A self-contained motion capture platform for e-textiles

Jacob Simmons

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Thomas L. Martin, Co-Chair
Mark Jones, Co-Chair
Peter Athanas

August 13, 2010
Blacksburg, Virginia

Keywords: E-textiles, Wearable Computing, Inertial Measurement Units

A self-contained motion capture platform for e-textiles

Jacob Simmons

ABSTRACT

Wearable computers and e-textiles are increasingly prevalent in today's society. Motion capture is one of many potential applications for on-body electronic systems. Self-contained motion capture applications require information from sensors distributed throughout the body on a "smart" garment. Therefore, this thesis presents the design of a flexible hardware platform for e-textile motion capture applications. This thesis also presents software for one such application, namely, tracking the pose or relative position of body limbs. The accuracy of this solution is compared to an industrial optical motion capture system. The combined hardware and software design are successful at collecting and processing motion capture data in the context of an e-textile jumpsuit.

Acknowledgements

I have many people to thank for their support and assistance in the completion of this work.

I would like to thank Dr. Tom Martin, my advisor and committee co-chair, for his advice, support, and many hours invested in this project and in me. Without his efforts I could not have succeeded.

I would like to thank Dr. Mark Jones for serving as my committee co-chair, and for his constant critiquing and guidance throughout this process.

I would like to thank Dr. Peter Athanas for serving on my committee.

I would like to thank Dr. Thurmon Lockhart for allowing the use of his motion capture equipment to perform many of the tests in this thesis.

I would like to thank Robert Lewis and David Uliana, for their efforts in helping me work through problems, find fresh ideas, and collect results.

I would like to thank Dr. William Baumann, for his guidance regarding proper implementation of Kalman filtering.

I would like to thank Bob Lineberry for his help with printed circuit board design and soldering tips.

I would like to thank my family and friends, for supporting me outside the lab and keeping me focused and motivated.

This thesis is dedicated to Megan Salaj, Susan Hedgepeth, and Maxwell Hedgepeth.

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0447741. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation(NSF).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Wearable computers and electronic textiles (e-textiles) are increasingly prevalent in today's society. With portable music players, laptop computers, and cell phones becoming virtually ubiquitous, day-to-day reliance on wearable computers is already a reality for many. Virginia Tech's e-textiles lab has already accomplished the task of creating a self-contained smart garment system [1]. In the previous versions of this e-textile platform, the sensor capabilities included harvesting and transmitting of simple inertial measurement data, but this information was only used for the purposes of pattern recognition [2] or detecting their own position relative to the garment itself [3]. With those successes, motivation is high to expand the e-textile capabilities into other applications, among them self-contained motion capture.

Motion capture is a particularly useful application for wearable computing because the framework of an e-textile affords a unique opportunity to track a body with local sensors. In particular, self-contained motion capture allows for the garment wearer's movements to be recorded in any location, and stored on the e-textile itself for later processing if necessary. Without the constraint of needing a lab for processing or being limited to working in rooms of optical motion capture cameras, self-contained motion capture in the context of an e-textile allows for flexibility and mobility at a time when these design aspects are at a premium.

1.2 Contributions

This thesis presents the design and implementation of a self-contained motion capture system with an emphasis on e-textile feasibility. This design improves on previous iterations by tracking the pose of individual limbs and, indirectly, an entire user's body using only gyroscope and compass sensors. The flexibility of this framework allows it to be customized easily and leaves the design open for further development. The overall effectiveness of this design was demonstrated on Virginia Tech's e-textile jumpsuit with an OpenGL animation program, as well as in many simpler, repeatable off-body tests.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 outlines the background information needed to place this research in a proper context. Chapter 3 is a detailed description of the hardware required to implement this framework. Chapter 4 provides an overview of the software developed for all levels of this project. Chapter 5 covers the results of the experiments and an examination of the system's performance. Finally, Chapter 6 summarizes the contributions of this thesis and conclusions drawn from this research, and explores avenues for attempting future work on this project.

Chapter 2

Related Work

This chapter presents previous work which relates to this thesis. Many concepts from these endeavors were used to develop the platform described in this thesis, and for specific design of the hardware and software necessary to complete the e-textile prototype. This chapter begins with an overview of inertial measurement motion capture systems, and how their development relates to this work. Then, the process of Kalman filtering and its potential applications are described. Finally, the previous projects completed in the Virginia Tech E-Textiles Laboratory relevant to the work done in this thesis are dicussed.

2.1 Inertial Measurement Motion Capture Systems

Inertial measurement motion capture systems use sensors to track a body's movement and orientation. Typically these systems use some combination of accelerometers, gyroscopes, and magnetometers to determine the position and orientation of each sensor.

In addition to inertial measurement unit (IMU) based designs, motion capture systems can be developed using optics, magnetic field sensors, or computer vision algorithms. These systems may integrate IMUs into their design, or attempt to forego local sensors completely [4]. However, due to the inclusion of external hardware, and stricter requirements in terms of setup, these approaches are generally much more expensive and less portable. Relatively inexpensive and highly mobile solutions for motion capture based on IMUs have been demonstrated in applications ranging from medical [5] to gaming [6]. The XSens MVN Motion Capture suit is one successful commercial option also based on inertial measurements, but costs around $60,000 to

purchase [7].

A common issue with traditional IMU-based motion capture designs is measurement drift. Because computation of positions and joint angles relies on integrating raw acceleration or angular velocity, small errors in individual measurements become drastically compounded and accumulate over time. Circumventing the issue of sensor drift has been done primarily by external means, such as corrections involving cameras [8], GPS [9], or optic markers [10]. Additionally, constraining the detectable motions to include "still times" at which the sensors can be corrected can periodically alleviate this problem. While many previous motion-capture designs have represented the state of sensors using traditional Newtonian position, velocity, and acceleration vectors, our implementation instead tracks the orientation of each sensor in terms of heading, pitch, and roll angles. By combining the less frequent but highly accurate and absolute readings of the digital compass with the angular velocity updates provided by the gyroscope, our design both provides for a local method of self-correction as well as limits the maximum possible drift due to accumulated error. A similar design tracking orientation angles instead of position estimates has been demonstrated to be successful [11].

Previous work done in inertial measurement motion capture also attempts to limit the impact of these issues, but usually does so at the expense of increased constraints in motion or narrowed scope. The sportSemble sensor system developed by MIT uses a similar hardware configuration to our design to track the movements of baseball players, but limits its focus to exploring the biomedical effects of short, high-intensity bursts of activity on specific parts of the body rather than full-body motion capture over longer time periods [5]. Other experiments focus exclusively on gait detection [12] or gesture recognition [13]. The clothing designed in [14] tracks known movements in different users based on similarities in acceleration. In consideration of the work done in this area up to this point, one of our goals was to capture pose over time accurately without needing external corrections.

Unlike many similar self-contained motion capture suit implementations such as the Xsens MVN suit, our design does not use external wires between sensors that contact the body underneath. Instead, all communication lines are woven directly into the fabric of the suit itself, allowing interacting with the jumpsuit to more closely resemble putting on and wearing actual clothing.

2.2 Kalman Filtering

A useful method for combining the inputs of multiple IMUs into one known set of state variables is the Kalman filter. A Kalman filter is a matrix-based method for finding a best estimate of a system's states and

outputs after accounting for noise in the process and measurements of that system. As the filter updates with each new set of measurements, it weights the value of the new measurements against its own existing estimate of the state based on the amount of noise present in the system. The Kalman filter works recursively, so each state estimate relies only on the previous state estimate and the current measurements. In addition to its estimate of the state, the Kalman filter maintains an estimate of the covariance of the error of this state, which in simple terms is how much weight the existing state should be given relative to the new state based on the most recent observations [15]. At each iteration, the Kalman matrices are updated in two stages, the predict stage and the update stage.

The Kalman filter predicts its state, x_{pred}, and its error covariance, P_{pred}, according to the following equations:

$$x_{pred_k} = Ax_{k-1} \tag{2.1a}$$

$$P_{pred_k} = AP_{k-1}A^T + Q \tag{2.1b}$$

where A is the matrix describing transition from the previous state to the current state, and Q is the covariance matrix of the process noise. While A and Q can vary with k, for the purposes of the work described here they are fixed.

The predictions made in this step can be used for calculations if an observation (in this case, a measurement) is not available. When an observation is available, the Kalman filter proceeds to the update phase, using the following equations:

$$K_k = P_{pred_k}C^T(CP_{pred_k}C^T + R) \tag{2.2a}$$

$$x_k = x_{k-1} + K_k(y_k - Cx_{k-1}) \tag{2.2b}$$

$$P_k = (I - K_kC)P_{k-1} \tag{2.2c}$$

Here, y_k is the measurement at time k, C is the matrix transforming states into observations(measurements), and R is the covariance of the measurement noise. The values x_k and P_k are the outputs of the filter. x_k is the filtered state for the update at time k, and P_k is the estimation of the noise covariance for this state at

time k. The term K is called the *Kalman gain* and directly determines the state and covariance estimates. When the update is completed, the filter can either wait for another observation (and another update), or advance the state another timestep by performing another prediction.

Kalman filters are important in motion capture implementations for a number of reasons. Most obviously they can attempt to account for errors present in the IMUs or other sensory devices in order to give more accurate tracking measurements. In addition, since many individual IMUs are quite noisy, filtering can prevent the bouncing back and forth to which accelerometer and gyroscope data is prone even while still, resulting in more stable visualizations of motion. When moving, the Kalman filter will thus also provide smoother tracking [16]. The "predict" stage of the Kalman filter is specifically useful, in that it can allow certain real-time applications to move ahead with a priori filter estimates of state that can be updated when additional observations are detected. Finally, Kalman filtering is capable of combining different types of sensors into one meaningful state to be processed. For instance, if a sensor's state is represented in the filter as position, velocity, and acceleration, each individual IMU on the sensor can be expressed in terms of its contribution to that state. This allows for easy synthesis of raw data without losing the ability to estimate more accurately, even in the case of combining visual and inertial measurement sensors [17].

2.3 Previous E-Textile Research at Virginia Tech

The Virginia Tech E-Textile Laboratory has performed many experiments in past years developing "smart" garments for a number of applications. In particular, the Virginia Tech E-Textile Laboratory has produced several previous versions of an e-textile geared towards motion capture. Research done by Quirk [18] demonstrated the feasibility of producing e-textiles with conductive material capable of transmitting signals through clothing. With an automated loom, Virginia Tech can produce these garments in-house.

The original garment design was a pair of pants using embedded wires and built for use with on-body accelerometers and gyroscopes [1]. The software for this garment attempted to match measured accelerations with expected results from a simulated environment. Later work with the pants focused on activity recognition [19], using a singular value decomposition algorithm to organize the data from the accelerometers and gyroscopes. The processed data could then be compared to a list of pre-defined activities and determine which of these activities was most likely taking place in real-time.

The e-textile platform was expanded to a full body jumpsuit in a later iteration of the design. The jumpsuit fixed some mechanical issues with the pants, as well as allowed for many additional features including USB-

compatible local sensors, on-body data processing, and wireless communication between the jumpsuit and a nearby PC via Bluetooth. The work done in [2] established the two-tiered model of on-body communication and data collection, by which the local or Tier 1 sensors on the garment serve only to harvest data and pass this data to the higher-level processing units, called Tier 2 sensors. Applications done with this garment in the past have included measuring local accelerations, activity recognition and matching in the same style as for the original pair of pants, and algorithms to automatically detect the limb to which a specified Tier 1 sensor is attached [3].

The work in this thesis uses the same full body jumpsuit as an e-textile platform. However, instead of simply storing data harvested from Tier 1 sensors, this project taps into the available on-suit information processing capacity to perform a large-scale application, specifically motion capture, on the suit itself. Whereas previous work with this e-textile focused mainly on identifying mathematical patterns in the data for applications like activity recognition, this implementation more aptly interprets the physical meaning of the IMU information, requiring that the Tier 2 processing unit be capable of synthesizing the data of multiple sensors in real-time to produce a tangible result.

Chapter 3

Hardware Overview

The system described in this thesis consists of two levels of processors connected by an on-fabric wired network, using the two-tiered framework developed in previous Virginia Tech e-textile research (see Chapter 2). The local sensors (also called Tier 1 sensors) contain the hardware necessary to acquire inertial motion data, package that data, and transmit the data to a higher-level hardware component. In order to meet many potential needs both in the current implementation and in future systems, the local sensors were designed to be as simple and flexible as possible.

The second level of this design consists of the central processing unit or units, also called Tier 2 sensors. In the design described by this thesis there is only one Tier 2 node; however, larger numbers of Tier 1 sensors may require multiple central processors.

The hardware for the design was chosen to meet the following criteria as closely as possible. Each Tier 1 sensor had to be capable of sensing acceleration and rotation in all three dimensions yet not be overly susceptible to drift errors. Additionally, sensors should not rely on input from other components or external sources to sense their own positions. Another key design priority was to have a uniform voltage for all of the Tier 1 sensor's components, thus allowing each sensor to have a single voltage regulator. Another important concern was local communication, so devices with digital interfaces, specifically I^2C, were preferred. Finally, the fairly common and expected concerns of low power usage and small physical size were considered as well in making a final decision.

3.1 Tier 1 Sensor Hardware

Tier 1 sensors contain the inertial measurement units used for motion capture, and a processor to collect this data and transmit it to the Tier 2 sensor. Each Tier 1 sensor is designed to be an on-body node for measuring local acceleration, rotation, and orientation.

3.1.1 Accelerometer

Each Tier 1 sensor uses one Analog Devices ADXL345 accelerometer. The ADXL345 senses on three axes and is precise to 4mg of acceleration. The range of the accelerometer can also be adjusted manually; this was important during the planning stages of the sensors because the exact upper bound on instantaneous acceleration was unknown. While high-g accelerations would be unreasonable to expect over a long period of time, it is conceivable that a limb might move at extremely high acceleration for a few brief data samples that would need to be captured. The ADXL345's range can be set at +/- 2g, 4g, 8g, or 16g. The communication is entirely digital and I^2C compatible. The chip can run at 3.3 V, which satisfied our requirement of low power and a single operating voltage for the entire sensor.

3.1.2 Gyroscope

In order to sense rotations in space, each local sensor is outfitted with two STMicroelectronics LP530AL 2-axis gyroscopes, as shown in Figure 3.1. Each of these chips senses rotation about the x- and z- axes, so by arranging them perpendicularly on the sensor board, one "x-axis" becomes the "true" y-axis for the sensor.

Choosing a gyroscope was a difficult task because at the time when the hardware was selected, no available gyroscope ICs met all of the desired criteria. The top priority for the sensor boards was to maintain a low, uniform operating voltage. Since no 3-axis gyroscopes were available that met the requirements, the decision was made to use two 2-axis gyroscopes instead. By using two gyroscopes with redundant z-axes, the sensor board would not need to have chips mounted vertically on its side; instead, both gyros lay flush to the board and on the same side. Another considerable drawback is that the output of the gyroscopes is an analog voltage, not a digital signal. This trade-off was deemed acceptable since analog-to-digital converter pins would be available on the local microprocessor. The LP530AL can sense angular velocity up to 1200 deg/s in normal mode, but can also be set to operate in 4x mode which drops the range to +/- 300 deg/s and increases the precision of the device. For these experiments the device was set to 4x mode.

