Low Cost Open Source Modal Virtual Environment Interfaces Using Full Body Motion Tracking and Hand Gesture Recognition

Matthew J. Marangoni
Wright State University

LOW COST OPEN SOURCE MODAL VIRTUAL ENVIRONMENT INTERFACES USING FULL BODY MOTION TRACKING AND HAND GESTURE RECOGNITION

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

MATTHEW J. MARANGONI
B.S. C.E.G./C.S., Wright State University, 2011

2013
Wright State University

WRIGHT STATE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

May 22, 2013

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Matthew J. Marangoni ENTITLED Low Cost Open Source Modal Virtual Environment Interfaces Using Full Body Motion Tracking and Hand Gesture Recognition BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

Thomas Wischgoll, Ph.D.
Thesis Director

Mateen Rizki, Ph.D.
Chair, Department of Computer Science and Engineering

Committee on
Final Examination

Thomas Wischgoll, Ph.D.

John Gallagher, Ph.D.

Michael Raymer, Ph.D.

R. William Ayres, Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Marangoni, Matthew. M.S. C.E.G, Department of Computer Science and Engineering, Wright State University, 2013. *Low Cost Open Source Modal Virtual Environment Interfaces Using Full Body Motion Tracking and Hand Gesture Recognition.*

Virtual environments provide insightful and meaningful ways to explore data sets through immersive experiences. One of the ways immersion is achieved is through natural interaction methods instead of only a keyboard and mouse. Intuitive tracking systems for natural interfaces suitable for such environments are often expensive. Recently however, devices such as gesture tracking gloves and skeletal tracking systems have emerged in the consumer market.

This project integrates gestural interfaces into an open source virtual reality toolkit using consumer grade input devices and generates a set of tools to enable multimodal gestural interface creation. The AnthroTronix AcceleGlove is used to augment body tracking data from a Microsoft Kinect with fine grained hand gesture data. The tools are found to be useful as a sample gestural interface is implemented using them. The project concludes by suggesting studies targeting gestural interfaces using such devices as well as other areas for further research.

Contents

List of Figures

List of Tables

Acknowledgment

This thesis would not have been possible without the help of many other people. I extend my thanks to my advisor Dr. Thomas Wischgoll for this opportunity as well as all his help and input throughout this process. Dr. Michael Raymer, and Dr. John Gallagher for serving on the committee for this thesis and providing excellent advice and input. Thank you to Dr. Mateen Rizki, Chair for the Department of Computer Science and Engineering and the department faculty and staff for providing me the opportunity to complete my degree at Wright State University.

Dedicated to

the people I owe everything:

Joseph and Geraldine Marangoni

and my family,

each of you,

for reasons infinite.

Introduction

This section introduces the project, provides supporting contextual information to the reader by surveying related works, the motivations behind this work, and introduces underlying concepts. Related works are surveyed to establish the context for this project and why it is important. Underlying concepts are then introduced to ensure that the reader is at least familiar with components critical to this project.

1.1 Background

As computer hardware, applications, and devices evolve, mouse and keyboard interfaces have increasing competition from devices such as the Microsoft Kinect and Nintendo Wii Remote. Enhancing and improving the way that users interact with computers is increasingly important as computers continue to expand their domain. Smart phones and tablets have popularized touch-based gesture interfaces while the Microsoft Kinect has made full body motion tracking and audio commands a common occurrence in the home. New interfaces offer alternatives that collect user input in ways that feel natural to the user and are more effective versus always using traditional input methods [6] [10]. Novel interfaces such as these have proven popular for console gaming but also have many other applications. One notable application that has emerged is interacting with virtual environments.

Virtual environments offer users more engaging and enveloping experiences when interacting with applications, especially large data visualizations [6] [10]. Large scale visualizations are often used for data sets that are unmanageable on standard desktop displays. Such visualizations often utilize virtual environments as a way to explore data in three dimensions and provide a more navigable visual interpretation of the data. This type of visualization allows users to take away more information and gain different insights into their data [10].Taking a tour of a new house or of a high resolution CT scan are not impossible on a standard monitor but navigating them where walking through a room is more directly simulated, user movements are used as input, may prove to be a much more informative and natural user experience. However, in order for the user to interact with the visualization an interface must be used and that often entails using a controller or motion tracking. Often setups like this are expensive but the Microsoft Kinect is able to offer a viable solution for full body motion tracking at the price of a consumer gaming peripheral [6].

Researchers have utilized the Microsoft Kinect to offer a full range of movement tracking, discerning human actions, and even hand tracking and hand gesture recognition [6] [12] [23] [24] [11]. These are significant advances and are creating substantial interest in gestural interfaces especially ones with a Kinect base. Yet the methods used are based on the hardware depth sensor data and image analysis performed on image data. If the user steps out of the field of vision or conditions change in such a way to compromise the reliability of the image analysis algorithms the user is left with an inaccessible system. This means the user is still constrained not necessarily to their own viewable area but instead constrained to the viewing area of the Kinect. Users are restricted in their movements by these aspects.

The approach taken in this project is to augment the strongest inherent feature of the Kinect - body tracking - with a glove that specializes in hand gesture recognition to create a modal gestural interface.

1.2 Related Work

There is a lot of research available covering both virtual environments and gesture-based interfaces. The two areas are often linked, as virtual environments are an area that has provided a natural use case for gesture based interfaces. Manipulation of immersive visualizations via gestural interfaces has gained substantial popularity as a result of recently released devices. Video game consoles and smart phones have been accompanied with novel ways of collecting user input. These different devices, such as the Nintendo Wii Remote, PlayStation Move, and Microsoft Kinect, have been explored heavily by researchers but most recently the Microsoft Kinect has received much attention due to its potential for gestural interfaces.

Illustrating the usefulness of virtual environments for interacting with large data sets, Kreylos et al. [10] provide a new set of tools to geoscientists in order to enhance their work flow. Focusing on the geoscientists normal tasks of 3D mapping, measuring topological changes, and investigating the movement of tectonic plates, the authors were able to tailor a virtual environment specifically to their users. The navigation of the visualization, the ability to draw and measure points in the data on to the visualization, the ability to be machine agnostic yet maintain the virtual environment, and the real-time display of the data were the functional goals. The results were that, without a formal study, the geoscientists they had utilize the system were tasked with performing the same tasks in both their usual work flow and the new virtual environment based one. Users were noted that without the use of stereoscopic displays present in the virtual environment, 3D objects were harder to recognize immediately. The virtual environment allowed a more natural identification of the 3D objects versus relying on movement of the visualization to determine in which plane a point exists. Another key result of the study is that the virtual environment succeeded in aiding the users to identify and locate an error in a simulation which the previous work flow tools were unable to assist the user. It is clear that the virtual environment was able to assist in the exploration of data sets to scientists not specializing in such systems. The immediate adaptation to the 3D environment is an illustration of the need for more immersive, more natural visualization tools and that need extends to the interfaces used for manipulating the visualization as well.

The Nintendo Wii Remote and Microsoft Kinect are compared by Francese et al. in using subjective terms such as usability and the user's sense of envelopment by the interface and experience [6]. Users were tasked with navigating a 3D environment using both input devices. The study found that the Microsoft Kinect was preferred overall due to a feeling of a natural interface and that users felt more connected with the device. Francese et al. noted their awareness of a novelty factor influencing the results and also recognized that while the Kinect was preferred the authors did not feel it was a viable solution for prolonged usage due to fatigue. Fatigue was also noted by Host et al. as a problematic source which lead to the suggestion of future systems focusing more on hand and finger recognition [7].

Kristensson et al., Ramirez-Giraldo et al., Li, and Ren et al. demonstrate methods of using the Microsoft Kinect to recognize hand gestures [11] [23] [12] [24]. Kristensson et al. focus on one and two handed gesture recognition. The users hand(s) must enter an "input zone" that delimits when the beginning and end of a gesture have occurred [11]. Implemented using an adaptation of a pen stroke and touch screen recognition system, the authors probabilistic continuous gesture recognition manages to achieve over 92.7 percent accuracy with one and two handed motions. The hands are tracked via the skeleton data retrieved from the Microsoft Kinect. Ramirez-Giraldo et al. propose and implement a system that identifies hand gestures by employing kernel based machine learning mechanisms to find relationships between hand trajectory samples [23]. Li proposes and implements a system that identifies the hands, identifies the fingers, and then classifies the gesture. An image of the hands is captured and RGB image pixels are mapped to each hand which is then followed by contour tracing to identify the fingers and calculates the fingertips based on the palm position and direction of each finger [12]. Gestures are then processed according to the extended finger count, extended finger name, and finally vector matching to determine the final classification of a gesture. The method by Li was able to achieve, when the same gesture was made with both hands, over 90 percent accuracy across a dictionary of 9 gestures [12]. Ren et al. were able to achieve 90.6 percent mean accuracy utilizing the depth sensor and a black belt to detect the hand in conjunction the Finger-Earth Mover Distance algorithm and some finger detection algorithms, to counteract the low resolution from the Microsoft Kinect to determine a measurement for the hand and then matches against the known gestures [24] [25] . Each methodology proposed differs, and in fact Ramirez-Giraldo et al. are recognizing dynamic gestures versus static gestures being recognized by Ren et al. and Li, yet each shares the commonality of utilizing the Microsoft Kinect depth sensor to create an active distance field in which the hands should be operating for detection. The results are more than just encouraging strides to the prospects of completely hands-only gestural interfaces that recognize both static and dynamic (temporal) gestures.

Rahman et al. demonstrate using the Kinect to recognize gestures in the car [22]. Using motion path recognition the user's motions are compared against a map of gestures representing multimedia device functionality. To play a random song, the user may shake their hand. As this does not induce visual search as a dashboard button would, the driver may keep their attention on the road. Motion recognition was measured as 89.2 percent and higher by Rahman et al. across a 10 gesture pool [22]. Using gestures to control media devices while driving could be a safer way to utilize systems in the car versus traditional button and knob methods. Users were also reported to find the system useful and a pleasure to use. Creating systems that utilize gestures in order to replace common interfaces, such as buttons, have been spurred forth by the low cost abilities offered by the Microsoft Kinect.

Hoste et al. developed a multimodal system that assisted a user entering text using speech [7]. The system allows users to speak the text input, increasing the usability over standard controller based interfaces and keyboards to a hands-free system, and then perform corrections on their input utilizing gestures. This system utilizes the Microsoft Kinect to perform the recognition of body gestures such as moving an arm upward to seek towards the start of the alphabet and a downward arm gesture to seek towards the end of the alphabet. The system performed better than the Microsoft Kinect exclusive interface and almost as well as the virtual keyboard with exclusively speech recognition registering the highest Words Per Minute (WPM). The authors however did note that the speech recognition was frustrating at times due to incorrect recognition. Clearly frustrating the user is going to dampen the experience and hinder any attempts at immersion. The results directed the authors to aim future research towards smaller gestures and focusing on finger gestures versus full body gestures due to fatigue and a very slow input speed versus traditional controller-based methods [7].

1.3 Purpose

The objective of this project is to supplement full body motion tracking with hand gesture recognition in order to provide the ability to create a modal gestural interface for a virtual environment that is low cost and accessible. Combining the two tracking methods provides high sensitivity to both fine motor movements of the hand (e.g. extending the index finger) and full body motions (e.g. waving an arm). The utilization of Open Source Software (OSS) and consumer grade devices in this project yields a low cost solution with high accessibility for exploration, utilization, and future developments.

The Microsoft Kinect is able to provide a lot of useful tracking data but as many papers cite, the device is not particularly well suited for fine grained hand and finger tracking or hand gesture recognition due to the low resolution imagery both from the depth camera and the image camera. This has lead researchers such as Kristensson et al., Ramirez-Giraldo et al., Li, and Ren et al. to use image analysis and related techniques on the data retrieved in order to perform such identification and distinctions [11] [23] [12] [24]. Many, if not all, hand detection methods using the Microsoft Kinect rely on the depth sensor data to create an invisible field between two distances in which the hands are active and generally assumed to be the only data present. In most cases this is manually tuned and though automatically tuning it appears feasible, when a user turns around, moves their hands out of the active zone by accident, or performs the same gesture but does not have their hand facing the camera perfectly (sideways gesture perhaps) the recognition process may be derailed. A device such as a mouse or the AnthroTronix AcceleGlove is not subject to such restrictions; thus while the user may have to wear a glove, the user may also move around in a less constrained manner and through an extended space.

The Microsoft Kinect creates an active zone which constrains the user. Such constraints on the user, both spatially and relating to their actions, must be compared against current interface solutions; in this case, a traditional game controller is the standard and such a device has recognition of input, hardware flaws notwithstanding, of approximately 100 percent. When input is incorrectly determined, it is generally not software recognition failing - there is minimal to none involved in recognition - it is the fault of the user. Short of inferring the user's intent, this is a desirable state because the responsibility now is placed squarely on the user. If a hand gesture recognition system achieved 99 percent accuracy and was used to replace your steering wheel, 1 percent of the time when the driver attempted to turn right the vehicle would not turn right. The example is extreme but the concept is the the same. A game or computer that is frustrating to use due to the gestural interface, may not be used at all. This is another reason this project moves the hand gesture recognition responsibility to a device that is better suited for hand gesture data. By combining the two devices, this project can relax some of the constraints on the user. The user is not required to keep their hands at a specific distance but are required to make the correct hand gesture.

Tracking systems are often quite expensive but the advent of the Microsoft Kinect and AnthroTronix AcceleGlove have brought low cost, accessible, and usable hardware into the field. Making use of such hardware allows this project to maintain the low cost objective. The software cost is negated by using OSS, further diminishing the overall project cost. A low cost solution is important as it encourages others to experiment and further the research versus creating a prohibitively expensive entrance fee to the field. Students, researchers, and hobbyists alike can reconstruct this entire project at minimal cost.

Underlying Concepts

This section introduces concepts that are at the crux of this project. Virtual environments are introduced as they are the primary visualization environment target of this project. Gestural interfaces are covered as they are the primary form of interaction and control applied to the visualizations. The hardware utilized in this project is introduced and the underlying method for static gesture recognition, Support Vector Machines (SVM), is presented.

2.1 Virtual Environments

Virtual environments are highly immersive computer-based visualizations that allow users to interact with a virtual world - a virtual reality. Virtual environments often incorporate large 3D capable displays or head mounted displays, such as the upcoming Oculus Rift, to give users a more immersive experience by surrounding them and placing them inside the world [20]. Haptic feedback, surround sound systems, and body tracking may all be integrated into such a system and any other elements that enhance the users a feeling of being immersed inside a virtual world.

An interesting recent example of a complex system utilizing many sensory stimuli somewhat outside of an academic context, including pain, is the ”Ultimate Battlefield 3 Simulator” shown on The Gadget Show [27]. The system presented allows users to play a video game but have it monitor their movements via omni-directional treadmills and a Microsoft Kinect. The system also reacts to users taking enemy fire by correspondingly firing paintballs at the user from approximately the appropriate direction. Combat simulations such as these are just one example of how virtual environments can be used to heavily enhance and create immersive interactive visualizations.

