked around. The error bars are the standard deviation calculated on the readings. Each reading was taken after 10 steps had elapsed, taking a time of approximately 2 minutes and 20 seconds or 20,000 system ticks. Most evaluations on the robot are complete in 1 minute; only a few take up to 2 minutes. Note that while there consistently a change in the zero

point over the robot's run, the change itself is not consistent (even whether it is up or down!). It is also usually within the larger of the two standard deviations measured during the zero point calculations. Finally, when tuning, the smallest error used to make a correction is over 400 - much greater than any shift noted. Thus, zero point calculations at the beginning of every run mean that a zero point calculation will almost never be more than a couple minutes out of date. This ensures a reasonably accurate zero point for the balancing algorithm to work with.



(a) X (Sagittal) (b) Y (Coronal)

Figure 4.5: 10 Zero Point readings, taken at the beginning and end of Lillian walking 10 steps.

4.4.2 Data Filtering

Initial accelerometer readings indicated that the accelerometer could only produce a new reading approximately every system tick, as shown in Table 4.1. Notice that of the 20 readings shown, there are 10 distinct system ticks, and 12 distinct accelerometer readings. Therefore, all readings are used in creating the running average, so as

include as much data as possible at the lowest levels. While this could potentially be a waste of processing power, readings taken during actual walking (as in Table 4.2 are taken approximately once every system tick, at roughly the same rate new data is processed by the accelerometers.

Reading	SystemTick	X Reading	YReading
0	30414	19696	24859
1	30414	19738	24859
2	30415	19738	24881
3	30415	19739	24877
4	30416	19739	24877
5	30416	19731	24865
6	30417	19731	24865
7	30417	19741	24864
8	30418	19732	24864
9	30418	19732	24874
10	30419	19726	24885
11	30420	19726	24885
12	30420	19716	24894
13	30421	19716	24894
14	30421	19737	24891
15	30422	19728	24891
16	30422	19728	24881
17	30423	19727	24866
18	30423	19727	24866
19	30424	19720	24863
20	30424	19720	24863

Table 4.1: Accelerometer readings taken while the robot was not moving, showing the maximum speed at which the accelerometer can produce new readings.

4.4.3 Calculating the Running Average

Upon connecting the accelerometer to the controller board, testing was immediately carried out to determine the variability of the sensor. The accelerometer, unattached to the robot except for two thin wires, was placed on a stable and rel-

Reading	SystemTick	X Reading	YReading
0	335	19395	19711
1	0	19617	19515
2	1	19617	19515
3	1	19617	19515
4	1	19841	19464
5	2	20503	20185
6	3	19626	19505
7	4	20412	20255
8	5	19485	19582
9	6	19656	19622
10	7	19349	19354
11	8	19651	19292
12	9	19847	19485
13	10	19570	19179
14	11	21268	20828
15	12	19593	19197
16	13	19529	19331
17	14	20090	19692
18	15	19904	19190
19	16	20007	19510
20	17	19655	19694

Table 4.2: Accelerometer readings taken while the robot was walking, showing the maximum speed at which the accelerometer can produce new readings while the controller is processing the control information.

atively level desk, and readings were taken. The resulting jitter spanned a band of approximately 30-50. While not a huge range, a still reading would ideally be as close to a straight band as possible. The readings here are the exact ones taken from the accelerometer itself, as are the units.

The data collected was graphed with a running averages of 1 (unchanged), 5, 10, 15, 20, 25, 30 and 35 to inspect the changes between them. Figure 4.6 contains comparison graphs of these charts, blown up to provide a good view of the data. A running average of 10 was chosen, as the data is smoothed out nicely, yet still retains identifying features. (This is best noticed by comparing the area before time 500, which has a still section, then a small jag, possibly from bumping the desk.)

With the accelerometer data smoothed out using a running average, and calibrated with a zero point, the data can now be used to aid in movement correction.

4.5 Correction

Correction is handled in two parts. The first is to compare the readings to the desired ones, and come up with the appropriate correction amounts. Only a general overview of the mechanics involved is given here; the algorithms are discussed in Chapter 5. The second part is to then apply the corrections to the actual movement of the robot. The readings are compared and the appropriate corrections are determined in the correctMovementWithAccel() method. Corrections are applied using the adjustTarget() method, as explained in the next section.

The correctMovementWithAccel() method begins by calculating the error on the readings against the desired reading for the current position in the gait. Next, it

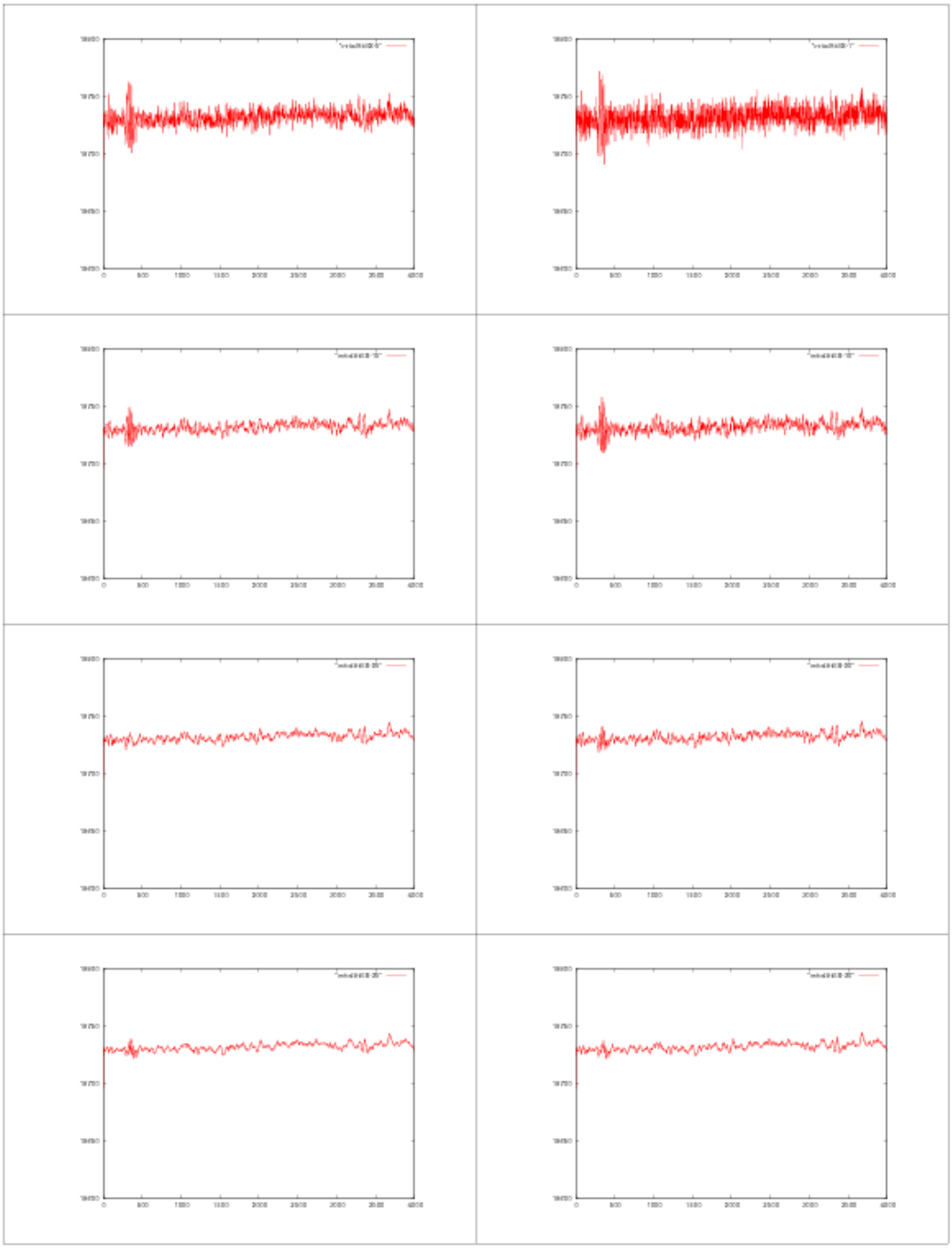


Figure 4.6: Running averages plotted by number of steps, for still readings before and after movement.

checks if any corrections are allowed at this point in time using the tuned correction rate. If corrections are allowed, the error is adjusted using the tuned gains, and then used to calculate the necessary correction amounts.

Error is calculated by comparing the current averaged reading from the accelerometer against the desired reading. The desired reading is calculated from a series of key points that describe a gait. Each point consists of a time (from 0 to 100), and an accelerometer reading. The exact reading desired for any particular point in the gait is linearly interpolated from the nearest key points stored. (Calculations allow for the readings to wrap around from 99 to 0.) Once the desired point has been calculated, it is subtracted from the current reading. This gives a positive error for high readings, and a negative error for low readings, as intuitively would be expected.

Once the error has been calculated, the algorithm checks if a correction is allowed at this point in time. Currently, tuned correction rates cause time delays to exist between corrections, reducing the oscillation. Hence, the time of the last correction is recorded, and if the time since then is equal to or longer than the minimum delay required, a correction may be made. Otherwise, the rest of the method is skipped, and no corrections are calculated or applied.

If enough time has passed and a correction is allowed, the amount of correction required must be determined. For every joint (ankle, knee or hip), in each movement plane (sagittal or coronal), and for each accelerometer plane (X or Y), there is a gain, for each correction method (T, PID or H). The correction is multiplied by the gain (usually 1 over several hundred) to find the appropriate correction for the given joint, plane and accelerometer reading. Each joint is also marked as positive or negative for

each movement and accelerometer plane. The correction is then either left untouched or multiplied by -1 to ensure that the joint moves in the proper direction for the error recorded. Finally, if a correction of 1 or greater has been made to the joint, a flag is set to indicate a non-zero correction has been calculated.

4.6 Application

Once the corrections have been calculated, they must be applied to the robot's movement. `adjustTarget` takes a value and a duration, and applies it to the given servo's target information. There are two main options for how to correct movement when it is linearly interpolated. The first is to adjust the end target of the movement, but leave the end target untouched: adjusting the posture. The second is to adjust the immediate position, and adjust the end target accordingly: adjusting the position. [1]

Adjusting the end target without adjusting the immediate position means that all corrections are applied more to the end of a movement, and take a while to show their effects. Therefore, all adjustments made have a significant lag before the effects show, making corrections much more difficult.

Adjusting the immediate position as well as the end target simply "bumps" the gait pattern up a notch for the particular joint. This allows the effect of any corrections made to be instantly felt, and removes almost all of the time lag inherent in adjusting only the end target.

[1] Technically, there are three: the third is to adjust the immediate position and not the end target. Due to the re-interpolation of the current position done each tick by the servo movement routines, the position would immediately revert to its previous position, with a net effect of leaving the servo untouched.

While the `adjustTarget` method allows for both types of corrections to be made (depending on the duration passed in), I adjust the posture for the gait patterns, and the position for the balance corrections.

4.7 Summary

This chapter detailed the basis for and thought behind the algorithms used to control the servos and the accelerometer, create the gaits and implement the balancing on Lillian. Servos are controlled using key control points, with linear interpolation used to determine the exact current placement of the servo in question. The accelerometer sensor is read in, and goes through a variety of processing. The initial readings are filtered using a running average, and a zero point is calibrated, allowing walks to be compared to each other. The processed readings are then used to create corrections for the gait, using the desired correction method. These corrections are then applied directly to the servo movements. The correction methods used to interpret the accelerometer readings are described in the next chapter, Chapter 5.

Chapter 5

Tuning the Balancing Algorithms

This chapter details the algorithms used to balance the robots, and the methodology used to tune them. The two main correction methods that must be tuned here are the PID and threshold methods; hybrid correction is based on these two methods being tuned already. These methods depend on a well calibrated walk to correct the walk well. Methods of quantifying the goodness of a walk are listed, and then the controllers are tuned through a series of tests, beginning with an exploration of the likely setting ranges with a tilting platform. The tuned controller settings and rates are further refined through balancing a single joint as the robot walks, and then on multiple joints.

On Lillian, each joint is uniquely identified by a three letter acronym consisting of the leg the joint is on (L or R), the joint being controlled (A: ankle, K: Knee, H: Hip), and the plane in which the joint is moving (C: Coronal, S: Sagittal, T: Transversal). The joints are thus identified as LAC, RAC, LAS, RAS, LKS, RKS, LHS and RHS. The three most important joint sets in this thesis are C: the LAC/RAC

joints, which control the C plane, sensed by the accelerometer's Y axis readings; XAS: the LAS/RAS joints in the S plane, and XKS: LKS/RKS which controls the knees in the S plane, both sensed by the accelerometer's X axis readings.

5.1 Control Methods

Without balancing corrections to fine-tune the robot's walk, anything from changing surfaces to ramps can present major obstacles to robotic movement. A control algorithm is needed to modify or adjust the robot's gait to account for changes in the environment. These changes are detected by the accelerometer's sensory readings, which must then be converted into corrections to the walk.

There are many potential algorithms available to convert sensory readings into motion corrections. I used three: a standard PID controller, a threshold based controller, and a hybrid version of the two. These are all implemented as reflexes, similar to the way humans balance. This section will describe each one in greater detail.

5.1.1 PID

The PID controller is based on the concept of target sensor readings, or a baseline. A PID controller will correct the actual reading by calculating the difference from the target point, and basing corrections on that error. Most of the corrections are made by the proportional (P) component of the controller. Lesser corrections are made by the integral (I) and derivative (D) terms. Controllers thus start as a basic P controller, and are further fine-tuned to create P, PD, PI, or PID controllers.

For an accelerometer based PID control, a baseline is created by either taking a

sample from previously programmed good motions (eg, a walking gait), or setting the baseline to be unmoving (stand). The closer the baseline is to the center of the actual readings, the better the corrections prove to be. Corrections are based on the error — the difference between the desired target reading and the current one. The error is multiplied by the gain (error * gain-numerator / gain-denominator). If the absolute difference is greater or equal to 1, the calculated correction is made.

For Lillian, corrections are made to each set of joints, for each axis, individually. (A set of joints refers to the corresponding left and right leg servos at the same joint positions.) That means that for each of the four joint sets, there are six different gain settings (both numerator and denominator) to store: one for each axis (X and Y), for P, I and D.

Further, the time delay between each correction turned out to be much more important than previously realized. Both the X and Y axis had to be tested for how frequently corrections could be made without causing oscillation due to the frequency of corrections - tuning the correction rate. Normally, this can be solely done by manipulating P, but with the humanoid robot, this proved untenable. Setting P to any gain that reacted to robot movement caused overcorrections and oscillation; once set outside this area, no corrections were made. Hence, the rate of corrections was adjusted, and this proved to be effective in controlling the robot. Here, the correction rate is adjusted by controlling the number of time steps that must pass between corrections are allowed again. Thus a correction rate of 1 indicates corrections are allowed on every time step, whereas a correction rate of 4 indicates that corrections will not be allowed again until 4 time steps have passed. This allows the robot

time to react to the corrections implemented by the balancing reflex, before further corrections are allowed. Thus, oscillation is prevented, and corrections are applied and the accelerometer is given time to sense the changes before more corrections are applied.

5.1.2 T

Threshold-based T balancing is based on the methods developed for Tao-Pie-Pie. The corrections are minimal, making it more difficult to over-compensate, though it is thus more liable to undercorrection. Corrections are only a change of one servo setting, of a possible 256 settings. T corrections are technically a simple variant of on/off deadband controllers [de Vegte, 1994] – deadband referring to a preset area where corrections are not applied, and on/off indicating that correction is simply turned on when outside the deadband, and off when inside. House heating is usually controlled by on/off deadband controllers.

T controllers are tuned in multiple steps. The first step is determine the deadband area, where no corrections will be made. For simplicity, this deadband is centered on the the PID baseline, but with a greater range to stop corrections from causing oscillation when the robot is standing still. The second step is to determine the range at which corrections should be applied. This allows for some extra leniency in determining correction values, and a simple way of adjusting the deadband range. Further, this provides a simpler means to adjust settings, and eventually to compare or combine them with PID settings.

As with the PID controller, joint corrections were assumed to be coupled across

the joints. Each set of joints had an error setting for each plane, allowing joints to potentially correct for multiple errors simultaneously. Corrections also had to be spaced over time. Allowing corrections at any time caused corrections to be made to the robot before the effects of previous corrections reached the sensors, resulting in instant and severe oscillation.

5.1.3 Hybrid

Both methods listed above have their own faults: PID, a quick reaction but a tendency to overcorrect, and T less of a tendency to overcorrect, but unable to react quickly to changes. Thus a hybrid (H) method was theorized and implemented, intending to incorporate the best of both methods. The hybrid method uses a threshold correction for smaller corrections, but allows the corrections to become larger (PID based) if the error is large enough. Theoretically, this should allow for the best of both methods to be combined into one.

5.2 Calibration

The first step in correcting the gait is to determine the characteristics of a good or desirable gait. This information can be used to create a *baseline*, used by the corrective methods as the desired reading or reading range from the sensor input. Generally, a baseline is created from readings taken from a known good gait. As a good gait will be stable and repeatable, readings over multiple steps will create a band that can be simply copied and turned into a baseline. Future walks can then be manipulated to follow the preset baseline, regardless of the terrain.

It is essential to the accuracy of the controller to calibrate the baseline carefully. Previous baselines were transferred from older walks (eg., before a hardware change), with the hypothesis that the parameters of any good walk would be similar enough for corrections. This proved not to be the case: differences between the walks were quite likely to cause falls or instability in the robot's gait. As a general rule, the more accurate the baseline is to a good walk created on the robot in its current physical state, the better the corrections. Changes can be made after the baseline has been calibrated, but this should not be attempted without a fairly good understanding of how the changes will affect the robot.

Some changes generally attempted when manipulating baselines are to smooth out oscillations apparent in the baseline, or extreme readings in any direction. The robot needs to lift up its feet to clear steps or sudden changes in height, necessitating large shifts to either side, but there is a fine line between the angle needed to lift a foot sufficiently high, and an unstable angle that may cause the robot to fall or become unbalanced. Thus, these leans to one side or the other may be adjusted to allow for enough foot clearance, but less instability.

5.3 Measurements

While analysis of a robotic gait is often subjective, there are several measurable factors that should improve. These can be used to measure the effectiveness of an algorithm as well, as an effective algorithm should noticeably improve a gait based on these criteria. First, the robot should be more robust. Second, the gait should conform more closely to the baseline. Third, the robot should move faster and farther.

Fourth, the development time creating gaits should decrease. The importance of each of these traits, as well as how to measure them, is described below.

5.3.1 Sensor Plots

Initial measurements began by looking at plots of the accelerometer readings from the robot. Stability, repeatability and conformity could be determined simply by looking at graphs, and comparing them with readings from other walks. By recording the time step of each reading during the walk, multiple steps could be layered on top of each other, allowing the steps taken during a single walk trial to be measured directly with the other steps. This also provides an indication of variance, as less variation between steps indicates a more stable, repetitive walk.

Stability was noticeable, as falls or strong leans showed up as large spikes on the graph. The least stable walks showed a large range of readings over the entire walk, while relatively stable walks would have only small ranges, or few areas in the walk with large ranges of readings. Conformity could also be measured by overlaying the baseline or thresholds directly on top of the walk readings.

In terms of tuning, the graphs also could be used in early testing to observe signs of oscillation caused by the correction method.

Graphs, however, became difficult to judge as the corrections improved. It becomes very subjective, to determine which of two graphs is better when both are reasonably decent walks, and further to determine which corrections are quantitatively better. The SAE (described in the next subsection) was implemented to address this problem.

5.3.2 SAE

The Sum of Absolute Error (SAE) is a quantitative measurement used throughout this thesis to determine the relative goodness of varying walks. As all the correction methods have a baseline (or a set of thresholds), deviations from the baseline can be measured, and used to directly compare one trial with another. Summing the absolute errors allows the total deviations from the baseline to be measured. The greater the total deviation, the less the walk conforms to the baseline, and the less the corrections are helping. The less deviation, the more the walk conforms to the baseline, and the better the corrections.

The Sum of Squared Error (SSE) is commonly used to evaluate the fit of a line to multiple data points. Here, it could be used to record the distance from the baseline or the threshold. Because the error is squared, the values are absolute, allowing the distance from the line to be recorded. However, as the values are squared in terms of error from the line, values close to the line are insignificant (and generally ignored) in comparison to values further from the line. This would cause many of the data points collected to be ignored because of outliers, and give excess weight to the occasional faulty reading that would be incompatible with its actual importance. Therefore, the SAE was used.

SAE readings can then be plotted to look for trends, as well as comparing trials. Further, these calculations give a good indication of the stability and conformity of the walks. Thus, they are a good way to quantitatively and directly compare both the gait and the algorithm.