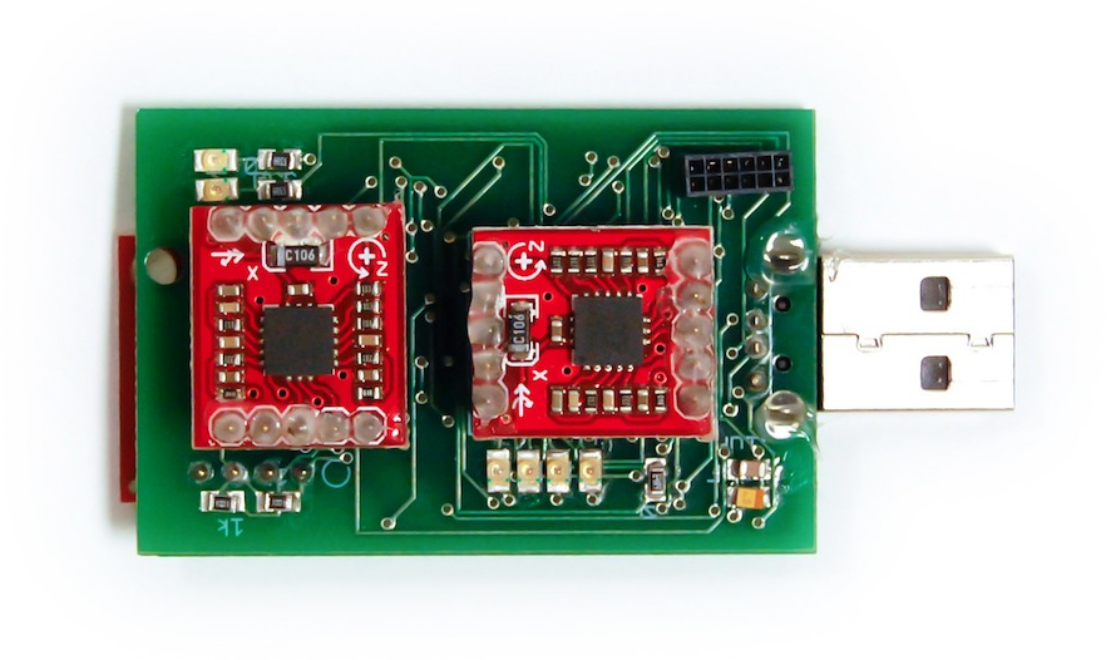

Figure 3.1: Perpendicular arrangement of two 2-D gyroscopes.

3.1.3 Digital Compass

Each local sensor uses one Honeywell HMC6343 digital compass to periodically provide an absolute orientation of that sensor in space. This allows individual sensors to self-correct with each compass update, negating the effect of drift.

While relatively expensive compared to the accelerometer and the gyroscopes, the HMC6343 chip offers several advantages. The digital compass compensates internally for its own tilt to sense heading, pitch, and roll angles. This means that regardless of how the compass chip is mounted on the sensor board, the compass can give an accurate snapshot of the orientation. The compass's maximum update rate is 10Hz, so its data is intermittent and insufficient, on its own, to capture the user's motion. However, correcting the accumulated state of the sensor with periodic compass updates improves the overall accuracy of the calculated orientation by keeping errors in individual gyroscope measurements from building up over time.

Another important feature of the compass is that there are three starting orientations to choose from. The compass can report its orientation relative to a level starting point, or from an upright starting point. The upright orientations can be chosen with the compass edge as "forward" or the front as "forward". In this

thesis the orientation is measured in the level mode, except when the magnitude of the pitch angle is greater than 60 degrees. In this case it is more appropriate to use an upright orientation; this experiment uses the upright front orientation under these circumstances.

3.1.4 Microprocessor

Once the inertial measurement units were selected for the sensor board, the requirements for data processing and both local and external communication became much clearer. Specifically, the microprocessor for each Tier 1 sensor needed to be easily programmable and have memory capabilities for the program and computations needed to control the IMUs. Because two analog gyroscope chips were used to detect rotation, and five outputs are associated with each chip, the microprocessor would need no fewer than ten analog-to-digital conversion channels. These channels also needed to have at least ten bits of resolution to be precise enough for computations.

Another limiting factor in choosing a microprocessor was the communication capabilities. In order to read data from the digital components, the microprocessor needs access to an I^2C bus. However, a second, independent I^2C bus is necessary to communicate with the central processing unit of the entire system.

The sensor boards in this design use the low-power PIC18LF6722 microprocessor. This chip features twelve 10-bit ADC channels, 1 megabit of flash memory, 128k program memory, and 4k of SRAM. One drawback of the PIC is its relatively large physical size; however, this tradeoff was deemed acceptable in order to meet the other requirements of the system. The completed Tier 1 sensor is shown in Figure 3.2.

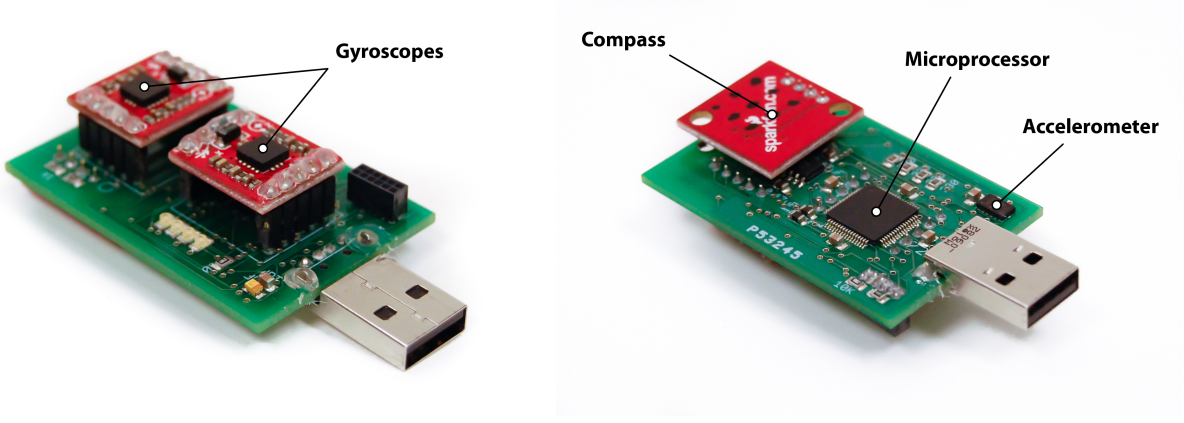

Figure 3.2: A Tier 1 sensor

.

3.2 Tier 2 Sensor Hardware

Tier 2 sensors are responsible for interpreting and organizing the information received from the local nodes and serve as the central processing units of the system. Therefore, Tier 2 sensors require more processing power and storage, and must be capable of communicating continually with a large number of Tier 1 sensors.

The Tier 2 sensor used in this design is unchanged from the previous projects at Virginia Tech that have used this e-textile jumpsuit. The information about the Tier 2 sensor's design is included here only as a reference; no further contributions to its design are made in this thesis. For a more thorough explanation of the development of the Tier 2 sensor components, please see the previous work done by Chong [2]. The previous versions of the garment developed at the Virginia Tech E-textiles Lab have successfully used the Verdex 400xmbt motherboard from Gumstix as the processing unit of the Tier 2 sensor. The Gumstix has the capacity to run a full Linux operating system and is also compatible with USB and Bluetooth. In addition to performing the necessary on-suit processing for real-time applications, the Gumstix board also has the capability to store several minutes of data into a file for later use, depending on the number of Tier 1 sensors and the data rate. Like the Gumstix motherboard, the I^2C to serial converter board is a necessary component of a past design of the Tier 2 sensor. This board lies parallel to the converter board and serves only to convert the incoming I^2C data from the Tier 1 sensors into universal asynchronous receiver/transmitter (UART) serial data. The I^2C to serial converter board is also a Gumstix product.

3.3 Jumpsuit e-Textile

The jumpsuit used as an e-textile platform for this work is the same garment used in previous work done at the Virginia Tech e-textiles lab. The jumpsuit is shown with Tier 1 and Tier 2 sensors attached in Figure 3.3. The jumpsuit is outfitted with nine Tier 1 sensors: one on each forearm, each upper arm, each thigh, and each shin, and one sensor on the torso.

3.4 Sensor Calibration

Each Tier 1 sensor contains four IMUs: one accelerometer, two gyroscopes, and one digital compass. This section outlines how to calibrate each device to ensure the data from each sensor is as accurate as possible.

Figure 3.3: The jumpsuit used as an e-textile platform for this thesis.

3.4.1 Motivation for Calibration

Each Tier 1 sensor is expected to provide accurate and up to date about its position and orientation. This is especially important in the context of the e-textile application for this design, because each limb or joint of the body being tracked is described by a single sensor. Also, joint positions between limbs are placed assuming the limbs' calculation of orientation are accurate.

Specifically, precision in the accelerometers is more critical than in any of the other sensors. In order to determine a wearer's position, each acceleration reading must be integrated into velocity, which is then integrated into position. However, this process means that even a slight error in acceleration measurement will drastically affect the end result calculation of position.

For example, consider the case where the measurement error in the accelerometer is a constant $0.05g$. Assuming $g = 9.81$ m/s^2, this error is 0.4905 m/s^2, which means the measured velocity will have an error component of 0.4905 m/s * t, where t is in seconds. It is easy to see that this now-linear component will cause the accuracy of the velocity to rapidly deteriorate. A second integration of this function to position will cause the measurement error to be quadratic.

Additional sources of error arise due to the actual soldering of the Tier 1 sensor breadboard. The IMUs may not be exactly flush with the board or each other, particularly the compass which has its own breakout board that is mounted about an inch off the main section of the board. The end result of these placement issues is that a "flat" position as seen by the compass may be a slight tilt in the other sensors, or vice versa. This can cause small errors that cannot be detected via software.

The best way to address these issues is to calibrate each Tier 1 sensor. While some of the errors will still exist in smaller magnitude, the overall gains in accuracy of motion capture from a calibration process are substantial, and at a relatively small cost of time and equipment.

3.4.2 Accelerometer Calibration

Each accelerometer has two sets of values that must be calibrated as precisely as possible. The first set is the accelerometer sensitivity, which is defined to be the amount of real acceleration (in mg) described by each bit of the digital reading. The sensitivity along each axis is assumed to be unique. Also, each accelerometer is assumed to have a bias component in each direction as well. These two sets of data, covering all three axes, total six values that must be calculated and stored as the end product of the calibration routine.

The datasheet for the ADXL345 accelerometer gives the range of the sensitivity in each axis as 3.5mg to 4.3 mg, with 3.9mg being the typical value. The bias is listed as -150 mg to 150mg for the x- and y-axis, and -250 mg to 250 mg for the z-axis, with 0 mg being the typical value.

The calibration routine uses the one "ground truth" about the accelerometer, which is that when still, the magnitude of the measured acceleration vector should be exactly 1g. To find the sensitivities, the

accelerometer is placed in a variety of positions and held still while data is collected. Each of the samples contributes a term to an error function, which calculates how far the measured data deviates from the 1g magnitude mark, using the sensitivities and biases as variables. The calibrated sensitivities and biases can then be calculated by running a mathematical optimization on those variables to minimize the error function. For a more thorough discussion of the accelerometer calibration routines used in this thesis, please see Chapter 5.

3.4.3 Compass Calibration

The HMC6343 Honeywell digital compass can be calibrated for hard-iron distortions in the magnetic field. These errors can occur when the compass module is brought near magnetized metal [20]. Generally these errors manifest as heading angle discrepancies. Specifically, the compass's measurement of the heading angle will not change at the appropriate rate when the compass is turned. Continuing to change the heading once the measured heading angle stalls will cause the compass readings to eventually "jump" ahead to try to catch up with the true position. This discrepancy can be corrected by following the datasheet's hard-iron calibration routine, which involves two steady rotations and resets the magnometer offsets [20].

Error in the compass readings can also come about through sensor construction, when a discrepancy occurs between the placement of the compass's own breakout board and the sensor breadboard. In order to adjust for this potential error, the readings from the compass can be measured against a known orientation. By holding the sensor still so that gravity is the only force acting on the accelerometer, the now-calibrated accelerometer data can be converted to a set of expected orientation angles based on the direction of the gravity vector. In theory, these baseline angles should be equal to the compass readings, and any difference between these two sets of angles can be considered error due to the construction of the sensor board. In practice, however, the accelerometer calibration was not accurate enough to be used to correct potential orientation errors in the compass.

3.4.4 Gyroscope Calibration

A full-fledged calibration of the gyroscopes is less critical than for the other inertial measurement units. The gyroscopes only serve to provide orientation change estimates for a small number of samples in between compass updates. Therefore, the problem of drift error accumulating over time naturally has less of an impact, since any damage done to the sensor state by drift error is erased by the correction of the compass.

While not necessary for this specific implementation of the design, it is conceivable that an implementation with a much faster gyroscope read rate relative to the compass rate (and thus more gyroscope samples per compass update) would have more issues with drift. In this case a procedure similar to that listed for the accelerometer should be considered. For the experiments described in this thesis, the calibration of the gyroscopes was limited to corrections for the bias at the zero point with a series of still tests.

Chapter 4

Software Overview

The sections of this chapter are an overview of the software components and specific programs needed to perform effective motion capture on an e-textile platform. The programming of Tier 1 sensors and the communication between Tier 1 and Tier 2 sensors is briefly outlined, and then the method and rationale for the motion capture algorithm is discussed in detail.

4.1 Tier 1 Sensor Programming

All necessary software for the tier 1 nodes is programmed onto the PIC microprocessor through mounted headers using the PICkit programmer, shown in Figure 4.1. Because the purpose of the Tier 1 nodes is simply to gather and send information, this program simply loops infinitely, polling the digital components (the accelerometer and the compass) on the local I^2C bus, and reading the gyroscope data through the analog-to-digital converter channels. The data from these IMUs is combined into a single data packet that represents the information update for this sensor this cycle.

The data rate of each sensor is controlled by one of the PIC's countdown timers. When this counter reaches zero, a hardware interrupt is generated which sends the data packet to the Tier 2 sensor. This timer can be configured to set the frequency of data transfer; for the experiments in this thesis it is set at either 30 or 60 Hz.

Figure 4.1: A Tier 1 sensor's PIC being programmed by the PICkit.

4.2 System Communications

Each tier 1 sensor board communicates with the central gumstix hardware (tier 2 node) through a second, separate I^2C line. The tier 1 PICs are master devices that each control the slave I^2C-to-serial converter board in multi-master mode. The I^2C-to-serial board then performs the simple but necessary task of converting the incoming I^2C data to serial data to be processed by the gumstix board.

Due to the considerably slower data rate of the compass compared with the other devices, the tier 1 sensors must transmit two kinds of data packets - a less frequent "full update" consisting of the compass, accelerometer, and gyroscope data, and in between compass cycles, a "partial" or "half" update with only accelerometer and gyroscope readings. These packets are distinguished to the gumstix board by setting a specific bit in the first byte of the packet. Thus, right away, the slave device knows how many bytes to expect from this master. Figure 4.2 shows the breakdown of each individual data packet.