2.2 Gestural Interfaces

Human Computer Interaction has seen a lot of advancements in gestural interfaces. Recently these advances have been fueled largely by video games and consumer media outlets such as the Microsoft Xbox 360 and Microsoft Kinect. Users have taken to creating gestural interfaces as well as many other projects using the Microsoft Kinect. Gestural interfaces are based on a user's motion and position, actions that feel more natural instead of using the mouse and keyboard or other traditional paradigm input devices. People often use gestures in conversation to help convey their meaning. As a result, an interface that uses such gestures is to some degree an extension of natural conversation dynamics. Such interfaces do not require controllers and may not require fine motor movements thus they may be immediately applicable to users who find that those interfaces are ineffective. Moreover, gestural interfaces can assist with silent communication - whether the user is adept in sign language or in scenarios where noise is unacceptable. Sign language is an example of a gestural interface and there are devices, such as the AnthroTronix AcceleGlove used in this project, that are able to collect data that allows a computer to interpret such gestures. Examples of gestural interfaces are given by Kavakli et al. and Farhadi-Niaki et al. [8] [5]. See also 1.2.

2.3 Accelerometer Equipped Glove

The AnthroTronix AcceleGlove is a hand position capturing device based on accelerometers. It is a consumer grade product and collects movement and position data about the hand and fingers using the 6 accelerometers [1]. The glove is accompanied by a Software Development Kit (SDK) which includes a Java library that enables developers to interact with the glove and write their own applications. The provided SDK is compatible with the three major platforms: Linux, Mac OSX, and Windows. The kit also comes with a visualization tool that allows the recording, training, and testing of gestures [1]. The library that interfaces with the glove is OSS and upon request (in this case it took multiple requests) the code may be retrieved from AnthroTronix. The device is USB and/or Bluetooth with the additional wireless module, and utilizes a virtual serial port for communication of sensor data [1].

Figure 2.1: The reference axes for the AnthroTronix AcceleGlove

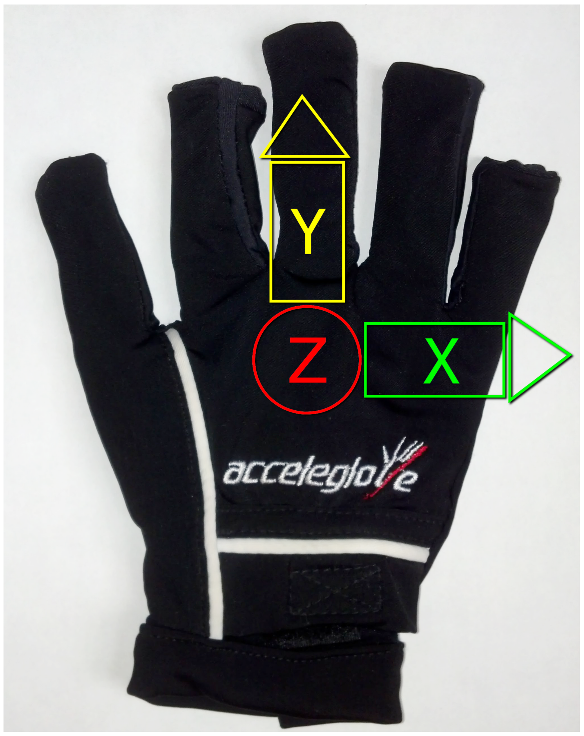


Figure 2.1 shows the glove and the orientation of the axes. The back of the hand is shown, the thumb is at the far left. As displayed, the positive Z-axis is perpendicular to the palm, the positive X-axis and positive Y-axis are perpendicular to each other as well as the Z-axis. Correspondingly, Left handed gloves are also made and there are also 3 sizes

of the glove. The wireless module of the glove utilizes Bluetooth, can be attached with an included strap to the user, and is rechargeable using an included AC adapter.

There is a small circuit board above the back of the palm in a Velcro pocket to which the sensors and communication interfaces are connected. Each of the sensors is positioned on the back of each digit towards the tip of each finger and one on the back of the palm. Each sensor reports 3 axes of data: X, Y, and Z which measure the position and motion of the sensor. The data collection mode, format of the data reported, and other features may be controlled by various commands that can be sent to the glove. When retrieving data from the glove, 18 values are reported.

The AnthroTronix AcceleGlove was chosen for this project for a few reasons. The primary reason is that the glove was outfitted for and targeted towards hand gesture recognition. The demo software included with the glove performs recognition of American Sign Language (ASL). The cost of the glove was low, 500 USD and the additional wireless module was 500 USD which makes the glove available to consumers, especially without the wireless module. The glove was readily accessible through the labs. There was also the potential to retrieve the source code to the library utilizing the glove from AnthroTronix which could be useful for other applications, debugging, and prototyping.

2.3.1 Data Accuracy

Two common areas of concern with accelerometers, especially lower cost accelerometers, are noise and drift. These two issues are standard properties associated with accelerometers and as such are present in specification sheets. Therefore drift and noise must be anticipated, examined, and if necessary accounted for when using accelerometers.

In the context of our recognition system the noise could be mitigated using a low pass filter or a smoothing filter, such as a Kalman filter. If there was substantial noise in the data it could potentially increase the space occupied by the gesture thus having adverse affects on the gesture recognition. For smaller gesture sets this may not prove problematic if the spaces occupied were separated by substantial and clear gaps. In this project's use case, gesture recognition needs to be able to switch between modalities, not necessarily offer an optimal solution to gesture recognition as it can be replaced with a different algorithm.

Drift however would mean the sensors measurements degraded over time and this would certainly lead to usability issues for long term sessions of visualizations and potentially render the device useless in practical scenarios. A reasonable session would be thirty to sixty minutes allowing for rests for user's eyes and arms. A more taxing and thorough goal would be eight hours assuming the user was utilizing this for an occupation and the device was not reinitialized during this period.

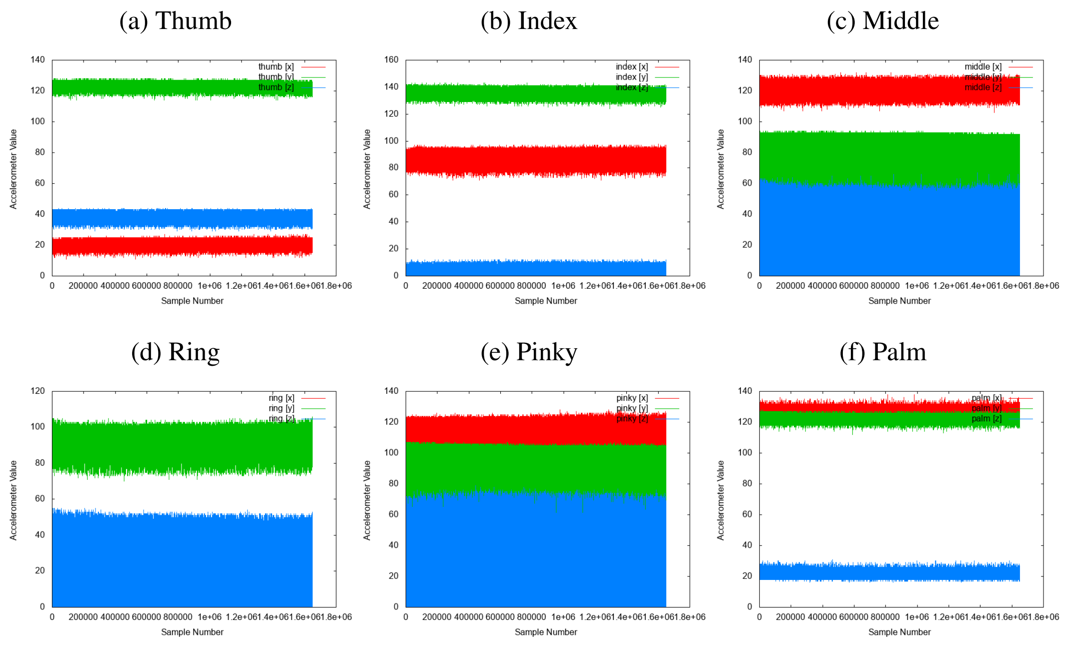
As noted, normally a specification sheet for the accelerometer in question would be provided and express such data about the device. As the model indicators on the accelerometers inside the glove were removed, the seller was contacted and due to a potentially infinite response time a simple experiment was designed and implemented.

A microcontroller would be connected to a servo in order to control it's rotation. The servo wheel would then have an arm attached to it and strings would be attached from the arm to 2 fingers of a glove. More than one finger is used for this test to ensure that the accelerometer was not an exception. Only 2 fingers were used in order to have some fingers remaining relatively still. The fabric of the glove, movement in the string, and vibrations would still impact the other sensors on the glove but as this is what would happen when a user is using the glove it is an accurate approximation. Moreover, if the sensors experience drift it will be noticeable despite those small discrepancies if it is significant enough to impact the usability of the device.

When the servo moved, the arm would lift the fingers accordingly, and then the servo reversed direction and would rest the fingers. Using this setup the same motion could be simulated effectively for an extended period of time. The arm rotated 90 degrees, pulling up the two digits, paused for 1 second, rotated negative 90 degrees, setting down the two digits, and paused for 1 second, and repeated this sequence for a period of 4.097 days. Data from the glove was collected as fast as possible using the current development versions of the libraries at the time.

The digits used in this experiment were the middle and pinky. This allowed ample fabric space that the thumb would remain relatively stationary while the other fingers would be impacted accordingly which is visible in the samples. At the end of the sample collection period the data was graphed and examined. Overall it appears as though no significant drift or noise variations occurred that would greatly influence the outcome of the system.

Figure 2.2: The first 8 hours of data collection per finger

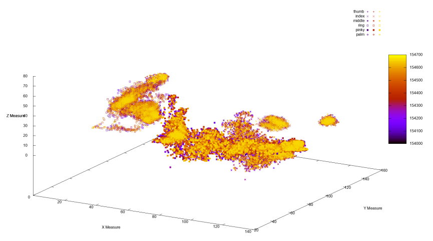


(a) Thumb (b) Index (c) Middle (d) Ring (e) Pinky (f) Palm

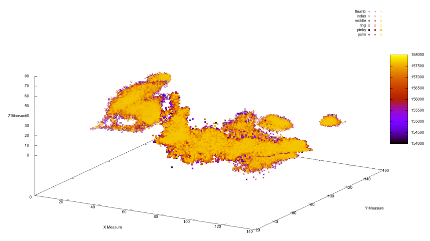
Graphs for 8 hours of sampling were generated and are presented in Figure 2.2. The graphs show clear distinctions between axis values in some cases such as Figure 2.2a and Figure 2.2b. In some cases, primarily the sensors which were near the moving areas, the graphs do not have clear separation between axes and as a result, further graphs have been generated per axis for some of the sensors. See Appendix B.1 for enlarged versions of the graphs. The figures show that each accelerometer experienced a relatively constant amount of noise during each time frame and that substantial drift was absent. From the data collection used in Figure 2.2, data for 4.097 days was collected and a total of 20286974 samples. This equates to a capture rate of approximately 57.31 Hz.

Figure 2.3: The first 8 hours of data collection from all accelerometers, colored by sample number.

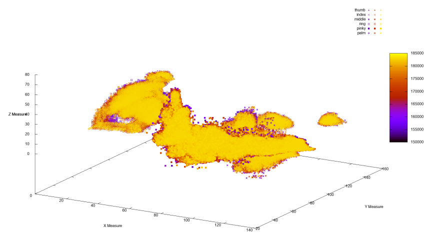
(a) 10 minutes



(b) 60 minutes



(c) 8 hours



Another view of the same data provides a similar insight. A 3D plot of all the accelerometer data is provided in Figure 2.3 at the sample intervals of 10 minutes in 2.3a, 60 minutes in 2.3b, and 8 hours in 2.3c. If the devices were susceptible to drift and noise that would substantially impact their usage in this project, the image after 8 hours would exhibit a much different point cloud shape than the image after 10 minutes. The data points are also colored by the sample number, thus the points with dark colors are the oldest points and the brighter colors are the newest data points. It is also likely that there would be more darker colors visible than the large amount of bright colors if the readings were becoming further distorted over the period of time. However, the general contour of the point cloud is established in the 10 minute sample set and the rest of the images fill in the form with higher point density but do not substantially expand the contour of the shape. See Appendix A.1 for enlarged versions of the graphs.

2.4 Microsoft Kinect

The Microsoft Kinect was released for the Microsoft Xbox 360 gaming console offering fully body motion tracking. The device has subsequently seen many developments in applications outside of the console and instead using computers. Due to the low cost and advanced functionality versus anything else on the market, the Microsoft Kinect provides opportunities for novel gestural interfaces to be created for a wide array of applications.

Figure 2.4: The Microsoft Kinect

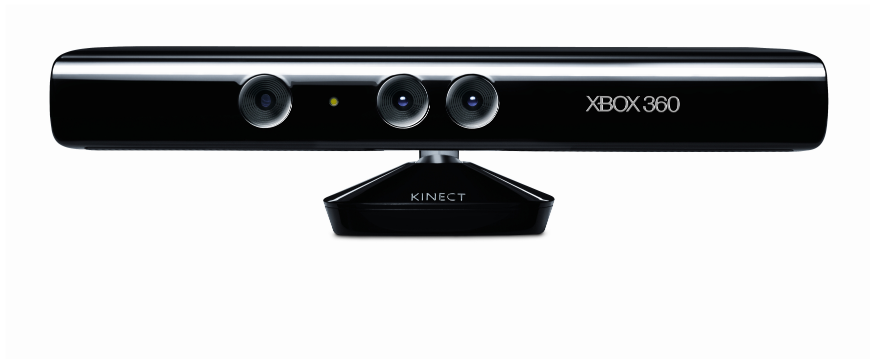


Figure 2.4 shows the Microsoft Kinect, courtesy of [18]. The Microsoft Kinect comes outfitted with an RGB camera, an infrared (IR) depth sensor and IR emitter, four microphones in an array, and a 3-axis accelerometer. From left to right in Figure 2.4, the first lens is for the IR emitter, the second glass lens is for the RGB camera, and the third lens is for the depth sensor. In the base of the device there is a motor that also allows for $\pm 27^\circ$ vertical tilt [16].

The RGB camera can provide color RGB data at up to a resolution of 1280 x 960, allowing the recording of video as well as photos of the Field of View (FOV). The FOV provided is 43° x 57° (vertical x horizontal). The device also can provide 30 frames per second (FPS) for both RGB camera data and IR depth sensor data at a resolution of 640 x 480 [14]. Table 2.1 shows the available formats for images from the RGB camera [14].

Table 2.1: Microsoft Kinect RGB Camera Data Formats

Resolution (Width x Height in pixels)	**Frames per second**	**Data Type**
640 x 480	30	Raw Bayer
640 x 480	30	RGB
640 x 480	15	Raw YUV
640 x 480	15	YUV
1280 x 960	12	Raw Bayer
1280 x 960	12	RGB

The IR depth sensor provides depth data at up to a resolution of 640 x 480. By projecting IR light from the emitter, reading the data back in through the depth sensor, and then evaluating the results the Microsoft Kinect is able to give users access to the distance of objects within the FOV relative to the device. The FOV provided is 43° x 57° (vertical x horizontal) [15]. The depth sensor has an operating distance of 800mm-4000mm, and the Kinect for Windows variant may go into ”Near Mode” decreasing the distance to 500mm-3000mm [17]. Usable modes for depth data collection are shown in Table 2.2 [15]. There is also a data stream that collects an image of the IR depth sensors view at a resolution of 640 x 480 at 30 frames per second [14].

Table 2.2: Microsoft Kinect IR Depth Sensor Data Formats

Resolution (Width x Height in pixels)	**Frames per second**
80 x 60	30
320 x 240	30
640 x 480	30

The Microsoft Kinect was chosen for this project for multiple reasons. The ability to track the movement of a user with a single USB device and minimal hardware setup made this device an immediate candidate for this project. At a price of 149.99 USD the device lets this project maintain a low cost objective. At such a low price point and due to the popularity of the gaming console, the Microsoft Kinect is also a well suited solution as the user may in fact already own the device and if not, it can be readily purchased from multiple vendors.