5.4 Tuning

Once a walk has been calibrated, it must be tuned. This thesis distinguishes between calibrating (the act of creating a baseline for a gait), and tuning (adjusting the gains and settings on the balancing algorithms). Experience has shown that tuning is less tied to a particular gait than calibration. Once the robot has been properly calibrated for a gait, physical repairs to the robot such as servo replacement will alter the tuning very little. However, the actual tuning can be rather time consuming. As with most balancing approaches, I have tuned the robot for only one balancing plane, before complicating matters with multiple planes, inclines, or uneven terrain. Further, as differing gait disturbances produce oscillation at differing points in the tuning, small increments of complexity are necessary to allow for the robot to be properly tuned to each new task. As many researchers use only one plane, or do not specify a method by which to tune for increasingly complicated balancing during a gait, I will explain how I found it easiest and most effective to tune Lillian. While I relate a logical sequence of testing, I often found myself having to go back a few steps and retune an earlier test in order to prepare for or double check a later tuning.

5.4.1 Stand

The initial tuning is extremely simple. The corrective methods are simply tuned to not oscillate when the robot is standing completely still (regardless of the running average, some jitter will appear on the sensor, as well as the robot, at this point). This gives a minimal value at which to start tuning the PID method, and a range to use in calculating the threshold bounds. The threshold boundaries will be set slightly

(a) Flat (b) Tilted

Figure 5.1: Balancing platform for testing when flat (a) and tilted (b).

under where compensations cease to make the robot jitter when standing still, with the gain beginning just outside the boundaries. Maximum PID gains will also be set to allow the robot to stand without jittering.

5.4.2 One axis - Tilt

The next step is for the robot to stand still while the surface under it tilts. Obviously, the robot must now start correcting for deviations from its desired pattern. In order to minimize possible complicating factors, the robot will correct in one axis at a time, attempting to maintain a steady sensor reading of zero. A moving platform that only moves in one plane at a given time (like a teeter-totter) was constructed (see Figure 5.1), and the robot placed upon it.

5.4.3 One axis - Walk

Once the robot can remain stable while tilting, a more complicated sensor pattern can be given to the robot, and the tuning adjusted. Following a walk baseline instead of a steady line adds another level of complexity to the balancing, exposing previously hidden oscillation, as the balancing must compensate for unexpected readings at many levels and speeds. Once again, the robot is returned to a flat surface, to lessen the complexity involved in the balancing.

5.4.4 Two axis - Tilt

Balancing in two axes is much more difficult than any of the previous tasks, as any oscillation (or even overly quick corrections) in one axis can produce a rebound and perhaps oscillation in the other axis. Thus, two axes are not tested until balancing is working effectively in one axis. Again, balancing starts with a simple tilting platform, allowing compensations to be made to a basic pattern before anything more complicated is introduced.

5.4.5 Two axis - Walk

Once satisfied that the robot is balancing well, I can again use a more complicated pattern - walking on a even surface while correcting two axes. As with the rest of the tuning, a simpler case is tried first, then uneven terrain. Balancing is performed on a simple walk, on a flat surface to begin with, checking that corrections will only be made if necessary (in this case, if the baseline is different from the walk). The tuning, however, is limited in this thesis to walking on a flat surface. Further tests are used

to evaluate the now-tuned algorithms instead of continuing to use them to tune the robot.

5.5 Tuning Configurations

Tuning configuration began with P and T methods on the tilting platform. The test results from the tests were used to further define P and T tests for walking. The best of these results were then used to test the hybrid method. The best of all results were then used for the randomized perturbed walks, and the stepping field tests — the final evaluation.

Standing tests showed that Lillian would oscillate while standing still with a one degree correction for an error reading of 300, while remaining mostly still for a one degree correction for an error reading of 350. Tests were thus begun in this range, and reducing the gain throughout. The thresholds were set at ±200, allowing a little leeway in further tests. P, having no boundaries, simply began testing at 350.

5.5.1 Tilting

Testing on the tilting platform began with the P and T correction methods applied to individual joints (XAS, XKS, and C), and then the best results were used to create further walking tests. Balancing on a moving platform is an excellent way to determine if a robot can compensate for errors. The platform was tilted from -30° to +30°, from a starting position of 0°, with an angular velocity of 240° per minute. A metronome was used to count time steps during the testing, regulating the velocity. Initial tests had been proposed with angular velocities of 60°, 120° and 180° degrees

(a) C Tilt (b) S Tilt

Figure 5.2: Tilting: Lillian is show tilting in first the C plane, and then the S plane.

per minute, but the velocities were so slow that differences between the settings were difficult to determine. Further, the velocities did not approach the speed of a walking gait. Figure 5.2 shows Lillian on the tilting platform, completing tests runs in both the C and S planes.

The tests were coarse grained, looking at correction values of 150, 450, 750 and 1050, with correction rates of 1, 4, and 7 timesteps required between corrections. As this was intended to provide an overview of the balancing approaches, each test was run only once. This allowed a general understanding of the problem area, leaving finer investigations for more complex testing.

Data from the tests on controlling multiple joints while tilting is also included in this section. Manipulation of the robot revealed that adjusting the robot by one servo setting at the knee, and one servo setting at the ankle caused the robot to adjust its torso forward or backward, while remaining standing up. This meant that initial standing tests could simply use one S setting to complete a coarse grained

examination of the settings and rates.

Due to the previous tests results, a correction rate of 1 was not considered, being replaced by a rate of 10, to allow for increased interactions between the joints controlled. Similarly, a setting of 150 was removed, but otherwise the same control settings previously used (450, 750 and 1050) remained.

5.5.2 Walking - Single Joint

Results from the tilt testing (see section 5.6) were used to create parameters for a series of tests to be run on a simple walk gait, and further explore more interesting looking values, while culling those that were simply untenable. Testing began with T, and results from those tests further pruned the tests run on the P method (as P is much more likely to react than T, and thus produce oscillation or poor results, and possibly damage the robot). Table 5.1 shows the configurations chosen for the testing. Tests were run on the XAS and XKS joints using the threshold method, beginning at a setting of 250, which was increased in steps of 100 until the final test value of 950 was reached. These tests were repeated with rates of 4, 5, 6, and 7. Testing on the ankle for the Y axis was similar, with a range of 150 up to 850, again in steps of 100, and repeated under rates of 2, 3, 4, and 5. Each of these tests was run 3 times to reduce the possibility of outliers.

For P testing, the ranges tested were similar to those of T. As PID is based on a single centerline, testing ranges appear higher than those of thresholds, but because T corrections are based on distance outside the boundaries, the actual error from the baseline would be similar in both methods. PID testing began by testing only the P

T					P				
Joint	Low	High	Step	Delay	Joint	Low	High	Step	Delay
XAS	250	950	100	4 – 7	XAS	450	1150	100	6 – 7
XKS	250	950	100	4 – 7	XKS	450	1150	100	6 – 7
C	150	850	100	2 – 5	C	350	1050	100	4 – 5

Table 5.1: Initial configurations for Walking Tests: Tests for a joint under the given correction method are shown with the low and high end of the tested range, the size of the step used to test within the range, and the range of correction rates each control setting was tested upon. Only P settings were originally testing for the PID controller.

setting, as it is the most important one for tuning the controller. P testing for the S plane began at 350 for both the XAS and XKS joints. The upper limit was 1150, once again with a step of 100 in between settings. Only two rates were originally tested under this correction method — 6 and 7 — as the other lower values were shown to cause oscillation under threshold testing, and offered little or no chance of improvement under P corrections. C was similarly only tested for two correction rates, 4 and 5, on a range of 250 through to 1050, with a step size of 100. The 250 test setting was removed after the first test run, due to oscillation. After initial testing was completed, higher correction rates and control settings looked promising, so the tests were expanded to include rates up to 9 and control settings up to 1250 for the XAS joint. Based on the results of these tests, the XKS tests only looked at correction rates of 8 and 9, and cancelled the 250 tests in their first run, due to their extremely bad results. The C tests similarly did not test a control setting of 150, or a rate of 2, as the initial run showed them to be extremely poor.

Following the testing of the P setting, the best setting from each joint was taken and tested with a range of differing D settings. This allowed a reasonable D to be

Joint	Low	High	Step	Delay
XAS	250	1250	100	6 – 9
XKS	250	950	100	8 – 9
C	250	850	100	3 – 5

Table 5.2: Revised P Walking Tests: Tests for a joint under the given correction method are shown with the low and high end of the tested range, the size of the step used to test within the range, and the range of correction rates each control setting was tested upon. Only P settings were originally testing for the PID controller.

calculated for use in further testing. The settings tested for D are shown in Table 5.3. A new test, based on the tilting test, was created to test the derivative setting. The robot was given a threshold that stayed at +1000 for the first half of the gait, and then moved to -1000 for the second half. No movements were included in this gait, so the robot would only be able the adjust to the desired accelerometer readings based on the movement corrections implemented. It was then possible to look for the overshoot caused (or avoided) by varying the D parameter. These tests were only run once each, as they were a coarser grained exploration of the search area.

Joint	P	D Low	D High	D Step
XAS	850	50	500	50
XKS	750	50	250	50
C	550	50	100	25
	550	100	200	50

Table 5.3: Configurations for Derivative Tests: Derivative tests for a joint are shown with the P setting chosen for the tests, the low and high end of the tested derivative range, and the size of the step used to test within the range.

5.5.3 Walking - Multiple Joints

Once the results from the walking tests were collected and analyzed, the best of the various single joint settings were used to create tests for multiple joints being corrected simultaneously. The results from the single joint tests were analyzed, and the top tiers of results decided upon. These were results that improved on the baseline where possible, and the best results available otherwise. The top tier was divided into two (best and good). For testing the multiple joint settings, each of the best joint settings was paired with each of the best and good settings for another joint. This allowed for a reasonably thorough testing without factorial explosion of the test cases. Each of these tests were run three times, to reduce the noise in the data. Further, S joints were first tested and combined with each other (XAS and XKS) before adding in a second plane with the C ankle joint. This follows the ideology of the prior tests in adding in as little complexity as possible to each test to get the clearest possible picture of the effects of each new factor.

For D, a similar process was carried out, as the best setting for multiple joints was tested against a range of the best settings determined in the prior D testing. Both P settings were enabled, and then one D setting at a time was enabled for each range. Finally, both D settings were turned on, and the better of both ranges used simultaneously to compare the benefits of D on multiple joints. These tests were all run three times each, to reduce the possibility of noise interfering in the results.

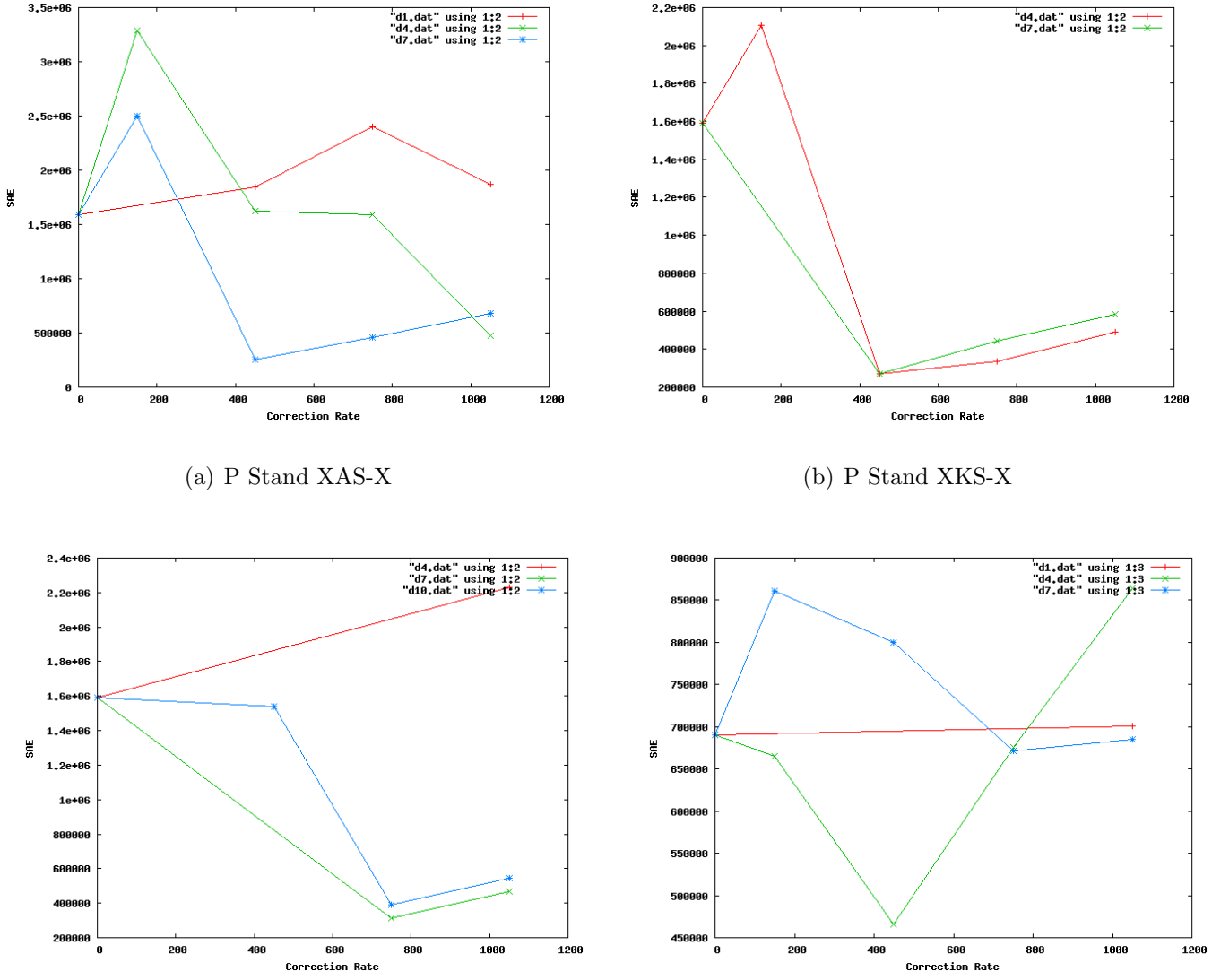
5.6 Tuning Results & Analysis

Each set of tests was run and analyzed before moving on to the next set. This allowed the findings from the previous sets to be used to refine future tests, as is shown in the tuning configurations in Section 5.4. The results of the different tuning tests are examined here, with the reasoning used to refine the tuning as the tests progress.

5.6.1 Tilting

Figure 5.3 shows the results of testing the P algorithm on the balancing platform. Results are shown for each joint set (XAS, XKS, and C), and for one combination of joints (S). Only the most relevant data is shown in this set of graphs; for the data recorded in the other plane, see Appendix 8, Section 8.1. Graphs in Figure 5.3 show the data organized by the correction rate, while Figure 5.4 shows the same data ordered on the control setting. Missing data points indicate that tests were unable to be completed due to the extreme movements caused by the correction settings.

Every graph in Figure 5.3 shows its worst results with a correction rate of 1. Moving to a higher rate (either 4 or 7) results in significant improvement over no corrections, and the lower correction rates. The best rates differ between joints; the XAS graphs show a decided improvement when the rate is increased from 4 to 7, while the remaining graphs show minor differences between a rate of 4 or 7. Looking at Figure 5.4, the general trend remains that larger settings (i.e., higher control settings) provide more conformity than lesser ones. This comes with a caveat, as the XKS graph clearly shows that simply increasing the control setting does not improve



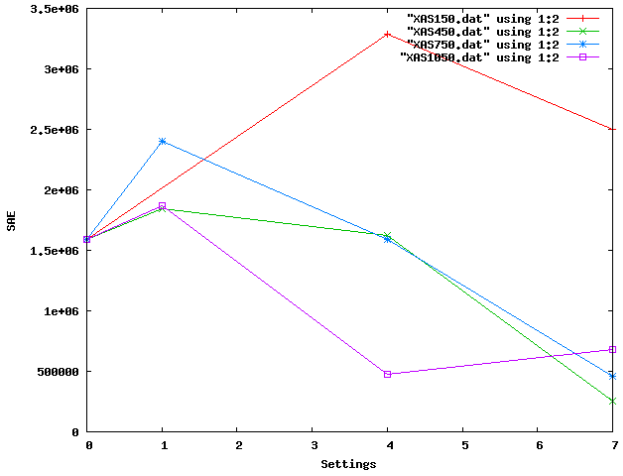
(a) P Stand XAS-X

(b) P Stand XKS-X

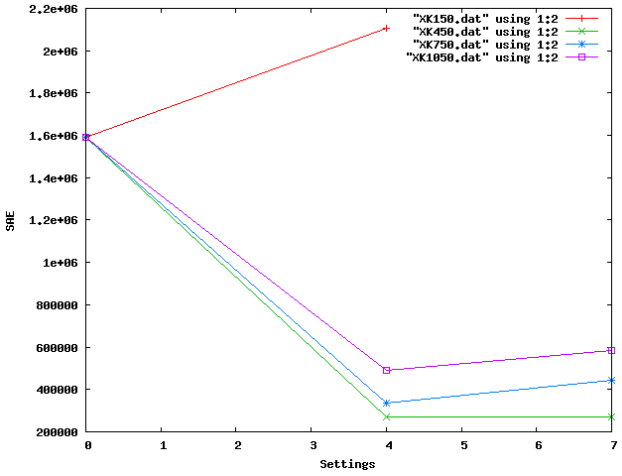
(c) P Stand X-X

(d) P Stand C-Y

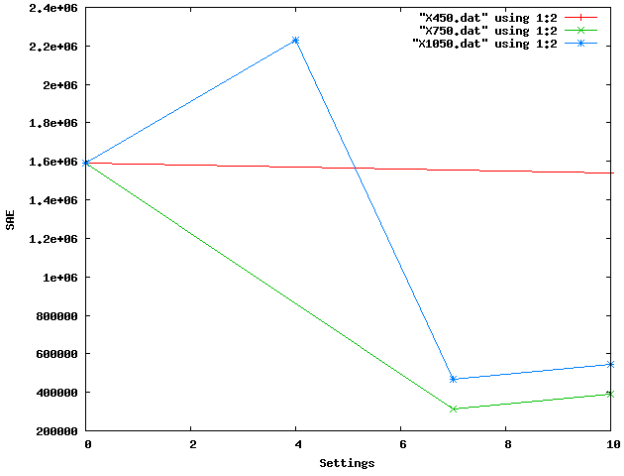
Figure 5.3: P test results for a tilting platform, by correction rate



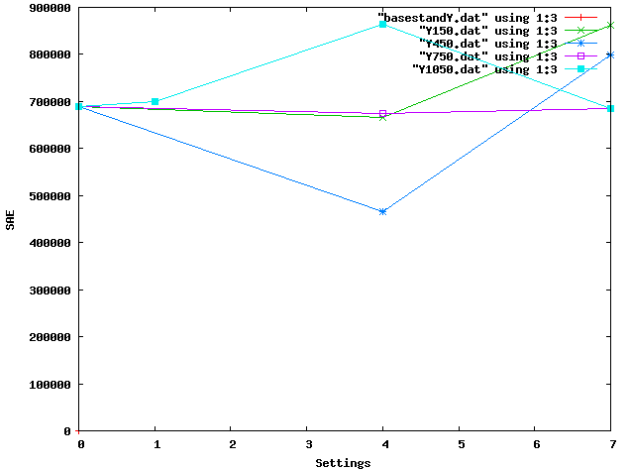
(a) P Stand XAS-X



(b) P Stand XKS-X



(c) P Stand X-X



(d) P Stand C-Y

Figure 5.4: P test results for a tilting platform, by setting

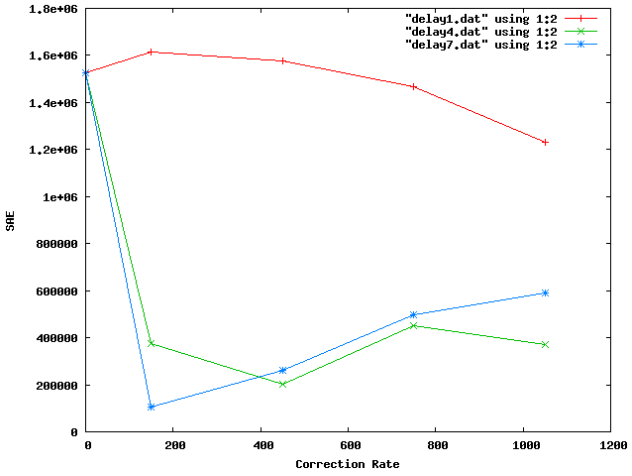
performance; in the XKS graph, increasing the control setting past 450 decreased the measured performance. The XAS graph (especially the 1050 line) similarly shows that increasing rate for a given setting doesn't necessarily improve it. Therefore, while there may be a relation between correction rate and setting control, it is not a linear one. General combinations of settings (in this rough grained testing) appear to show areas for further testing, but also show that these areas may not be easily related to each other.

Figure 5.3 (c) shows the effects of controlling both XAS and XKS, coupling the joint settings in the P algorithm. In general, a correction rate of 4 was useless (all the tests but the one at 1050 had to be cancelled). There was relatively little difference between rates of 7 and 10, with a rate of 7 being slightly better overall. The higher control settings were also more effective, with the second highest setting of 750 providing better results than 1050. This would indicate that further testing should examine higher rates and settings with two joints than with one, a not unexpected result.