As a tier 2 processing unit, the gumstix board has the information and processing power available to perform

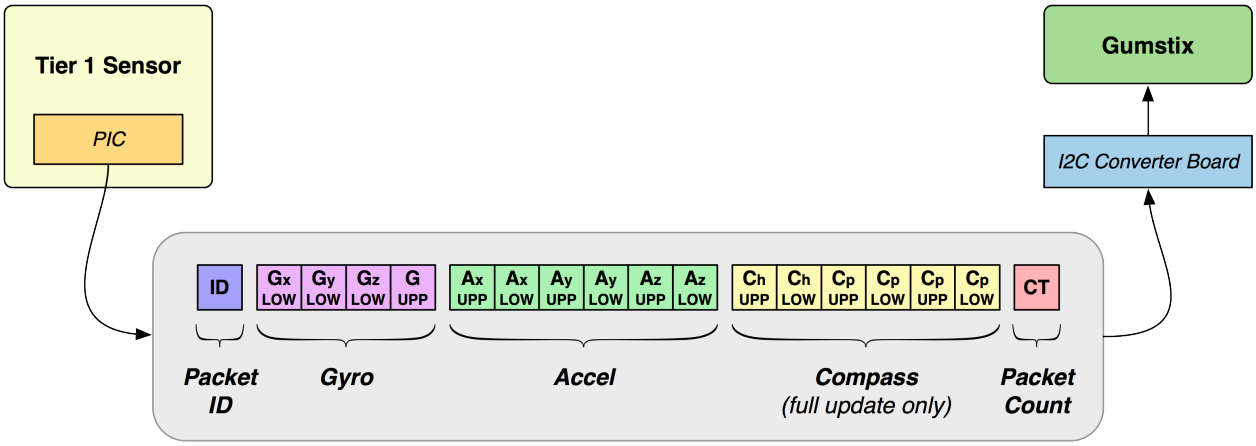

Figure 4.2: Packet structure for on-suit communication.

significant computations and/or analysis of the incoming data from multiple tier 1 nodes if necessary or desired. However, for the experiments in this thesis, the gumstix board is mainly used for processing, organizing, and storing the orientation information of each individual tier 1 node.

4.3 Motion Capture Algorithm

The primary device for finding each tier 1 sensor's orientation is the digital compass. The compass is far more accurate and reliable than either of the other inertial measurement units available, and requires no conversion or comparison with past data because it gives the absolute orientation of the sensor. Unfortunately, the maximum speed for the digital compass on each sensor is 10Hz, which for many real-time applications is an insufficient update rate. In addition, even simple, brief movements can be missed or misrepresented by relying on only the compass information.

In order to improve this performance, the system uses the gyroscope to provide some estimate of the orientation during the intermittent time between compass updates. Because the gyroscopes measure angular velocity in all three directions, a reasonable approximation of the change of the orientation angles can be constructed from the gyroscope data, and at a much faster sampling rate than the compass. The process for maintaining the orientation state of each sensor is to convert each gyroscope sample of angular velocity into the proper frame of reference; that is, from *xyz* angles to heading-pitch-roll angles. Then since these values represent velocity, we integrate the values once to find the change in angular position, and add these changes to the existing state of the sensor.

Because the compass data is the most accurate due to drift and other errors associated with repeated integrations, each compass update will correct the sensor's state by simply overwriting the stored orientation

with the compass data. Future intermittent gyroscope samples will then be accumulated into this new stored state.

4.4 Sensor Orientation Representation

The state of each sensor can be calculated in a number of ways; in this design the state is determined in terms of the heading, pitch, and roll angles as calculated at each update. Each limb of the e-textile is then described visually according to the orientation of the sensor on that limb. This is more synergistic with the compass than quaternions or *xyz*-coordinates, since the compass information can be read directly as a state update. In addition, each local gyroscope update can be easily converted to changes in heading, pitch, and roll angles, as discussed below. The state of each sensor is then estimated by accumulating the angle changes from incoming gyroscope data, until another compass update is read.

Sample	Compass Update?	Sensor State
1	Yes	c_1
2	No	$c_1 + \int g_2$
3	No	$c_1 + \int g_2 + \int g_3$
4	Yes	c_4
5	No	$c_4 + \int g_5$

Table 4.1: Example state calculations, assuming 30Hz data rate and 10Hz compass rate. In this table, c_i and g_i are the compass reading and the gyroscope data, respectively, in sample i.

The default setting for each digital compass is level orientation, meaning each angle is reported relative to a level orientation (heading, pitch and roll of 0°). This can be changed in real-time to the "upright front" orientation if necessary to avoid gimbal lock. In order to support the potential to switch back and forth between level and upright orientations in real-time, the Tier 2 sensor maintains the state of each Tier 1 sensor in both frames of reference, and chooses which set of orientation angles to report to the top-level application for a given Tier 1 sensor based on a bit set in the id byte of the most recent data received from that sensor.

Each sensor's orientation state is stored on the gumstix central processing unit. This eliminates the need for any system-level calculations or storage on the tier 1 sensors. This keeps the local nodes in line with our stated goal, which is that their only purpose should be for harvesting and transmitting data.

4.5 Mapping Gyroscope Data to Orientation Angle Changes

A key issue with the strategy outlined above is the difference between what the gyroscopes measure (angular velocity about the x, y, and z axes of the gyroscope chip itself) and what information is needed from this data (the change in position, in terms of heading, pitch, and roll angles). Especially troublesome is that the axes of rotation about which the velocity is calculated change depending on the existing orientation of the sensor board. Figure 4.3 shows this problem.

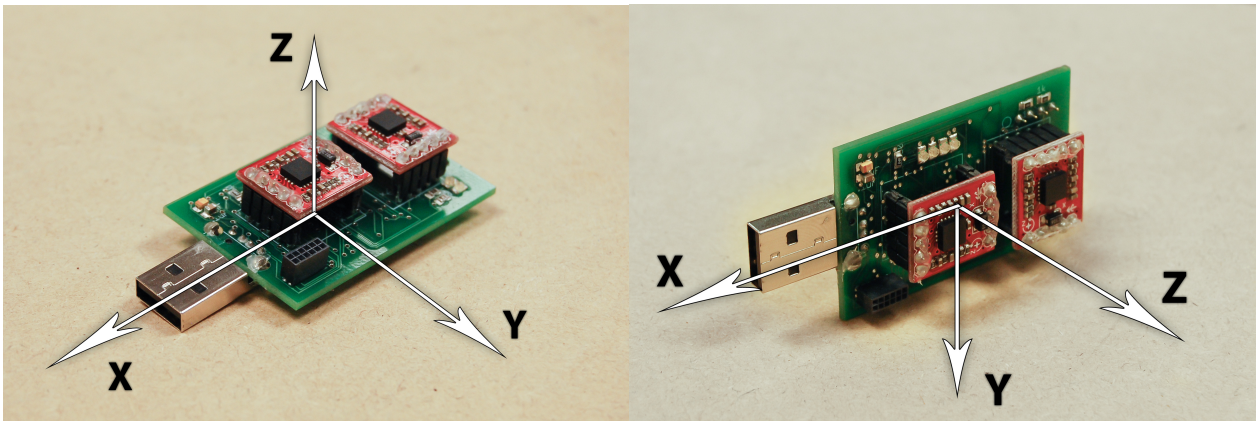

Figure 4.3: Depending on the sensor's orientation, local axes of rotation can have vastly different meanings in terms of absolute position.

One constant, however, is the key to an accurate mapping of one set of data to the other. The roll angle of the sensor board, as it is currently designed, will always correspond to rotation about the gyroscope's x-axis regardless of orientation. This is because the roll is the "twist" angle of the sensor. In order to change the sensor board's roll angle, the board must be rotated about the vector bisecting the board, and this vector will always lie along the x-axis of the gyroscope due to how the chip is mounted.

Given this constraint, we can see that the roll angle effectively describes how the y- and z-axes of the gyroscope are twisted in relation to the board itself. In other words, the changes in heading and pitch described by each gyroscope update are contained in the angular velocities about the y- and z-axes, but which rotation corresponds to which orientation angle is a function of the roll.

4.5.1 Derivation of Angle Change Magnitudes from Angular Velocity

Assume that the sensor board is free to rotate about its local x- and y-axes, but that the z-axis is locked in place. As discussed previously, rotation about the x-axis will represent the angle by which the board is

twisted about itself, which is the roll angle. Thus the roll of the sensor can clearly be freely changed.

However, quantifying rotation about the y-axis in terms of orientation angles is not so straightforward. It is simple to demonstrate that if the sensor is "level" (roll of 0°, or 180° inverted) that a rotation about the y-axis will change the pitch angle, and not change the heading angle at all. A quick verification of this property is shown in Figure 4.4. But suppose the board is rotated about its x-axis until the roll reaches 90°. Visually, this would mean the board is tilted onto its side. Now the local y-axis has rotated 90° as well. Movement about the local y-axis will now be reflected in the heading angle, as demonstrated visually in Figure 4.3 and in the experiment depicted in Figure 4.5. It is evident, then, that by changing the roll of the sensor, one changes how a given y-axis rotation is decomposed into pitch and heading angles.

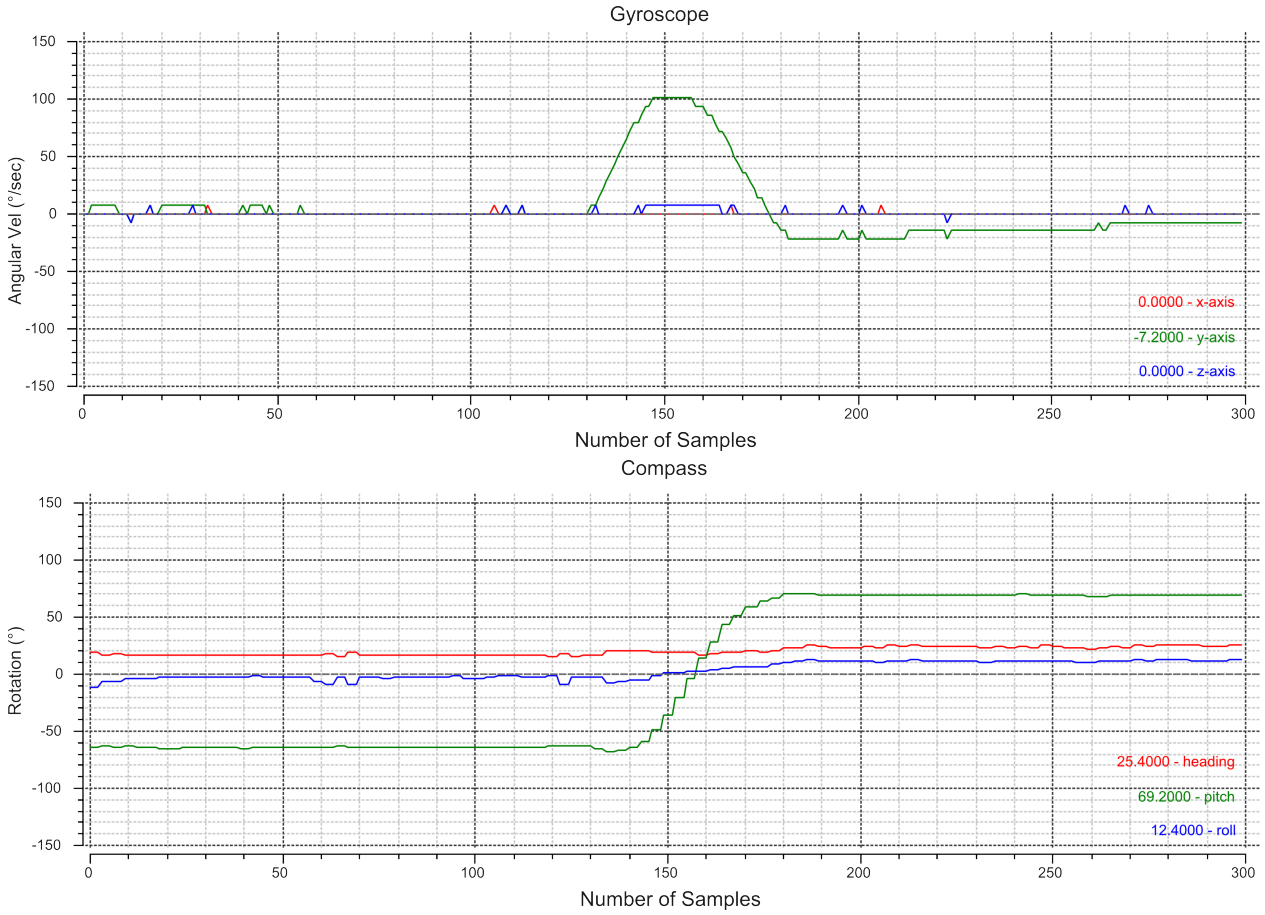

Figure 4.4: When the roll is at 0°, rotation about the y-axis changes only pitch.

For roll angles not so conveniently at right angles, the decomposition of a y-axis rotation into pitch and heading angles can be represented by a trigonomentric function. Knowing that the magnitude of the changes in the pitch and heading angles due to the y-axis must sum to the total number of degrees rotated about the y-axis, the individual orientation angle changes can be calculated as:

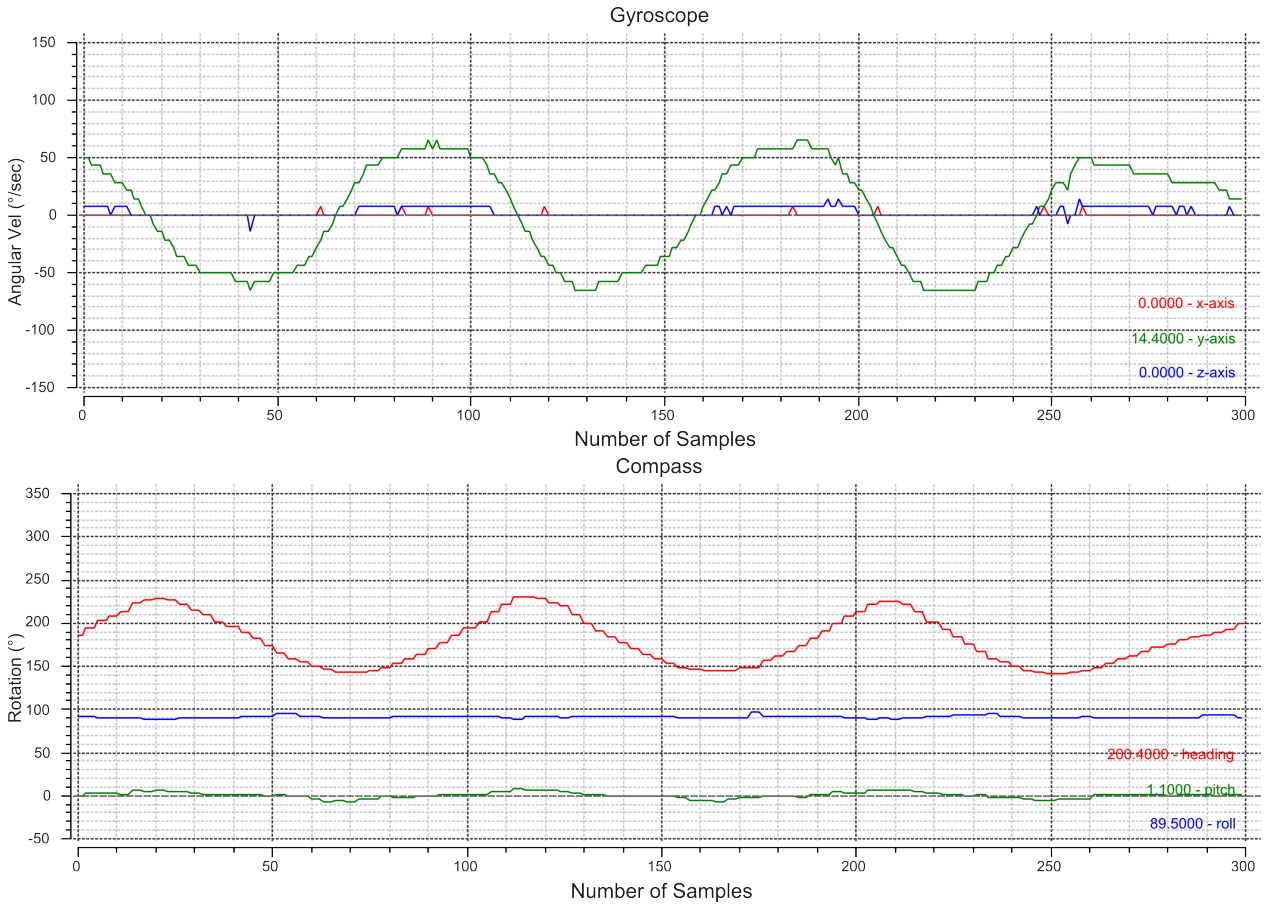

Figure 4.5: When the roll is at 90°, rotation about the y-axis changes only heading.