2.5 Support Vector Machines

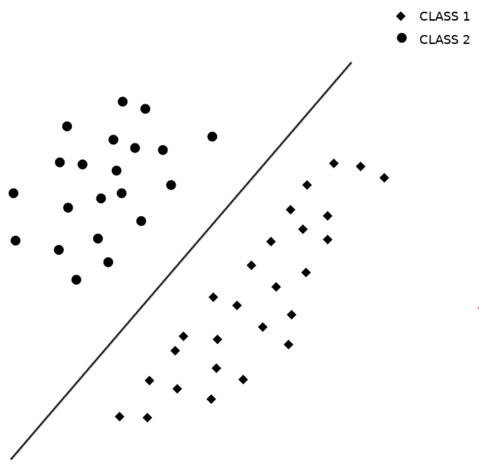
In order to determine when a user is making a hand gesture a method of recognition was to be chosen. As the focus of this project is not on optimal gesture recognition, a viable recognition method was sought versus developing an optimal one. Support Vector Machines (SVM), or Support Vector Networks as referenced by Cortes et al. are machine learning methods utilized for pattern recognition [4]. SVMs are supervised and able to perform linear and non-linear classifications efficiently. Having been heavily researched starting in the 1990s, the Support Vector Machine is a well-known approach in the area of machine learning [4]. The use of SVMs by Cortes et al. show that even early implementations are effective in performing hand writing recognition versus other methods at the time [4].

Support Vector Machines or Support Vector Networks are a formalization of a very intuitive approach to classification. Consider a binary classification problem. A set of unknown data must be separated into two classes, Class 1 and Class 2 (it is common to use ± 1 as boundaries), based on features. SVMs attempt to create a maximal and optimal boundary between the two sets of data based on their features [4]. This objective appears intuitive when viewed in 2D (See Figure 2.5a) and may be extended to higher dimensions.

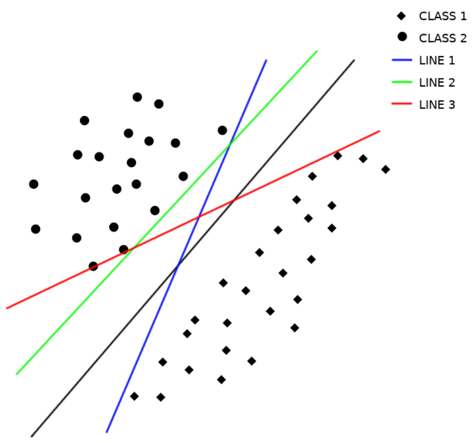
The line separating the two classes of data visible in Figure 2.5a is the optimal hyperplane that SVM seeks. There are many potential hyperplanes that separate the data, as shown in Figure 2.5b. The 3 labeled hyperplanes also separate the two classes of data but are non-optimal. As new data is processed, incorrect classifications will clearly occur due to the unequal proximity between the hyperplane and a data class. In order to determine the optimal hyperplane separating the data classes, it is clear that the distance between the two classes should be maximal [4].

Figure 2.5: SVM Hyperplane and Margin Optimization

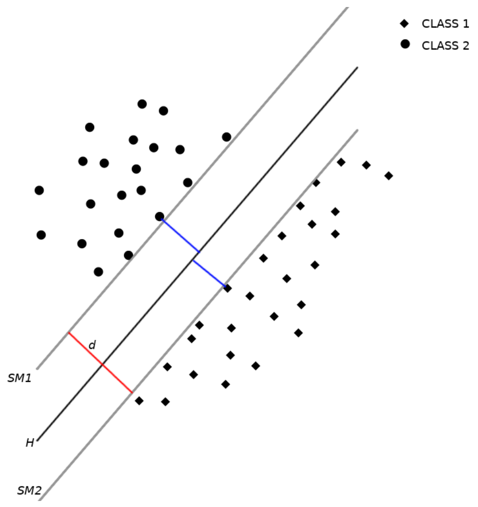
(a) Optimal and Intuitive Hyperplane



(b) Other Hyperplanes



(c) Margin Optimization and Support Vectors



SVM determines this maximal separation by finding a hyperplane that is as far from both data classes as possible. The points used in this calculation determine the placement of the hyperplane and are called Support Vectors. The support vectors are identifiable in Figure 2.5c as the points closest to lines SM1 and SM2 - the points that are closest to the opposite class. These are Support Vectors as their removal would move the hyperplane as opposed to removing other points which would not impact hyperplane placement. By solving a quadratic programming problem maximizing the margin between support vectors, an SVM model is able to represent the optimal hyperplane between classes [4]. At this stage input may be classified and where it falls relative to the hyperplane will determine its classification.

In many cases data is not trivially separated, quite the opposite of Figure 2.5a which is easily linearly separated (which is visually observable). In some cases data is heavily meshed in the input space, some sets of data may be trivially separated in their input dimension whereas other sets, such as the gesture data shown in Figure 2.7 display meshing that clearly causes issues. There may also be noise or valid data that does not conform to the usual pedigree of the rest of the labeled data. Both of these situations need to be addressed. As a result, SVMs incorporate different features, such as kernel functions and soft margins, in order to address data separability. Through the use of these elements SVMs are afforded flexibility and gain substantial generalization [4] [13].

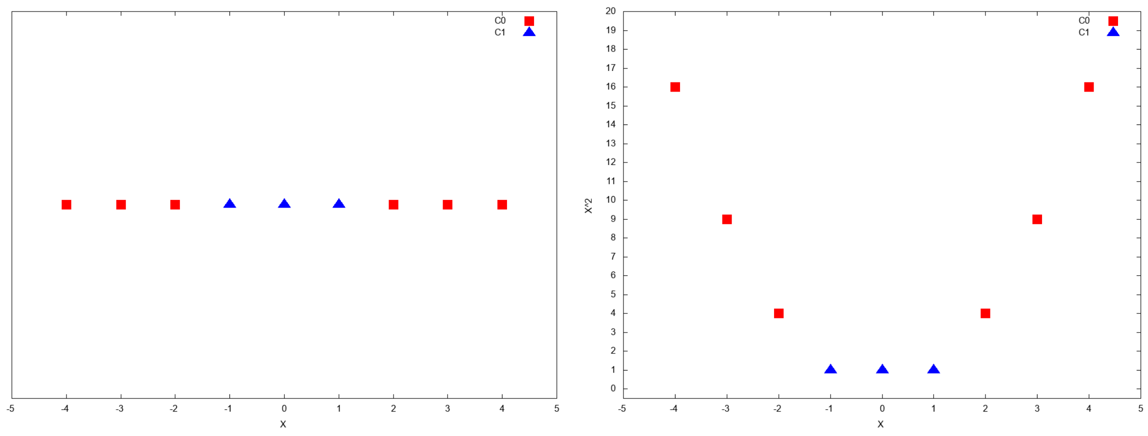
Similar to other methods that also utilize kernels, SVMs use kernel functions to map low dimensional data into higher dimension spaces. The goal is to map the feature space to a higher dimension in order to find linear separability in the data. Data that may not be separable in the input space may be separable in a higher feature space. With a properly designed kernel the mapping may be made implicit to the kernel and any explicit calculations of the mapping are avoided completely by nature of the dependency of the linear classification on the dot product. The exploitation of this mathematical property is generally referred to as the ”kernel trick” [13]. Such usage of kernels allows a SVM to classify data that is not linearly separable but instead are non-linearly separable. This method is also computationally advantageous as, through the kernel trick, calculations are not performed in higher dimensions thus drastically reducing the calculations necessary.

A common example is using variations of the XOR problem to show that some data can be linearly separated once mapped into a a different feature space. Another example is shown in Figure 2.6. Figure 2.6a shows a graph of two 1D classes, $C0$ and $C1$, where $C0 = (-4, -3, -2, 2, 3, 4)$, $C1 = (-1, 0, 1)$. It is visible that the data is not linearly separable in 1D but through using a kernel that establishes $x \mapsto (x, x^2)$, as shown in

Figure 2.6b, the data becomes linearly separable in the higher feature space.

Figure 2.6: Non-linear Classification

(a) Input Space (1D) (b) Mapped Feature Space (2D)

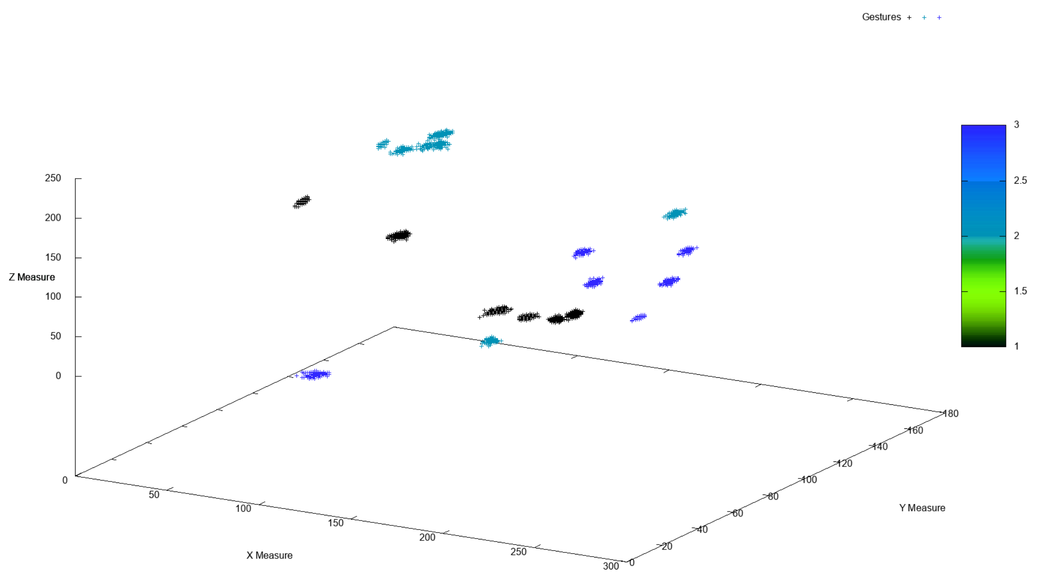


Soft margins are a way to allow SVMs to cope with noisy data, making a better approximation while accepting some errors. The idea is to reduce the influence of such outliers on the placement of the hyperplane by excluding the points. A soft margin classifier may then create an optimal hyperplane separating the training data without being incorrectly influenced by noise [4].

SVMs have many applications with real world data. As real data often does not have binary categories, SVMs have been adapted to perform multi-class classifications, which is how they are used in this project. There are many schemes available to integrate multi-class classifications into SVMs and the library chosen for this project supports such classifications [3]. This project only utilizes them in a common classification capacity. Other applications are outside the scope of this paper however it is important to note that other applications do exist.

An example of the data being analyzed using SVM for gesture recognition is shown in 3D in Figure 2.7. Figure 2.7 shows the x,y, and z axis values of each sensor over a series of samples, colored per gesture. It is clear to distinguish each sensor in this manner. The SVM algorithm does not make the distinction between sensors but instead treats each gesture as a set of features (3 features from each sensor) for classification.

Figure 2.7: Plot of 3 Gestures Accelerometer Data Colored By Gesture



Implementation

This section discusses the specifics of the implementation of the modal interface. Challenges that were faced in each area of the project are addressed and further notes and details for each subsection are provided. How the devices in the system communicate is explored and each of the input systems, hand gesture recognition and body tracking, are detailed. Finally, the integration of the input devices with the virtual environment to create a usable interface is discussed.

3.1 Device Communication

This section discusses the types of communication used when transporting tracking data. The AnthroTronix AcceleGlove and Microsoft Kinect both require their own protocols. The glove utilizes a virtual serial port interface to report data in response to specific commands. The Microsoft Kinect is able to transmit data via USB through the use of middleware like FAAST [26]. Using FAAST, VRUI is able to consume the data via TCP/IP from the Microsoft Kinect into the virtual environment. Through the implementation of a custom tool for VRUI the data from the AnthroTronix AcceleGlove is able to be read and utilized in conjunction with the Microsoft Kinect data.

3.1.1 Hand Position Data Retrieval

Hand position data is provided to the visualization system by the user via manipulating the AnthroTronix AcceleGlove (See Section 2.3). The SDK that was paired with the glove was written in Java and while it performed fine on the Linux based visualization systems, the visualization environment utilizes C and C++. After some investigation, the use of Java Native Interface (JNI) and other mechanisms to transfer data between the two languages was abandoned. It would introduce another language and a layer of added complexity to the overall system that was unnecessary. The glove could communicate via a virtual serial port thus it was decided that an interface to the glove would be constructed and then utilized to retrieve the data from the glove. The company was contacted in order to retrieve the source code to get some idea of how the system worked as they had clearly researched the problem already however at this time the source code was irretrievable. Inside of the user guide however the serial connection information was present, commands were listed that the glove would respond to, and substantial information about the data format was provided. Enough information was given to create the interface. Based on that information, a serial communications library (sport) and a library that interfaced with the glove (aglove) were constructed. The aglove library uses sport to perform the sending and receiving of data but conceals the serial interface from the developer. From the developers perspective they are interfacing with the glove. Some issues were encountered, such as data format listings being vague or incorrect in the manual however the manual that was being used was also an earlier version than the one found in the updated user manual [1]. In the AnthroTronix User Guide a second command protocol with more rigorous specifications is provided and the conversion to this command protocol is an area for future development (See Section 4.2) [1].

The libraries were designed to aid in rapid development in order to lower system integration efforts. In the current version of the glove interface, the code shown in Figure 3.1

Figure 3.1: Glove Connection Code

```cpp
#include "Glove.hpp"

int main(int argc, char * argv[]) {
    Glove g;

    while(g.isConnected()) {
        g.binaryQuery();
        Glove::HandPosition const * hp = g.getCur();
    }

    g.disconnect();
    return 0;
}
```

is all that is needed to create a connection to the glove and retrieve a hand position.

3.1.2 Skeletal Position Data Retrieval

In order to retrieve data about the movement of a user, users skeletons are detected and tracked using the Microsoft Kinect. The data about the movement of the skeleton is then passed into the system and utilized for manipulating the virtual environment. In this system the virtual environment does not support input from the Microsoft Kinect as-is - this is not an uncommon issue by any means. There are many many games, especially those designed for the PC realm that are not setup to ingest body tracking information.

In order to work around this issue, this project uses the Flexible Action and Articulated Skeleton Toolkit (FAAST) which eases the integration of depth sensors that support the OpenNI standard. By including a Virtual Reality Peripheral Network (VRPN) server, FAAST is able to communicate the skeletal information it accesses from the Natural Interface Technology for End-User (NiTE), a middleware developed by PrimeSense that performs body detection and tracking duties as well as other tasks, to the virtual environment where it is then possible to use that data to manipulate the environment. This is possible as the virtual environment used in this project is compatible with VRPN. FAAST also features the ability to emulate input devices based on user actions [26]. This feature however proved problematic and as a result was not utilized. Using FAASTs VRPN server on a Windows machine with a Kinect, the data is then transferred to a daemon running Linux for the virtual environment and the inputs are made available to utilize in conjunction with the glove data and gesture recognition system.

3.2 Gesture Recognition

Gesture recognition was part of the SDK that accompanied the glove. However, as it is not being used, recognition needed to be implemented as part of the project (See Subsection 3.1.1). Previous research efforts had developed and experimented with gesture recognition using the idea of identifying simple linear relationships, somewhat similar to linear SVM. While considering multiple gesture recognition algorithms for this project, another attempt to retrieve the source code from AnthroTronix for their glove SDK was made and was successful. Upon examining their source code it was discovered that a SVM library was being utilized to perform their gesture classifications. In the end, as SVM was what was being used by the manufacturer of the glove and research lead to the conclusion that SVM was a suitable gesture recognition approach, SVMs were chosen. Investigating viable SVM libraries lead to libSVM [3].