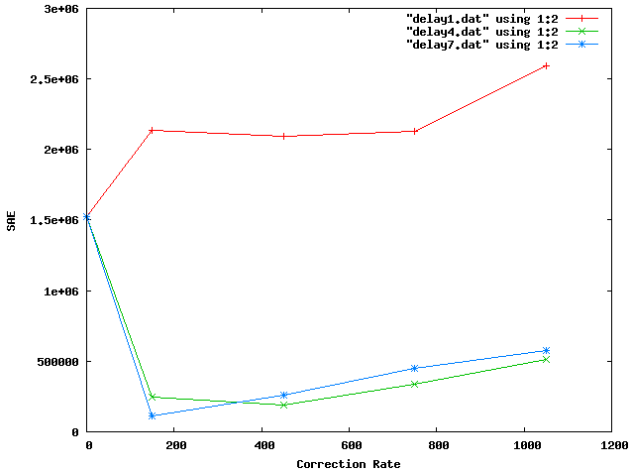
Figures 5.5 and 5.6 show the results of the tilt testing for the threshold method. As with the P results, only the most relevant results for the joints tested are shown, with the full set of graphs available in Appendix 8, Section 8.1.

The first major point to notice with the T graphs is the greater range of useful settings. While the tests with a correction rate of 1 were generally poor, most other settings are an improvement on the baseline SAE. Thresholds thus provide a greater range of useful settings to choose from than the P method.

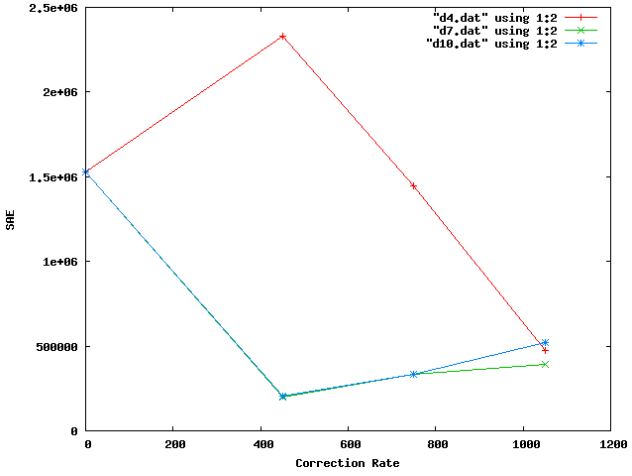
Similar to the P tests, tests for the threshold method show the greatest differences



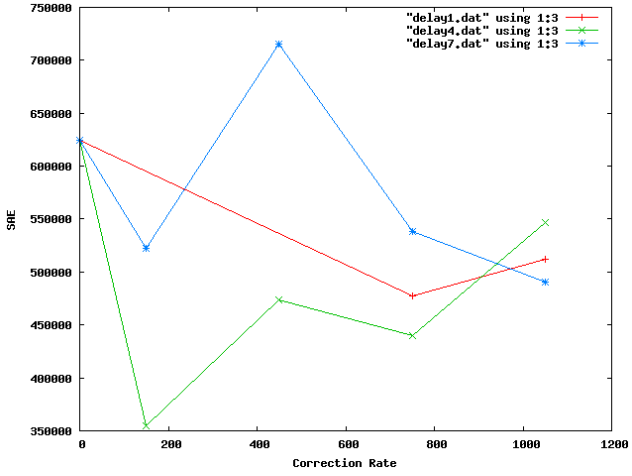
(a) T Stand XAS-X



(b) T Stand XKS-X

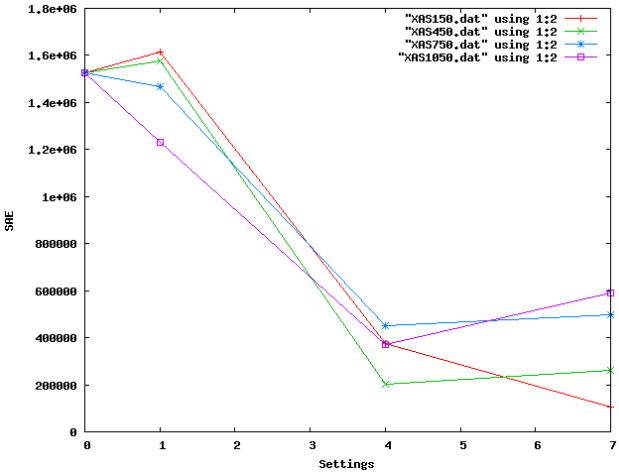


(c) T Stand X-X

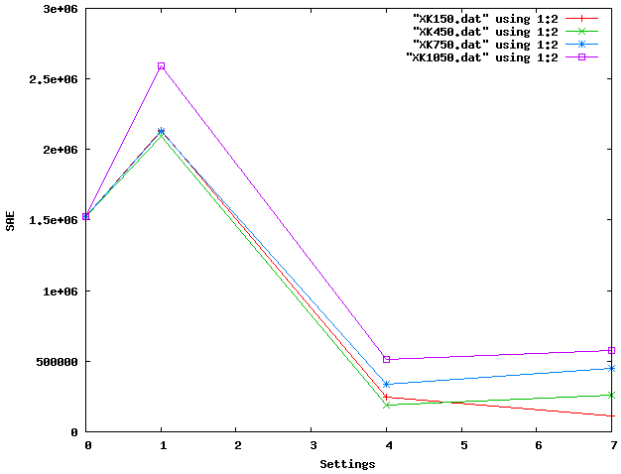


(d) T Stand C-Y

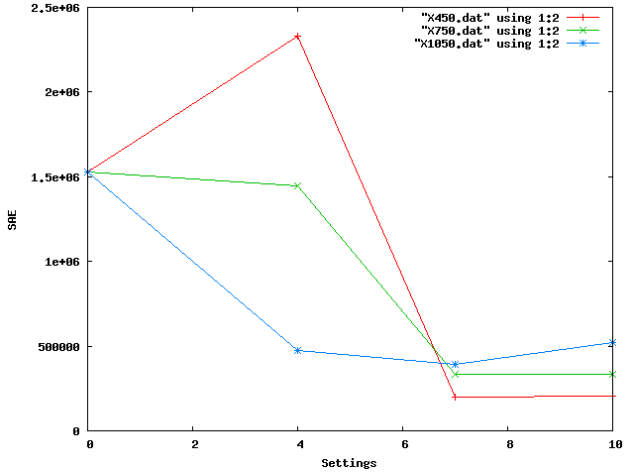
Figure 5.5: T test results on the balancing platform, by delay.



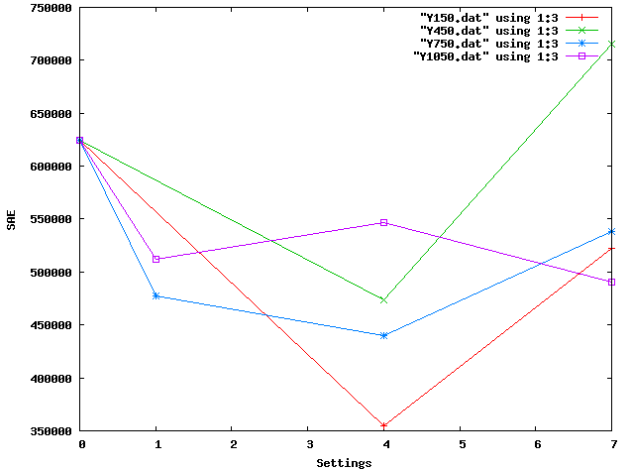
(a) T Stand XAS-X



(b) T Stand XKS-X



(c) T Stand X-X



(d) T Stand C-Y

Figure 5.6: T test results on the balancing platform, by setting.

appear from changing the correction rates. This only affects the measurements up to a certain point, however, as the XAS, XKS, and S graphs all show very minimal differences between a rate of 4 and 7, though both of these are significantly better than a rate of 1. The C graph shows that too low a correction rate can actually be detrimental to the balancing, as the rate of 7 is noticeably worse in general than the rate of 4 or 1. Looking at the data arranged by setting, the XAS, XKS and S graphs once again display similar trends. For these graphs, a setting of 450 is generally best, and higher settings do not improve the balancing. The C graph is generally best around a value of 750, though it displays one major low point with the 1050 setting.

The T results for coupling the S joints show a similar result to what was found with the P algorithm: controlling multiple joints will require higher settings and rates than a single joint. Here, a rate of 4 is much worse than the higher rates of 7 and 10. The setting of 450 still appears best overall, though with a lesser improvement over the higher settings than previously resulted.

These graphs and values will be used to help create a set of experiments for the next set of tests, exploring balancing constraints applied to walk data.

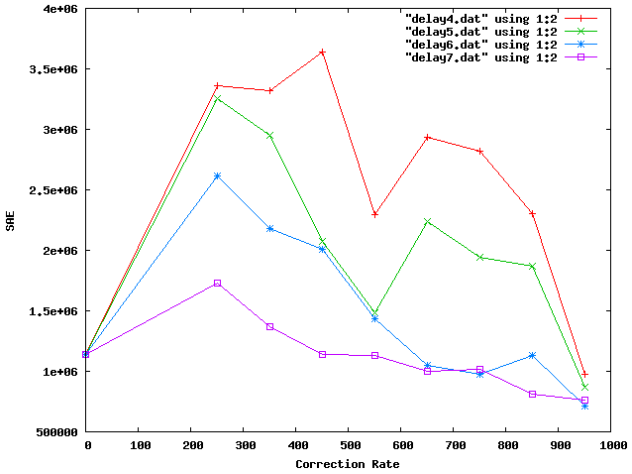
5.6.2 Walking - Single Joint

Once the tilting experiments were complete, the tuning ranges found were used to create experiments with the robot walking. Only a single joint was corrected to allow for clearer test results. The knee and the two ankle joints were tested. Results for both the T and P controllers are discussed here.

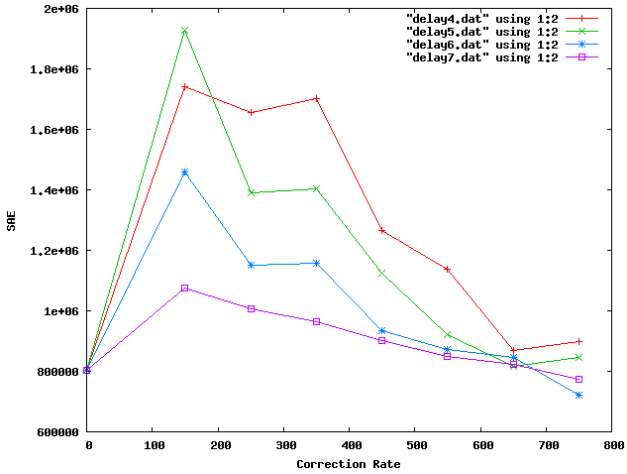
T Controller

Testing T while walking confirmed that the general tendency of the corrections was to improve the walk most at the highest settings. Numerous settings improved on the basic walk, but they tended to cluster around the higher control settings and correction rates. No directly linear relationship was observable however, as is illustrated by the XAC results. For the control setting of 950, all tested rate values were better than the original walk, while at the other end of the scale, only the highest correction rate of 7 for the median control value of 550 was better than the uncorrected walk. One would then expect that the intermediary control setting values of 850, 750 and 650 would gradually and linearly increase the rate values that were better than no corrections. While these three values are better than no correction with a rate of 6 or 7, all three are significantly worse than no corrections with a rate setting of 5. This argues against a linear relationship between correction rate and control settings, and suggests instead an exponential relationship which would introduce "pockets" of good control settings, as shown.

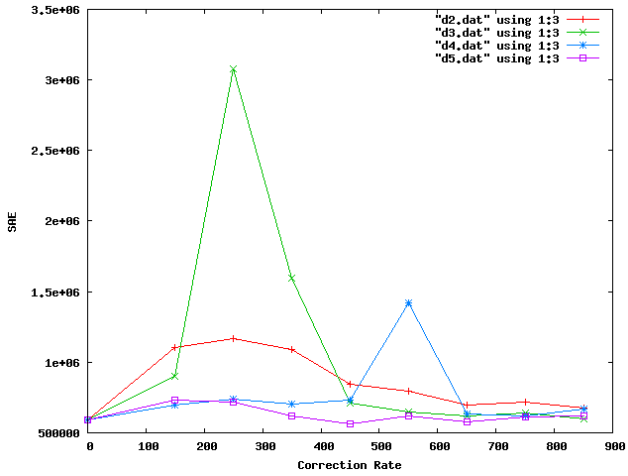
The XAS control settings for 650, 750 and 950 (as shown in the XAS-Setting graph) all plateau between a rate of 6 and 7, suggesting that there is no use further expanding the tested rate areas. The control settings all appear to converge on the same value at the highest end of the testing area, with the correction rate making relatively little difference at all in the test results. This would suggest that the corrections are made infrequently enough because of the settings that how often corrections are made makes only a minor difference. The best results all appear to be at a rate of 7, with settings of 850 and 950. There are a number of control



(a) T Walk XAS-X

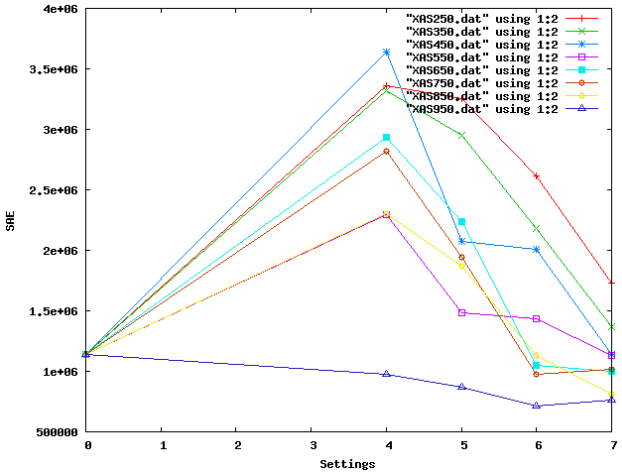


(b) T Walk XKS-X

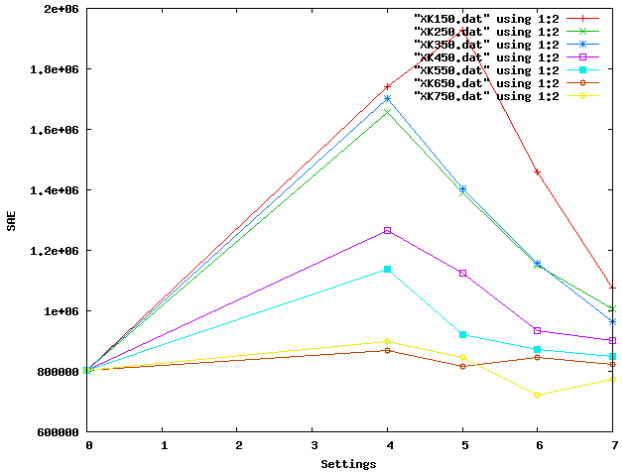


(c) T Walk C-Y

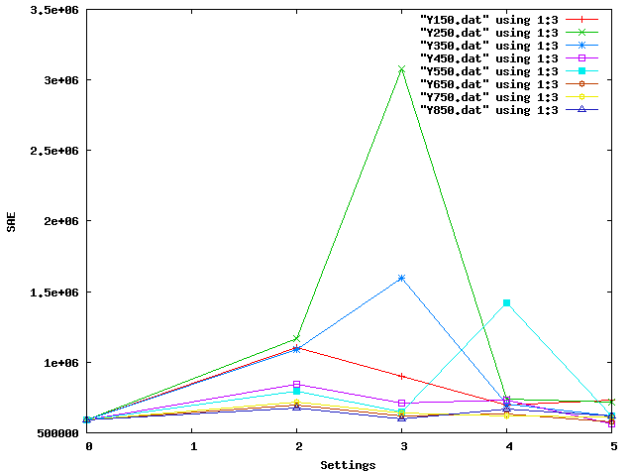
Figure 5.7: T test results while walking, by delay.



(a) T Walk XAS-X



(b) T Walk XKS-X



(c) T Walk C-Y

Figure 5.8: T test results while walking, by setting.

combinations that improve the walk, mostly those with a rate of 6 or 7, or a control setting of 950.

The threshold control applied to the XKS joint does not show as much of an improvement over the original walk as the XAS joint did. Only two settings (750 with rates of 6 or 7) both show an improvement on the uncorrected walk. Looking at the XKS-Settings graph shows that the most significant factor in improving the walk is the correction rate. The control settings overlap each other and show less improvement between settings than with the rate. However, a setting of 650 or especially 750 shows that these settings are generally about as good as the uncorrected walk, and occasionally better. The tested rates plateau at the end of the test range, and do not appear to indicate further testing would yield improved results, while the control settings all currently improve as they increase. Further testing of these control settings could be beneficial, but would not necessarily result in better settings.

T testing on C shows minimal improvement over the uncorrected walk. As the rates get higher, the corrections improve slightly, but the C-Delay graphs shows the rates plateauing at 4 and after. When sorted by settings, the corrections improve slightly after 450, but not much. Two settings are actual improvements on the uncorrected walk - 450 and 650 at a rate of 5.

Due to the non-linear relationship between the rate and setting values, it is possible that an untested combination of settings (higher more likely than lower) would produce better results than any of those already tested. However, general trends are evident in the graphs of the initial test results that indicate the best value ranges have already been tested.

P Controller

The P controller showed "pockets" of good control results, similar to the threshold controller's results. Overall, however, few test combinations improved on the uncorrected walk, and only for C.

Test results for the P controller on the XAS showed that no settings tested were able to improve on the uncorrected walk. The original testing area was greatly enlarged to explore higher settings and rates, but the combinations of the highest settings and rates appear to plateau, suggesting further tests would not be useful. In fact, the highest rate setting of 9 is actually worse, in over half the cases, than the next highest rate setting of 8. This lends support to the existence of a non-linear relationship between rate and settings in the P controller as well as in the threshold controller. Looking at the control settings, the higher settings improve the control, but appear to plateau (or even worsen) on the two highest settings, for the two highest rate settings, suggesting that further testing would not improve the control.

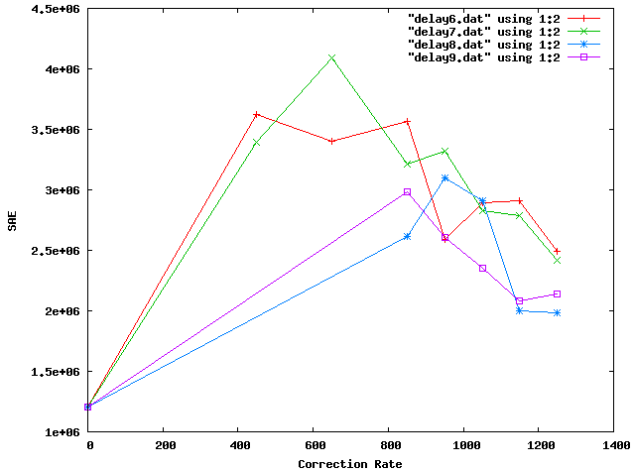
Testing on the XKS joint also found no settings that could improve upon the uncorrected walk. The differing correction rates showed minimal differences between the two settings, with the control settings showing more variation. The best setting was still 950, with the rate affecting it very slightly. Most of the other values tested clustered around the same range, except for one bad outlier, set at 550 with a rate of 8. The original tests at a control setting of 350 had to be cancelled, due to the oscillation in the robot's movements. Overall, the XKS P corrections affected the total walk less than the XAS, but were unable to improve the walk.

C tests show a noticeable difference from either of the S joints. Every setting

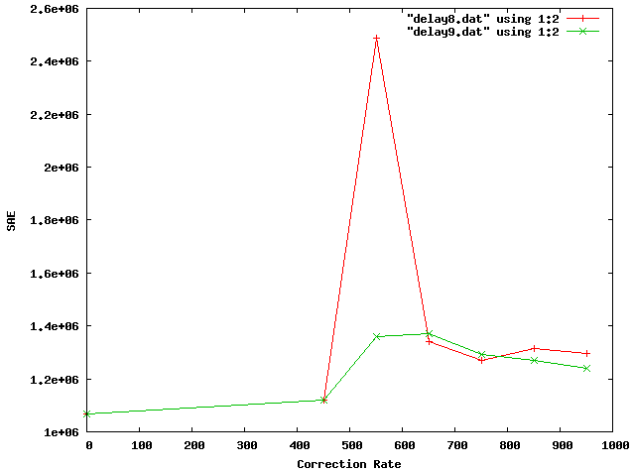
for C improved on the original walk. Part of this improvement is due to the robot's feet moving to a position that effectively stops swaying from side to side. Results show a distinct improvement on moving from a rate of 3 to a rate of 4, but little or no improvement with a rate of 5 or greater. The best correction setting, especially for the two higher rates, is 550, but any setting except 250 and 350 show a good improvement over the uncorrected walk.