$$|\text{pitch}_y| = \text{rot}_y cos^2\theta \tag{4.1a}$$

$$|\text{heading}_y| = \text{rot}_y sin^2\theta \tag{4.1b}$$

where θ is the roll angle, rot_y is the total rotation about the y-axis as calculated by integrating the gyroscope reading about that axis, and $|\text{pitch}_y|$ and $|\text{heading}_y|$ are the magnitude of the pitch and heading angle changes due to that rotation, respectively. Note that if the two equations are added together, the result is

$$\begin{aligned} |\text{pitch}_y| + |\text{heading}_y| &= \text{rot}_y cos^2\theta + \text{rot}_y sin^2\theta \\ &= \text{rot}_y(cos^2\theta + sin^2\theta) \\ &= \text{rot}_y \end{aligned} \tag{4.2}$$

which is expected because, as mentioned above, the magnitude of the pitch and heading contributions for the y-axis must add up to the entire rotation angle about the y-axis.

As a demonstration of this principle, at a roll of 45°, the expected orientation change with a rotation about either the gyroscope's y-axis or z-axis would be an equal split across the heading and pitch orientations, since $cos^2(45°) = sin^2(45°) = 0.5$. This even division of the angle contributions is demonstrated in Figure 4.6.

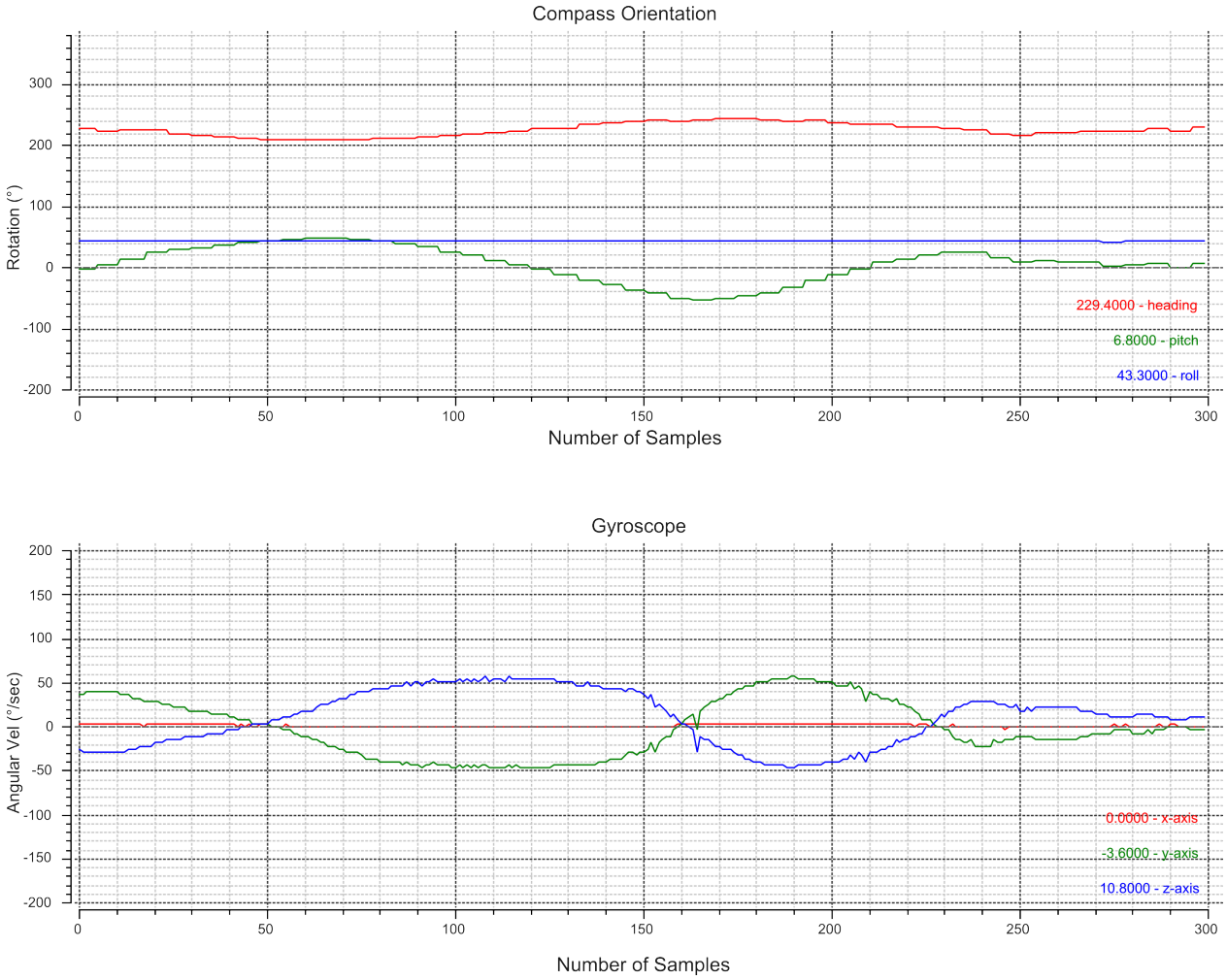

Figure 4.6: When the roll is at 45°, a pitch change is reflected in the y- and z-axes in equal amounts.

This derivation thus far has focused on only a hypothetical rotation about the sensor's y-axis. However, the same principles can be applied to the z-axis rotations as well. The only difference is that the magnitude of pitch will be related to the sine of the roll angle, and the magnitude of heading will depend on its cosine (the opposite of the y-axis calculations).

$$|\text{pitch}_z| = \text{rot}_z sin^2\theta \tag{4.3a}$$

$$|\text{heading}_z| = \text{rot}_z cos^2\theta \tag{4.3b}$$

The final step is to combine the orientation angle changes from y-axis and z-axis rotations. This is done simply by adding together the contributions to get the magnitude of the total angle change.

$$|\Delta_{pitch}| = |\text{pitch}_y + \text{pitch}_z| = \text{rot}_y cos^2\theta + \text{rot}_z sin^2\theta \tag{4.4a}$$

$$|\Delta_{heading}| = |\text{heading}_y| + |\text{heading}_z| = \text{rot}_y sin^2\theta + \text{rot}_z cos^2\theta \tag{4.4b}$$

4.5.2 Direction Multipliers

However, finding the magnitude of the angle changes is only part of the solution. As the sensor's roll changes, it becomes necessary to negate some of the gyroscope readings to keep them in line with the frame of reference of the orientation angles. Since negation of a rotation is equivalent to a change of direction, the terms of the pitch and heading equations must be negated when the local gyroscope readings are reversed due to the roll of the sensor.

For example, when the sensor board is level and "upright" (roll of 0°), a positive rotation about the y-axis will translate to a positive change in pitch–exactly what is needed. However, when the sensor's roll is 180° ("upside down" to an observer), the gyroscope's frame of reference has been inverted, so that a rotation about the y-axis that is read as positive will actually result in a negative pitch. In order to correct for situations like this, each angle contribution must be multiplied by a direction term that is a function of the roll angle.

$$\rho_1 = \begin{cases} 1 & \text{if } cos(\text{roll}) > 0 \\ -1 & \text{otherwise} \end{cases} \tag{4.5a}$$

$$\rho_2 = \begin{cases} 1 & \text{if } sin(\text{roll}) > 0 \\ -1 & \text{otherwise} \end{cases} \tag{4.5b}$$

Each term in Equation 4.4 is multiplied by one of these two direction multipliers. The multipliers reflect the quadrants the roll angle lies in when a specific rotation is positive or negative. For example, when the roll angle is between -90° and 90°, a positive rotation about the y-axis will contribute a positive value to pitch, because the board is right side up. When the roll angle is less than -90° or greater than 90°, the board is upside down and any positive reading from the gyroscope about the y-axis needs to be reversed to yield the correct negative pitch adjustment. Based on this information, the term for the y-axis rotation contribuition in Δ_p should be positive when cos(roll) is positive, so this term's multiplier is ρ_1.

The finalized equations for updating the orientation state from the gyroscope are thus:

$$\Delta_{roll} = rot_x \tag{4.6a}$$

$$\Delta_{pitch} = \rho_1 rot_y cos^2(\text{roll}) - \rho_2 rot_z sin^2(\text{roll}) \tag{4.6b}$$

$$\Delta_{heading} = \rho_2 rot_y sin^2(\text{roll}) + \rho_1 rot_z cos^2(\text{roll}) \tag{4.6c}$$

Once Δ_{pitch} and $\Delta_{heading}$ have been calculated, they are simply added to the existing state's pitch and heading, respectively. The rotation about the x-axis is separately integrated and added to the existing roll angle. The accumulated heading, pitch, and roll angles then become the new stored state for this sensor, until another data packet is received.

4.6 Kalman Filtering

After the updated heading, pitch, and roll have been computed, to further enhance the quality of the results, the new state information is processed by a Kalman filter. The Tier 2 sensor maintains the Kalman filter state information for each individual Tier 1 sensor.

Kalman filters (see Chapter 2.2) attempt to account for random variations in the measurements of a system. In this specific design, for instance, the purpose is to smooth out the individual orientation and position changes that occur with each new data packet. Because position is based on only orientation, in many cases even small fluctuations in the data can produce large changes in position. The Kalman filter will produce its best estimate of the state of this sensor, which draws from both the existing previous orientation state, and the new state as calculated from the gyroscope data.

There are some issues associated with including the Kalman filter. The covariance of the process noise and measurement noise is difficult to calculate with any certainty, especially because it may vary with what kinds of movements the sensor is experiencing. For example, measurement noise covariance may be low when the compass and gyroscope are still or moving slowly and steadily, but brief, sharp motions may increase the error covariance.

Choosing suitable noise covariance parameters is a non-trivial process. If the measurement noise covariances are chosen to be very small, the filter will weight the measured data more heavily, negating the entire reason for including the filter. If the measurement noise covariance is realtively high, the filter will resist all but the most gradual changes in state, which will manifest as lag as over time as the filter tries to "catch up" with the measurements. Figure 4.7 shows the effect of altering one of the noise components on the filter performance.

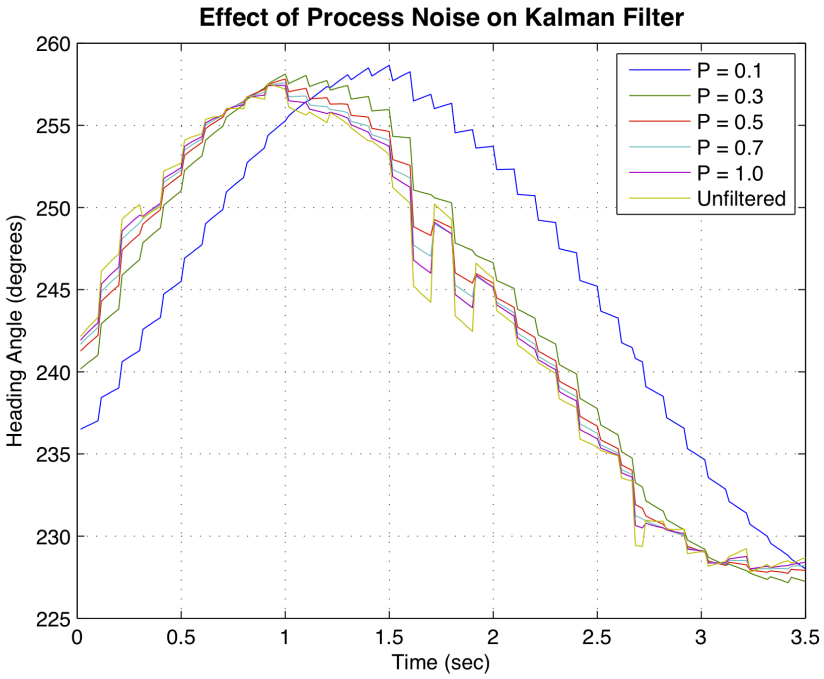

Figure 4.7: Increasing the process noise (P) causes the sensor to weight measured values more heavily.

4.7 Converting Orientation to Position Vectors

Having demonstrated the capability to maintain the orientation of individual tier 1 sensors using only the compass with periodic gyroscope updates, the goal then becomes to convert the orientation angles for each sensor into a unit position vector describing the direction of that sensor. If this can be accomplished, all

that remains for a working motion capture system is to set the proper length for these vectors, and, most importantly, organize the vectors in space relative to one another to correspond to actual limb positions and movements.

One important difference between orientation and position is that when determining the *xyz*-coordinates of the unit vector, the roll angle will not be a factor. Because the roll represents the "twist" angle of the sensor, and twisting a vector has no effect on its position in absolute space, the roll angle can be completely ignored when computing position of the vector. This leaves the heading and pitch angles to determine the position in 3-space. Similar to the way spherical coordinates describe a direction with only two angles[21], the unit position vector can be calculated as [*cos*(heading) * *cos*(pitch), *sin*(heading) * *cos*(pitch), *sin*(pitch)]. In this representation, the heading angle is much like the θ angle in a polar coordinate system-it is the angle around the z-axis. The pitch angle then corresponds to how far "up" the z-axis to rotate the position vector.

There is one specific exception to this method–the torso sensor. Because the single sensor on the torso describes the shape of a rectangle, not a vector, the roll must be considered when orienting the torso. Just as twisting the trunk of one's body causes the shoulders (and all attached limbs) to rotate in space, the roll of the torso sensor, in addition to the heading and pitch angles, will have an impact on the full-body motion capture result. This is apparent when animating the body limb by limb, as discussed in the next section.