Having placed well in competitions for machine learning, having sufficient documentation, providing sample data sets for demos, being OSS, and being able to maintain the low cost objective libSVM was an obvious candidate. LibSVM was found to have extended SVM to offer probability estimates as well as containing supporting tools for accurate and simple model generation, training, and prediction. These tools are used in order to generate well-fitting recognition models from collected hand gesture data. Moreover, it is not the goal of this project to provide a new gesture recognition algorithm but instead to create an enhanced modal gestural interface based on augmenting the Microsoft Kinect with hand gesture data. This projects interface is such that another gesture recognition algorithm may replace the SVM implementation or the library may offer multiple recognition algorithms.

In order to generate the SVM model used for gesture recognition, a tool was developed that allows the user to input multiple gestures. Due to the need to regenerate the model each time the collection is changed, an easy to use tool was useful. Using methods suggested by Chang et al. with the accompanying Python scripts the data is scaled, trained using a model supporting probabilities, and tested [2]. At this point the model and scaling files have been generated and output. In order to perform prediction, these two files must be provided to the glove interface library and the prediction can then be performed. While this process can take some time, by using the tools provided the process is largely automated and has the added benefit of giving insight as to the accuracy of the recognition model once testing is completed. This process could be automated and integrated further, see Section 4.2. An example of an application that performs continuous gesture recognition and retrieves the label of the predicted class is shown in 3.2.

Figure 3.2: Glove Recognition Code

```
#include "GloveManager.hpp"

int main(int argc, char* argv[]) {
    std::string devicePath(argv[1]);
    std::string scalingFilePath(argv[2]);
    std::string modelFilePath(argv[3]);

    GloveManager * gm = new GloveManager(devicePath, ...
        modelFilePath, scalingFilePath);

    while (gm->getGlove()->isConnected()) {
        gm->getGlove()->binaryQuery();
        double predictedClassLabel = ...
            gm->predict(*(gm->getGlove()->getCur()));
    }

    delete (gm);

    return (0);
}
```

3.3 Virtual Environment Integration

The virtual environment toolkit chosen for this project is Virtual Reality User Interface (VRUI) version 2.2. A large advantage of this toolkit is it offers abstractions for input, displays, and data sources [9]. When switching between virtual environments with different display setups (e.g. a desktop monitor moving to a 3 wall environment) users are able to run the same application made for VRUI and not compromise usability in the transfer. Also, VRUI is readily available, OSS, maintains the low cost goal, and familiarity with VRUI was present. The abstractions VRUI implements lends some extra flexibility that is utilized in this project, specifically for input devices. The user level VRUI application usually communicates with the VRDevicesDaemon - a daemon that input devices are connected to and which serves the respective input data to the clients. At the client level, users may make their own tools, and can configure them as they please. The tools consume the input data served and respond as programmed. This decoupling means that multiple users may use the same environment input devices but the way that the input manipulates the visualization can differ per user. This project utilizes that decoupling in a way that is perhaps not intended but serves the goals of this project.

Figure 3.3: System Data Flow

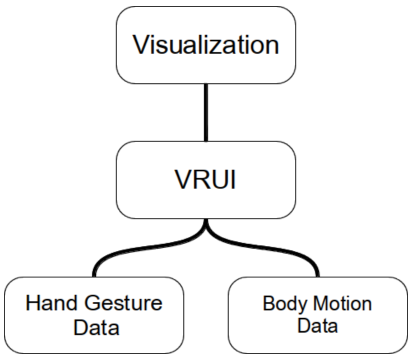


Figure 3.3 shows the data flow relationship between the elements involved in this project. Both data sources are ingested in their respective ways into VRUI. Inside VRUI, a custom tool has been created that reads the input from both input devices, the Microsoft

Kinect from the device daemon and the AnthroTronix AcceleGlove via an instance of the library created in this project, and acts upon them as the visualization is utilized.

While this methodology is an ”inelegant” method of adding a new input device to VRUI, as the input is collected directly from the client instead of the device input daemon, it is an appropriate solution in this scenario. By including the AnthroTronix AcceleGlove connection code inside the tool, the user is able to connect multiple gloves to the same machine, and it is determined per client giving extra flexibility for variation between clients. This also means that in a classroom environment, students may modify their own tool implementations yet still use the standard Microsoft Kinect setup available to them on the lab machines. This also means that instead of recompiling both the device daemon on the input machine as well as the visualization on the client machines, only the client machines need recompilation when changes occur.

A modal interface is then programmed into the tool, thus when the user selects the tool after initializing their visualization they enter a mode in which their body movements are tracked by the Microsoft Kinect and the AnthroTronix AcceleGlove is tracking their hand gestures. In testing, the glove was used to create the different modalities of ”pan”, ”rotate”, and ”null” (or ”NOP”). A ”zoom” modality was also being added but had been postponed in favor of the intuitiveness experienced using the Microsoft Kinect depth sensor. The gestures associated with modalities were an upright closed fist, a thumbs up, and a high five gesture.

Assume the user is wearing the glove on their right hand for the following explanation of use cases for the modal gestural interface.

If the user makes a closed fist with the back of the palm parallel to their body (the ”upright closed fist” gesture) with their right hand, they enter the ”null” modality. In this modality, no user actions are acted upon by the Microsoft Kinect until the user leaves this modality by making another gesture. This modality was viewed as necessary to avoid the user from accidentally entering a mode in which they had not intended. The probabilities for each gestures recognition are retrievable and may also be used to provide a minimum acceptable probability to enter a mode.

If the user makes an open palm facing parallel to their body (the ”high five” gesture) with their right hand, they enter the ”pan” modality. At this point any movements they make with their left hand are tracked by the environment and interpreted as translations of the visualization. If the user moves their hand left and right when facing the Microsoft Kinect, the visualization translates left and right. The panning mode also utilizes the depth sensors and if the user moves their left hand perpendicular to the device, the depth cameras register this and correspondingly pan the visualization closer or farther away from the camera. As a result, this may simulate a zoom or scaling affect yet it is still panning. Just as with a mouse, if the user reaches the edge of the FOV for the Microsoft Kinect, they may make the ”null” gesture and position themselves in the FOV, then make the high five gesture and resume panning. This modality switch, after a small amount of time, becomes smooth, similar to the switching between reverse, neutral, and drive in a car.

If the user makes a closed fist with their thumb pointing to the sky (the ”thumbs up” gesture) with their right hand, they enter the ”rotate” modality. At this point any movements the user makes with their left hand are translated to rotations. As the user moves their left hand from right to left, the same rotation occurs in the virtual environment.

The accuracy of the gesture recognition relies on libSVM and the appropriateness of the accelerometer data for characterizing gestures. As gesture sets may be different, model parameters may be adjusted, and the training sets used to create a recognition model may be insufficient or incorrect, recognition can vary wildly. Using a thumbs up and thumbs down gesture may generate some confusion in recognition and yet a fist and a sideways thumbs up (gesture between thumbs up and thumbs down, a 45° rotation) may provide excellent separability for SVM. This is all able to be visualized and tested beforehand using provided tools. Throughout the many gesture sets generated, most were achieving recognition on test data sets in the 92 to 100 percent range and subsequently using the resulting interface reflected such high degrees of test data accuracy.

Discussion

In this section the results of the project are discussed. The gestural interface, the tools created, and overall project are presented within a context according to the purpose of this project. Future development ideas in hardware and software which range from targeting this project specifically to gestural interface systems in general are discussed.

4.1 Results

The purpose of this project was to develop an accessible, low cost, modal gestural interface for virtual environments by supplementing full body motion tracking with hand gesture recognition. This project utilizes OSS that is freely available almost exclusively, Microsoft Windows (which is still highly accessible) on the tracking machine being the exception. The libraries generated in this project will also be available as OSS. The reliance upon OSS also has a direct impact on meeting the goal of a low cost implementation.

By utilizing free OSS such as VRUI, FAAST, and Linux as well as licensing the produced libraries as OSS the cost of replicating such an environment is dramatically reduced. At the time of writing, from NewEgg.com, Microsoft Windows 7 Home Premium SP1 64-bit is 99 USD, a 1080p 42 inch LCD TV is 469.99 USD, and a Microsoft Kinect is 149.99 USD. The AnthroTronix AcceleGlove cost 500 USD [19]. Generally, any standard computer setup may be used that supports a relatively modern kernel and can run Windows 7 (it is also possible that this may be done using Virtual Machines (VM)) due to the flexibility of VRUI and the devices are USB. Large screens such as a TV or projector may be used in many cases. While the cost may vary significantly depending on screens acquired, computers owned, assuming the user has absolutely nothing and purchases two small laptops with dedicated graphics cards at 399.99 USD each, the full system would cost 2018.96 USD sans shipping and handling. If the user were to choose a 3D capable headmount such as the Oculus Rift, the cost would be lowered by 169.99 USD (totaling 1848.97 USD) and stereoscopic 3D would be added [20]. For comparison, the cost of OptiTracks Motive body tracking software alone is 2499 USD and still requires purchasing of other hardware and requires users to wear marker spheres [21].

The Microsoft Kinect and AnthroTronix AcceleGlove were successfully combined to manipulate a virtual environment. An example gestural modal interface was implemented in Section 3.3 and modal control of the virtual environment was achieved. The user is able to switch modality by changing the gesture made with the glove. The user's alternate hand (the hand not wearing the glove) is tracked by the Microsoft Kinect and serves as a way to provide input appropriate to the current modality. The libraries created substantially shorten the time for a developer to quickly engineer their own gestural interface using the same devices. Replacing the gesture recognition algorithm is also quite accessible. A simple interface is provided for prediction such that previously constructed gestural interfaces may not have to be adjusted if the recognition algorithm is replaced.

The presented sample gestural modality (See Section3.3) was tried with another user to gauge the initial impression of the interface and a small idea if the recognition of the gesture model would remain the same user to user. Recognition worked quite well through the user swap. The user also was able to quickly adapt to use the interface. This is another benefit of the implementation of the VRUI tool. It allows users to create their own control schemes for visualizations using such a modal interface without interfering with other users control schemes. If one user finds that a gestural interface is inadequate or unnatural, they may simply modify their own tool and have a completely new interface for the same virtual environment. Further studies are necessary to gauge the potential reception of this device to a broad user base (See Section4.2).

Dr. Thomas Wischgoll and Wright State University have provided students with access to the visualization facilities and hardware needed to fully utilize this project. As this project is OSS, students are now able to utilize a library that helps them make gestural interfaces, incorporate features into class projects, research gestural interaction, or experiment however else they deem appropriate. They may also contribute to the libraries and improve their efficacy.

4.2 Future Research and Improvements

In this section the potential for further research will be explored. The opportunity for additional research is present in many elements involved in modal gestural interfaces. Each area of potential research will be accompanied by a short description. The suggested areas for future development are broadly classified as hardware, software, and gestural interface usability. The scope of improvements will be both this project as well as in general for modal gestural interfaces.

4.2.1 Hardware

This project exists in part because of the shortcomings of the Microsoft Kinect. Low resolution images, low resolution depth sensors, and a limited field of view lead to image analysis and inferring hand positions as well as constrained user interactions. If the hardware is improved, substantially higher resolution data is provided, then it may be possible to replace a traditional controller with a Microsoft Kinect. Hardware level implementations of tracking algorithms may also prove fruitful.

As shown in Appendix A, the data that is retrieved from low cost accelerometers is noisy. While this level of noise is acceptable in many applications that is not to say that many applications would not benefit from reduced noise and that other applications may become feasible with reduced noise. Cheaper high quality accelerometers would certainly be a welcome development.

4.2.2 Software

Adding a different mode that converts the values that are used for gesture recognition to rotation invariant measurements could potentially improve the interface substantially in terms of usability. It seems quite possible to convert the positions of the fingers to offsets after taking into account the position of the palm sensor. Once this is done, the palm sensor may be ignored for gesture recognition and only the fingers used as features for SVM. As it stands, the data used is the position base data from the glove without any such offset calculations which means that flipping the hand over while still using the same finger gesture can potentially change the gesture. If this were successfully implemented the resulting system could then also use the gloved hand movement data as well as that of the associated arm from the Microsoft Kinect as further input. This would restore two hand interactions instead of the current state where one hand is used for mode selection and the other hand is used for manipulation with respect to the current modality.

Implementation of a filter to help reduce the noise may increase the accuracy and stability of the SVM predictions. The filter that was discussed most frequently for this purpose was a Kalman filter as it should be robust enough to provide smoothed data while remaining responsive and efficient. While smoothing out the data may reduce the sensitivity to extremely minute movements, it must be examined if it will cause problems or not and may be heavily dependent upon the gesture library that is being tested against.

Dynamic (temporal) hand gesture recognition has been heavily researched. Adding an implementation for use with this project would add another modality of user input. Dynamic gestures could be recognized using either the Microsoft Kinect or the AnthroTronix

AcceleGlove.

While American Sign Language (ASL) is a standardized set of gestures, their application to virtual environments is not natural. A formal standardized set of natural gestures, similar to those present on touch screen applications, for navigating virtual environments would be extremely useful. If such a standard were also supplemented with data sets of those gestures being performed it would then be much easier to perform meaningful measurements on recognition systems and the usability of interfaces.

The library which was created to interact with the AcceleGlove uses an older simple communication format, the newer advanced communication format is supported by both the wireless and wired devices. Implementing this would create a unified communication protocol and simplify programming level details.

The development of tools to aid in the use and creation of gestural interfaces would also be quite useful. Displaying the user's current modality would be an extremely helpful aid for users. Developing tools that allowed the user to enter a modality to record new gestures with sufficient samples for usable accuracy, regenerate the recognition model, and then reinitialize the gestural interface and allow the user continue in the virtual environment with their new gestures are also interesting ideas. The collection of a sufficient number of samples as well as the optimization of model generation would be interesting research problems.

4.2.3 Gestural Interfaces

From experiences with gestural interfaces using the Microsoft Kinect, it would be interesting to see multiple studies of a significant group of users for a prolonged period of time that are required to use a gestural interface. Often in research and media gestural interfaces are hailed as the future for Human-Computer Interaction and replacements for the mouse and keyboard. It would be interesting to see a study that removes the novelty and perceived effectiveness of the interface and then evaluates the users perceptions, especially quantifying factors such as fatigue or the potential for Repetitive Stress Injuries (RSI).

Conclusion

The objective of this project was to provide the capability of using gestural interfaces to manipulate virtual environments at minimal cost and with high accessibility. Using the Microsoft Kinect as a body motion tracker and an AnthroTronix AcceleGlove as input devices, the body movements of a user were used to control a gesture in a specific modality while hand gestures were recognized as changes in modality. Through the use of this consumer grade hardware and additional open source software the cost of the project was kept at a low level. The use of readily available hardware and software also leads to high accessibility, allowing anyone to recreate such an environment and proceed to create gestural interfaces or further the research. Using this project a gestural interface was created and the initial suggestion is that the software provides a useful set of tools to create gestural interfaces and that such interfaces are useful. Formal studies of the usability of specific gestural interfaces are needed. The distinction must be made when evaluating this project, between the libraries and software being useful versus the specific implemented gestural interface for that test. A complete study where users compare when they think they are making a gesture versus what the recognition system recognizes at that time may provide a useful metric for usability. Areas for future research and improvement were discussed. This project achieved its goals and was able to successfully integrate gestural interfaces with a virtual environment.