In general, the S joints (XAS and XKS) both showed worse results with the P controller. Visual examination suggests that any correction made by the P controller is simply too strong a correction, once an error is noted. Further, the corrections were worst on the S joints, those with the more complex interactions,and more likely to over-react to strong corrections. The robot is especially sensitive to sway in the sagittal plane, as the multiple joints in the leg provide weak points where the sway-based torque can overpower the servo motors. The coronal plane of the robot has no axes of rotation outside the ankle joint, making it is much less susceptible to sway. Thus, the coronal plane is better able to accommodate handle small overshoots without oscillation. This could explain why the P method works so well on the coronal plane, but not on the sagittal plane.

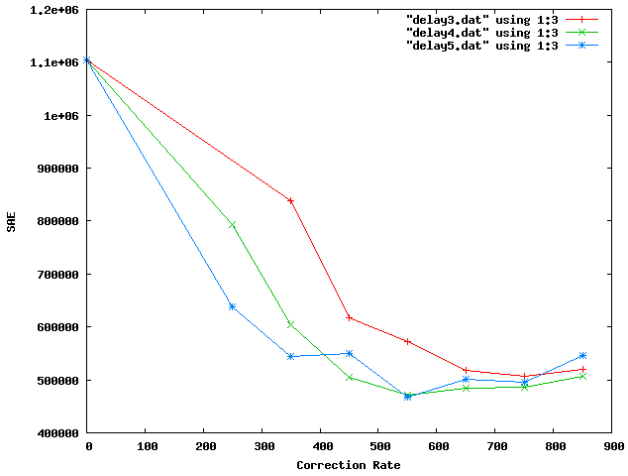
It is possible that further testing will locate minima beyond what has already been discovered, but unlikely. During these walks, corrections were already insensitive enough that points in the walk where corrections should have been made were noticeable.



(a) P Walk XAS-X

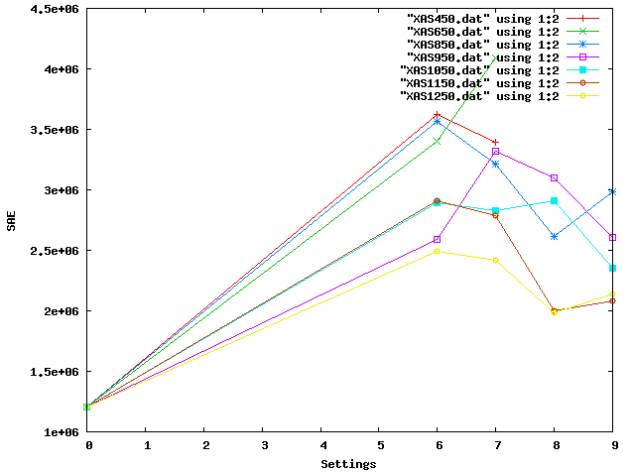


(b) P Walk XKS-X

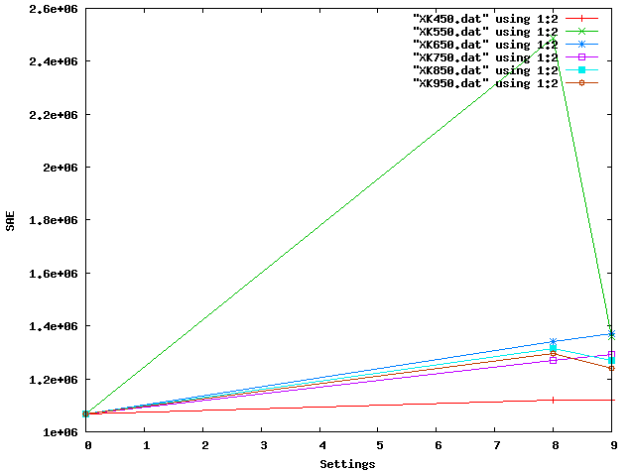


(c) P Walk C-Y

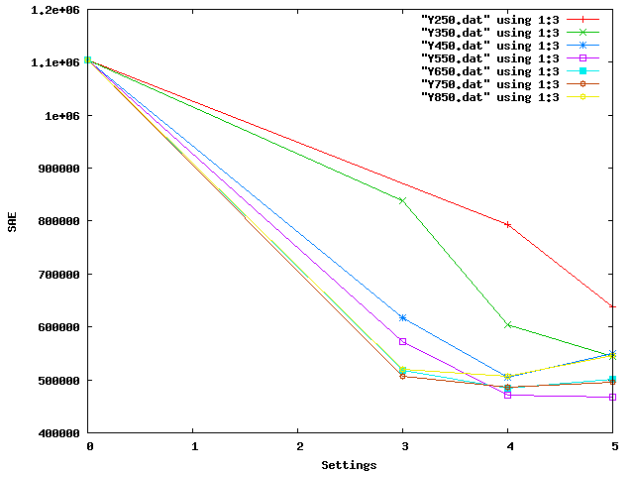
Figure 5.9: P test results while walking, by correction rate



(a) P Walk XAS-X



(b) P Walk XKS-X



(c) P Walk C-Y

Figure 5.10: P test results while walking, by setting

Tuning D

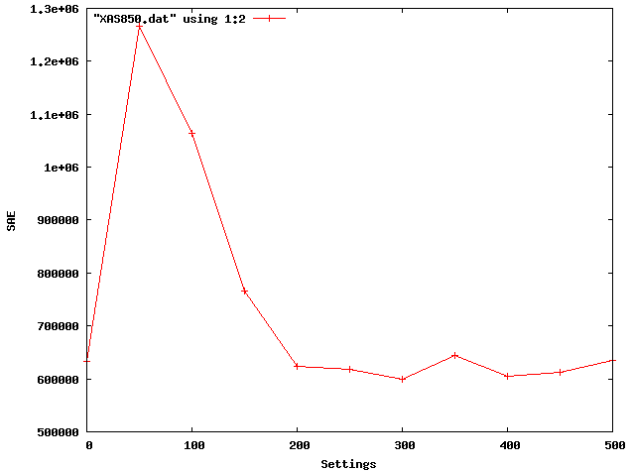
Most corrections implemented by a PID controller are due to P. However, the next common setting to tune is D, to avoid overshoot. After tuning P, therefore, each joint was tested for various settings of D to see if the controller would improve by hitting the desired setpoint more quickly, or worsen by increasing the tendency to overshooting.

The SAE was calculated based on the baseline given to the robot, shown in Figure 5.11. Each set of tests was also monitored visually, and this data was used to choose a setting.

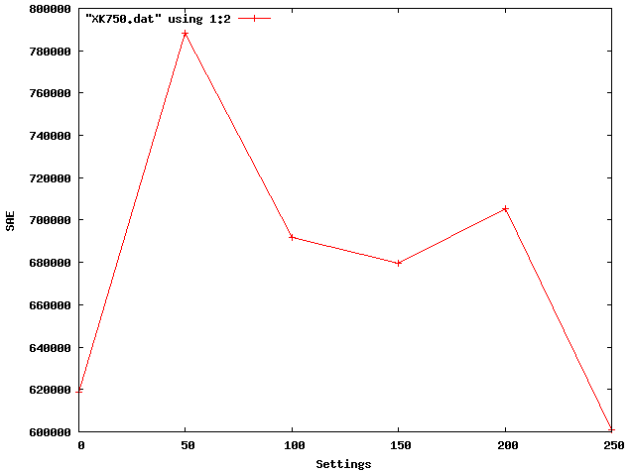
The XAS graphs shows that any setting from 200 on up is relatively good, and that most of those settings are slightly better than corrections with no set D. Visual inspection noticed no oscillation from a setting of 250 and upwards, and chose 450 as the best setting.

The XKS SAE graph shows that 250 is the best setting, followed by 150, though only 250 is actually an improvement on the uncorrected walk. Visually, 150 was the only setting without minor oscillations at the forward lean, leaving XKS unimproved by D.

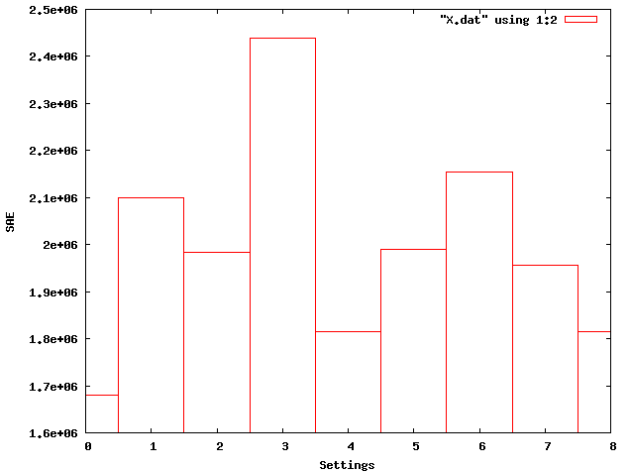
C produced little improvement with the tested values. Visually, the range of values tested (25, 50, 75, 100, 150 and 200) showed no oscillation, and appeared very similar for values of 75 and up. However, the SAE show that none of these values improve the uncorrected walk. Based on visual observation, the value of 150 was picked for D, as it gave the smoothest appearance of change when the robot moved.



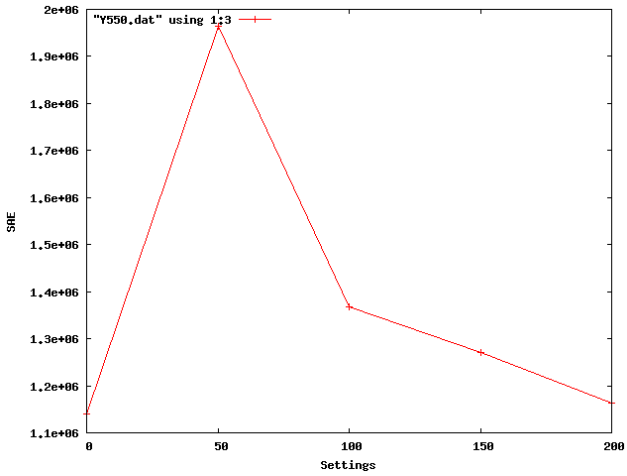
(a) D XAS-X



(b) D XKS-X



(c) D X-X



(d) D C-Y

Figure 5.11: PID Tuning of D, by joint, showing the relevant plane.

5.6.3 Walking - Multiple Joints

Once single joints were tested, the best settings found could be combined to determine whether multiple corrections would still improve the walk, or whether it would simply be too much correction. The first set of multiple joints was simply both joints in the sagittal plane. Combining the XKS and XAS joints let the S plane be tested against multiple joint corrections. This increased the difficulty of the corrections, as corrections now should still correct for the particular joint. However, the danger now posed was that both corrections together would cause not minor oscillations and overshoots, as with a single joint, but that both corrections would react to the corrections on the other joint, and recompensate for that movement.

Adding a second plane, with C, further increased this danger. Not only were the multiple joints potentially working against each other, but now corrections in one plane could negatively impact the other plane. The results from adding the C joint to the two S joints are given in this section.

T Controller

Figure 5.12 (a) displays the results of tuning the two S joints, in the S plane. These combinations were taken from the best of each of the XAS and XKS tests, and joined together. Approximately half of these combinations improve upon the walk: (tw-XAS850-XK550-d7, tw-XAS950-XK750-d6, tw-XAS950-XK750-d7, tw-XAS950-XK650-d6, tw-XAS850-XK750-d6, tw-XAS950-XK650-d7, tw-XAS850-XK750-d7). These combinations tend to be drawn from the higher XAS values, and also lean to the higher XKS values, leading to two conclusions. The first is that higher settings (so less in-

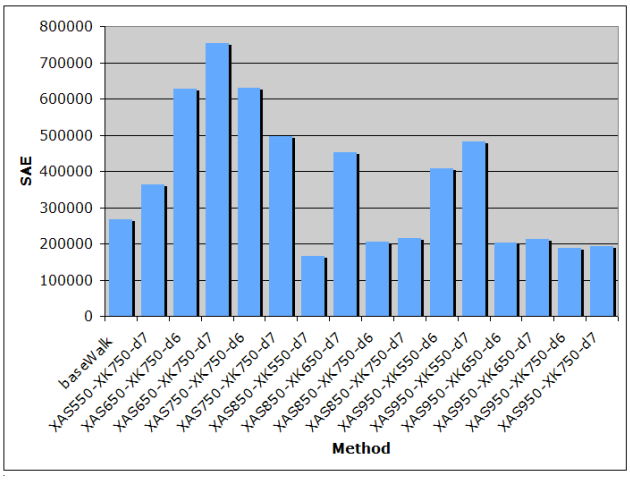
terference) is more useful as more corrections are undertaken. The second is that the XAS is potentially more disruptive, as it is almost uniformly best in combination only with the higher values. The XKS joint is making more corrective actions, but the XAS corrections are more influential in the walk.

The next step was to add the best C settings to the best S settings, seeing if further corrections would improve the walk. Figure 5.12 (b) and (c) show the results of adding C. In every possible case, the correction worsened the walk. In almost all cases, the higher C setting improved the corrections applied, suggesting that the less the C corrections were applied, the better the walk was. This applied for both the X and Y axes. Obviously, the interactions between the two planes were sufficiently bad as to overcome any possible benefits caused by the correction in either plane.

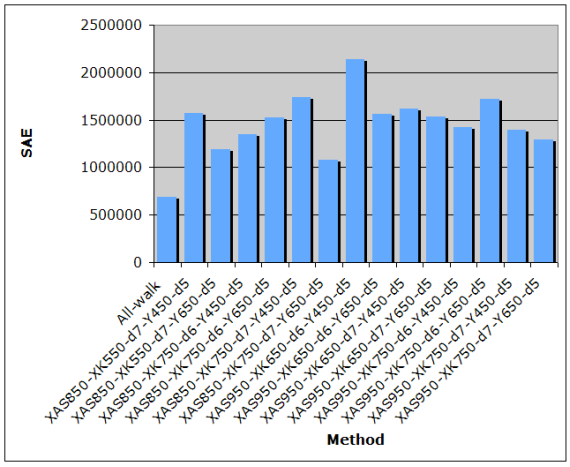
P controller

The P controller showed less workable combinations than did the T one. Only 3 combinations of settings (pw-XAS1250-XK850-d9, pw-XAS1250-XK950-d9, and pw-XAS1150-XK950-d9) were an improvement on the uncorrected walk. Once again, they were among the highest settings tested, further validating the conclusions above that increased joints being corrected require less corrections made to each joint.

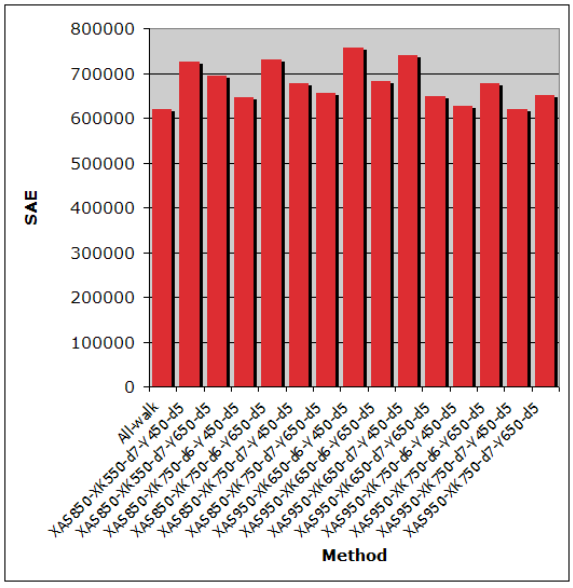
Adding in the C corrections only increased this tendency to overcorrection. None of the dual plane corrections improved on the uncorrected walk in the S plane, similar to the T corrections. Unlike the T algorithm, however, the P corrections did show an improvement on the C plane in most cases. An explanation for this phenomenon could be that the P algorithm, not being limited in its size of correction, is better



(a) X - X



(b) All - X



(c) All - Y

Figure 5.12: T tuning of multiple joints, in the S plane (XKS and XAS joints), and the SC planes (XKS, XAS and C joints). Trials are named by method, XKS and XAS joints with setting values, then S rates, C joint settings and C rates.

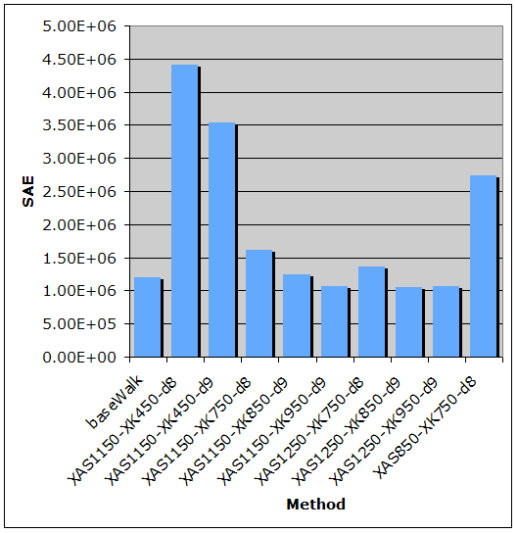
able to handle the disturbances made to the C plane with the C corrections than the minimal corrections allowed in the T algorithm. The S corrections were worsened however, as the P controller was not able to correct for both S and C in the S plane, having two joints to rebound off of each other instead of the single C joint.

Tuning D

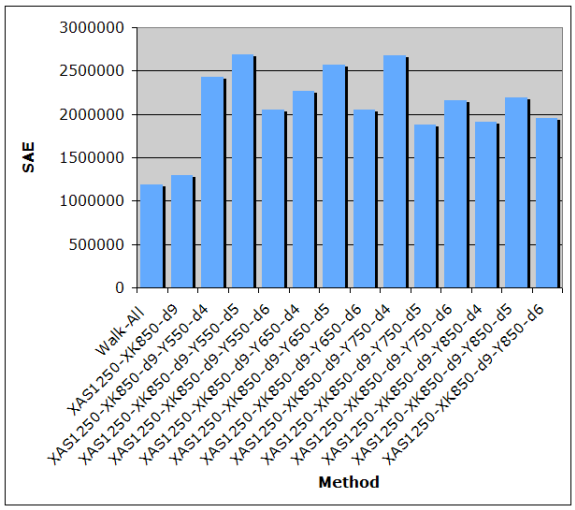
Further testing and tuning of D was carried out on a range of settings chosen from both XAS and XKS, to evaluate the settings in concert with multiple corrections occurring. The best P corrections for XAS and XKS (1250 and 850, with a rate of 9, respectively) were used in tandem, and then D was added in. The results (shown in Figure 5.14) indicate that no D value tested improved the walk as compared to the uncorrected walk or the S-corrected walk. The test results for pw-XAS1250-D450-XK850-d9 (D of 450 applied to the XAS joint) were very similar to the uncorrected walk, but not an improvement. Applying D to just one joint was, as might be expected from prior test results, better than applying D to both joints. Combining two of the better D settings and applying them both as corrections simply reduced the effectiveness of all the corrections. As D did not improve the tuning in any respect, it was not tested further.

5.7 Summary

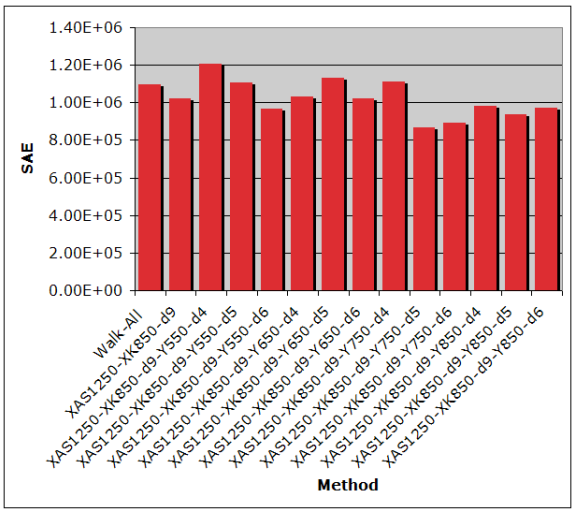
This chapter detailed the basis for and thought behind the algorithms used to control the balancing on Lillian, as well as the tests used to tune these algorithms. While this research is based an algorithm created for earlier robots in the lab, it goes



(a) X - X

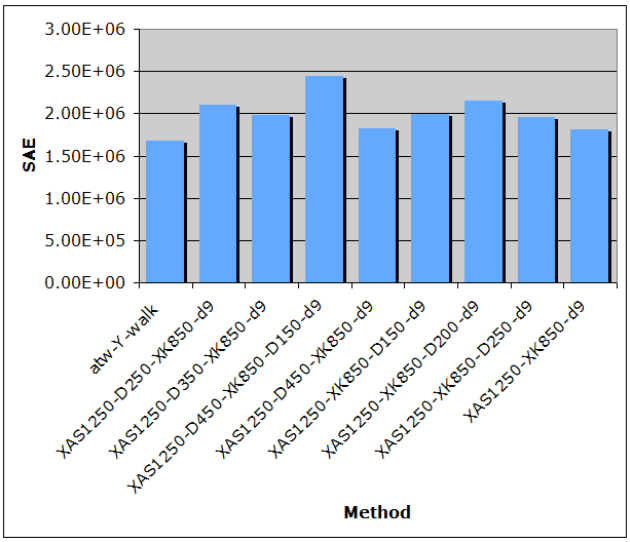


(b) All - X



(c) All - Y

Figure 5.13: P tuning of multiple joints, first S (XKS and XAS), and then the SC planes (XKS, XAS and C joints). Trials are named by method, XKS and XAS joints with setting values, then S rates, C joint settings and C rates.