4.8 Animation Programming

Since the system described in this thesis is designed for usage with e-textiles, animation of a body (comprised of limb segments) is a highly desirable application. Partially to demonstrate the design's feasibility as a wearable motion capture system, and also to serve as a test bench for various on-suit communication experiments, an OpenGL software program was developed to handle and animate the processed data from the gumstix board. The animation program receives the orientation angles delivered by each individual tier 1 sensor from the gumstix board and performs the conversion steps outlined above to create an *xyz* vector for the corresponding limb.

The animation algorithm results in a simple stick figure representation which is constantly refreshed by new incoming data from the gumstix. Rendering the figure begins with a simple rectangular prism centered at the origin; this represents the wearer's torso and has a predefined length, width, and height. The torso is then rotated according to the heading, pitch, and roll angles of the torso sensor. Following these rotations, the coordinates of the four joints corresponding to the shoulders and hips are calculated from the same set

of angles.

The limbs of the figure are arranged from inside out. Working within the constraint that limbs must remain attached to their body joints, each limb is drawn using the previous limb's endpoint as its own origin. For instance, the left shoulder joint location is used as the origin for plotting the left upper arm. Referring to this point as p_{LS}, and the *xyz*-coordinate vector of the left upper arm as v_{LU}, with the left upper arm having pre-determined length L, the left upper arm limb can be drawn simply as a line segment between the point p_{LS} and the point p_{LS} + (L * v_{LU}). Denoting this endpoint, which corresponds to the wearer's left elbow, as p_{LE}, it is clear that this point will then be the starting point for the rendering of the left lower arm, or left forearm. The left forearm is a segment drawn from p_{LE} to the point resulting in summing p_{LE} with the appropriately scaled left forearm *xyz*-vector.

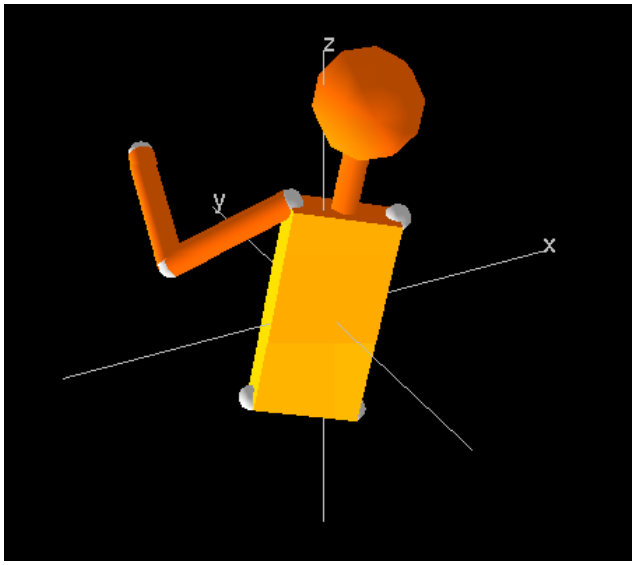

Figure 4.8: OpenGL animation program, showing the torso and left arm.

As shown in Figure 4.8, this process results in an animated figure that adheres to the constraints of the body (how limbs are known to be connected) without processing the data from multiple limbs together. This allows the design to stay flexible as more sensors are added, as long as the program is provided with the location and connection structure of the "new" body.

Chapter 5

Results

This chapter summarizes the experiments performed to validate the hardware and software design described in the previous chapters. First, tests are carried out to show the Tier 1 sensor's capability to measure static orientations. Next, experiments are performed comparing the Tier 1 sensor's performance to that of an industrial optical motion capture system. Next, the improvement in performance by using the Kalman filter is demonstrated. Finally, an acceleration calibration procedure is discussed and its success is quantified.

5.1 Static Sensor Performance

The tests in this section demonstrate the effectiveness of the Tier 1 sensors at measuring static orientations. These experiments are made repeatable by using mechanical setups to mark known positions and angle changes. The results are evaluated based on the error between the sensor's measured orientation and the correct predetermined orientation. Since the accelerometer and gyroscopes measure position and angle changes, static sensor tests can only demonstrate the ability of the compass to provide accurate orientation data. While these tests are somewhat limited, proof of this capability is extremely important because the compass's role is to correct errors the system incurs by using the other inertial measurement units. An analysis of compass performance in static positions should show how true these corrections will be.

5.1.1 Limb representation with a single sensor

The simplest way to test a single sensor's performance is to attach the sensor to a mechanical apparatus that can be rotated freely. To achieve this, one Tier 1 sensor board was affixed to a bike wheel. This allowed one orientation angle (in this case, the pitch) to be varied as the sensor was placed in a number of designated positions. In order to measure the effectiveness of the sensor, each measured orientation was converted to a unit vector and compared to the expected unit vector for this position in space. The expected unit vector at each step is determined by adding the amount of change in the pitch angle in each step to the initial compass reading. The error is quantified by the angle (in degrees) between the measured and expected unit vector. The results for this experiment are shown in Table 5.1.

Table 5.1: Static position measuremnts for a single sensor through a series of pitch angle rotations.

Pitch Angle	Measured Orientation			Measured Unit Vector			Expected Unit Vector			Error (deg)
Change	Heading	Pitch	Roll	x	y	z	x	y	z	
start	33.3843	-0.2792	2.4506	0.835	0.550	-0.005	0.835	0.550	-0.005	n/a
30	44.043	29.8458	3.727	0.623	0.603	0.498	0.725	0.478	0.496	9.25
60	47.5204	60.2271	3.1907	0.335	0.366	0.868	0.421	0.277	0.864	7.08
90	144.1808	88.9598	97.6568	-0.015	0.011	1.000	0.004	0.003	1.000	1.17
-30	25.7323	-30.8851	3.0449	0.773	0.373	-0.513	0.721	0.475	-0.504	6.61
-60	18.8457	-61.2368	5.0966	0.455	0.155	-0.877	0.414	0.273	-0.868	7.15
-90	249.8354	-87.149	131.6839	-0.017	-0.047	-0.999	-0.004	-0.003	-1.000	2.63

A representation of these results in plot form is shown in Figure 5.1. Note that as the pitch angle approaches $\pm 90°$, the heading and roll angles sometimes undergo drastic changes. This is reasonable because at these extreme points even small errors can cause large variations in the heading and roll. Even with these changes, however, the error angle between the expected and measured orientations remains low, with a worst-case error of under $10°$. This test has demonstrated that the digital compass can provide an accurate measure of orientation, and by extension, track the orientation of a limb when attached to the body. While the compass update rate is slow compared to the other IMUs, the data from the compass is reliable enough to correct the drift errors present in the faster sensors.

5.1.2 Joint representation with two sensors

Having measured the performance of a single Tier 1 sensor, a reasonable extension of this test is to examine the behavior of two sensors hinged together like a body joint. For the specific application of self-contained motion capture, the capability to measure the orientation of joints is extremely important. If the orientation of two limbs connected by a joint can be accurately calculated, then by iteration it is possible to know the pose of the wearer's entire body.

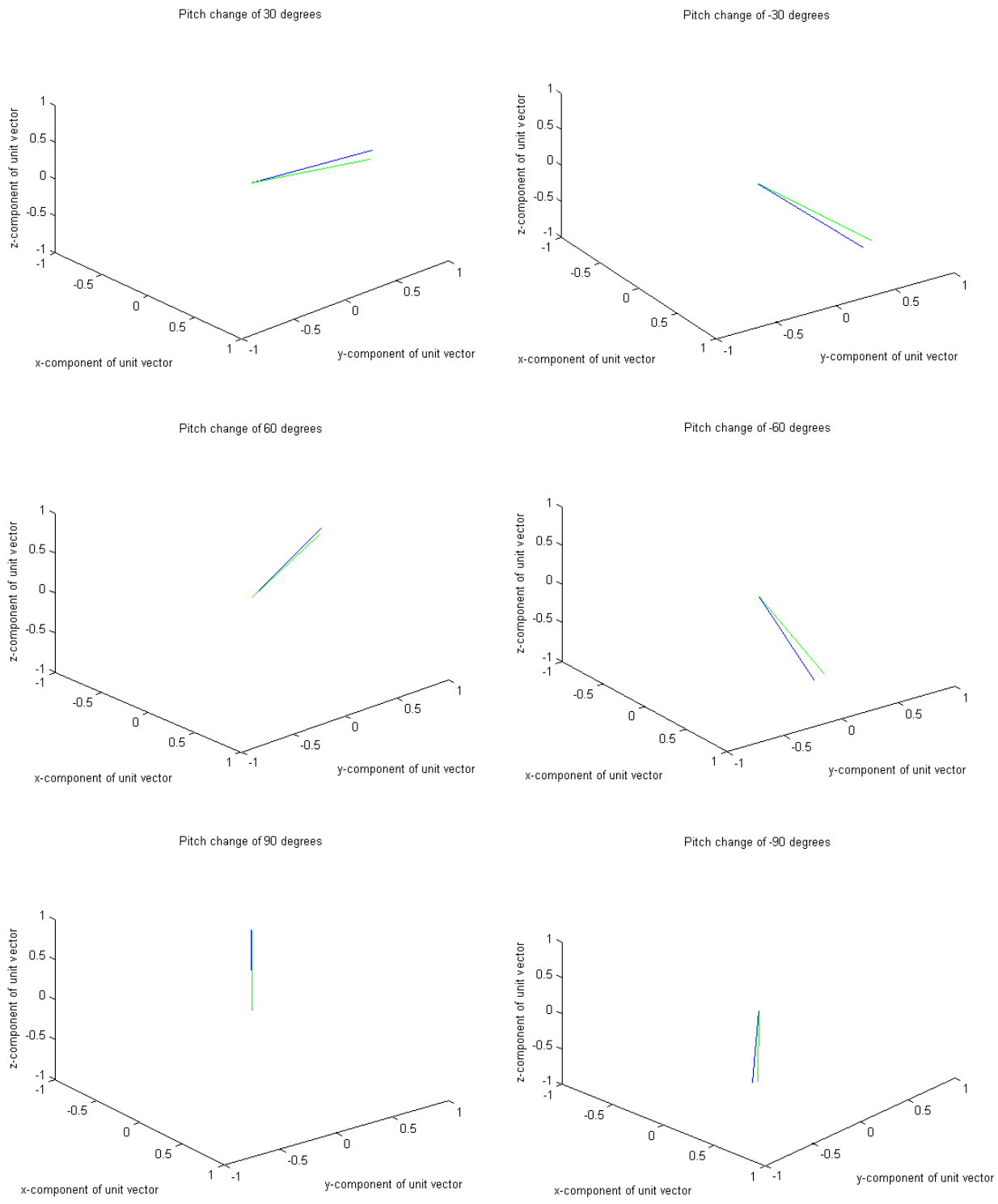

Figure 5.1: Plots of the expected (green) and measured (blue) unit vectors in the direction of sensor orientation for each test in Table 5.1.

In order to test the accuracy of the two sensors acting as a body joint, the sensor corresponding to the lower limb was placed in a number of static positions, each with a different pitch angle but with constant heading

and roll angles. The sensor tracking the upper limb of the joint was rotated in such a way as to change heading without altering pitch and roll. In order to accomodate the need to change different angles for each sensor, a test apparatus with predetermined static positions was constructed, as shown in Figure 5.2. In this test setup, the sensor corresponding to the lower limb is placed on the bike wheel as in the previous test, allowing its pitch to be changed freely. The sensor corresponding to the upper limb is placed level on a wooden block. Changing the heading of this block will also change the heading of the lower limb sensor, which is the expected behavior of a body joint.

As in the previous experiment, each measurement's accuracy is determined by the angle between the unit vector described by the measured orientation angles and the unit vector in the direction of the expected orientation. The expected orientation is calculated as a change in the heading and pitch angles from the inital compass measurement. Changing the heading of the upper limb sensor should cause an equal heading change in the lower limb sensor, while rotating the lower limb sensor about the pitch axis should have no effect on the upper limb. Therefore, to compute the expected orientation angles, the heading rotation in the upper limb is added to the initial measurement of *both* sensors; then, the pitch change in the lower limb is added to only the lower limb sensor. The resulting measurements and angle errors are shown in Table 5.2.

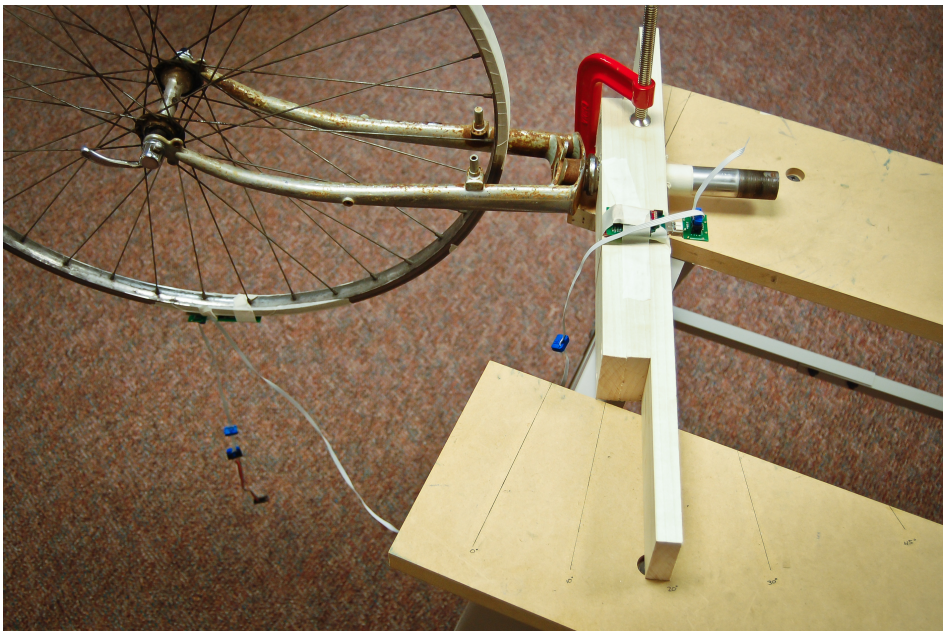

Figure 5.2: Test apparatus for static joint positions.

Each sensor performs to expectations. The upper limb sensor tracks the heading angle change with an error of less than 3° at each step. The lower limb sensor, which must accomodate both its own pitch change and a heading rotation from the movement of the upper limb, has a worst-case error of 11.4°, with typical values being less than 6°. While these orientation measurements are not exact, a 10° error in limb orientation with

Table 5.2: Static position measurements for two sensors, with heading and pitch angle changes.