Bibliography

[1] AnthroTronix, "Acceleglove user guide," AnthroTronix, January 2011, version 1.1.1. [Online]. Available: http://www.acceleglove.com/AcceleGloveUserGuide.pdf

[2] C.-C. Chang, C.-W. Hsu, and C.-J. Lin, "A practical guide to support vector classification," April 2010. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[3] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[5] F. Farhadi-Niaki, R. GhasemAghaei, and A. Arya, "Empirical study of a vision-based depth-sensitive human-computer interaction system," in *Proceedings of the 10th asia pacific conference on Computer human interaction*, ser. APCHI '12. New York, NY, USA: ACM, 2012, pp. 101–108. [Online]. Available: http://doi.acm.org/10.1145/2350046.2350070

[6] R. Francese, I. Passero, and G. Tortora, "Wiimote and kinect: gestural user interfaces add a natural third dimension to hci," in *Proceedings of the*

International Working Conference on Advanced Visual Interfaces, ser. AVI '12. New York, NY, USA: ACM, 2012, pp. 116–123. [Online]. Available: http://doi.acm.org/10.1145/2254556.2254580

[7] L. Hoste, B. Dumas, and B. Signer, "Speeg: a multimodal speech- and gesture-based text input solution," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ser. AVI '12. New York, NY, USA: ACM, 2012, pp. 156–163. [Online]. Available: http://doi.acm.org/10.1145/2254556.2254585

[8] M. Kavakli, M. Taylor, and A. Trapeznikov, "Designing in virtual reality (desire): a gesture-based interface," in *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, ser. DIMEA '07. New York, NY, USA: ACM, 2007, pp. 131–136. [Online]. Available: http://doi.acm.org/10.1145/1306813.1306842

[9] O. Kreylos, "Environment-independent vr development," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science, G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, F. Porikli, J. Peters, J. Klosowski, L. Arns, Y. Chun, T.-M. Rhyne, and L. Monroe, Eds. Springer Berlin Heidelberg, 2008, vol. 5358, pp. 901–912. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89639-5_86

[10] O. Kreylos, G. Bawden, T. Bernardin, M. I. Billen, E. S. Cowgill, R. D. Gold, B. Hamann, M. Jadamec, L. H. Kellogg, O. G. Staadt, and D. Y. Sumner, "Enabling scientific workflows in virtual reality," in *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, ser. VRCIA '06. New York, NY, USA: ACM, 2006, pp. 155–162. [Online]. Available: http://doi.acm.org/10.1145/1128923.1128948

[11] P. O. Kristensson, T. Nicholson, and A. Quigley, "Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors,"

in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, ser. IUI '12. New York, NY, USA: ACM, 2012, pp. 89–92. [Online]. Available: http://doi.acm.org/10.1145/2166966.2166983

[12] Y. Li, "Hand gesture recognition using kinect," in *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, 2012, pp. 196–199.

[13] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008. [Online]. Available: http://www-nlp.stanford.edu/IR-book/

[14] Microsoft, "Colorimageformat enumeration," 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/microsoft.kinect.colorimageformat.aspx

[15] ——, "Depthimageformat enumeration," 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/microsoft.kinect.depthimageformat.aspx

[16] ——, "Kinect for windows sensor components and specifications," 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/jj131033.aspx

[17] ——, "Kinect sensor," 2013. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh438998.aspx

[18] ——, "Press image of the microsoft kinect," 2013. [Online]. Available: http://www.microsoft.com/en-us/news/imagegallery/products/default.aspx

[19] NewEgg, April 2013. [Online]. Available: http://www.newegg.com

[20] Oculus VR, "The oculus rift," 2013. [Online]. Available: http://www.oculusvr.com/

[21] OptiTrack, "Motive:body unified optical tracking architecture." April 2013. [Online]. Available: http://www.naturalpoint.com/optitrack/products/motive/body/

[22] A. M. Rahman, J. Saboune, and A. El Saddik, "Motion-path based in car gesture control of the multimedia devices," in *Proceedings of the first ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, ser. DIVANet '11. New York, NY, USA: ACM, 2011, pp. 69–76. [Online]. Available: http://doi.acm.org/10.1145/2069000.2069013

[23] D. Ramirez-Giraldo, S. Molina-Giraldo, A. Alvarez-Meza, G. Daza-Santacoloma, and G. Castellanos-Dominguez, "Kernel based hand gesture recognition using kinect sensor," in *Image, Signal Processing, and Artificial Vision (STSIVA), 2012 XVII Symposium of*, 2012, pp. 158–161.

[24] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM international conference on Multimedia*, ser. MM '11. New York, NY, USA: ACM, 2011, pp. 759–760. [Online]. Available: http://doi.acm.org/10.1145/2072298.2072443

[25] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," in *Proceedings of the 19th ACM international conference on Multimedia*, ser. MM '11. New York, NY, USA: ACM, 2011, pp. 1093–1096. [Online]. Available: http://doi.acm.org/10.1145/2072298.2071946

[26] E. Suma, B. Lange, A. Rizzo, D. Krum, and M. Bolas, "Faast: The flexible action and articulated skeleton toolkit," in *Virtual Reality Conference (VR), 2011 IEEE*, 2011, pp. 247–248.

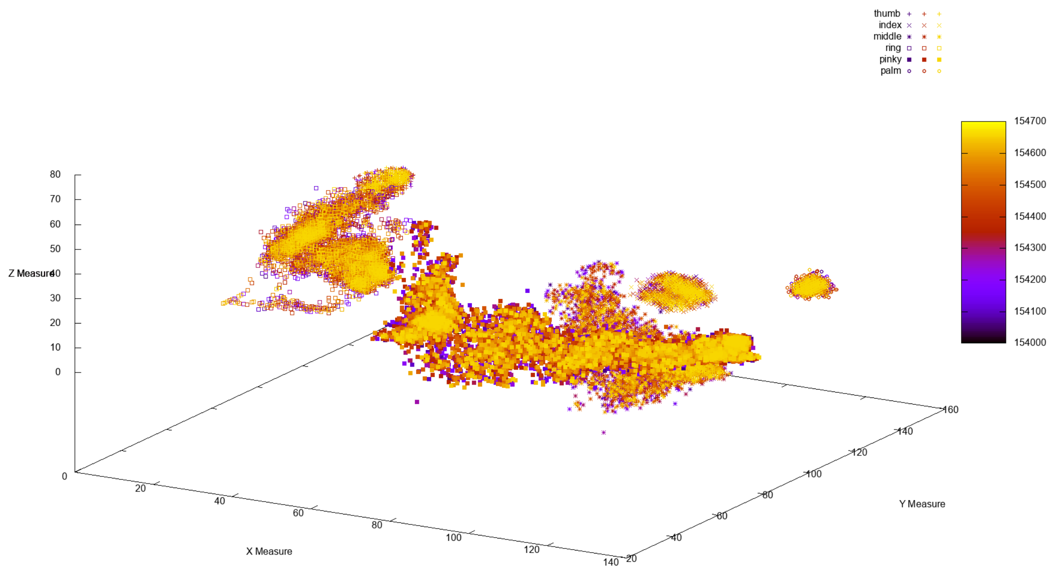
[27] The Gadget Show, "Ultimate battlefield 3 simulator - build & test (full video) - the gadget show," October 2011. [Online]. Available: http://youtu.be/eg8Bh5iI2WY

Appendix A

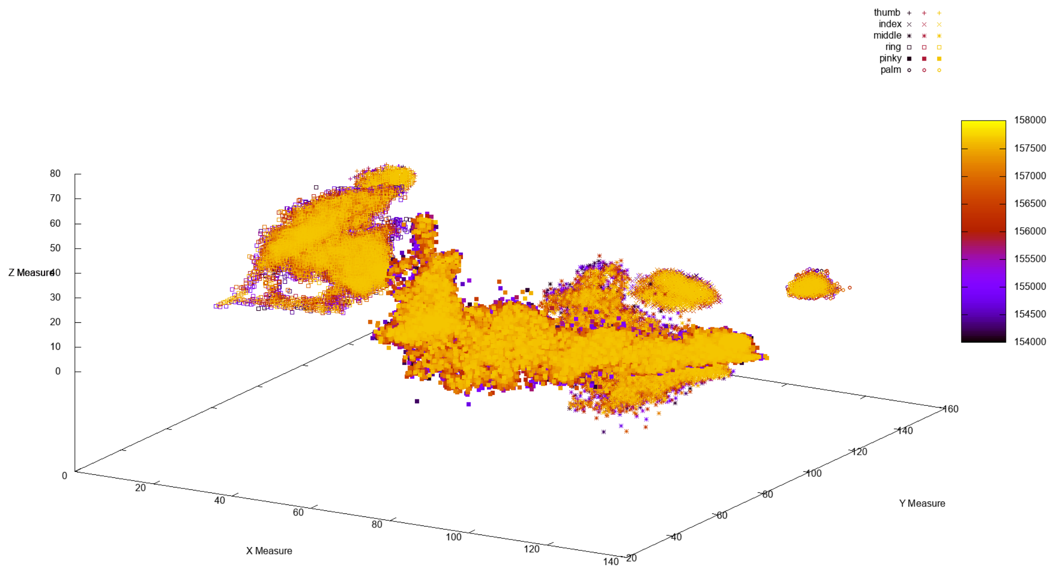
A.1 Noise and Drift 3D Graphs: All Sensors

Figure A.1: Graph of all sensors after a time period, colored by sample number (time).

(a) 10 minutes



(b) 60 minutes



(c) 8 hours

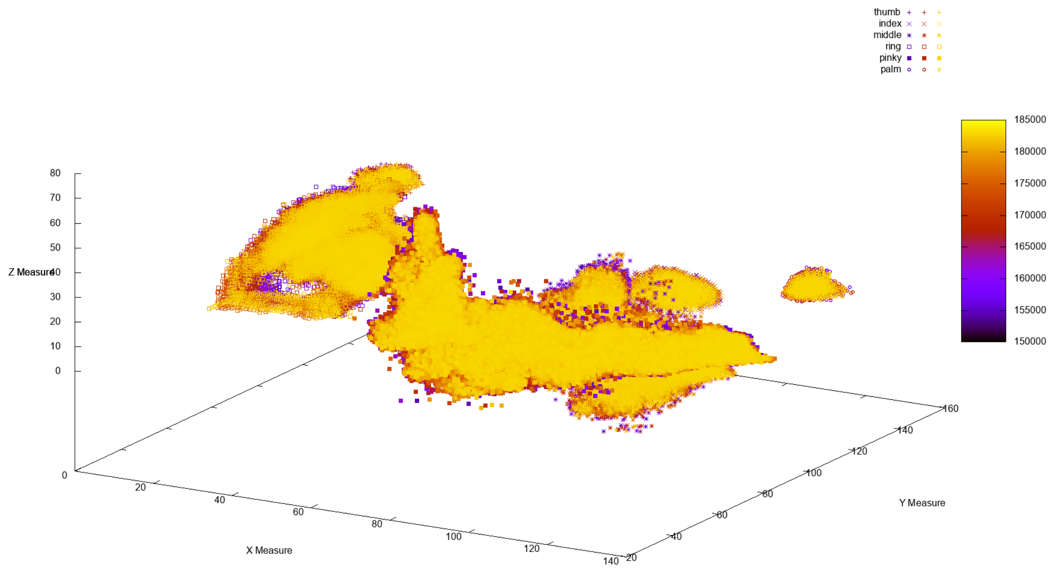
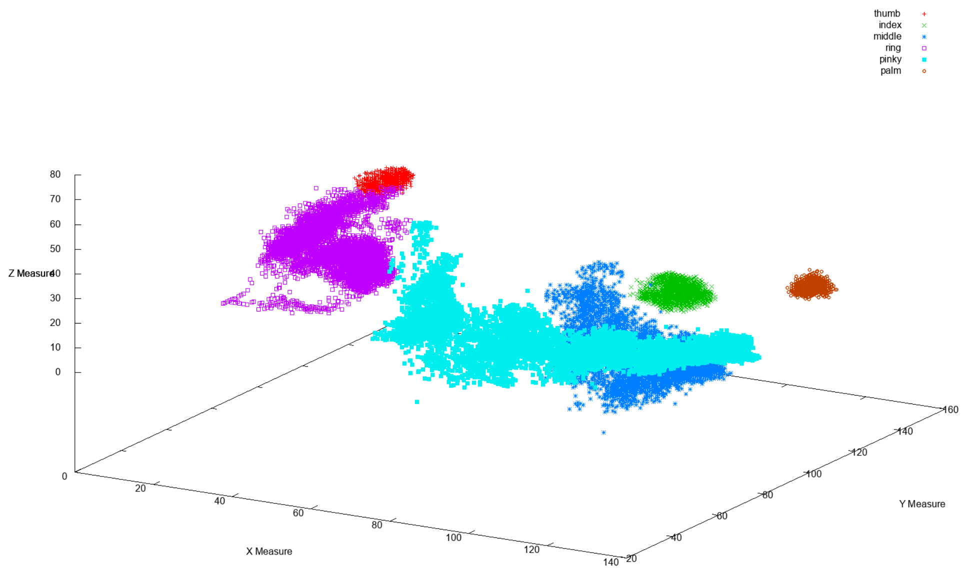
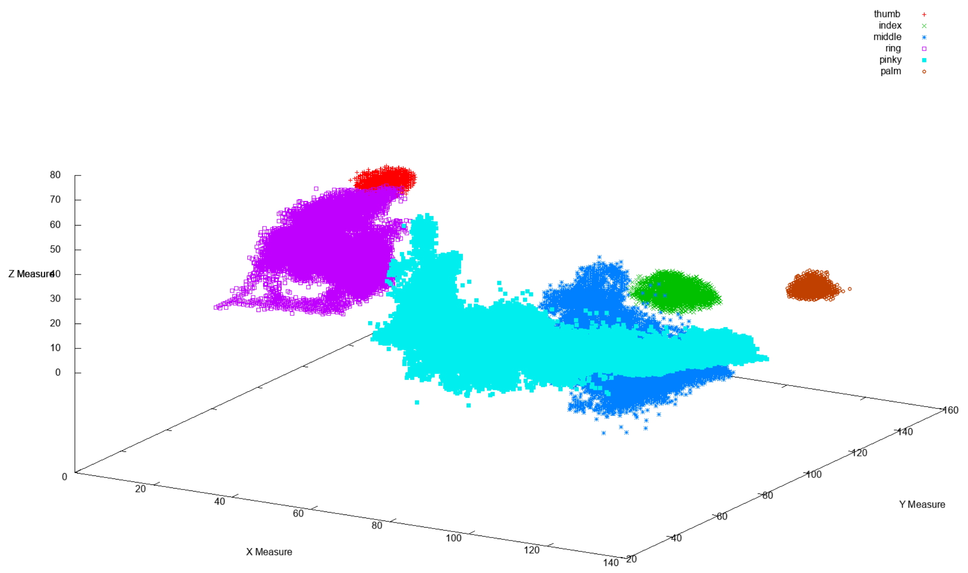


Figure A.1: Graph of all sensors after a time period, colored by sensor.