(a) X - X

Figure 5.14: PD D tuning of multiple joints in the S plane (XKS and XAS). Trials are named by method, XKS and XAS joints with setting values, then S rates, C joint settings and C rates.

beyond the algorithm previously used. Three algorithms are examined in this thesis: PID, T and H. PID uses a percentage of the error to correct the robot's movement. T uses a deadband, then a flat correction after the deadband is left. H uses a deadband with a flat correction to a threshold point; after that is broken, a percentage of the error is used. Each of these methods must be independently tuned. The process used to tune them, ranging from simple one axis movements to complex multi-axis movements is described, and the results of the tuning are shown and analyzed.

While settings appear non-linear, the results of ranges tested range from oscillation to undercorrection. In general, control settings affected the walk more than the correction rates, but a waiting period between corrections was necessary to prevent oscillation. More settings improved on the tilting than the walking, while no improvement was noticed with corrections applied to multiple joints.

The next chapter will examine the strengths and weaknesses of the best tunings determined here.

Chapter 6

Evaluation

This chapter explains the tests used to evaluate the various balancing methods implemented and tuned on Lillian. Tests include randomly perturbing the walk to varying degrees, and running Lillian over a stepping field. The results of these tests are discussed and analyzed.

6.1 Test Framework

Tests were run at the University to compare the gaits of both static and dynamic control systems. I ran walking trials in the lab, with perturbed walks and walking on an uneven surface consisting of a specially constructed field (see Figure 6.5). Each trial was run three times, reducing as much as possible any outside influences.

Each method was tested using settings determined in Chapter 5. Initial trials directly compared the best of all possible settings (by individual joint, multiple joints, and P and PD controllers). Any setting that improved on the basic walk gait in the

plane it is correcting in was used in further tests. Those that did not will be left, as if they make a simple walk degrade, they would not improve on more complicated gaits or terrains.

6.1.1 Varied Gaits

Initial walking trials only required the robot to walk across an even surface for a standard period of time, approximately twenty seconds. The robot had to walk both with and without balancing algorithms for gaits of varying goodness. Gaits were adjusted by randomly varying the control points of one good gait over a spread of 5 or 10 set points, at multiple points throughout the gait. The disturbances were applied to all joints (hips, knees, and ankles) as the balancing control assumes that the movements of the joints are coupled. These gaits were then analyzed for stability.

6.1.2 Uneven Terrain

Uneven terrain was tested by using a stepping field, based on the one created for FIRA [2005]. Figure 6.5 shows a picture of the constructed field. The height difference between neighbouring areas of the stepping field is 3 mm — the thickness of an average piece of cardboard, and about 1% of Lillian's total height. The height difference between two neighbouring pieces is never more than 3 mm, but the height may change more than once over the length or width of Lillian's foot.

6.2 Experiments

As mentioned in Chapter 5, a battery of tests were used, including tilting and walking tests, as well as walking tests on uneven surfaces. Testing was independent for each correction method, threshold (T), P and PD, to begin with. Hybrid (H) tests were left for the later tests, as it was a more sophisticated method requiring both T and P or PD to be tuned previously. Testing also began with a single joint, in a single axis, progressively moving to multiple joints, and finally multiple joints in multiple planes. In the same manner, testing began with simple tilt tests, before moving on to correcting walks, then randomized walks, and finally walking on the stepping field.

6.2.1 Comparing All Results

The sheer number of tests carried out to tune the controllers on the robot make it somewhat difficult to determine what settings or sets of settings are actually beneficial. To cope with this difficulty, I took the best of each category of testing for all methods, P, PD and T, and directly compared them all against each other. While the methods have different methods of calculating the error, and hence SAE, it is useful here to analyze them both by the same method. As the thresholds were set equidistantly from the PID baseline, the SAE was calculated as the distance from the center on all trials. This allows for direct comparison of the different methods.

The settings chosen for the P and PD controllers were those individually best for each joint, the best combination of settings for just the X joints, the best setting of all the joints (X and Y), and the best P and PD controllers for Y. No PD settings for any of the X joints were chosen, as prior testing did not show them to improve the

Name	Method	XAS	XKS	X Delay	YAC	YAC Delay
P-All	P	1250	850	9	750	5
P-X	P	1250	850	9		
P-XKS	P		450	8		
P-XAS-1	P	850		8		
P-XAS-2*	P	1250		8		
P-YAC	P				550	5
P-YD	PD			550	5 (D 150)	
T-All-1	T	950	750	7	450	5
T-All-2	T	850	750	7	650	5
T-X	T	850	550	7		
T-XKS	T		750	6		
T-XAS	T	950		6		
T-XKS-YAC	T		750	6	450	5
T-XAS-YAC	T	950		6	450	5
T-YAC	T				450	5
Hy-All-1	H: T	950	750	7	450	5
	H: P	1250	850	9	750	5
Hy-All-2	H: T	850	750	7	650	5
	H: P	1250	850	9	750	5
Hy-XAS-1	H: T	950		6		
	H: P	850		8		
Hy-XAS-2*	H: T	950		6		
	H: P	1250		8		
Hy-YAC*	H: T				450	5
	H: P				550	5

Table 6.1: Configurations for side-by-side comparisons of the best walking tests results. Top results from each set of tuning configurations are shown for PID and T methods, and several H tuning combinations are included. Configurations consist of the two XAS and XKS settings, as well as the X rate, the YAC setting and the YAC rate.

P controllers for single joints of X. As the best settings for any single X joint were not found to improve on the basic walk, no further combinations of settings involving them were chosen.

T settings to test were chosen similarly to those for the P and PD controllers. The best setting for each joint individually was chosen, and the best of the combinations of X and XY joints. Two XY settings were chosen, as one improved on the basic walk for Y, while the other was the best set of corrections for X. Two additional settings were chosen for testing as well: XAS + YAC and XKS + YAC. This decision was made because one of the XY settings was the same as the three best individual joint settings. Further, each of the single joint settings (unlike the P and PD controllers) actually improved on the basic walk, causing speculation that a single X joint setting with a YAC setting may prove an effective balancing setting.

The H controller is dependent on the combination of the best of the T and P or PD controllers. Hence, it was not tested up until this point in time, when the best settings for the simpler controllers have already been determined. The H controller is now tested using any combination of settings that actually improve on the basic walk, for both basic controllers, and the combination of the XY controllers, for comparison's sake. Thus, the H controller is tested on the top two T XY controllers, combined with the top P XY controller, the combination of the top XAS settings for both P and T, and the combination of the YAC controllers.

Several of the control settings have an asterisk in their name: this marks tests run at a later time. An incorrect result skewed which settings were chosen to be tested, and so several test groups were ignored which should have been tested. These were

retested later, along with another base walk reading for comparison, and are marked in all graphs with an asterisk.

Figure 6.1 shows the results of directly comparing all the chosen methods. Less than half actually improve upon the uncorrected walk, in the plane(s) that they are correcting for. This criteria is used to select settings for the final evaluations. The settings that improve the X SAE and actually correct for X are P-XAS-1, P-XAS-2, Hy-XAS-1, Hy-XAS-2, T-XAS and T-XKS. Many settings improve the YAC SAE, but if they also correct for X and did not improve the X SAE, they were not chosen for further testing. Thus, the only methods chosen for YAC were T-YAC, P-YAC, and Hy-YAC: the simplest correction methods of all those involving YAC. These settings are listed in greater detail in Table 6.2.

Name	Method	XAS	XKS	X Delay	YAC	YAC Delay
P-XAS	P	850		8		
P-XAS-2*	P	1250		8		
P-YAC	P				550	5
T-XKS	T		750	6		
T-XAS	T	950		6		
T-YAC	T				450	5
Hy-XAS-1	H: T	950		6		
	H: P	850		8		
Hy-XAS-2*	H: T	950		6		
	H: P	1250		8		
Hy-YAC*	H: T				450	5
	H: P				550	5

Table 6.2: Configurations for the random walking tests. These configurations were chosen from the best results of the previous tests.

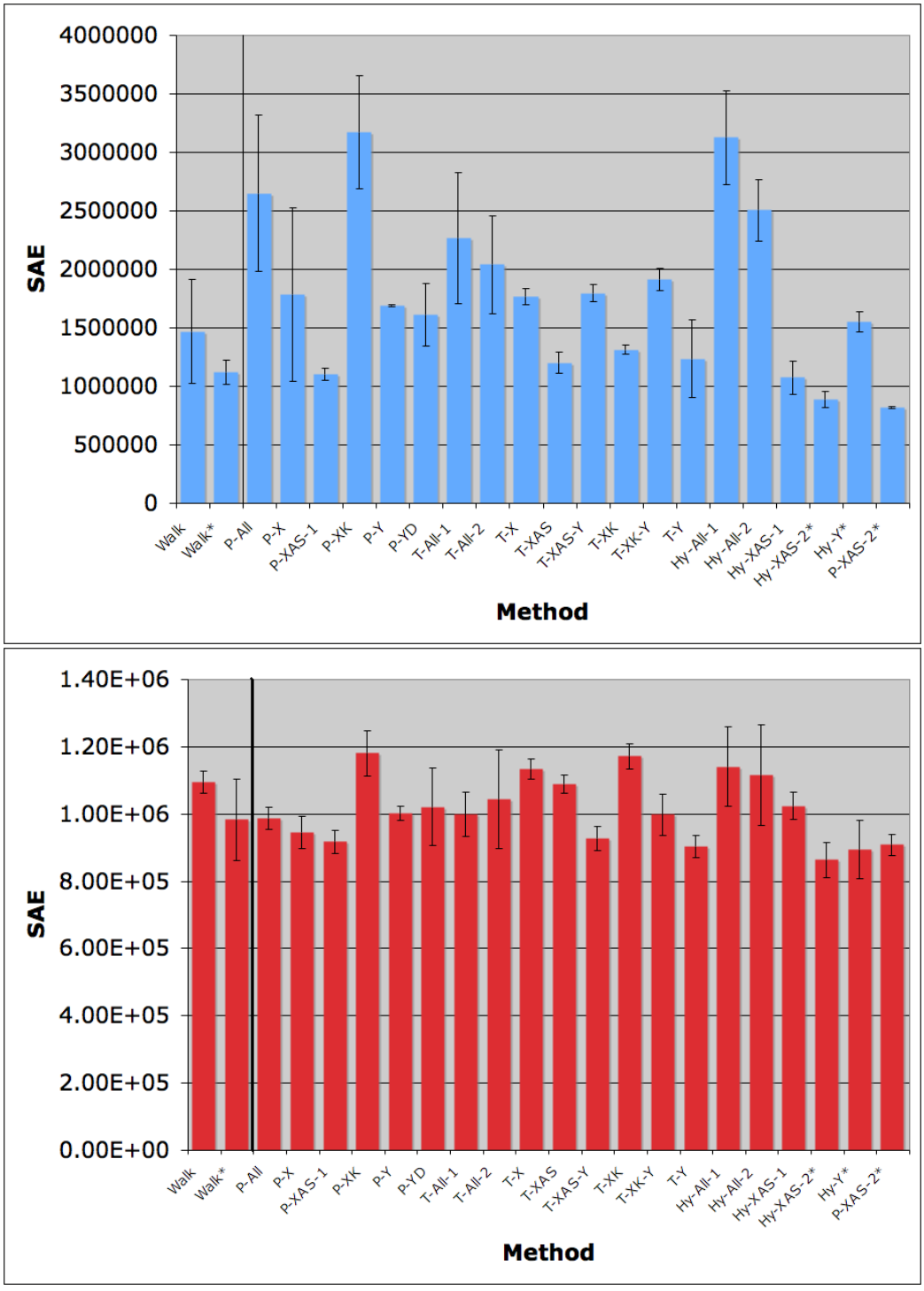
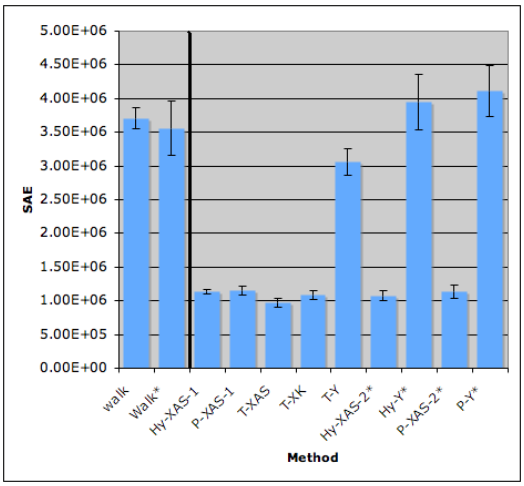


Figure 6.1: Random walking test results, by method and settings.

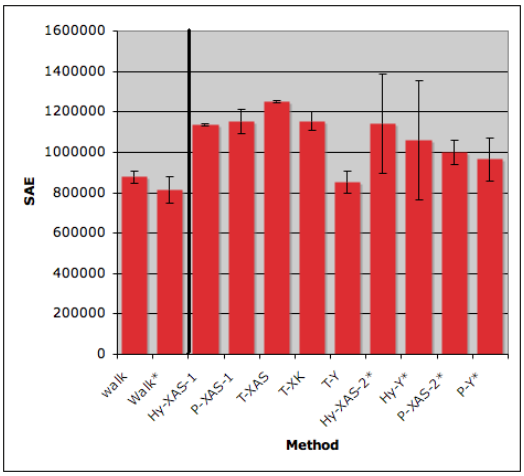
6.2.2 Random Walks

The random walks were created by perturbing Lillian's walk patterns by a random value. These random values were uniformly generated in a range of either [-2 ..2] (a spread of 5) or [-5..5] (a spread of 10). The values generated were then added to the walk motions, meaning that the randomness is applied to the walk approximately once per second. The walk pattern initializes to a basic stance, and all changes (i.e., through the course of a gait) thereafter are simply applied as adjustments to the current position. Since Lillian never returns to the original position, adding randomness to the walk means that the longer the robot walks, the worse the walk becomes, as errors compound upon each other. This makes the balancing much more difficult than if errors were reset, but also more realistic, and especially when considering the effect a ramp would have on the robot's stance. Further, the initial position was also slightly perturbed (up to 5% for each joint position) to throw the walk further off balance. All random values were pre-generated, as there is no random number generator on the Eyebot. This allowed for the randomness to remain constant across each set of tests. The robot was run on a flat surface, with the perturbed walk parameters, three times for each set of control method parameters.

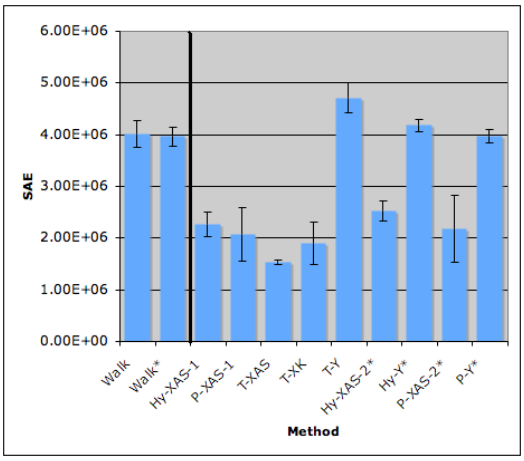
As can be seen in the graphs in Figure 6.2, any correction method makes an incredible difference in the perturbation of the walk. Figure 6.3 shows an example of the end state of the robot in this test for both an uncorrected and a corrected run. This is best noticed in the lesser perturbations of the Random Walk 5 graphs (Figure 6.2 (a) and (b)). The SAE of the methods correcting in the X plane is a third to a quarter of the uncorrected methods. The correction only in the Y plane is not



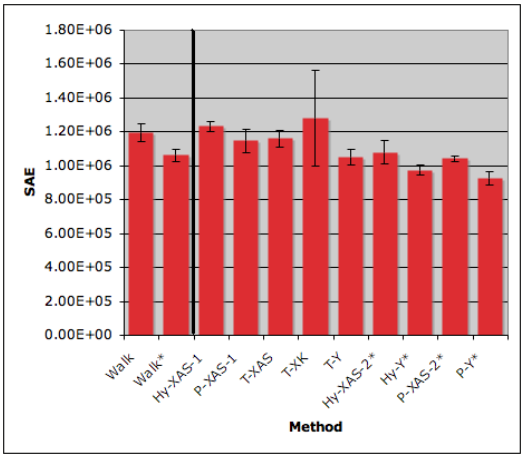
(a) Random Walk 5: X



(b) Random Walk 5: Y



(c) Random Walk 10: X



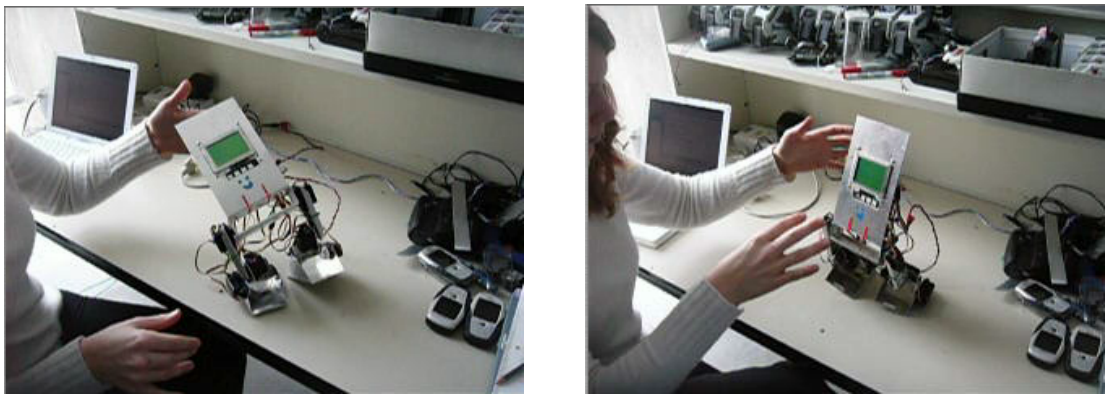
(d) Random Walk 10: Y

Figure 6.2: Random walking test results, by method and settings.

(a) Uncorrected (b) H Corrected

Figure 6.3: 5 Perturbation random walking test pictures, showing the robot near the end of the test run for both an uncorrected and a H corrected run.

as impressive; YAC corrections simply allow for the walk to remain approximately as good as it was without corrections, though the corrections solely in the X plane do worsen the YAC SAE values. The differences between the correction methods in this test, while they exist, are much less than simply the difference between the corrected and uncorrected tests. T has two of the three lowest SAEs in the X plane, with the H method having the other 2 of the 4 lowest SAEs. The P methods follow, still improving on the uncorrected walk, and finally the various YAC correction methods. All three correction methods were tested on the XAS joint, and T here clearly shows itself to be the superior method. For T itself, the best joint control comes from the T method controlling the XAS joint; the XKS joint is not as effective. T control on the YAC joint is not extremely effective; it improves the walk slightly in both planes, but not as much as the X control. This is partly due to most of the imbalances created by the perturbations arising in the X plane; the YAC plane, having only one joint to



(a) Uncorrected (b) T Corrected

Figure 6.4: 10 Perturbation random walking test pictures, showing the robot near the end of the test run for both an uncorrected and a T corrected run.

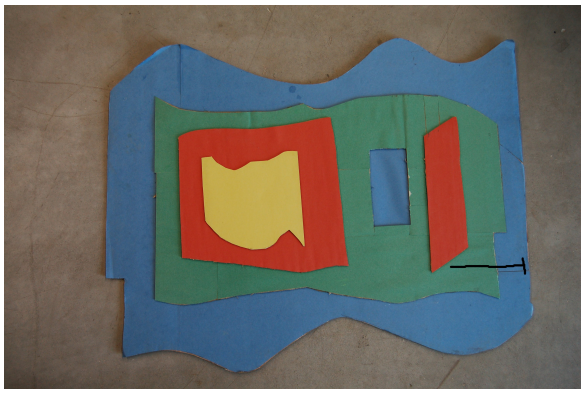
control, is naturally more resistant to disturbances in the walk pattern. H correction in this case appears to bridge the gap between T and P: it's not as effective as T, but more effective than P.