$\Delta_{heading}$	Δ_{pitch}	Upper Limb Orientation			Error Angle	Lower Limb Orientation			Error Angle
(upper limb)	(lower limb)	Heading	Pitch	Roll	(degrees)	Heading	Pitch	Roll	(degrees)
0	0	14.0964	-4.0617	-1.387	—	179.9064	12.2932	175.7148	—
0	30	12.6373	-3.1058	-2.2009	1.7419	178.0607	40.2008	174.5672	2.5106
0	60	12.6583	-2.5307	-2.0942	2.0988	172.7363	70.2075	167.8834	3.1048
0	90	11.7497	-2.5194	-2.3023	2.8049	28.5731	79.5913	21.7161	5.8770
10	0	2.7293	-2.8301	-2.0108	1.8382	167.9652	12.1944	176.3015	1.8996
10	30	4.0148	-2.5227	-1.7026	1.5412	165.657	41.5103	174.5921	3.2579
10	60	3.1358	-3.0978	-2.1029	1.3595	160.0861	70.1945	168.1322	3.7846
10	90	2.9238	-2.6233	-1.949	1.8545	16.6685	79.6689	21.9547	5.5432
20	0	353.5481	-2.7211	-2.0952	1.4480	157.0912	12.1015	176.1517	2.7583
20	30	354.7053	-2.6312	-1.9033	1.5543	153.7671	41.5985	174.586	4.6178
20	60	354.2732	-2.8223	-1.9108	1.2519	148.1615	69.9072	167.4993	4.4794
20	90	353.7203	-2.5183	-2.0886	1.5884	5.0684	79.5624	21.8581	5.2434
30	0	345.7802	-2.6358	-1.899	2.2042	145.4937	11.9739	176.2	4.3259
30	30	346.0423	-2.4791	-2.0007	2.5057	142.9298	39.9408	173.9087	5.7560
30	60	346.1645	-2.5145	-1.9455	2.5800	135.8981	69.8614	168.4971	5.1358
30	90	346.3323	-2.352	-1.8221	2.8118	353.1568	79.698	21.7363	4.9277
45	0	334.7688	-2.123	-2.1294	2.0517	129.221	11.4972	176.3577	10.4855
45	30	335.043	-2.376	-2.0835	1.9325	124.6499	41.3074	174.2478	11.4005
45	60	334.8798	-2.3299	-2.0965	1.9002	120.0908	69.7951	167.8922	6.8654
45	90	332.9364	-3.8205	-1.9974	1.1821	340.0725	79.6467	23.3094	4.3787

a limb length of 1 ft (30.48 cm) will have an absolute position error of 2.09 in (5.31 cm). This should be adequate for all but the most precise full-body motion capture applications.

5.2 Optical Motion Capture Tests

In order to demonstrate the motion capture capabilities of the Tier 1 sensors, an experiment was performed comparing the data from a single sensor to data produced by the Qualisys optical motion capture system. The Qualisys system uses an array of cameras strategically placed about a room which track a number of reflective balls. The system then records the absolute *xyz* coordinates in space for each of the balls over a predetermined length of time.

5.2.1 Motivation and Methodology

The purpose of these tests is to evaluate the accuracy of a Tier 1 sensor with regards to motion capture. Unlike the tests done in static positions, these experiments can demonstrate how well the gyroscope performs at tracking change in orientation in between compass updates. Additionally, the Qualisys system allows for an industry-tested, reliable baseline against which to measure the results.

For these tests, one Tier 1 sensor board was fixed on a yardstick, along with two of the reflective balls, one on either end of the sensor board about one foot apart. In order to compare the two systems' results, the

Tier 1 sensor data was converted to unit vector form and stored on the Gumstix. The optical motion capture data was converted to vector from as well, by subtracting the xyz coordinates of the two reflective balls as recorded by the system and then normalizing the resulting vector to get a unit vector in the direction of the orientation.

In order to place the two results vectors in the same frame of reference, an offset was applied to the sensor data during the post-processing phase of the tests. The need for this offset arises from the fact that while the optical capture system has a pre-defined x-axis and y-axis in space, the Tier 1 sensor's orientation is converted to xyz components with the x-axis pointing towards magnetic north. (Thus, a heading of 0° corresponds to an orientation vector in the direction of the x-axis.) In order to correct for this, the heading of the Tier 1 sensor was taken facing the Qualisys system's positive x-axis. This value was then subtracted from each sensor reading to match up the xy-planes.

The results for these tests were measured in terms of the error angle between the two vectors. The reason for using this measure, instead of others such as error in xyz positions, is that error in the vectors can sometimes be misleading, especially near extreme positions of the vector. If one component of the vector is dominant, even small discrepancies can register as >100% error. By contrast, the error angle is more descriptive of exactly how far off one measure is from the other, especially because the raw sensor data will be recorded in orientation angles. The error in the motion capture system is between 1-2mm [22]. At a distance of 1 ft (30.48 cm) between the two optical balls, this translates to a worst-case error of about 0.752° that can be present in the optical motion capture system in any given measurement.

5.2.2 Pitch Change Test

The first test conducted was a simple change in pitch, beginning at zero degrees and increasing, then returning back to the level position. This test lasted 15 seconds and was conducted at a data rate of 60Hz. This means each between each orientation correct supplied by the compass, the Tier 1 sensor relies on five gyroscope updates to track orientation changes.

Figure 5.3 shows the results of this test. The tracking is accurate within 20° throughout the duration of the sample. However, the performance is noticably better when the z-coordinate, corresponding to pitch, is more dominant. Because the heading angle determines the sensor's estimate of both x and y, this could mean that the sensor was slightly tilted, or it could mean that the compass's estimation of the heading was slightly inaccurate.

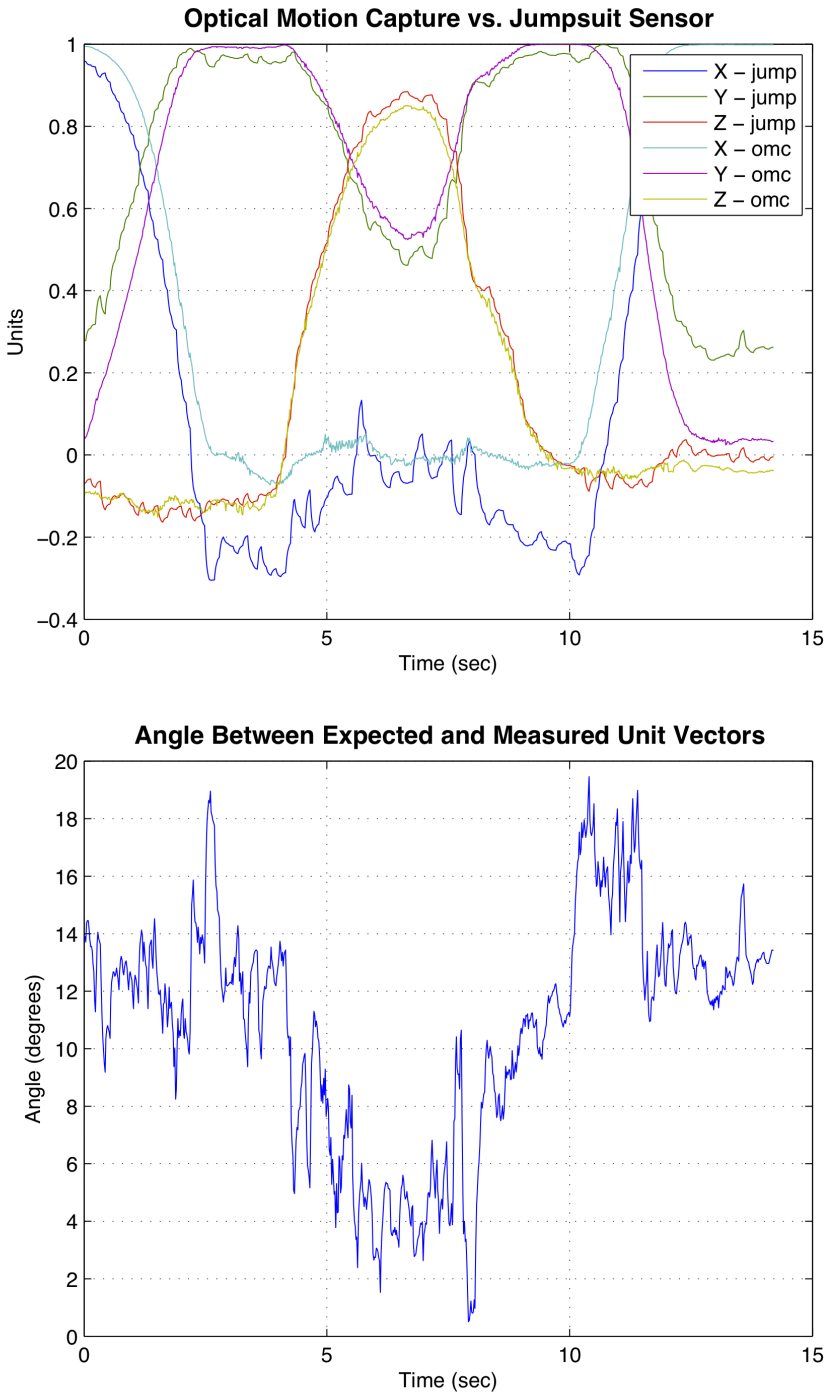

Figure 5.3: Test of a simple pitching motion against the Qualisys optical motion capture system.

5.2.3 Heading Change Test

Another experiment performed was a series of repeated heading changes while holding the yardstick level, over a full minute time period. This test was designed to detect any potential transient errors that might

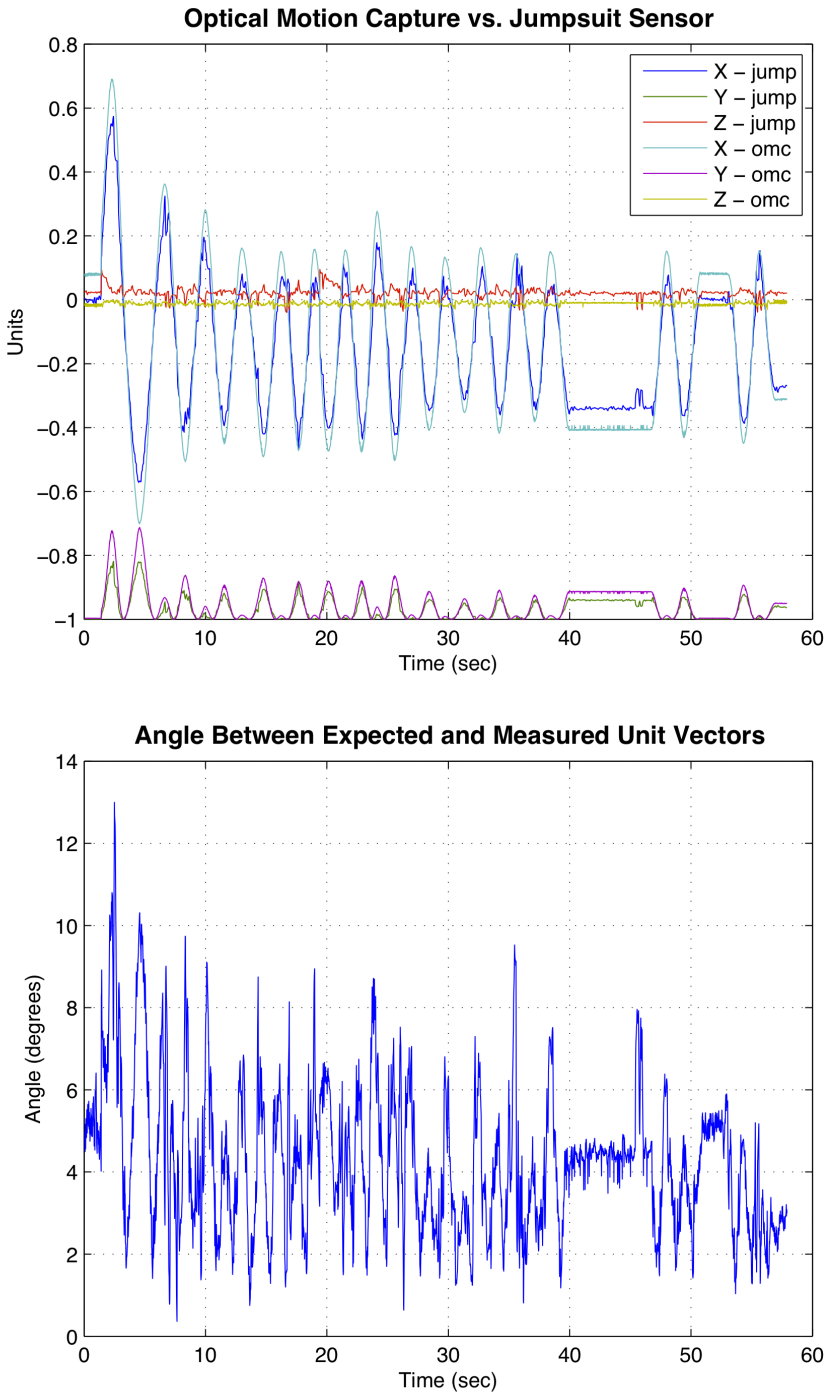

Figure 5.4: Longer duration test of heading change against the Qualisys system.

not be as apparent in a shorter test, and to demonstrate the effectiveness of the sensor motion capture algorithm over a longer range of time. The yardstick was held level, meaning pitch was approximately 0° for the duration of this test. This test was also performed with a data rate of 60Hz. The vectors and error angles for this test can be seen in Figure 5.4.

While the shapes of the two sets of data match up well, the Tier 1 sensor's x- and y-axis magnitudes do not reach the peaks of the optical motion capture system. Comparing the z-axis from each motion capture system shows that the sensor has a small positive value throughout the duration of the test, whereas the optical motion capture data reads almost exactly zero. This added value to the sensor's z-axis component will result in the x- and y-axis components being slightly smaller to preserve the unit length. The error angle shown in Figure 5.4 is almost certainly the result of these slight discrepancies.

5.2.4 Heading and Pitch Change Test

The motions captured in this test were a heading rotation in the xy-plane followed by a rotation upwards to change the pitch. This test was designed to show different kinds of motions within the same sample to verify that the orientation tracking algorithm can handle a variety of movements. The resulting motion capture comparison and error angle plots are in Figure 5.5. The worst case error for this experiment was about 12°, which corresponds to an error of about 2.5 inches (6.35 cm) for a limb of length 1 ft (30.48 cm).

5.3 Kalman Filter Effectiveness

The purpose of the test in this section was to demonstrate the effect of Kalman filtering on the motion capture data. The specific implementation of the Kalman filter is discussed in Chapter 4. The desired effect is an overall reduction in noise, and that the sensor be less prone to an occasional erroneous measurement. In order to measure the impact that Kalman filtering had on the motion capture algorithm, one data set from the tests alongside the optical motion capture system was processed both with and without the Kalman filter active. The graphs comparing both sets of end results to the optical motion capture data can be seen in Figure 5.6.

By simple inspection of the graphs, it is easy to see that a lot of the noise in the unfiltered data set is reduced by the Kalman filter. The error angle plot shows that the Kalman filter also results in a small but not insignificant reduction in error. Finally, at a time of about 6 seconds, there is a significant spike in the unfiltered data. The version of the data with the Kalman filter active reduces this sharp increase and avoids a greater discrepancy with the optical system. It can be concluded from this test that the Kalman filter has met all the goals of the filter, and is a worthwhile addition to the algorithm for motion capture.