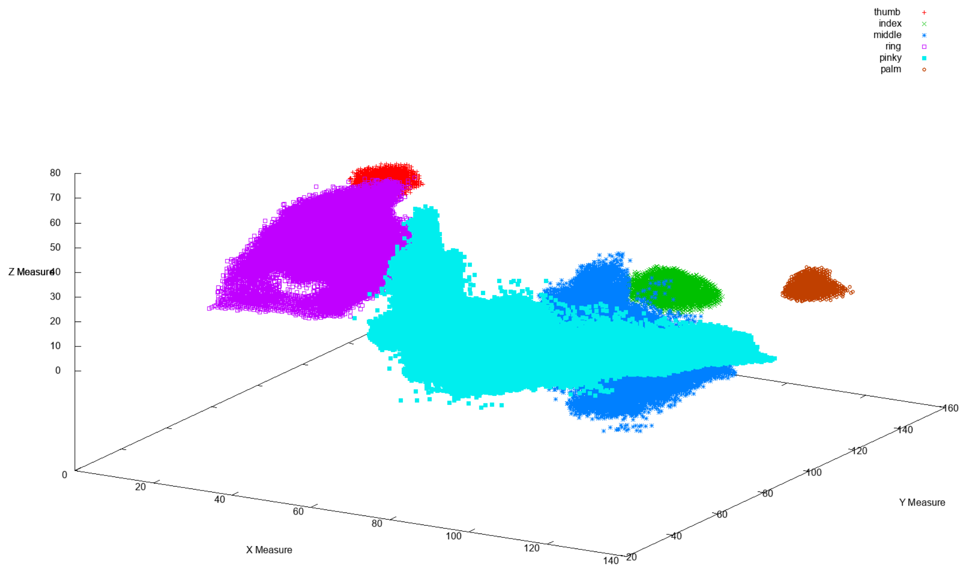
(a) 10 minutes



(b) 60 minutes



(c) 8 hours

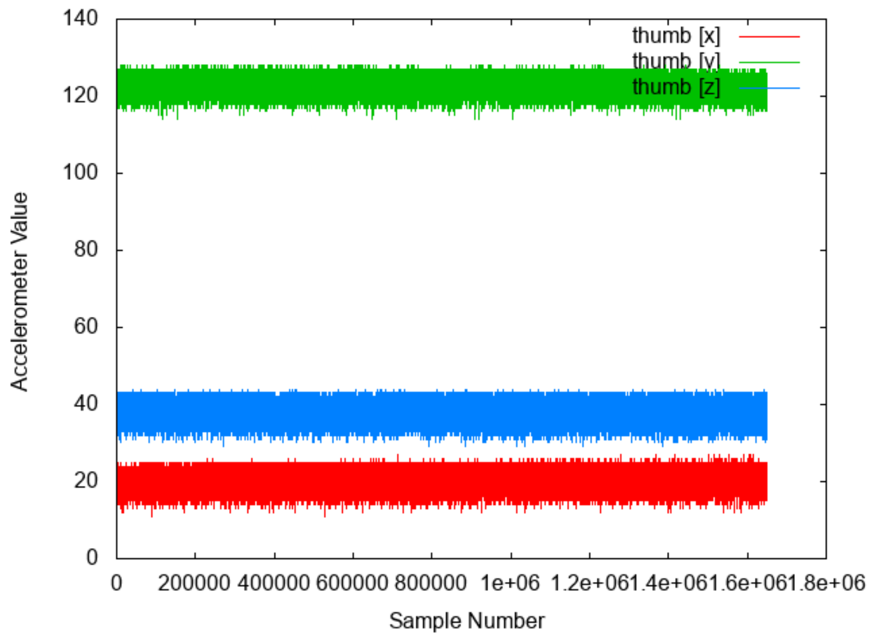

thumb
index
middle
ring
pinky
palm
Z Measure
X Measure
Y Measure
0
10
20
30
40
50
60
70
80
20
40
60
80
100
120
140
160

Appendix B

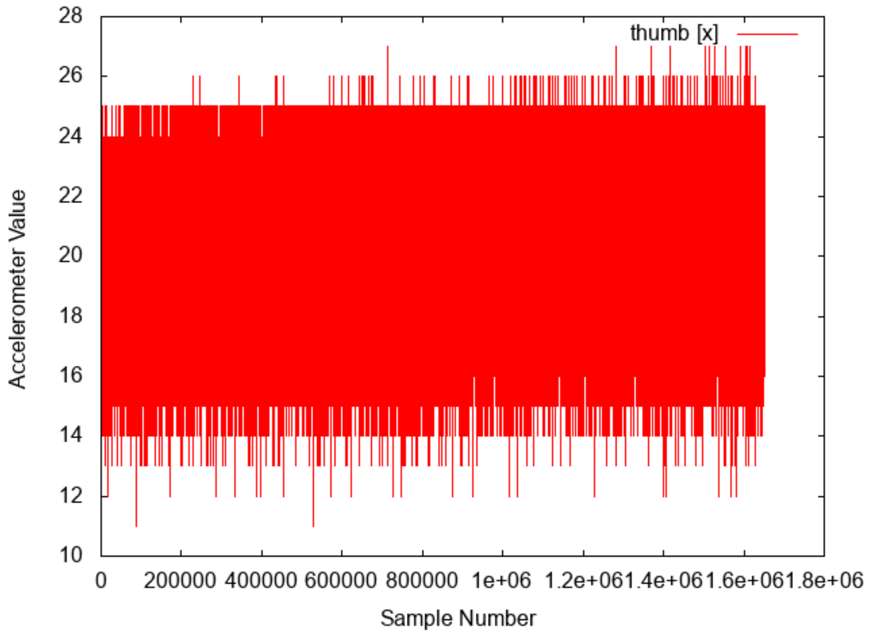
B.1 Noise and Drift 2D Graphs: 8 hours elapsed

Figure B.1: Graph of thumb data over 8 hours

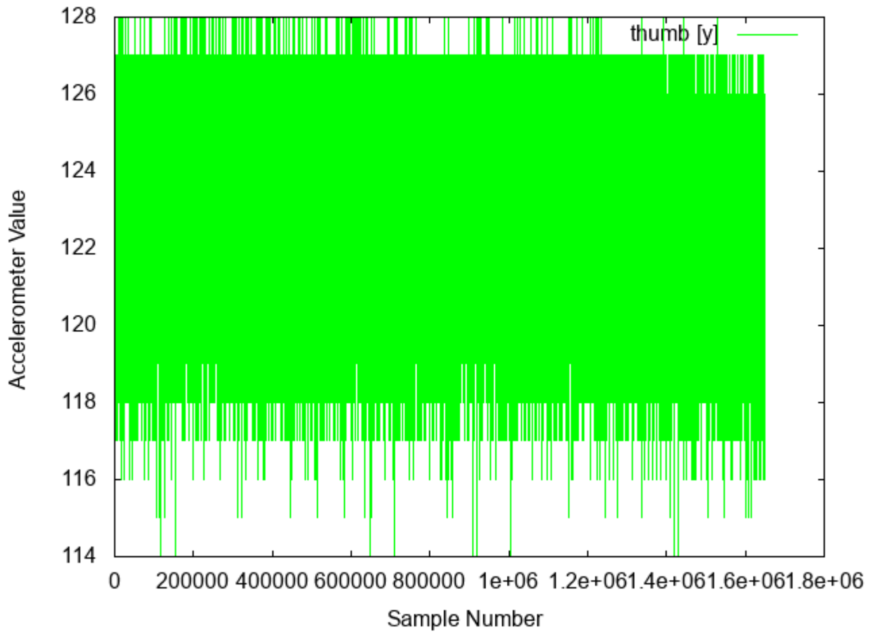
(a) Graph of thumb data over 8 hours



(b) Graph of thumb x-axis data over 8 hours



(c) Graph of thumb y-axis data over 8 hours



(d) Graph of thumb z-axis data over 8 hours

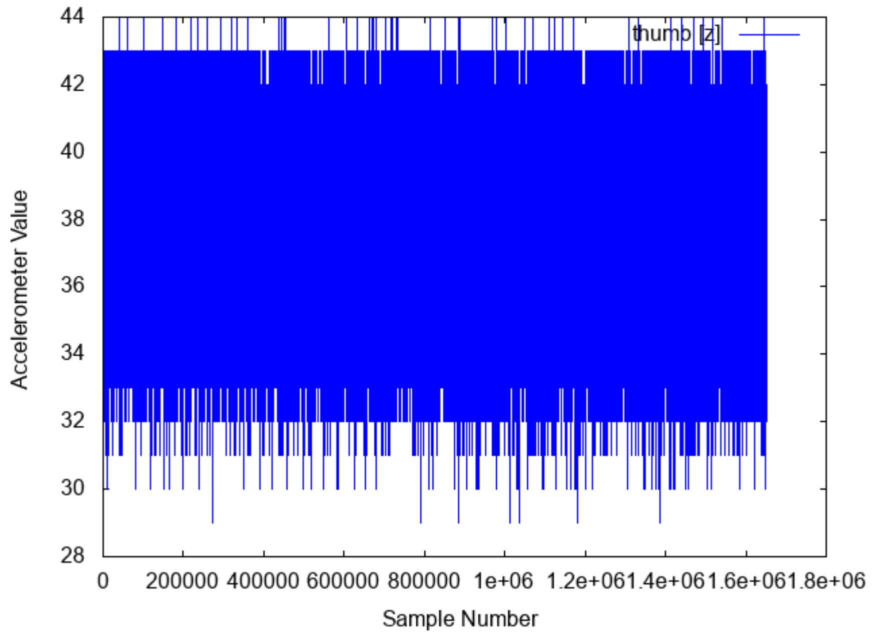
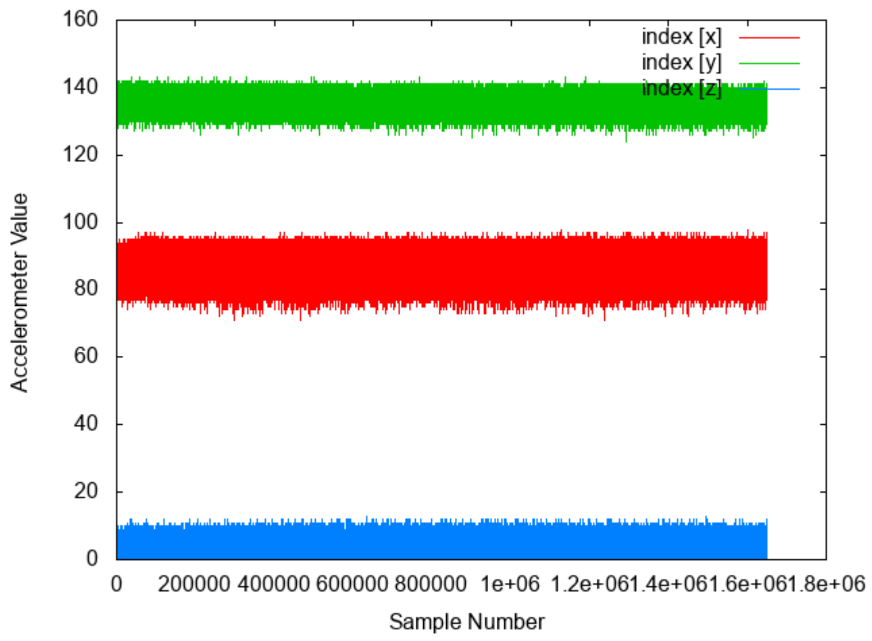
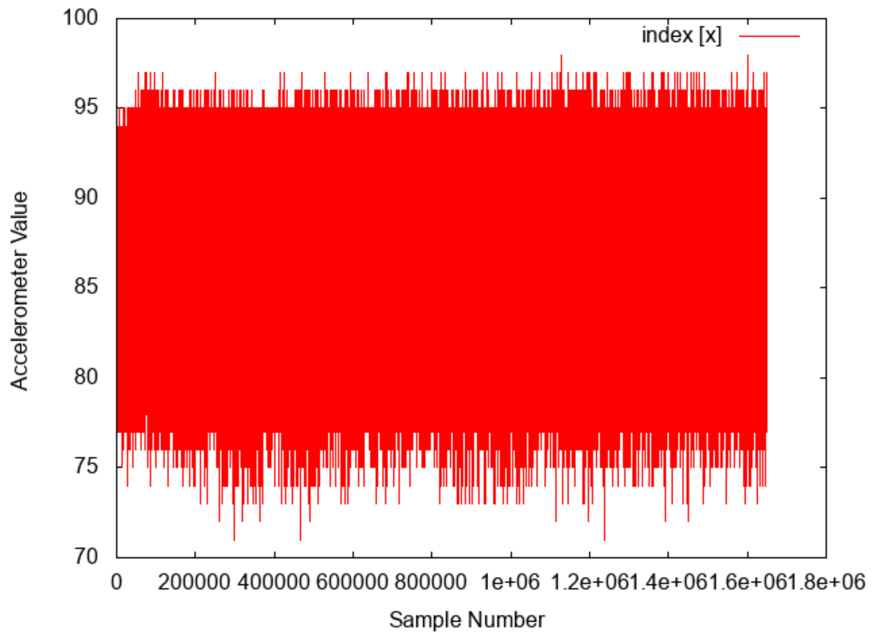


Figure B.2: Graph of index finger data over 8 hours

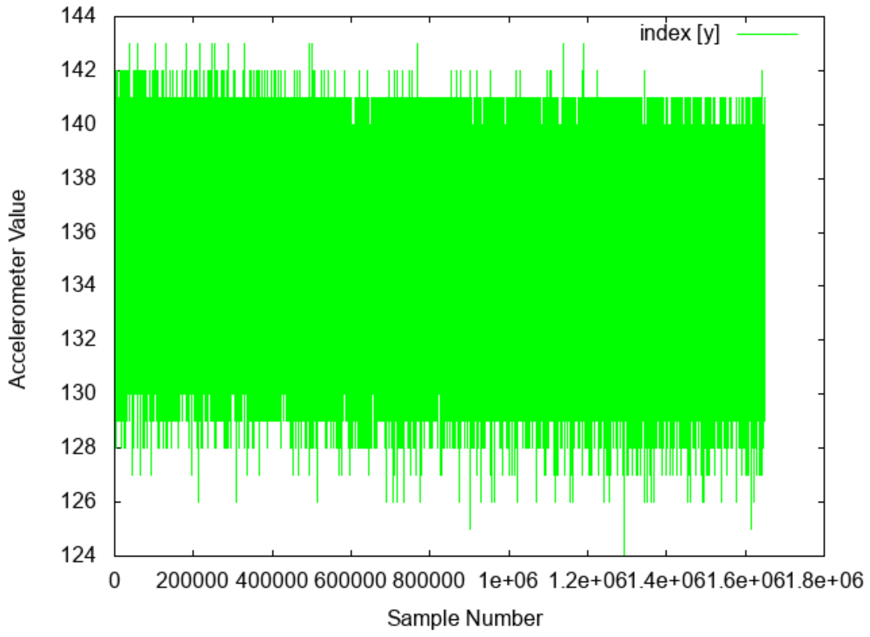
(a) Graph of index finger data over 8 hours