Increasing the perturbations to up to a spread of 10 caused the differences between the methods to become yet more pronounced. Figure 6.2 (c) and (d) show the results of testing with the greater perturbations, while Figure 6.4 shows the end results of Lillian for an uncorrected and a T corrected run of the robot. Once again, the T results are the best, but with a greater difference between them and the other methods. Any correction method still shows a marked improvement over the uncorrected walk, but the errors overall are larger than they were for the spread of 5, indicating that the correction methods are not as well able to cope with the error caused by the larger perturbations. The P method is the next best, with about half the error in the uncorrected walk. The H method is the worst (with slightly over half the error in the

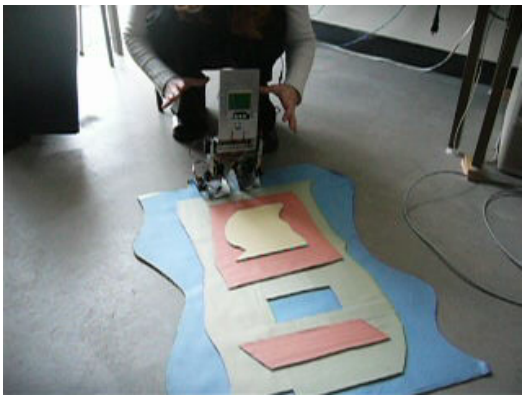
uncorrected walk). The T method reduces the error to a bit over a third of the original error, when applied the XAS joint as with the other two methods. For the T method as applied to the XKS joint, it is still an large improvement on the uncorrected walk, but is only slightly better than the other two correction methods on the XAS joint. The YAC corrections show more of an improvement on the uncorrected walk than with the lesser perturbations, but still do not make as much of a difference to the walk as do any of the X corrections. In fact, the YAC corrections manage to degrade the performance of the walk in the X plane, while the X corrections affect the YAC plane much less, and in some cases (T and P XAS) actually slightly improve the YAC component of the walk. Unlike in the X plane, the P method is the best, followed by the H method, and finally the T method. The possible reasons for this will be explored in section 6.3.

6.2.3 Stepping Field

The final test of the robot's balancing abilities was to run it on a stepping field, with highly uneven terrain. A stepping field was constructed, and the robot run across it. The height difference between neighbouring areas of the stepping field is 3 mm — the thickness of an average piece of cardboard, and about 1% of Lillian's total height. The height difference between two neighbouring pieces is never more than 3 mm, but the height may change more than once over the length or width of Lillian's foot. The starting location of the robot, as well as the most usual path walked by the robot is marked in Figure 6.5. Due to the balancing and the height differences in the terrain, the path could veer to either side over a trial.

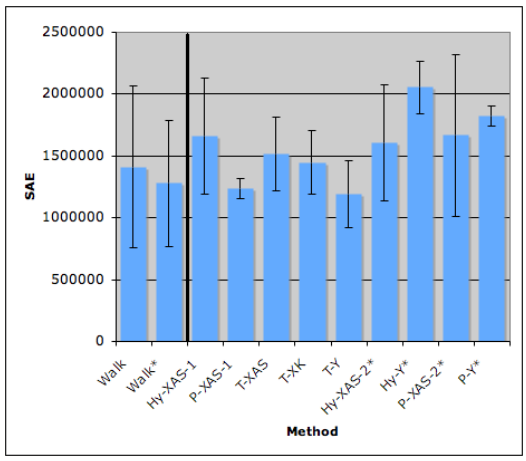


(a) Stepping Field

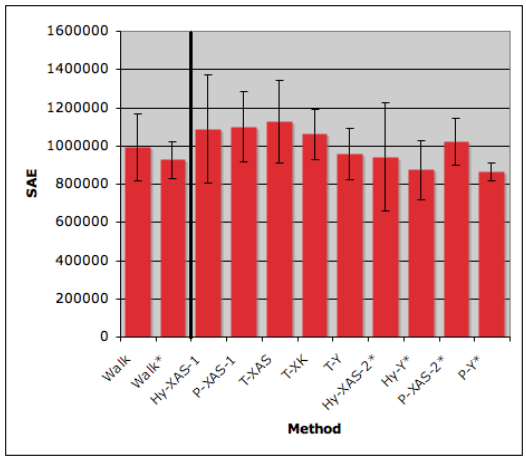


(b) Uncorrected Run

Figure 6.5: Uneven terrain: The starting location of the robot is marked by a vertical bar, and the general path of the robot is marked by a horizontal black line. A picture is shown of the robot traversing the terrain, with an uncorrected gait.



(a) Stepping Field: X



(b) Stepping Field: Y

Figure 6.6: Stepping field test results, by method and settings.

Results from the correction methods (see Figure 6.6) show that the balancing begins to break down with this test. For the methods based on the X plane, only the P corrections improve on the uncorrected walk. While the H methods actually worsen the SAE readings, the T corrections merely leave the walk mostly unchanged. Unlike previous tests, the T corrections on the XKS joints are actually slightly better than those of the XAS joint. Compared to the differences between the corrected and uncorrected walks in the previous perturbation test, however, all the differences between the results are relatively minor, indicating the X corrections are not making a large difference to the walk. As was expected, the lack of corrections to the Y plane by these methods is clearly shown, as the Y readings for all the X correction methods are worse than the uncorrected YAC walk. Unexpectedly, the T corrections for YAC improve on the uncorrected walk for both X and YAC, not just YAC. This sheds some light on the lack of improvement for the X corrections. Unlike the randomized perturbations test, where the main corrections to be made were to the X plane, here the most effective corrections are to the Y plane, and correcting in X without correcting YAC will not be very effective. This will be further commented upon in Section 6.3.

6.2.4 Further Results

One test that would be difficult to plan for is fatigue in the servo motors. Over time, servo motors will wear out as they provide actuation to the joints. This fatigue is not noticeable at first, but creeps slowly into the walk with initially minor errors, with the servos unable to reach or hold a joint position as requested. With enough use, the

fatigue becomes clearly visible in the walk, and the fatigued servo motor is identified and replaced. With a static walk, this fatigue appears as slight lurches or sways in the walk, worsening until the robot may fall while performing a gait that previously worked (though rarely does the fatigue continue this long without being noticed and fixed). With the correction algorithms turned on, it became almost impossible to detect a fatiguing motor on the robot until the motor was almost useless — i.e., to the point of causing falls in uncorrected walks. To determine if a joint was fatiguing, it was necessary to turn off the corrections, and move back to an uncorrected walk. The correction methods would compensate for the joint fatigue, and hide it from the researcher's eye. This did not happen only once, but many times. Approximately 25 servo motors were replaced over the course of this research, with balancing testing lasting approximately 10 months, yielding a servo failure rate of about one every two weeks. As the balancing reflexes became better tuned, it took progressively longer to notice servo fatigue, as indicated by the increasingly poor condition of the servos once fatigue was detected.

6.3 Analysis

A repeated measures analysis of variance (ANOVA) test was run on the final results, to ensure that variance within the trials was not invalidating the results. The ANOVA test indicated that there were no significant differences between the trial runs, and therefore, the results are not being invalidated by variance from the robot or the starting position.

The effort to directly compare all results with each other led to some interesting

observations. The more joints a controller attempted to control, the worse the balancing became. Similarly, more complicated controllers, such as moving from the P to the PD controller did not tend to improve the balancing. The simplest ideas, such as the T controller, were the most effective in controlling the walk, and maintaining the desired accelerometer readings. The simpler T controller also responded better (not as badly, that is) to adding more joints to the control algorithm, not overreacting as much as the P controller did when similarly controlling multiple joints.

This conclusion of complexity not being handled well was further born out by the random walks. Different joints were found to be more effective in controlling the balancing. The XAS joint is much more effective at controlling the robot's balance than any of the other tested joints. Further, as the random perturbations increase, the T controller shows more and more effectiveness over the P controller. The H controller is still better than no controller, but not as good as either of the simpler controllers.

The stepping field, surprisingly, upsets the previous trends. The P controller actually outperforms the T controller in this case. I believe that is because the T controller is unable to react quickly and strongly enough to the changes needed for the terrain changes. The random walks had a change of perhaps 5 or 10 servo settings over 1 second; the T controller allows approximately 10-12 servo setting changes per second. Thus the T controller could compensate for the changes in the walk, while the P controller tended to overreact. Once on the stepping field however, the changes would occur instantly, and require compensation of 5-10 servo settings (or perhaps more). The P controller could react instantly, while the T control would move more

slowly to correct. This slower reaction time would allow the robot enough time to create more inertia in the wrong direction, making it even more difficult for the robot to compensate for the leans.

A further explanation of the poorer results on the stepping field is due to the initial assumption that all joint actions are coupled. This is generally true of the testing, as with the randomized perturbations and the tilting, the robot's feet remain relatively aligned with each other. The stepping field, however, due to the unevenness of the terrain, allows for the feet to become misaligned, and this is not easily corrected by the balancing algorithms, as currently implemented.

6.4 Summary

This chapter has described and analyzed the tests that were used to test the three balancing algorithms described in the previous chapters. The best tunings from Chapter 5 were tested head to head against each other, and those that improved on the uncorrected walk were used for the final testing. These tests explored the results of the balancing algorithms (P, T and H) when correcting for random perturbations to the walk, and then correcting for the uneven terrain of a stepping field. Analysis of the results showed that overall the T algorithms are the best, but that P can be effective at times. In general, however, these algorithms are best for simpler corrections, and do not scale well to the complexities of controlling multiple joints or uneven terrain. My conclusions from this research will be presented in the next and final chapter of this thesis.

Chapter 7

Conclusion

The research undertaken for this thesis leads to some conclusions about the feasibility of only using an accelerometer, as well as the feasibility of the algorithms tested. While it is possible to use just an accelerometer and a simple control algorithm to balance a robot, this balancing is most effective in simpler domains (eg, a single axis tilting platform). The T algorithm's minimal corrections are more effective with simpler domains and multiple joints, while P starts improving on single joint control over differing terrain. This work leaves researchers with a thorough look at the basics of using only an accelerometer for balancing, and how much can be accomplished with just that sensor. It also provides a logical methodology to use in testing and evaluating balancing on humanoid robots. Future work could extend these tests by modifying the robot physically or by increasing the research into balancing with multiple axis and joints.

7.1 Answers to Research Questions

In Chapter 1, I asked the following research questions:

1. Can a balancing reflex that is a tightly coupled feedback loop implemented on a single motion sensor dynamically balance a small humanoid robot in real-time? If so, can the robot measurably improve its gait with this?

2. What balancing reflex should be used to dynamically balance a robot? What are the strengths and weaknesses of the various algorithms? How does the algorithm previously developed for the gyroscope compare to a standard control algorithm?

Yes, a motion based sensor with a tightly coupled feedback loop can provide enough information for a humanoid robot to actively balance, and to measurably improve its gait with this information. The sensory information provided by the accelerometer was enough to stop the robot from falling over when standing on a tilting surface, or to keep the robot upright when the walk was randomly perturbed. These tests prove that a balancing reflex does provide enough information for the robot to remain upright when it would otherwise have fallen over, thus actively balancing.

This thesis showed that simple algorithms can be used to balance a robot in simple situations, but that they become unable to handle the complexities of more complex situations (e.g., walking while controlling multiple joints). Both the T and PID algorithms showed impressive results on the tilting tests, with a broad range of settings providing beneficial corrections to the robot. Moving those corrections to a walk demonstrated the shortcomings of the algorithms, as the speed and complexity

of balancing a walking robot started to overcome the balancing capabilities of the controllers. Adding multiple joints to be controlled, or moving the robot to an uneven surface further demonstrated the inability of the balancing algorithms to compensate for the amount of variability in the walk. While the algorithms were not able to completely compensate for any surface, they did improve the walk noticeably against smaller, more regular changes, as shown by the random perturbation tests.

Part of the difficulty here is the inherent tendency of the robot to oscillate. In the sagittal plane especially, there are multiple joints controlled by servo motors, and the motors themselves may be overcome by the torque on the joint, rendering them ineffective for balancing. Further, these extra pivot points along the robot provide points of weakness where oscillation may develop, due to the sway inherently made possible by a joint. This means that corrections must be carefully made so as not to overshoot target values, which would further destabilize the robot. This limitation proved difficult for the control algorithms to overcome in more complex balancing situations.

To answer the second research question and decide what algorithm should be used to dynamically balance a robot with a motion based sensor, the strengths and weaknesses of the various algorithms should be discussed. No one algorithm was consistently best; rather, the most effective algorithm depended on the circumstances the robot was used in. Overall, the T method was best for slower, steadier changes, while the P controller responded better to occasional larger changes. The Hybrid method was never the best method, but was almost always in between the two other algorithms in terms of goodness; never the best, but regularly the runner-up.

In Chapter 4, I mentioned that XAS corrections have a greater influence on the gait than do XKS corrections. As XAS is also the only X joint corrected by all three methods, I'd like to take a closer look at it to examine side-by-side the results of the different methods on the sagittal plane. I will use the YAC joint to do the same for the coronal plane. Tables 7.1 and 7.2 shows the rankings of the various methods on each joint.

XAS			
Everything	Random Walk 5	Random Walk 10	Stepping Field
P	T	T	P
H	H	P	T
T	P	H	H

Table 7.1: Side-by-side rankings of the balancing algorithms on the XAS joint.

Table 7.1 shows that the Threshold algorithm is better on the random walks than P, but P is better on the walk and stepping field. H is never the best algorithm, but is usually in between the other two, indicating it is a more general algorithm. Threshold works better when the changes are slower, as with the random walks, as the minimal corrections it enforces can then compensate for the drift caused by the random perturbations. P, on the other hand, does not work as well with the drift, but reacts better to the larger changes required in the stepping field, when the robot needs to react more quickly to stepping on uneven terrain.

Table 7.2 shows the results of controlling the YAC joint with the three algorithms. Here, again, the best results are the T and P, but Tm is best when walking, and the smaller random perturbations, while P is best on the stepping field and the larger random perturbations. My hypothesis for the improved performance in P on

YAC			
Everything	Random Walk 5	Random Walk 10	Stepping Field
T	T	P	P
H	P	H	H
P	H	T	T

Table 7.2: Side-by-side rankings of the balancing algorithms on the YAC joint.

the Random Walk 10 is that the YAC joint is less sensitive to overshoot, and so a performance gain is noticed as opposed to on the X joints, where the overshoot is more likely to adversely affect the walk.

T gave the best results in the Stepping Field and Random Walk tests, in the X plane. Because of its minimal corrections, this method is best able to correct where overshoot is a problem, as is the case with Lillian in the sagittal plane. Thresholds also have the advantage of being an extremely simple method to tune. The threshold boundaries are constructed from a known good walk, and then an error threshold is set for corrections to begin at. Because a correction is made only when the error grows large enough to break the second threshold, it has a greater margin for error when deciding on where to put the second threshold, as compared to P.

P starts coming into its own on the more difficult tests. T cannot always compensate as quickly as P can for error, and so P results begin to improve on those of T. However, without the larger errors to compensate for, P appears to have a tendency to overcorrection, as is noticeable in the lower level tests conducted in Chapter 5, where many T values appear effective, but only a few P. Once a decent set of P tunings is found though, it may be slightly more effective on a broader scale than T.

The H method is never as good as either the T or P, but it generally gives a result in between the other two methods, indicating it does hybridize those methods. It is possible that this method successfully combined the best parts of both methods, but that because the methods appear to do better in different situations, it also inherited the less useful characteristics of the parent methods, meaning it is never quite as good as either of them — just potentially more robust. However, it is the longest to tune of any of the methods, which should be taken into account when deciding upon a balancing algorithm.

Overall, then, due to its ease of tuning and generally decent performance, I believe that T is the easiest and most useful choice for future balancing. However, while there are differences between the methods, for a single joint they all do improve upon the uncorrected walk, and thus any of these methods is better than none.

7.2 Implications

Creating a robot better able to balance over varying terrains with different speeds and gaits will enable today's robots to become closer to the robots we will use to aid us in the future. While other researchers are implementing balancing in their robots, the University of Manitoba — and this research in particular — is the first to attempt this with only gyroscopes or accelerometers. Usually, more advanced sensors are used. Hopefully, these cheaper devices will lead to a more robust and general solution that has been previously developed.

This research provides a jumping off point for others interested in creating robots that can walk where we can. Humanoids that cannot balance are confined to a more

academic circle of interest, as are high priced robots. Using fewer sensors and testing the limitations of the information such devices provide will allow others to know how much to expect of such a device. It will also make it easier for researchers to develop new code. Finally, with better balancing measures, walking gaits will be more robust than previously. This is a necessary step in allowing further research into what humanoids are capable of.

In order to determine how to do this, we must test out simple sensors, and it is best to begin with simple methods. This research provides an initial foundation for work looking into balancing with accelerometers, as it shows that it is possible to balance with only an accelerometer, and a simple control method. However, it also shows that these simple methods only work for reasonably simple balancing. More complex adjustments are not implemented well with these methods. Terrains such as the stepping field will require a more complicated (or at least more effective) means of balancing than either a PID or a threshold controller.

7.3 Future Work

There are several areas that can still be explored to learn more about the uses of accelerometers in balancing, and the algorithms used in this research.

An initial assumption was that all joint movement would be correlated — that is, that the ground would remain flat under the robot. This is generally the case in this research, especially with the tilting platform and the randomized perturbations, but is not always the case in more realistic world scenarios, such as is provided with the stepping field. Future work should investigate the possibilities and difficulties

involved with uncoupling the joint corrections, and allowing each foot or leg to be corrected separately.

A strength of the Threshold method that has been little used in this research is the ability to loosely or tightly bind the corrections in a particular timestep. Unlike PID control, which assumes that any deviation from the ideal, T allows the ideal to encompass a larger area, and not correct as quickly in certain areas, while correcting much more quickly in others. Determining which areas should be tightly bound and which ones more loosely bound is the the difficult part. The main research into the characteristics of a good walk deals with controlling the ZMP, but there is room here to investigate what data beyond the ZMP can be contributed by sensors. Adapting the threshold bounds has been used in prior work [McGrath et al., 2004b,a], but have not been investigated in this thesis. These bounds may improve the corrections made by T and improve its performance in the stepping field, at the least.

This research has looked at balancing reflexes as the sole means of correction. Now that their usefulness on their own has been established, they could be integrated into larger motion planning based methods, allowing greater usefulness over a larger variety of terrains. This would allow the robot to reflexively balance for smaller disturbances, but make larger adjustments as needed with a gait planning algorithm for larger scale disturbances.

Physical modifications to the robot could be made to test the robot's balancing abilities under more difficult circumstances. As mentioned in Chapter 3, a camera had originally been mounted on Lillian's head, but had to be removed due to the excess sway it added to the robot. A design issue inherent in building any humanoid

robot is the torque provided by the servos versus the momentum created by the movement of the robot, limiting the motions that can be made by the robot. Adding the camera back on Lillian would immediately present added challenges for balancing by further constraining the amount of torque than can be present in any motion, and therefore the set of possible motions, providing further means of testing the balancing algorithms.

A similar line of research would be to move the balancing algorithms to a more stable robot, such as a kit robot. This would remove the low cost platform used to create more robust algorithms, but the very stability of the new platform might provide a more precise comparison of the algorithms used. It is possible that they would still appear relatively similar, but moving to a kit robot would enable this to be determined with greater clarity. However, this would remove the ability to test the robustness of the algorithms in general, as a kit robot would have less inherent instability.

7.4 Summary

This thesis has investigated the utility of balancing reflexes implemented on an accelerometer for active balancing on a small low cost humanoid robot, as well as the advantages and disadvantages of several different algorithms used for a balancing reflex. While the simpler algorithms tested worked effectively for simple tasks, they became less effective as the complexity of the interactions being controlled increased. This research provides a basis for future research into accelerometers, single sensor balancing and balancing algorithms, as well as a series of tests that can be used

to test the effectiveness of a balancing algorithm. Future work on the effects of manipulating threshold boundaries, uncoupling the relationships assumed between the joints, physical modifications to the robot, and the effectiveness of more complex control algorithms should be the next steps to expand upon the research already presented here.