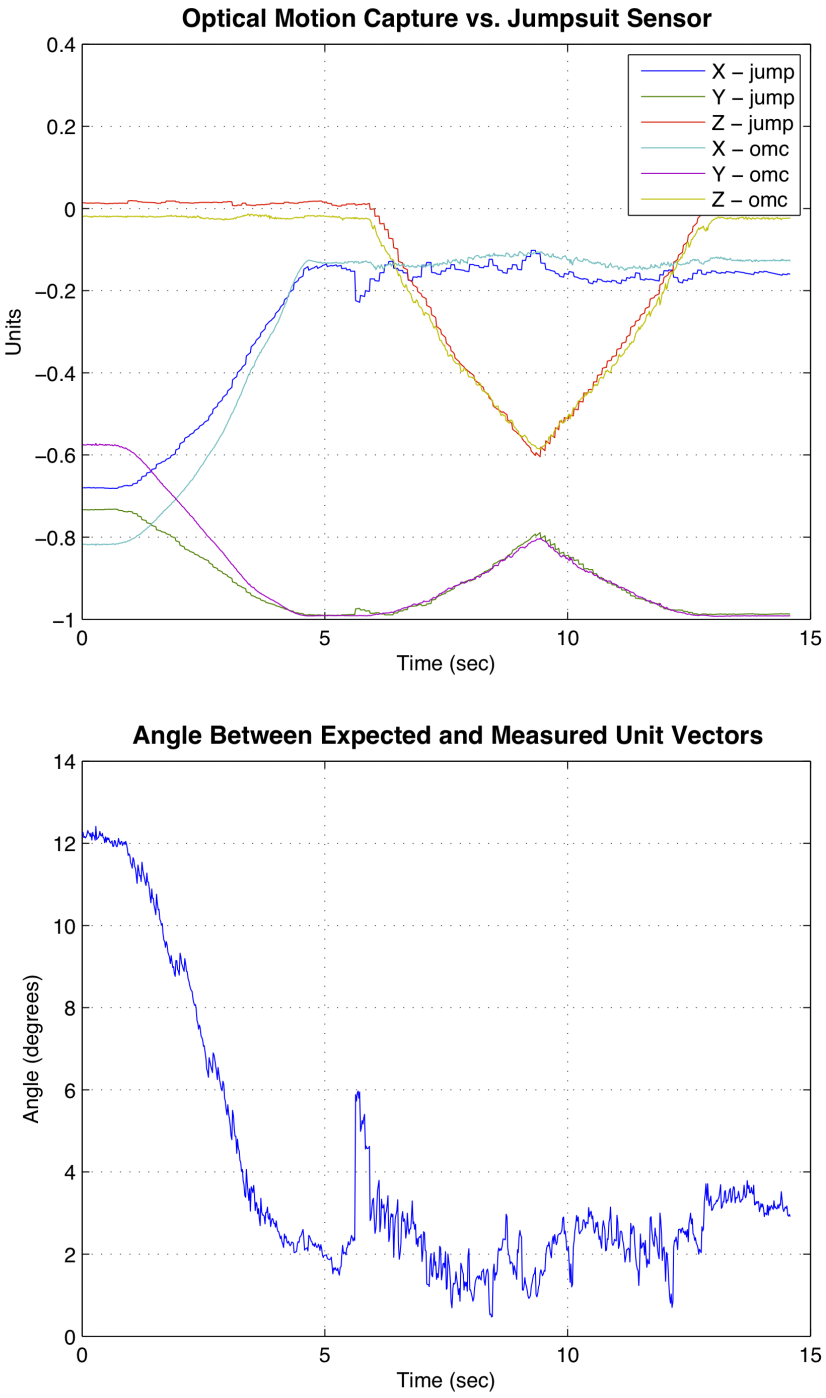

Figure 5.5: Heading motion followed by pitching motion with the Qualisys system.

5.4 Accelerometer Calibration Results

One of the stated goals of this thesis was to establish a hardware platform suitable for a wide variety of motion capture and e-textile applications. While the accelerometer was not a necessary component for

the orientation tracking algorithm, accelerometers are valuable for performing related tasks, such as body translation.

Due to the properties of accelerometers laid out in Chapter A.1, even slight errors in accelerometer readings can lead to much larger problems with drift, particularly when using the accelerometer to estimate velocity and position of a body or limb. The goal of the tests in this section is to explore the effect calibration can have on reducing the amount of drift error in the accelerometer.

5.4.1 Methodology

The calibration of the accelerometer sets six values: the x, y, and z sensitivities and the x, y, and z offsets (biases). The sensitivity of an accelerometer axis is the amount of acceleration described by a single digital bit. The datasheet lists these values as being typically 3.9mg, but as low as 3.5g and as high as 4.3mg. The offsets of each axis are the accelerometer output for the 0g point. These values are typically 0mg but can be as far off as $\pm$150mg for the x- and y-axes and $\pm$250mg for the z axis. While these margins may seem small, even these discrepancies can contribute to drift in a large enough quantity to lessen the accuracy of velocity and position estimates from the accelerometer.

To ensure these values were as accurate as possible, a calibration routine was developed relying on the fact that when still, an accelerometer should read an acceleration magnitude of 1g, since the only acceleration should be due to gravity. By placing the accelerometer in a variety of positions designed to explore the full range of potential gravity readings, an attempt was made to find the calibration procedure that yielded the best sensitivity and offset values. Since each accelerometer test used for calibration was done while the sensor was held still, each sample of each of these test positions contributed a term to the calculation of an error sum, in which the sensitivities and offsets of each axis were variables and error was determined to be the difference between the measured magnitude of the sample and 1g. The best values for each of these variables were determined by MATLAB's *fmincon* function, which minimizes the sum of the square of every error term. The results of this function were the calibrated sensitivites and offsets for each axis.

Four methods of calibrating the accelerometer were carried out. Once the values in question were computed, each set of values was assumed to be the "true" numbers for the accelerometer and then used to convert the accelerometer readings to actual acceleration data for a test set of different still orientations. The graph of the resulting measured accelerations across each test position is displayed with each method. The resulting sensitivites and offsets are displayed in Table 5.3. For reference, in fixed positions, the ideal result for

acceleration is $1g$.

Table 5.3: Resulting sensitivites and offsets from each of the four accelerometer calibration methods.

Calibration Method	x-axis sensitivity (mg)	y-axis sensitivity (mg)	z-axis sensitivity (mg)	x-axis offset (mg)	y-axis offset (mg)	z-axis offset (mg)
A	3.900	3.900	3.900	0.000	0.000	0.000
B	3.792	3.837	4.024	-8.716	-2.114	9.660
C	3.581	3.981	4.179	-9.689	-2.101	11.438
D	3.782	0.975	1.000	-9.158	-1.908	10.683

5.4.2 Calibration Method A: Datasheet Defaults

This section can be described as the control for this experiment, since the values used for the sensitivities and offsets were not calculated but instead taken to be the typical values of the datasheet. The results of the acceleration calculations for these values is shown in Figure 5.7. Effective calibration techniques should show improvement over these results.

5.4.3 Calibration Method B: Horizontal and Vertical Extremes

For the first experimental calibration method, the sensor was placed in both a level orientation, allowing gravity to be read as straight down or straight up, and a vertical orientation, meaning gravity was sensed along the x or y axis depending on rotation. This calibration method relies on capturing the instances of the gravity vector where the direction is most extreme. The measured accelerations of the still test cases using the sensitivites and offsets from this calibration method are in Figure 5.8.

5.4.4 Calibration Method C: Slanted Positions

In contrast to Method B, this calibration procedure places the sensor in a number of positions away from the extremes, distributing the gravity vector over all three axes. The positions used for calibration are slanted orientations facing in many different directions at several angles. The measured accelerations of the still test cases using the sensitivites and offsets from this calibration method are in Figure 5.9.

Table 5.4: Minimum, maximum, and average magnitude of the accelerometer's readings for each calibration method's results. The ideal values are $1g$.

Calibration Method	Maximum Magnitude (g)	Minimum Magnitude (g)	Average Magnitude (g)
A	1.079	0.9153	1.002
B	1.024	0.9722	1.001
C	1.071	0.9342	1.004
D	1.027	0.9748	0.9996

5.4.5 Calibration Method D: Slanted and Extreme Positions

For the final calibration method, a mix of the methods B and C was used. Method D includes some horizontal/vertical positions and some positions at an angle. There are an equal number of each kind of position, giving equal weight to each type of data. This method is intended to express the full range of potential still positions, but takes much more time than any of the other methods. The measured accelerations of the still test cases using the sensitivites and offsets from this calibration method are in Figure 5.10.

5.4.6 Analysis

The hypothesis was that either measuring on the extremes only or measuring only positions at an angle, without including the extreme cases would not be sufficient, and that the combination method, method D, would be the best method. Table 5.4 shows the worst-case magnitude error on each side of the target $1g$ mark, and the average magnitude of the still accelerations for each calibration method's sensitivites and offsets. According to these results, there is very little difference between method B and method D. While method C performs adequately at measuring specifcally slanted positions, its performance is far worse measuring horizontal or vertical orientations, with its worse-case error being more than twice that of the other non-default methods. It may be that the bias points, or axis offsets, of the accelerometer are more easily detected at the extreme positions used in method B, since two of the three axes are expected to be close to zero in these positions. This means that whatever value is measured for those axes becomes a much more substantial clue to what the offset for that axis should be. This would be one possible explanation for why the slanted orientations were of so little value in this calibration procedure.

5.4.7 Conclusions

While the software application discussed in this work did not use acceleration data, the accelerometer is an important component for other motion capture applications. Accuracy of the accelerometer in this context is

very important, because generally accelerometers are used to find information about movement in absolute space. This can be difficult to correct in real-time without outside assistance, making calibration with the intent to preemptively decrease measurement error as much as possible a high priority. These results have presented options for calibration routines for the accelerometer used in the project, as well as a potential guideline for developing others that may lead to higher accuracy.

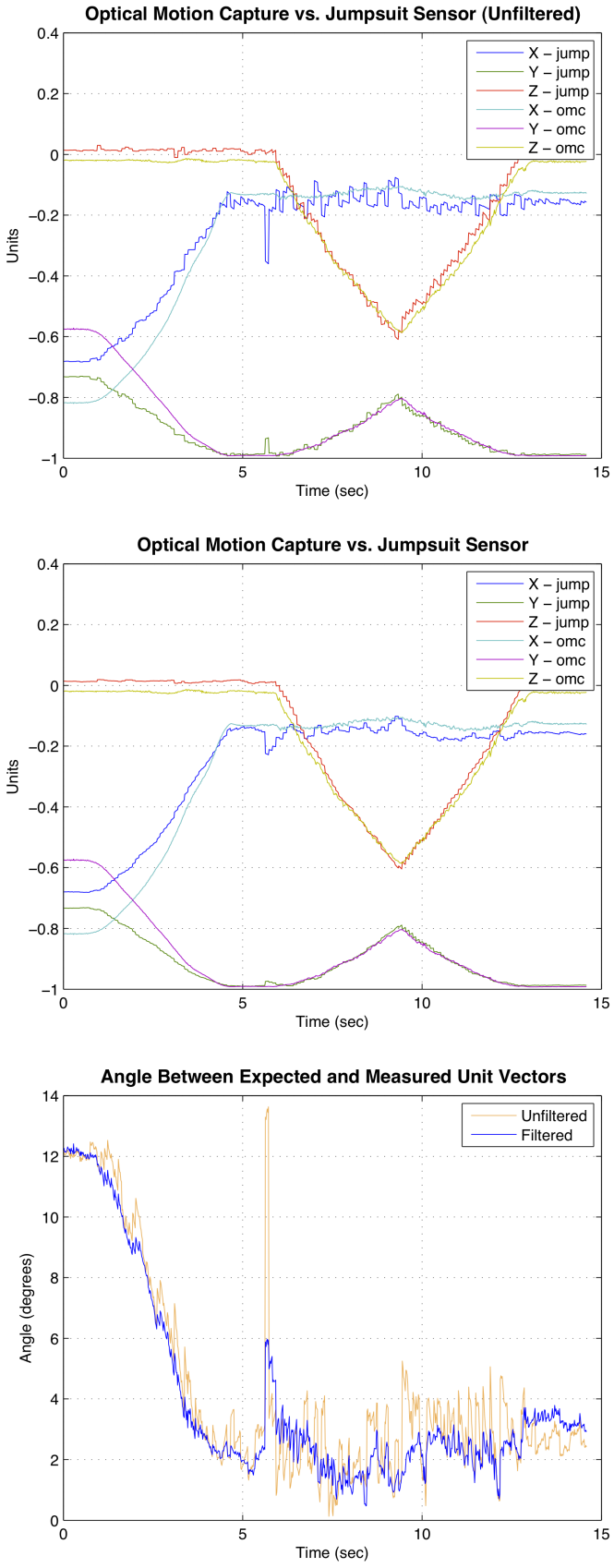

Figure 5.6: A comparison of motion capture accuracy, without the Kalman filter (top) and with the Kalman filter (middle), and a graph of the error angles for each method (bottom).

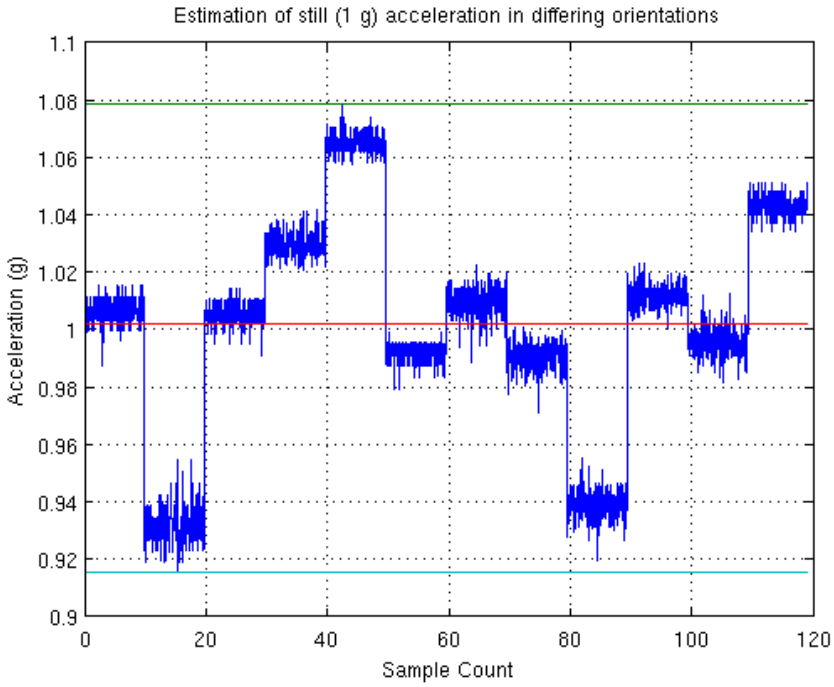

Figure 5.7: Results of still tests using calibration method A.

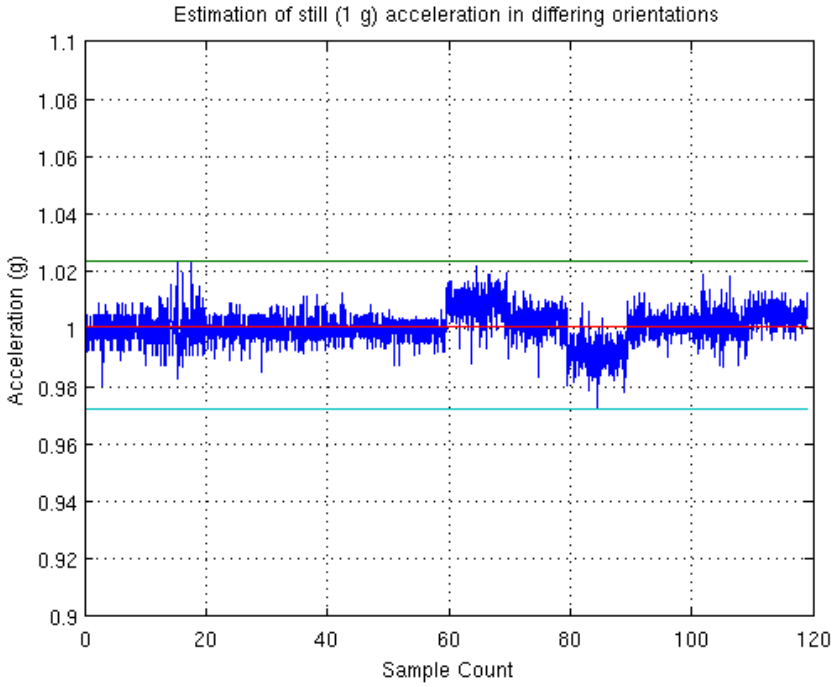

Figure 5.8: Results of still tests using calibration method B.