(b) Graph of index finger x-axis data over 8 hours



(c) Graph of index finger y-axis data over 8 hours



(d) Graph of index finger z-axis data over 8 hours

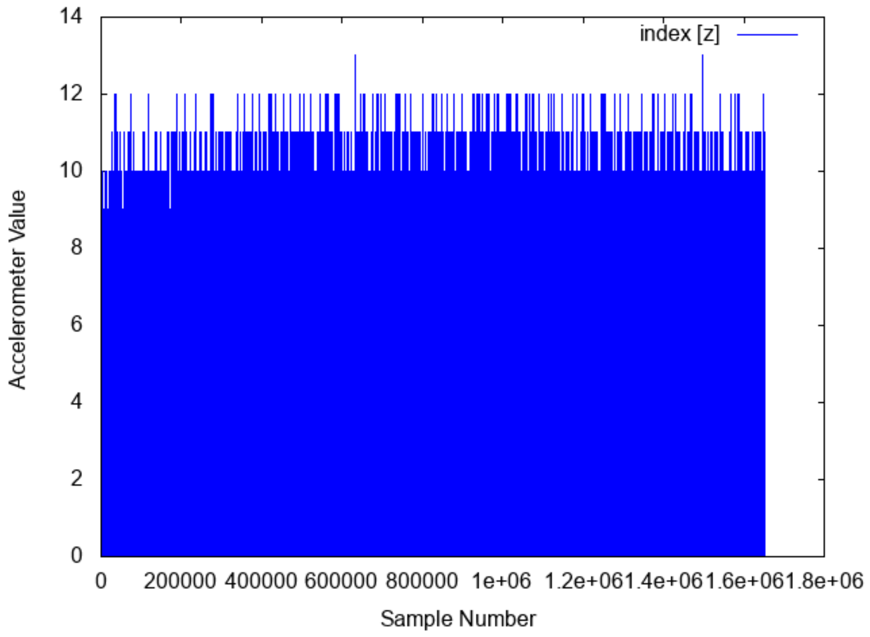
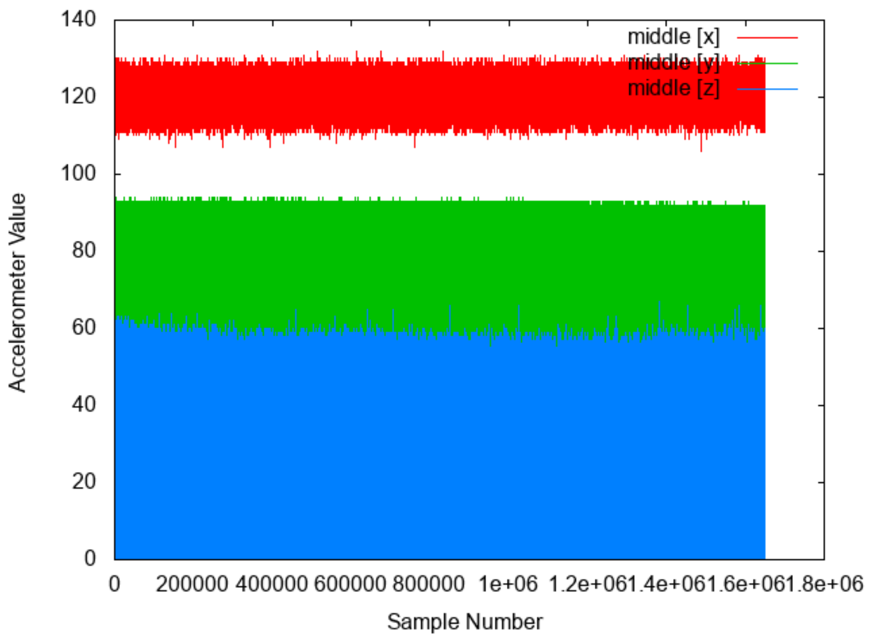
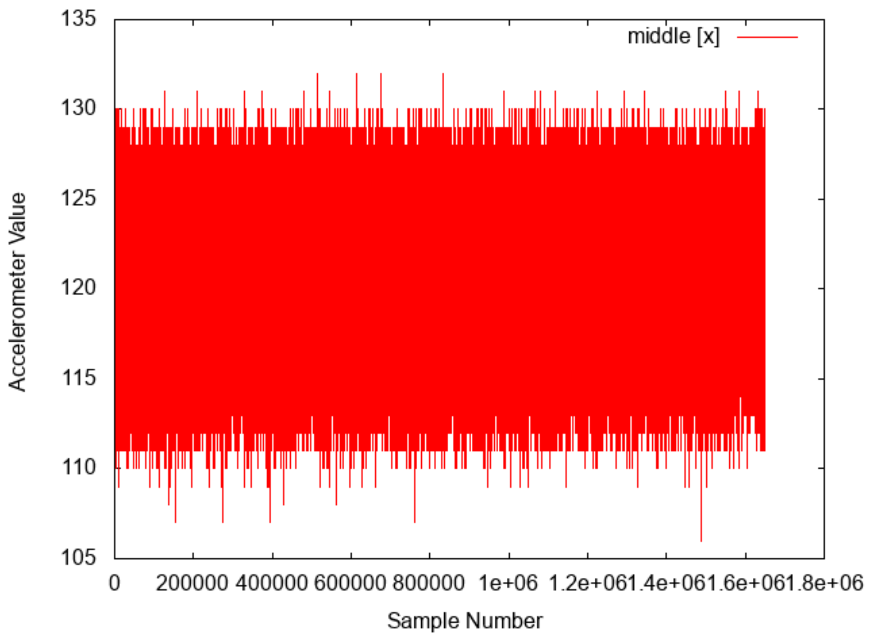


Figure B.3: Graph of middle finger data over 8 hours

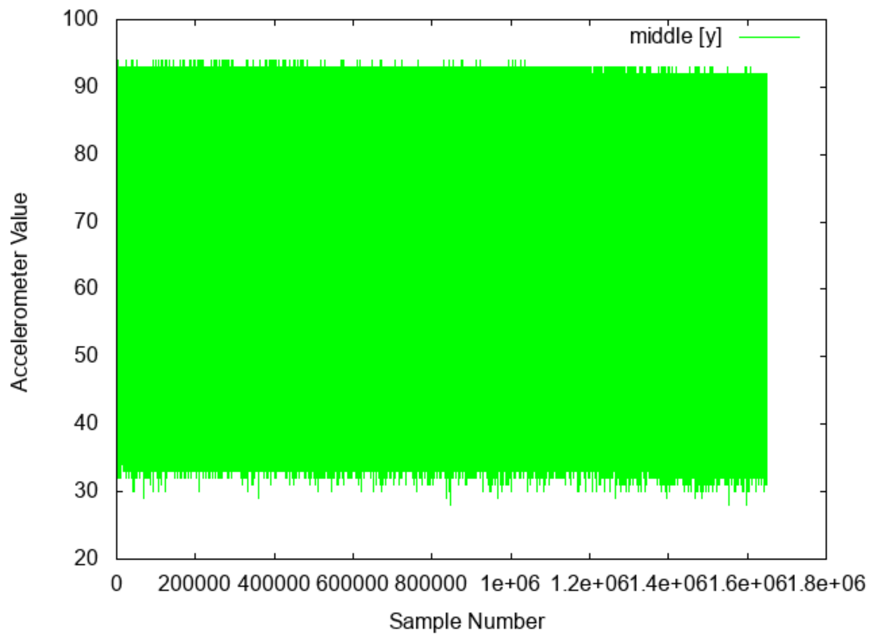
(a) Graph of middle finger data over 8 hours



(b) Graph of middle finger x-axis data over 8 hours



(c) Graph of middle finger y-axis data over 8 hours



(d) Graph of middle finger z-axis data over 8 hours

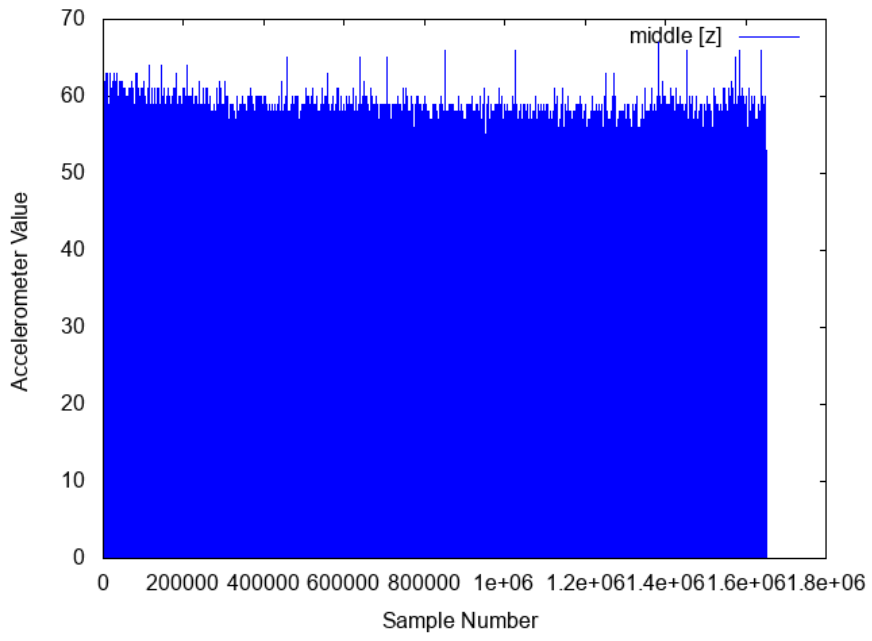
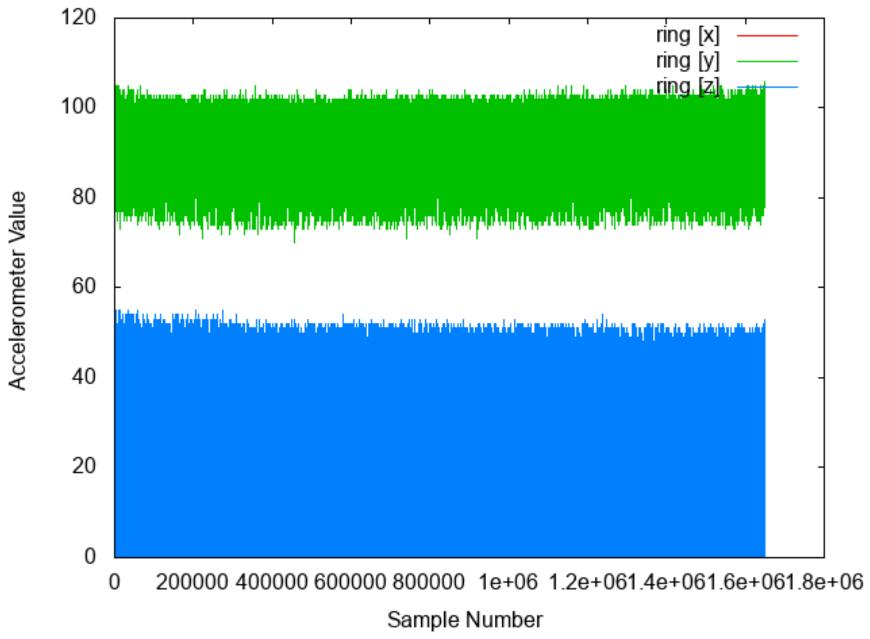
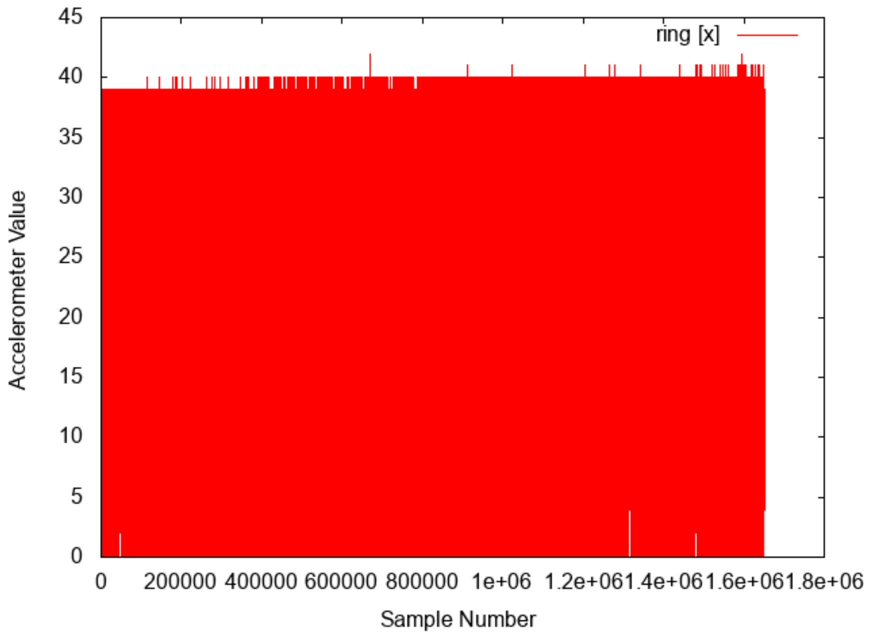


Figure B.4: Graph of ring finger data over 8 hours

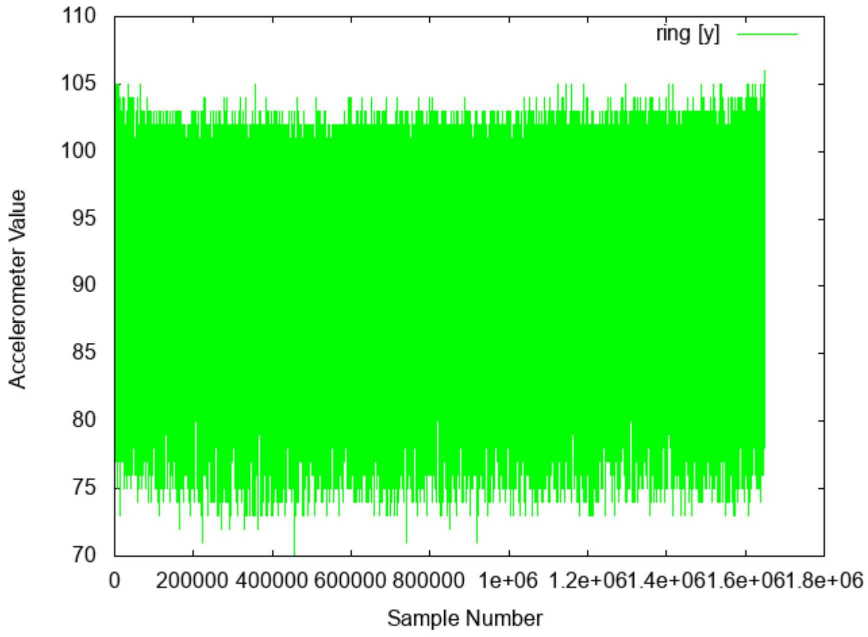
(a) Graph of ring finger data over 8 hours