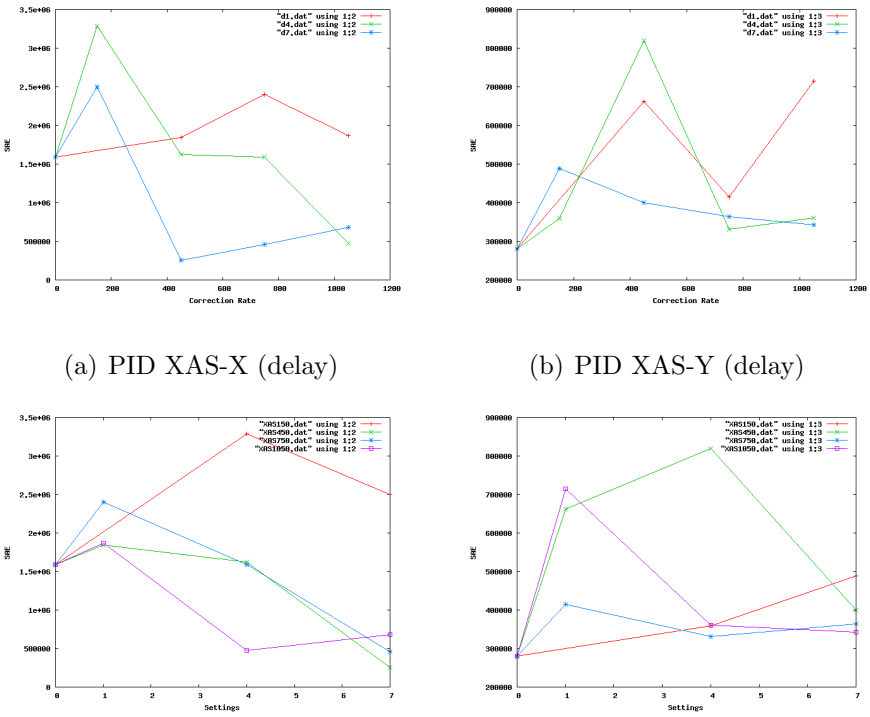
Chapter 8

Extra Data

X Reading	ZeroPoint	Min	Max	Variance
1	18615	18338	19193	268
2	18774	18555	19238	249
3	18787	18561	19537	237
4	18703	18574	18965	45
5	18890	18508	19697	310
6	18911	18715	19465	238
7	18960	18388	19779	312
8	18693	18619	18763	31
9	18969	18741	19689	283
10	18860	18797	18904	18

Y Reading	ZeroPoint	Min	Max	Variance
1	18260	17947	18840	264
2	18354	18155	18895	241
3	18382	18178	18192	235
4	18249	18139	18525	25
5	18493	18270	19255	301
6	18460	18251	19005	232
7	18534	18264	19328	298
8	18290	18196	18397	20
9	18590	18370	19292	276
10	18433	18372	18495	15

8.1 Full Tilt Test Data



(a) PID XAS-X (delay)

(b) PID XAS-Y (delay)

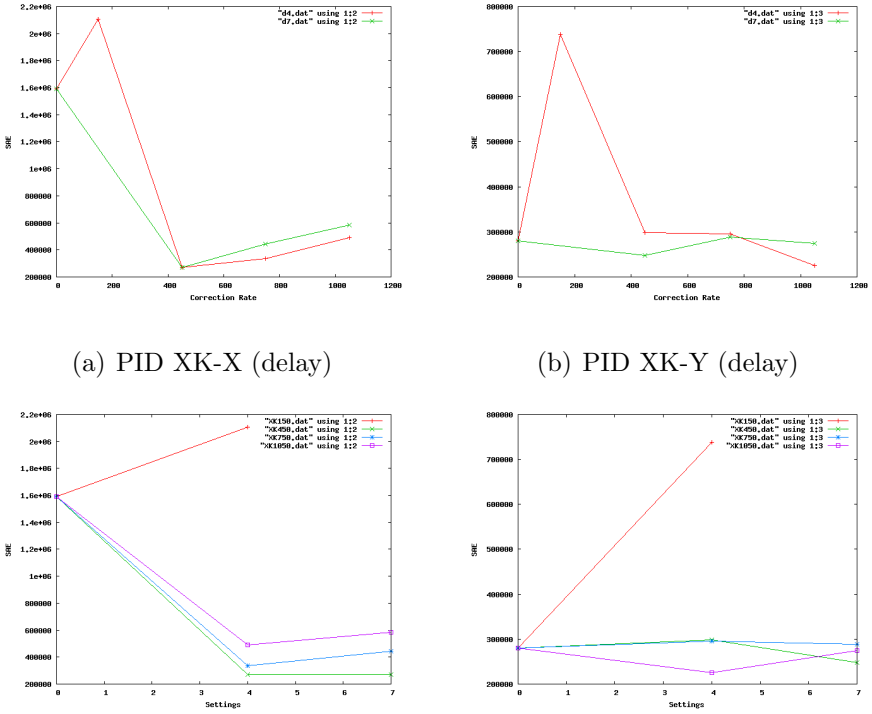
(c) PID XAS-X (settings)

(d) PID XAS-Y (settings)

Figure 8.1: PID tests on XAS.

8.2 Full Walk Test Data

8.3 Testing D



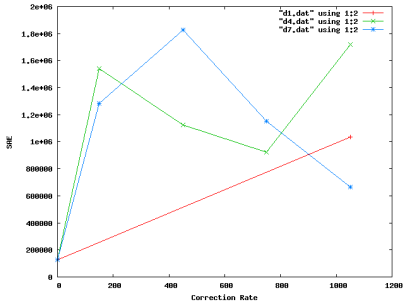
(a) PID XK-X (delay)

(b) PID XK-Y (delay)

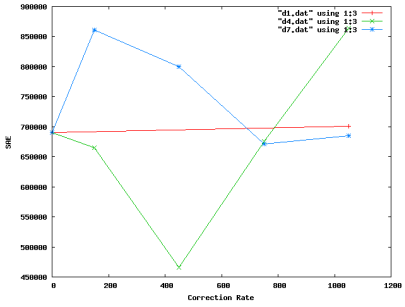
(c) PID XK-X (settings)

(d) PID XK-Y (settings)

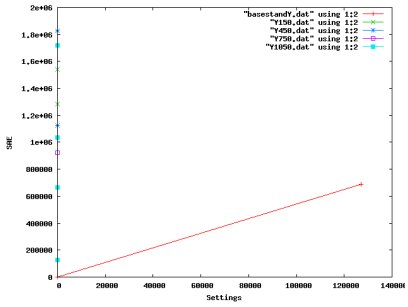
Figure 8.2: PID tests on XK.



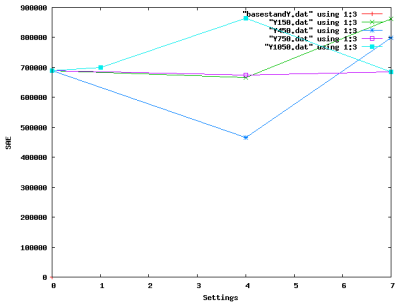
(a) PID Y-X (delay)



(b) PID Y-Y (delay)

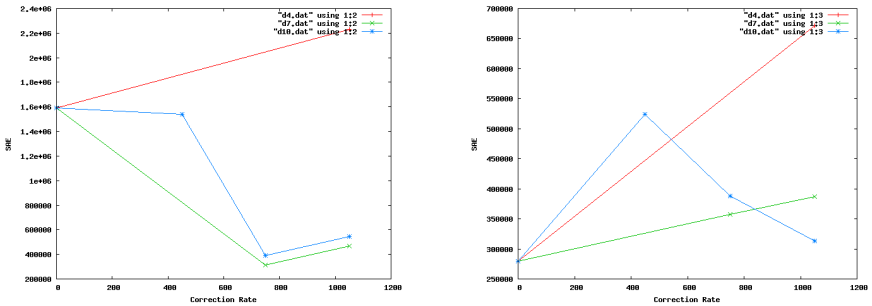


(c) PID Y-X (settings)



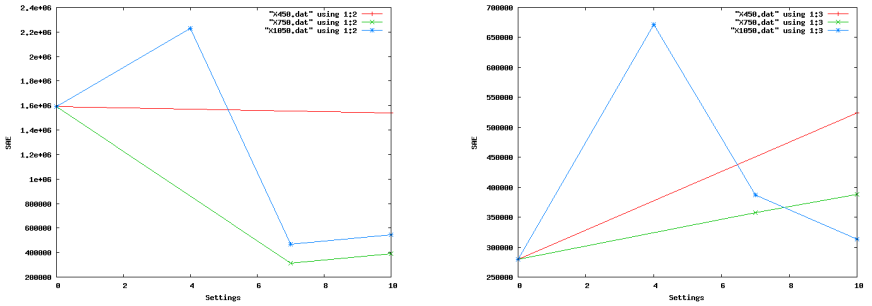
(d) PID Y-Y (settings)

Figure 8.3: PID tests on Y.



(a) PID X-X (delay)

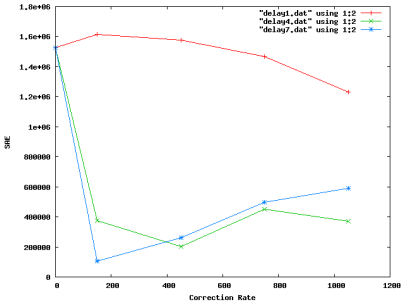
(b) PID X-Y (delay)



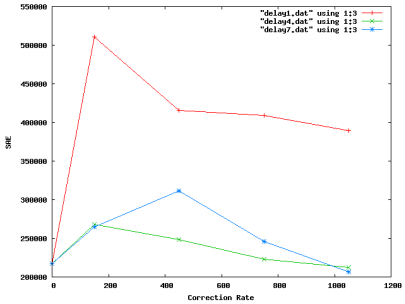
(c) PID X-X (settings)

(d) PID X-Y (settings)

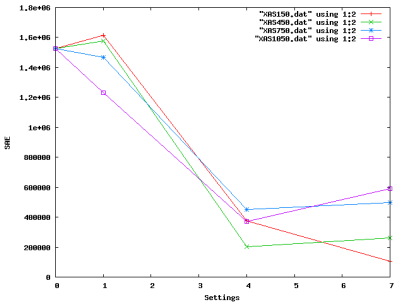
Figure 8.4: PID tests on X.



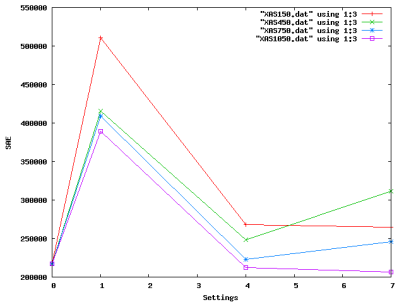
(a) Threshold XAS-X (delay)



(b) Threshold XAS-Y (delay)

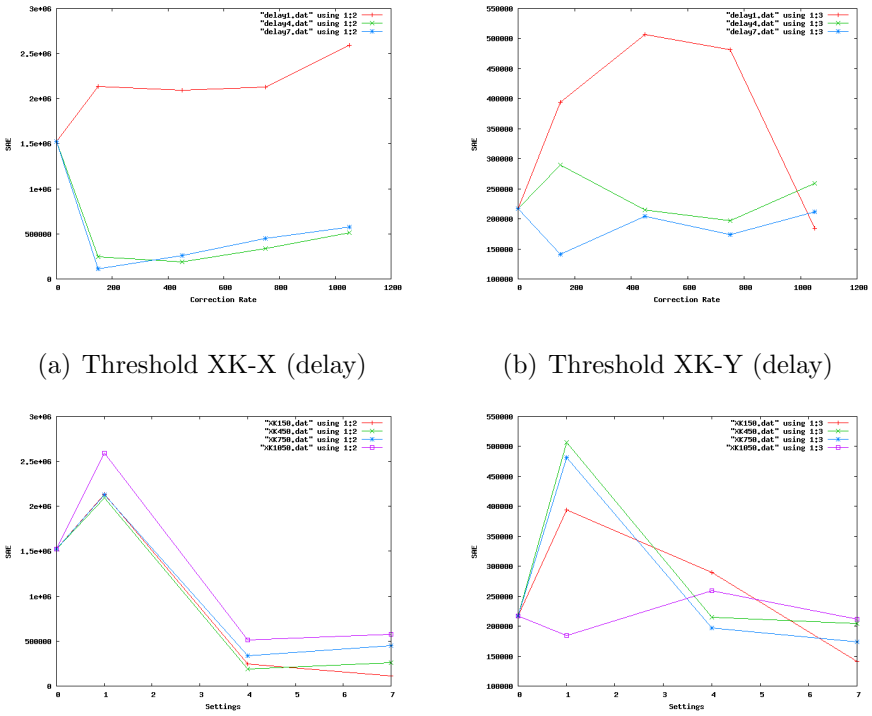


(c) Threshold XAS-X (settings)



(d) Threshold XAS-Y (settings)

Figure 8.5: Threshold tests on XAS.



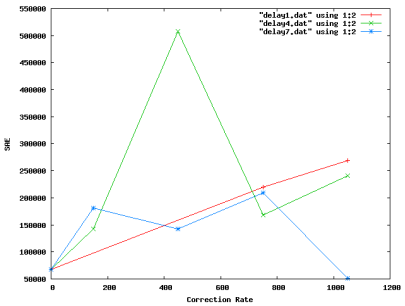
(a) Threshold XK-X (delay)

(b) Threshold XK-Y (delay)

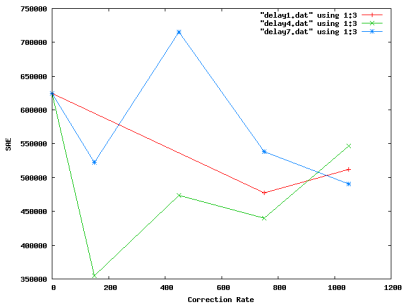
(c) Threshold XK-X (settings)

(d) Threshold XK-Y (settings)

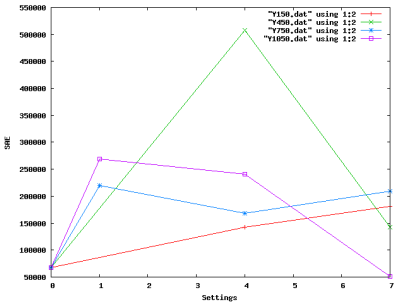
Figure 8.6: Threshold tests on XK.



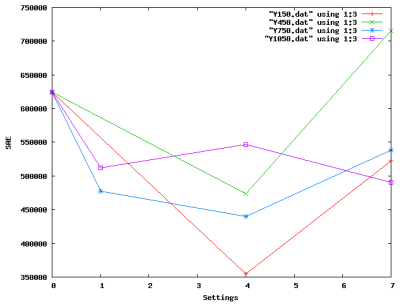
(a) Threshold Y-X (delay)



(b) Threshold Y-Y (delay)

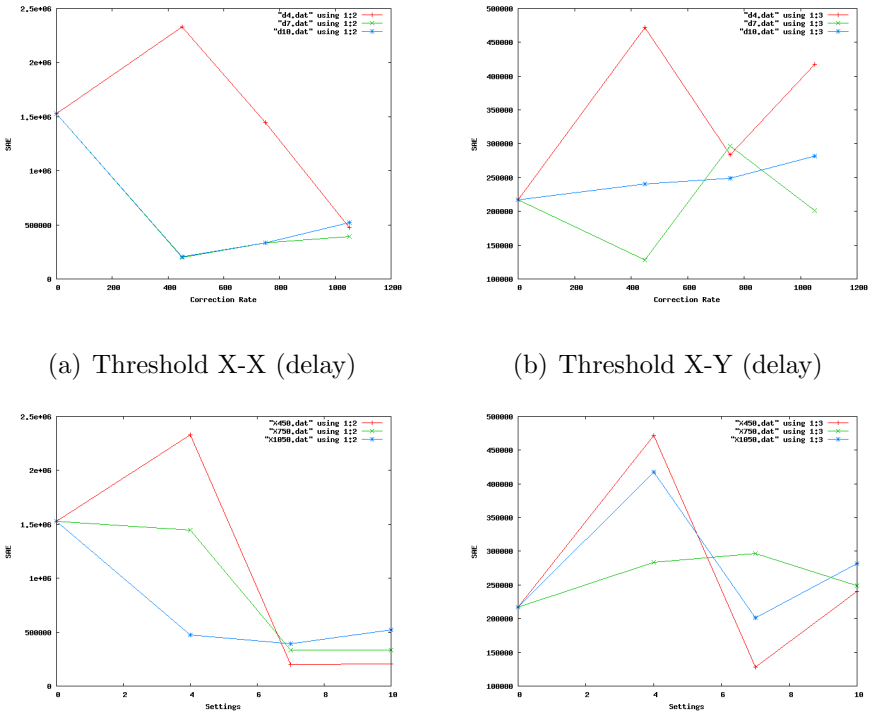


(c) Threshold Y-X (settings)



(d) Threshold Y-Y (settings)

Figure 8.7: Threshold tests on Y.



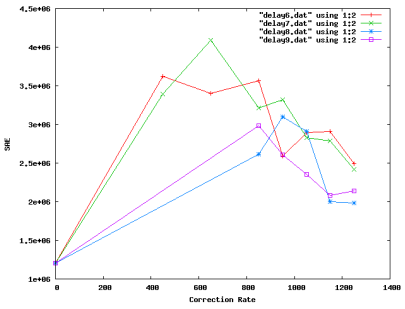
(a) Threshold X-X (delay)

(b) Threshold X-Y (delay)

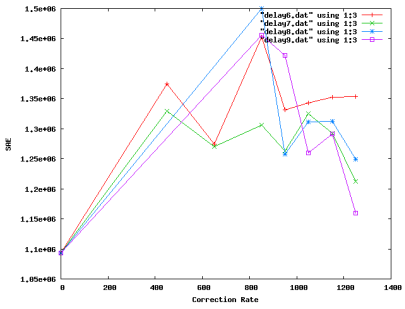
(c) Threshold X-X (settings)

(d) Threshold X-Y (settings)

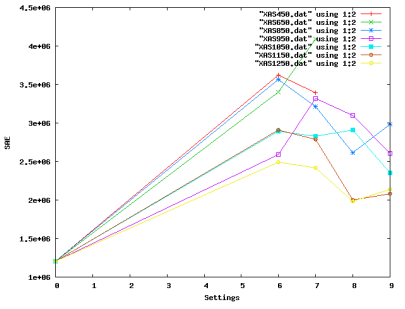
Figure 8.8: Threshold tests on X.



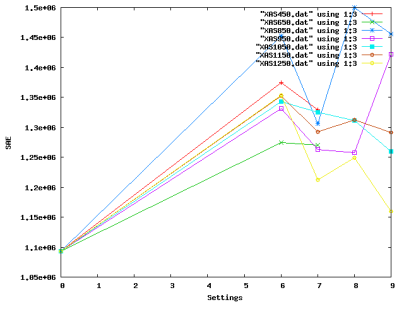
(a) PID XAS-P-X (delay)



(b) PID XAS-P-Y (delay)

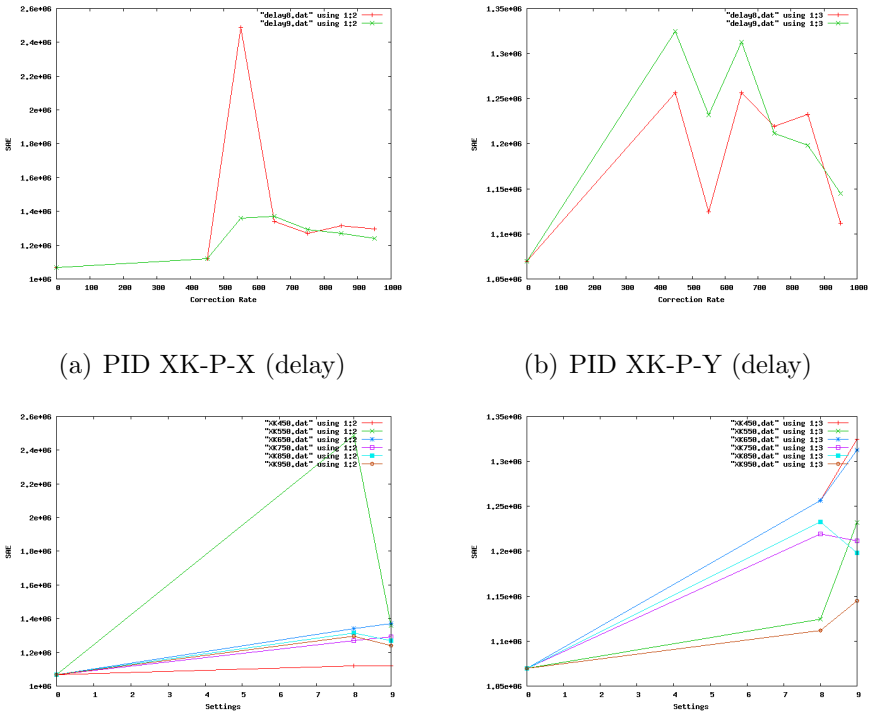


(c) PID XAS-P-X (settings)



(d) PID XAS-P-Y (settings)

Figure 8.9: PID P tests on XAS.



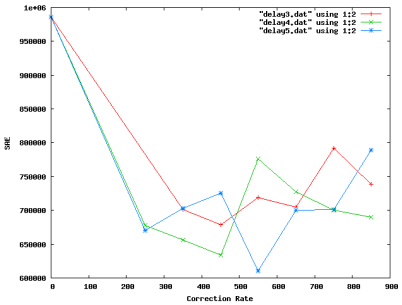
(a) PID XK-P-X (delay)

(b) PID XK-P-Y (delay)

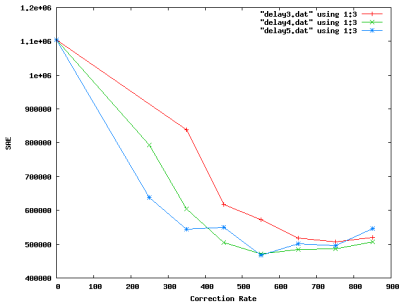
(c) PID XK-P-X (settings)

(d) PID XK-P-Y (settings)

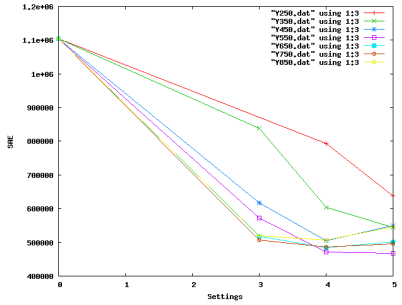
Figure 8.10: PID P tests on XK.