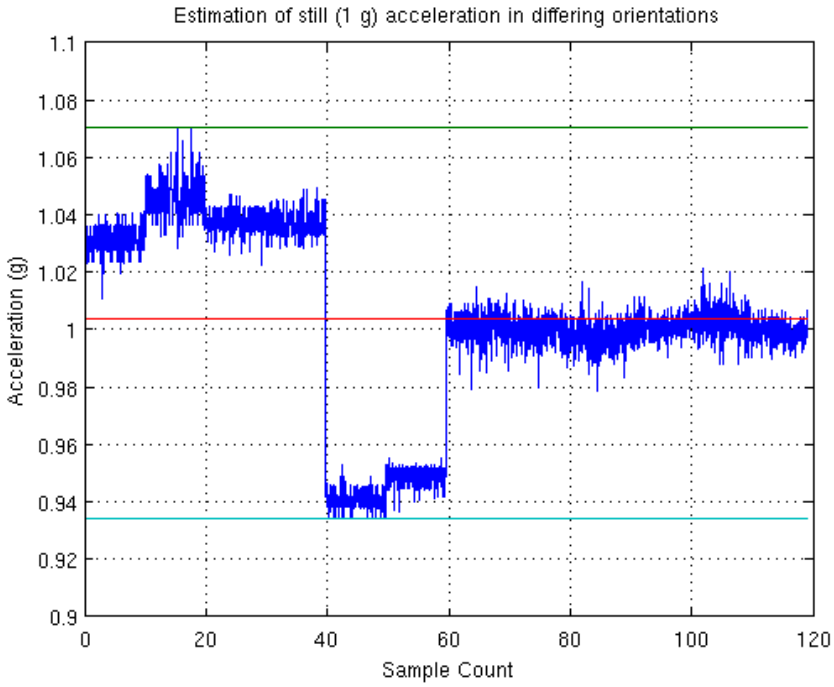

Figure 5.9: Results of still tests using calibration method C.

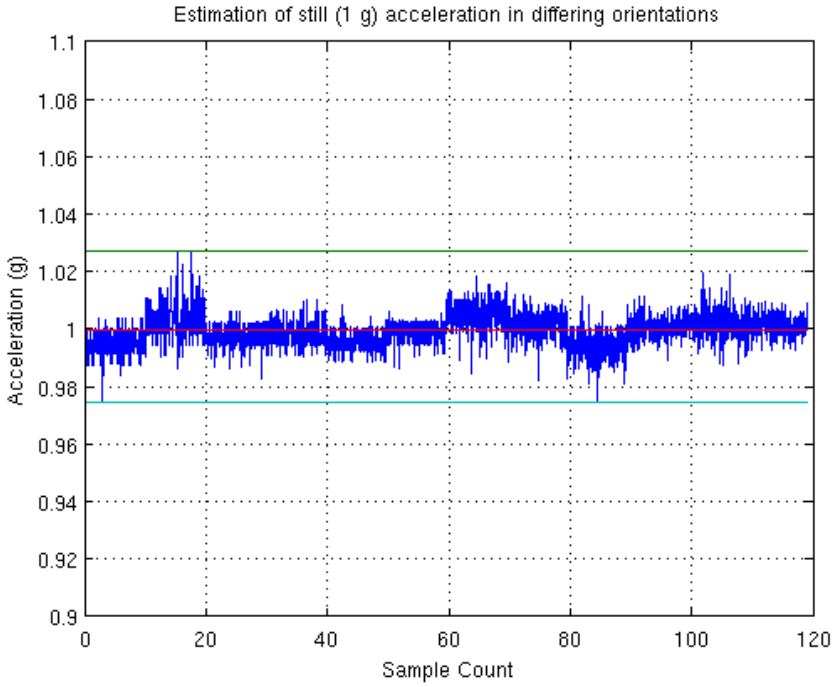

Figure 5.10: Results of still tests using calibration method D.

Chapter 6

Conclusion

This work establishes a hardware and software platform for self-contained, full body motion capture in the context of an e-textile. The system is flexible and allows for both the local sensors and the central processor to be customized for a variety of applications.

The experiments of this thesis demonstrate that local inertial measurement units are capable of accurately tracking limb and joint movements. These tests also validate the effectiveness of the algorithm combining compass and gyroscope data into an orientation state. In addition, this work has demonstrated that a calibration routine can have some effect at decreasing accelerometer error with the eventual purpose of reducing drift, though accelerometer drift will still persist in a smaller magnitude. The sum of these experiments has shown that this design is capable of accurate full body measurements compared to a commercial option for motion capture.

6.1 Future Work

One potential issue with this design is scalability. Adding additional Tier 1 sensors necessarily increases the communications capacity required on the e-textile itself, and may require adding additional Tier 2 sensors. Should this be the case, a method for communication between multiple Tier 2 sensors, each tracking and storing the data of unique Tier 1 sensors, would be necessary to assemble a complete picture of the user's body. One possible solution to this issue would be to redesign the current jumpsuit e-textile into a shirt and pants, which would in theory reduce the communication load on each garment by half, allowing either

more Tier 1 sensors or a faster sampling rate. This implementation would also more closely resemble modern clothing.

Another issue worth exploring is how each local sensor's state is stored. While this specific implementation uses orientation angles, a better approach may be to use quaternions. Quaternions are mathematical structures capable of representing orientation and rotation, and avoid the problem of gimbal lock that forced the use of two different compass orientations in this thesis. It is conceivable then that using quaternions to store Tier 1 sensor state information would be more reliable and be a higher performing design.

The hardware platform designed in this work also allows for expansion of the motion capture application. While the algorithm described in this thesis has been shown to be satisfactory for tracking relative movements, additional work could allow a wearer to be tracked in absolute space, potentially using the accelerometer and gyroscope data to calculate an estimate of translation. However, this would require overcoming the issue of drift which simple calibration was unable to solve in this thesis.

Finally, the scope of this work stops short of examining the effect the garment and its wearer can have on the effectiveness of the motion capture algorithm. Relatively large or small users can expect to see some discrepancy in measurements due to the loose-fitting nature of the garment, which causes the sensors to sometimes be displaced from their expected locations. While these errors likely cannot be eliminated completely, it should be possible to conduct a user study and determine the range of error incurred due to variance in user body types. These results could then allow future experiments to explore ways to configure the Tier 1 sensors to more effectively capture motion on a variety of users.

Bibliography

[1] J. Edminson, "Hardware architectures for software security," Master's thesis, Bradley Department of Electrical and Computer Engineering, Virginia Tech, 2006.

[2] J. Chong, "Activity recognition processing in a self-contained wearable system," Master's thesis, Bradley Department of Electrical and Computer Engineering, Virginia Tech, 2008.

[3] A. Love, "Automatically locating sensor position on an e-textile garment via pattern recognition," Master's thesis, Bradley Department of Electrical and Computer Engineering, Virginia Tech, 2009.

[4] P. Azad, A. Ude, T. Asfour, G. Cheng, and R. Dillmann, "Image-based markerless 3d human motion capture using multiple cues," *International Workshop on Vision Based Human-Root Interaction*, 2006.

[5] M. Lapinski, "A wearable, wireless sensor system for sports medicine," Master's thesis, Massachusetts Institute of Technology, 2008.

[6] Z. Luo, I.-M. Chen, K. D. Nguyen, W. Ni, K. Li, C. Gu, C. K. Lim, and S. H. Yeo, "A wearable sensor network for the control of virtual characters," *Advanced Intelligent Mechatronics*, pp. 1707–1712, July 2009.

[7] Xsens. Xsens mvn–inertial motion capture. [Online]. Available: http://www.xsens.com/en/general/mvn

[8] A. Sundaresan and R. Chellappa, "Markerless motion capture using multiple cameras," *Computer Vision for Interactive and Intelligent Environment*, pp. 15 –26, 2005.

[9] M. Brodie, A. Walmsley, and W. Page, "Fusion motion capture: a prototype system using inertial measurement units and gps for the biomechanical analysis of ski racing," *Sports Technology*, vol. 1, no. 1, pp. 17–28, 2008.

[10] Vicon. Vicon mx motion capture system. [Online]. Available: http://www.vicon.com/products/viconmx.html

[11] R. Guilemaud, Y. Caritu, D. David, F. Favre-Reguillon, D. Fontaine, and S. Bonnet, *Wearable eHealth Systems for Personalised Health Management: State of the Art and Future Challenges.* IOS Press, 2004, pp. 286–291.

[12] D. Gafurov, K. Helkala, and T. Søndrol, "Biometric gait authentication using accelerometer sensor," *Journal of Computers*, vol. 1, no. 7, pp. 51–59, November 2006.

[13] S. Lovell, "A system for real-time gesture recognition and classification of coordinated motion," Master's thesis, Massachusetts Institue of Technology, 2005.

[14] R. Slyper and J. Hodgins, "Action capture with accelerometers," *Symposium on Computer Animation*, pp. 193–198, 2008.

[15] W. T. Baumann, personal communication, 2009.

[16] C. Sul, S. Jung, and K. Wohn, "Synthesis of human motion using kalman filter."

[17] Y. Tao, H. Hu, and H. Zhou, "Integration of vision and inertial sensors for home-based motion rehabilitation," in *ICRA '05, IEEE International Conference on Robotics and Automation*, 2005.

[18] M. Quirk, "Inclusion of fabric properties in the design of electronic textiles," Master's thesis, Bradley Department of Electrical and Computer Engineering, Virginia Tech, 2009.

[19] V. Jolly, "Activity recognition using singular value decomposition," Master's thesis, Bradley Department of Electrical and Computer Engineering, Virginia Tech, 2006.

[20] Honeywell, "HMC6343 datasheet."

[21] Wolfram. Spherical coordinates. [Online]. Available: http://mathworld.wolfram.com/SphericalCoordinates.html

[22] X. Zhang, personal communication, 2010.

Appendix A

Accelerometer Drift

This appendix addresses a specific issue encountered when attempting to use the accelerometer to measure translation. The motion capture algorithm that is the key contribution of this thesis measures only pose, or relative position. This section is meant as a reference for attempts to perform absolute position calculations by integrating acceleration to compute velocity, which is necessary to extend the work done in this paper from relative movements to movements in absolute space.

A.1 Velocity from Acceleration

Since acceleration is the derivative of velocity, taking the integral (antiderivative) of the accelerometer readings gives an approximation of the velocity. A simple way to find the integral of acceleration is to calculate the area under the acceleration curve using a Riemann sum. As shown in Equation A.1, the change in velocity for a given sample can be calculated by multiplying the acceleration for that sample by the sample time. This would constitute a *right endpoint* Riemann sum; a more accurate calculation would be to average two accelerations and calculate based on the midpoints.

$$a = \frac{dv}{dt} \tag{A.1a}$$

$$dv = a \cdot dt \tag{A.1b}$$

As mentioned in Chapter , small errors in the acceleration are integrated into linear terms by this method. Therefore, a reasonable goal is to calibrate each accelerometer as accurately as possible, in the hopes of reducing the drift. While eliminating drift entirely is impossible, in applications where periodic corrections to velocity are available, calibration of the accelerometer can result in a pronounced increase in short-term accuracy.

A.2 Reducing Drift Through Calibration

The effectiveness of the calibration routine on reducing drift was tested on one sensor. By using the sensor's compass to find the true direction of gravity, and subtracting that gravity vector out of the accelerometer reading, the net acceleration due to non-gravity forces is what remains. Using a test data file taken while in a still pose, this net acceleration should be zero. The acceleration can then be integrated by way of a Riemann sum to approximate the velocity. Plotting this velocity as calculated by the calibration values for each method should illustrate the effect calibration has on drift error.

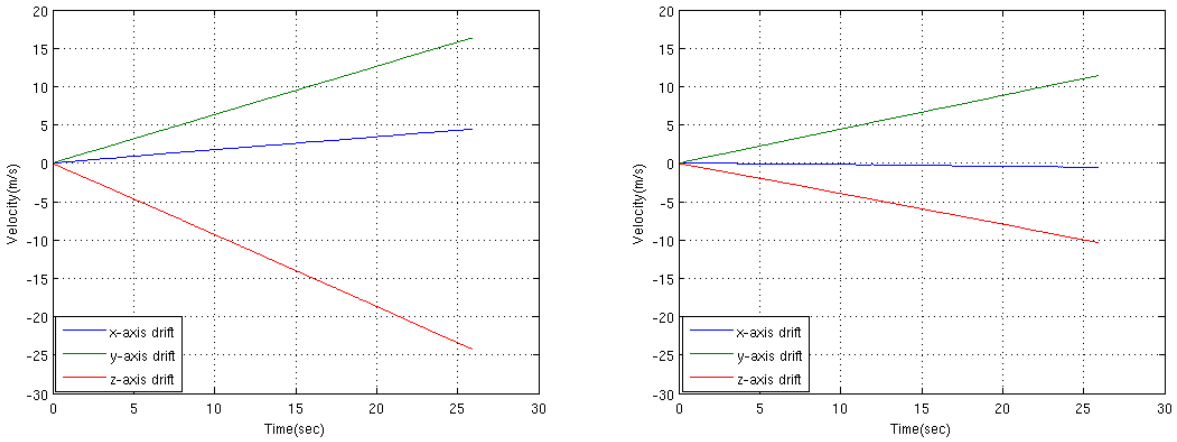

Figure A.1: Drift measurement using the datasheet defaults (left) and the best of the calibration methods (right). Each line represents the estimated velocity from computing the Riemann sum of the area under the acceleration.

The calibration procedure clearly offers an improvement in accelerometer drift over the datasheet values simply by increasing the accelerometer's accuracy, as demonstrated in Chapter 5. However, this process does not reduce drift enough for measuring translation velocity and position with the accelerometer to be a reality without some sort of external correction, much like the pose capture algorithm developed in this

thesis integrates the gyroscope to find angular position, and then corrects that estimate with the compass measurement.

An alternative to external correction, which might defeat the purpose of self-contained motion capture depending on the specific implementation, might be combining the acceleration readings from multiple sensors. While it is true that different limbs can experience different acceleration, using the Tier 2 sensor to synthesize measurements from related sensors (such as both parts of the leg) could be useful in recognizing certain movements like walking, running, or sitting. By working in tandem with the other components, and the Tier 2 sensor's determination of the pose and orientation changes, it may also be possible to narrow down the range of movement possibilities and make a best guess at translation. This strategy should be able to ascertain the direction, if not the magnitude, of translation for each limb.