(b) Graph of ring finger x-axis data over 8 hours



(c) Graph of ring finger y-axis data over 8 hours



(d) Graph of ring finger z-axis data over 8 hours

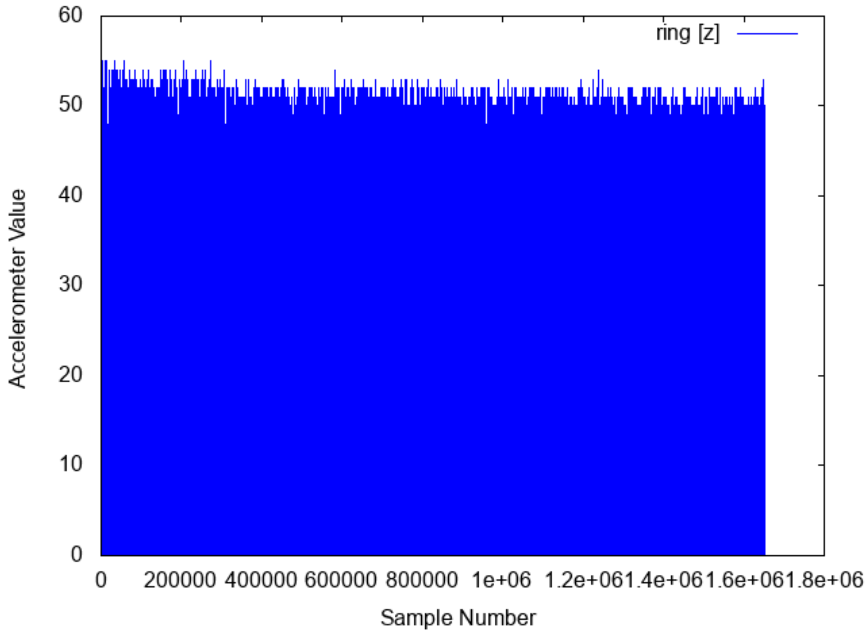
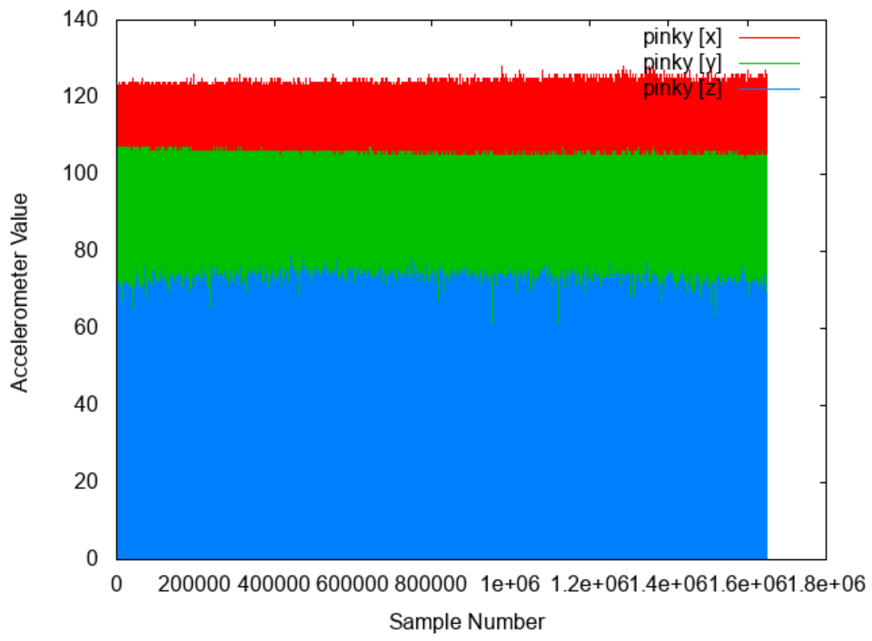
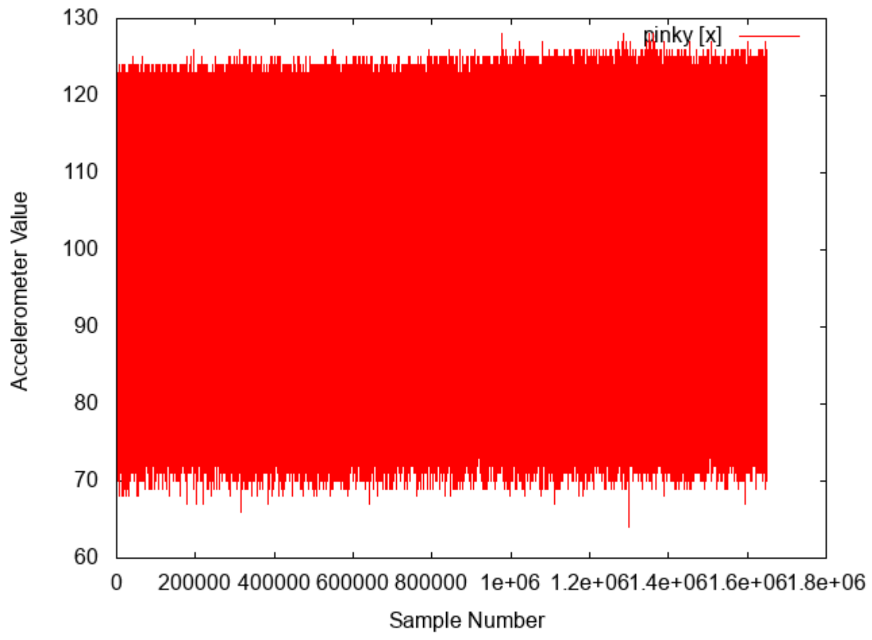


Figure B.5: Graph of pinky finger data over 8 hours

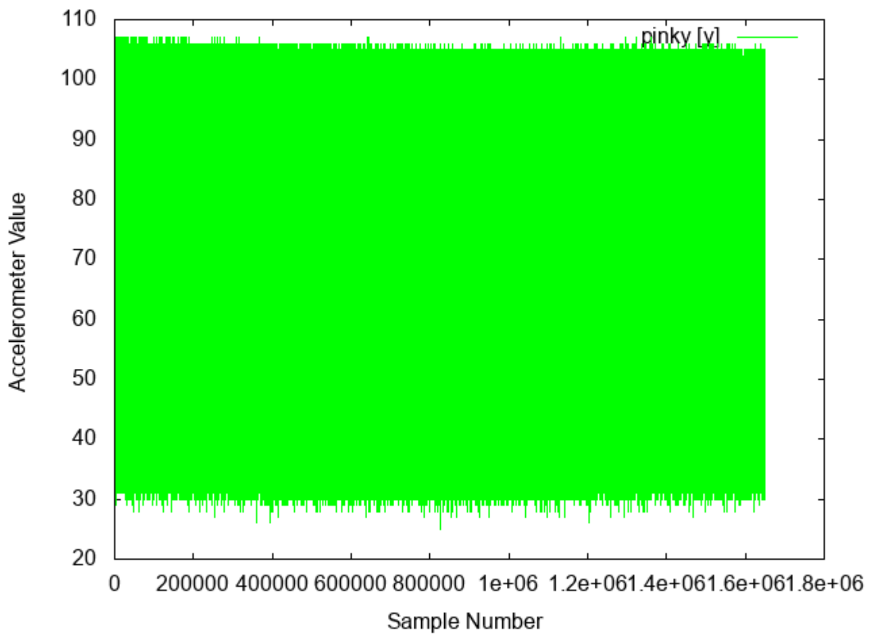
(a) Graph of pinky finger data over 8 hours



(b) Graph of pinky finger x-axis data over 8 hours



(c) Graph of pinky finger y-axis data over 8 hours



(d) Graph of pinky finger z-axis data over 8 hours

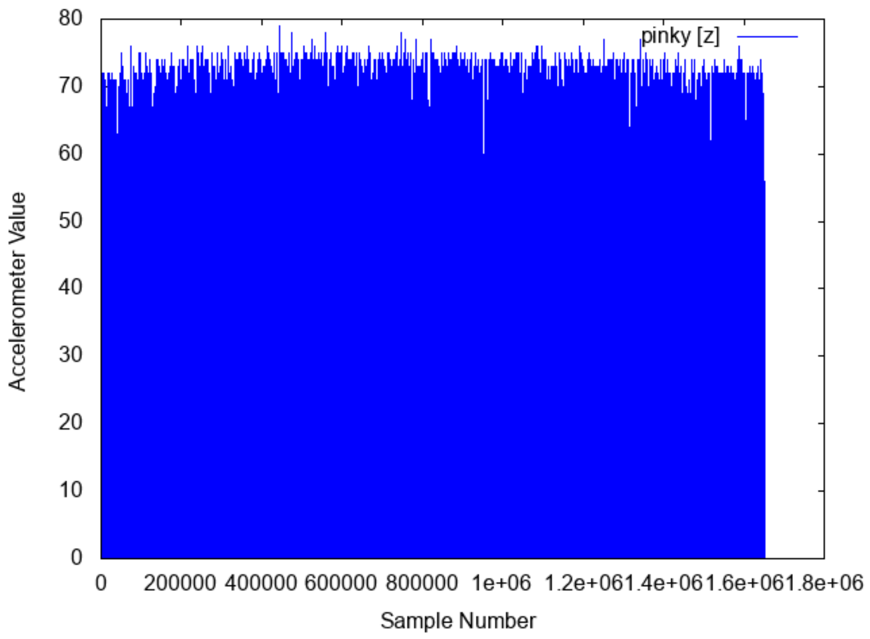
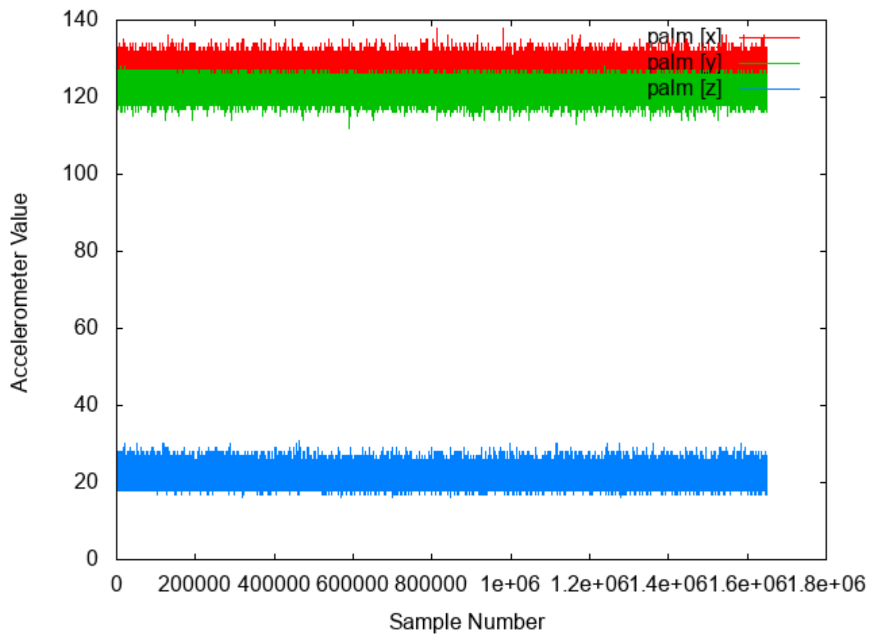
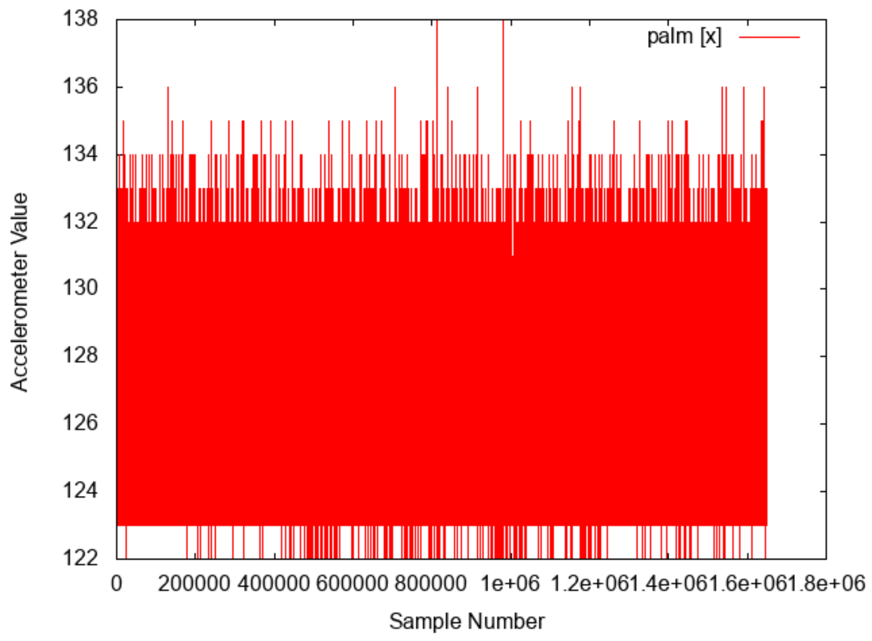


Figure B.6: Graph of palm data over 8 hours

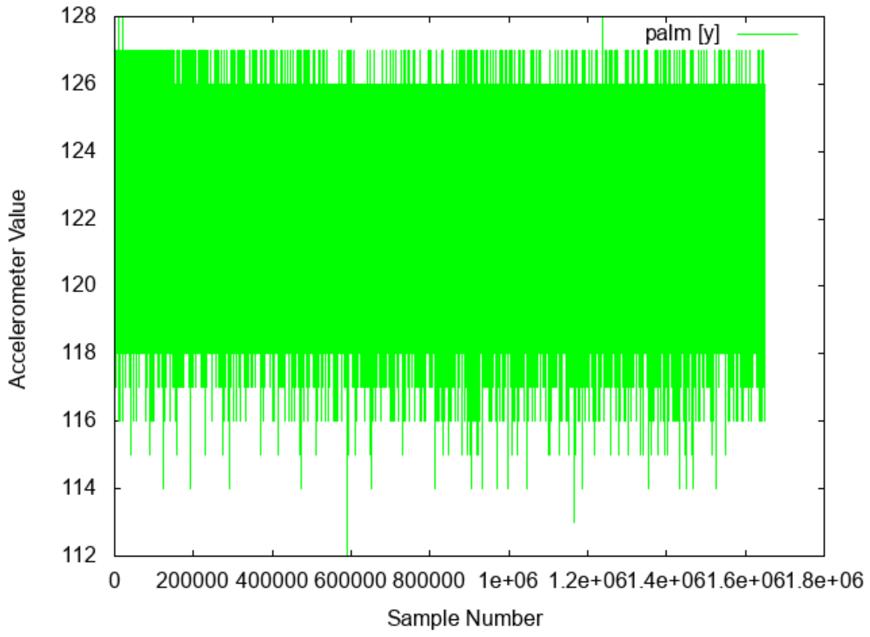
(a) Graph of palm data over 8 hours



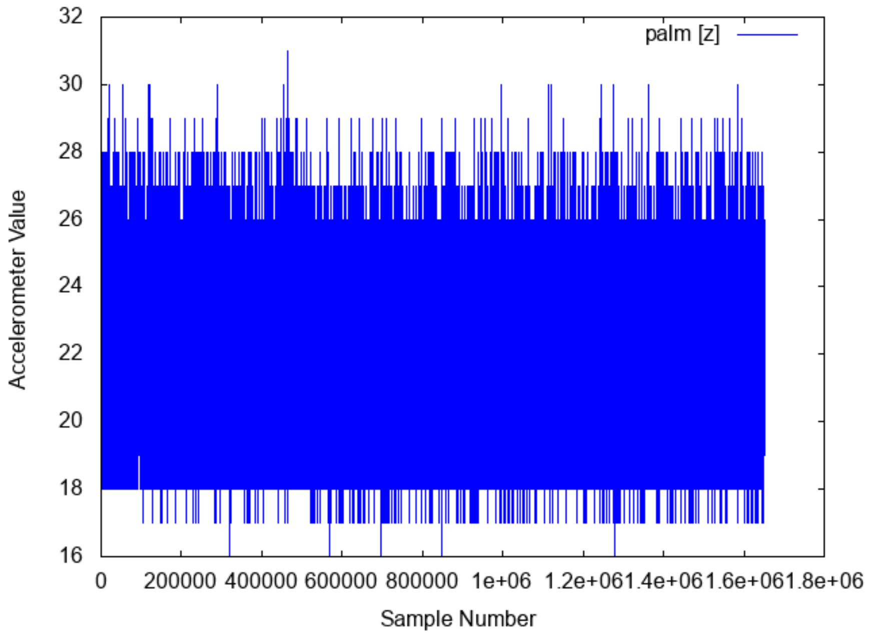
(b) Graph of palm x-axis data over 8 hours



(c) Graph of palm y-axis data over 8 hours



(d) Graph of palm z-axis data over 8 hours


palm [z]
Accelerometer Value
Sample Number
1e+06
1.2e+06
1.4e+06
1.6e+06
1.8e+06