(a) PID Y-P-X (delay)



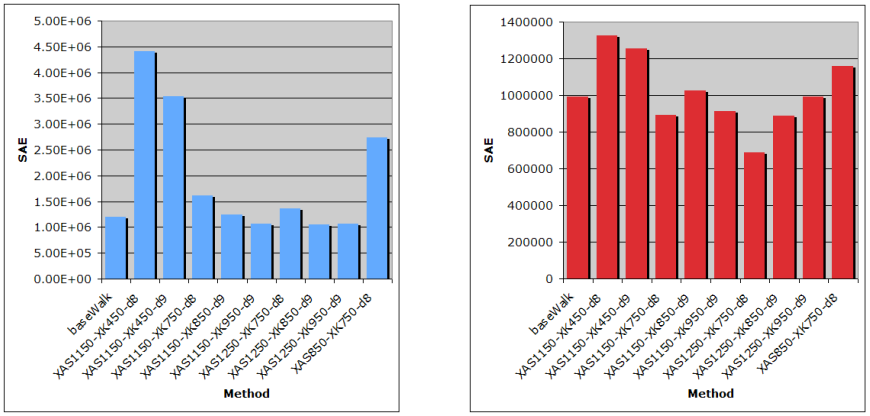
(b) PID Y-P-Y (delay)



(c) PID Y-P-X (settings)

(d) PID Y-P-Y (settings)

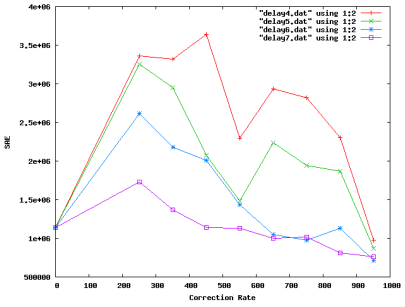
Figure 8.11: PID P tests on Y.



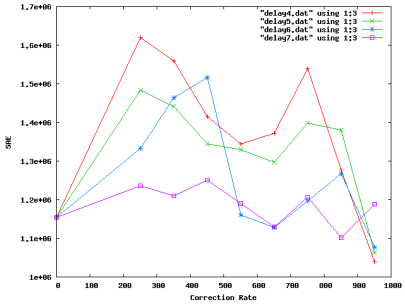
(a) PID X-P-X (delay)

(b) PID X-P-Y (delay)

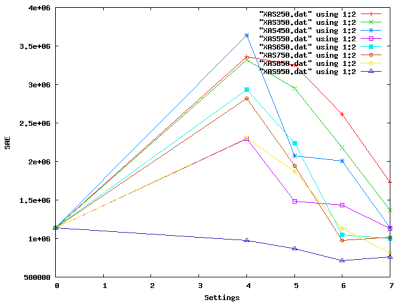
Figure 8.12: PID tests on X.



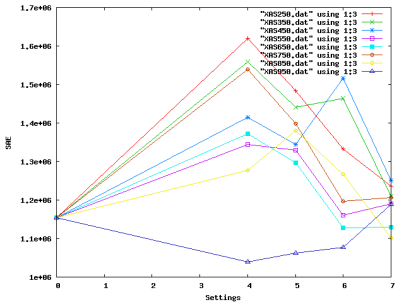
(a) Threshold XAS-X (delay)



(b) Threshold XAS-Y (delay)

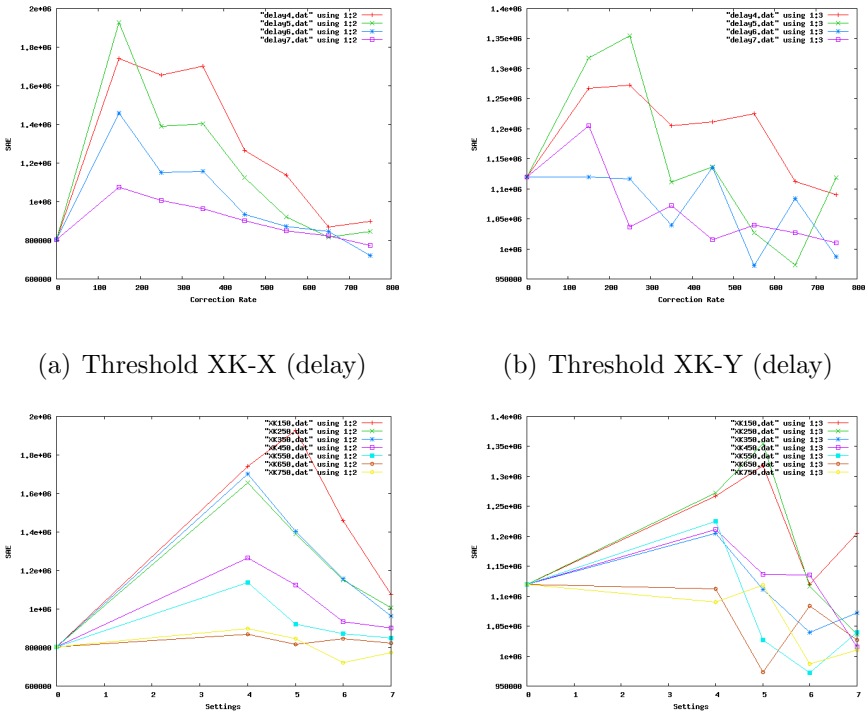


(c) Threshold XAS-X (settings)



(d) Threshold XAS-Y (settings)

Figure 8.13: Threshold tests on XAS.



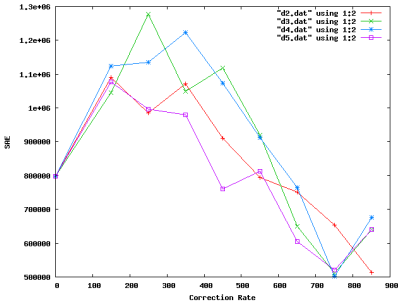
(a) Threshold XK-X (delay)

(b) Threshold XK-Y (delay)

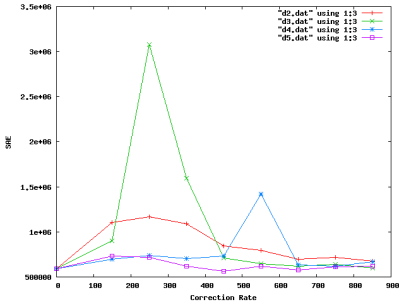
(c) Threshold XK-X (settings)

(d) Threshold XK-Y (settings)

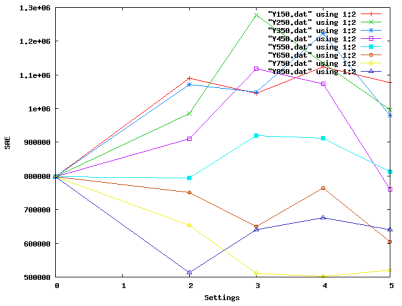
Figure 8.14: Threshold tests on XK.



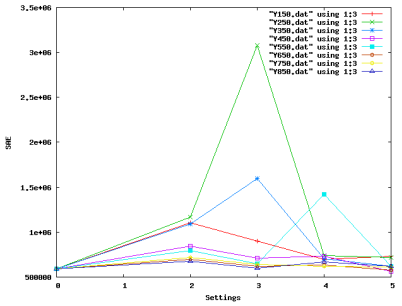
(a) Threshold Y-X (delay)



(b) Threshold Y-Y (delay)

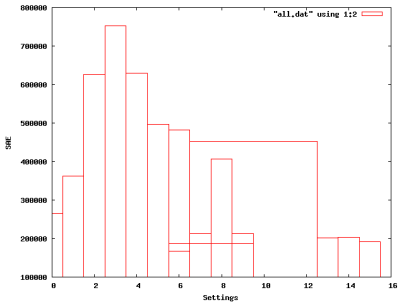


(c) Threshold Y-X (settings)

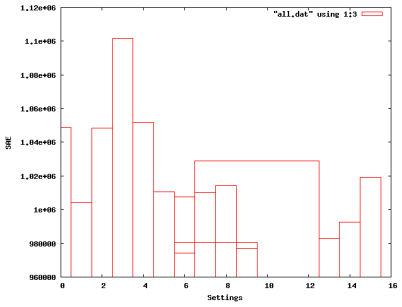


(d) Threshold Y-Y (settings)

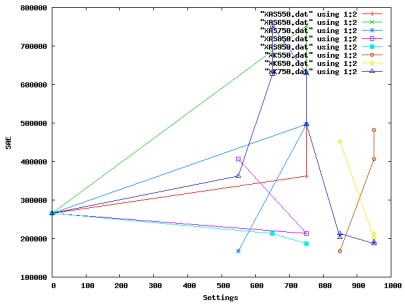
Figure 8.15: Threshold tests on Y.



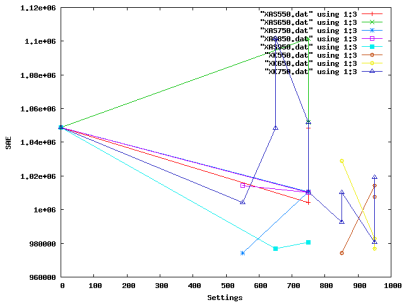
(a) Threshold X-X (delay)



(b) Threshold X-Y (delay)

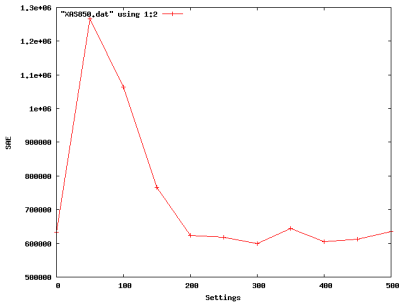


(c) Threshold X-X (settings)

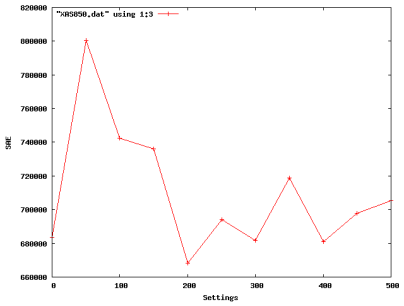


(d) Threshold X-Y (settings)

Figure 8.16: Threshold tests on X.

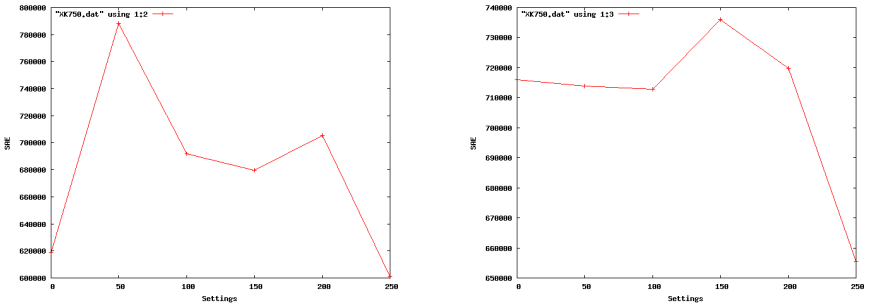


(a) PID XAS-D-X (delay)



(b) PID XAS-D-Y (delay)

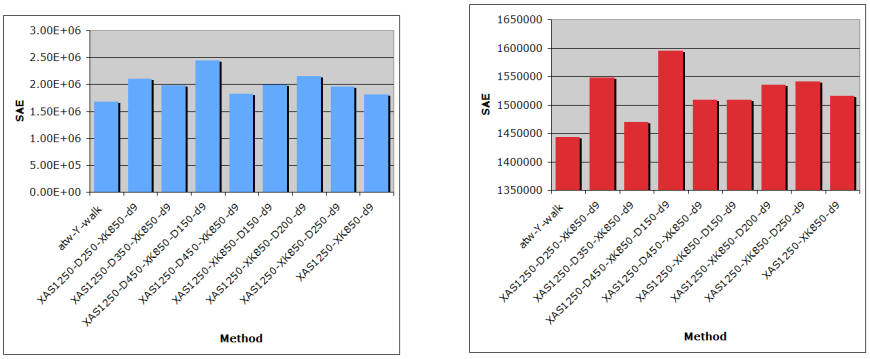
Figure 8.17: PID D tests on XAS.



(a) PID XK-D-X (delay)

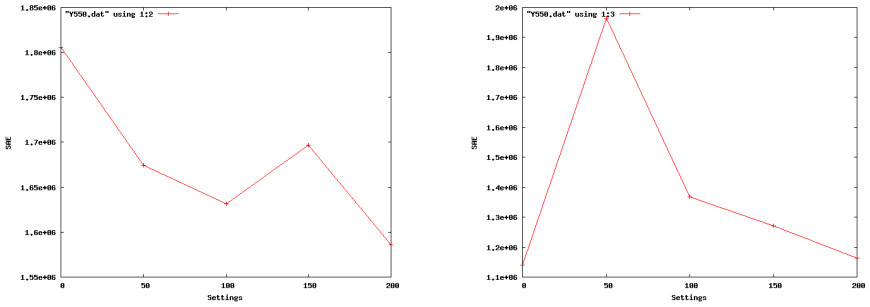
(b) PID XK-D-Y (delay)

Figure 8.18: PID D tests on XK.



(a) PID X-D-X (delay) (b) PID X-D-Y (delay)

Figure 8.19: PID D tests on X.



(a) PID Y-D-X (delay) (b) PID Y-D-Y (delay)

Figure 8.20: PID D tests on Y.

Bibliography

J. Baltes and S. McGrath. Tao-Pie-Pie. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

J. Baltes, S. McGrath, and J. Anderson. Feedback control of walking for a small humanoid robot. In *Proceedings of the FIRA World Congress*, Vienna, Austria, October 2003. Published on CD.

J. Baltes, S. McGrath, and J. Anderson. Tao-pie-pie humanoid robot. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.

Jacky Baltes and John Anderson. Humanoid robots: Hiro and daiguard-rs. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.

Jacky Baltes and Patrick Lam. Design of walking gaits for tao-pie-pie, a small humanoid robot. *Advanced Robotics*, 18(7):713–716, August 2004a.

Jacky Baltes and Patrick Lam. Design of walking gaits for Tao-Pie-Pie, a small humanoid robot. *Advanced Robotics*, 18(7):713–716, August 2004b.

Changjiu Zhou. Robo-Erectus. http://www.robo-erectus.org/. Accessed February 28, 2006.

J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, third edition, 2005.

John Van de Vegte. *Feedback Control Systems*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 3 edition, 1994.

Dr. David Dickman. Vestibular system primer. http://vestibular.wustl.edu/vestibular.html. Date accessed: March 22, 2007.

FIRA. FIRA Robot Soccer World Cup 2002. http://www.fira.net/firacup/2002.html, 2002. Accessed July 7, 2006.

FIRA. FIRA Robot Soccer World Cup 2003. http://www.ihrt.tuwien.ac.at/FIRAWM03/, 2003. Accessed February 28, 2006.

FIRA. Fira 2005 rules. http://www.fira.net/soccer/hurosot/HuroSot.pdf, 2005. Accessed March 8, 2005.

K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 1321–1326, May 1998.

HONDA. Asimo: Technology. http://world.honda.com/ASIMO/technology/spec.html, August 2003. Accessed February 28, 2006.

Q. Huang, S. Kajita, N. Koyachi, K. Kaneko, K. Yokoi, H. Arai, K. Komoriya, and K. Tanie. A high stability, smooth walking pattern for a biped robot. *Proceedings of the International Conference on Robotics and Automation*, pages 65–71, 1999.

Qiang Huang, We ming Zhang, and Kejie Li. Sensory reflex for biped humanoid walking. In *Proceedings of the 2004 International Conference on Intelligent Mechatronics and Automation*, 2004.

A. Iakovlev and A. Kritchoun. Team description ARNE. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

A. J. Ijspeert. *Design of artificial neural oscillatory circuits for the control of lamprey- and salamander-like locomotion using evolutionary algorithms*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, UK, 1998. URL `http://birg2.epfl.ch/publications/fulltext/ijspeert98c.pdf`.

Parallax Inc. Memsic 2125 accelerometer demo kit (#28017): Acceleration, tilt, and rotation measurement. http://www.parallax.com/dl/docs/prod/acc/memsickit.pdf. Accessed July 7, 2006.

S. Kagami, F. Kanehiro, Y. Tamiya, M. Inaga, and H. Inoue. Autobalancer: An online

dynamic balance compensation scheme for humanoid robots. In *Proceedings of the Fourth International Workshop on Algorithmic Foundations on Robotics*, 2000.

S. Kajita, K. Kaneko, K. Harada, F. Kaniehiro, K. Fujiwara, and H. Hirukawa. Biped walking on a low friction floor. *Proceedings of the International Conference on Intelligent Robots and Systems, 2004*, 4:3546 – 3552, 2004.

Fumio Kanehiro, Kenji Kaneko, Liyoshi Fujiwara, Kensuke Harada, Shuuji Kajita, Kazuhito Yokoi, Hirohisa Hirukawa, Kazuhiko Akachi, and Takakatsu Isozumi. The first humanoid robot that has the same size as a human and that can lie down and get up. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003.

D. Kee, G. Wyeth, A. Hood, and A. Drury. Guroo: Autonomous humanoid platform for walking gait research. In *Proceedings of the Conference on Autonomous Minirobots for Research and Edutainment (AMiRE2003)*, 2003.

J. Kim and J. Oh. Walking control of the humanoid platform KHR-1 based on torque feedback control. *Proceedings of the International Conference on Robotics and Automation*, 1:623–628, 2004.

James Kuffner, Satoshi Kagami, Koichi Nishiwaki, Masayuki Inuba, and Hirochika Inoue. Dynamically stable motion planning for humanoid robots. *Autonomous Robots*, 12:105 – 118, 2002.

James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inuba, and Hirochika Inoue. Motion planning for humanoid robots under obstacle and dynamic balance

constraints. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001.

Pasan Kulvanit, Bantoon Srisuwan, Khunit Siramee, Apinya Boonprakob, and Djitt Laowattana. Team kmutt: Team description paper. In *Proceedings of the RoboCup 2006*, 2006. Accessible at www.humanoidsoccer.org/qualification/KMUTT_TDP.pdf; accessed December 12, 2006.

Hun-ok Lim and Atsuo Takanishi. Compensatory motion control for a biped walking robot. *Robotica*, 23:1–11, 2005.

Toby Daniel Low. "active balance for a humanoid robot". Technical report, "University of Queensland", October 2003.

S. McGrath, J. Baltes, and J. Anderson. Active balancing in a small humanoid robot. In *Proceedings of the 2004 FIRA Robot World Congress*, Busan, Korea, October 2004a. FIRA. Published on CD.

S. McGrath, J. Baltes, and J. Anderson. Active balancing using gyroscopes for a small humanoid robot. In S. C. Mukhopadhyay and G. Sen Gupta, editors, *Second International Conference on Autonomous Robots and Agents (ICARA)*, pages 470–475. Massey University, December 2004b.

X. Mu and Q. Wu. Dynamic modeling and sliding mode control of a five-link biped during the double support phase. *Proceedings of the 2004 American Control Conference*, 3:2609–2614, 2004.

Harry Mueller. Six-servo humanoid walker. *Robot Magazine*, (3):48–49, 2006.

Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:79–91, 2004.

M. Ogino, Y. Katoh, M. Aono, M. Asada, and K. Hosoda. Reinforcement learning of humanoid rhythmic walking parameters based on visual information. *Advanced Robotics*, 18(7):677–697, 2004.

K. Okamoto and Y. Okamoto. Team "Foot-Prints". In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

Osaka University. Asada Lab. http://www.er.ams.eng.osaka-u.ac.jp/index-eg.html. Accessed February 28, 2006.

A. Pickel. Control for a biped robot with a minimal number of actuators. Master's thesis, University of Applied Sciences Koblenz and the University of Western Australia, 2003.

Jerry Pratt. *Exploiting Inherent Robustness and Natural Dynamics in the Control of Bipedal Walking Robots.* PhD thesis, Computer Science Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2000.

C. Rawichote, M. Aono, J. Ooga, M. Ogino, and M. Asada. Senchans 2003. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003:*

Robot Soccer World Cup VII, volume 3020 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

RoboCup. Robocup 2005 rules. http://er04.ams.eng.osaka-u.ac.jp/humanoid_webpage/humanoid.pdf, 2005. Accessed March 8, 2005.

C. Shou, P. Kong Yue, F. Seng Choy, and N. Ahmed. Robo-erectus: A soccer-playing humanoid robot. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

Neal Stephenson. *SnowCrash*. Bantam Spectra, 1992.

University of Western Australia. University of Western Australia Mobile Robot Lab. http://robotics.ee.uwa.edu.au/. Accessed February 28, 2006.

G. Wyeth and D. Kee. Distributed control of gait for a humanoid robot. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

Jin-ichi Yamaguchi, Atsuo Takanishi, and Ichiro Kato. Development of a biped walking robot adapting to a horizontally uneven surface. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1156–1163, 1994.

Nobuo Yamato, Yohei Akazawa, Hiroshi Ishiguro, Tomotaka Takahashi, Takeshi Maeda, Takuro Imagawa, Hitoshi Takayama, Noriaki Mitsunaga, and Takahiro Miyashita. Teamosaka. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi,

editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.

C. Zhou. Neuro-fuzzy gait synthesis with reinforcement learning for a biped walking robot. *Soft Computing*, 4:238–250, 2000.

C. Zhou and Q. Meng. Dynamic balance of a biped robot using fuzzy reinforcement learning agents. *Fuzzy Sets and Systems*, 134(1):169–187, 2003.