

A GESTURAL HUMAN COMPUTER INTERFACE FOR SMART HEALTH

A THESIS IN
Computer Science

Presented to the Faculty of the University
Of Missouri-Kansas City in partial fulfillment
Of the requirements for the degree

MASTER OF SCIENCE

By
SOWMYA GINJUPALLI

B.Tech, Acharya Nagarjuna University, 2009

Kansas City, Missouri
2012

©2012

SOWMYA GINJUPALLI

ALL RIGHTS RESERVED

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled “A Gestural Human Computer Interface for Smart Health,” presented by Sowmya Ginjupalli, candidate for the Master of Science degree, and hereby certify that in their opinion, it is worthy of acceptance.

Supervisory Committee

Deendayal Dinakarpandian, Ph.D., Committee Co-Chair
School of Computing and Engineering

Yugyung Lee, Ph.D., Committee Co-Chair
School of Computing and Engineering

Praveen Rao, Ph.D.
School of Computing and Engineering

A GESTURAL HUMAN COMPUTER INTERFACE FOR SMART HEALTH

Sowmya Ginjupalli, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2012

ABSTRACT

For centuries, man was forced to live a highly active lifestyle with food being a precious commodity. Technological advances in the past few decades have resulted in increasingly sedentary lifestyles and a surfeit of calorie dense foods. This has resulted in a global epidemic of obesity and a host of associated health problems. One way to address this problem is to incorporate a higher level of physical activity into the workday. The objective of this thesis is to design a low cost gestural human computer interface for the recognition of vigorous gestures. We demonstrate that an action vocabulary of eight intuitive gestures can be recognized by the use of inexpensive accelerometers and a computationally simple approach involving Principal Component Analysis and Naïve Bayes classification. The accuracy is comparable to more computationally intensive approaches. The actions can be mapped to commands for controlling commonly used applications like e-mail and customized to individual preferences. There is a significant rise in pulse rate during these actions comparable to light aerobic activity. This has the potential to mitigate the harmful effects of sedentary work habits by raising the rate of metabolism with minimal impact on productivity.

TABLE OF CONTENTS

Contents	Page
ABSTRACT	iii
ILLUSTRATIONS.....	vi
TABLES.....	ix
Chapter	
1. INTRODUCTION	
1.1 Motivation	1
1.2 Problem Statement	1
2. RELATED WORK	
2.1 Different Devices and Software for Gesture Recognition	3
2.2 Chronos Watch	6
2.3 Chronos Flying Mouse	9
3. MODEL FOR GESTURE RECOGNITION	
3.1 Principal Components	12
3.2 Principal Components for Gesture Recognition	16
3.3 Naïve Bayes Classifier and Multivariate Gaussian Distribution Probability Densities....	22
3.4 Naïve Bayes Classification for Gesture Recognition.....	27
4. APPLICATION	
4.1 Application Architecture	33
4.2 Acceleration Data Collection from Chronos Watch	34
4.3 Filtering of the Acceleration Data	38

4.4	Segmentation of Data.....	42
4.5	Description of the Gestures	49
5.IMPLEMENTATION		
5.1	Overview of Implementation	55
5.2	Training of Gestures	56
5.3	Recognition of Gestures	68
6.EVALUATION OF GESTURAL HUMAN COMPUTER INTERFACE FOR SMART HEALTH		
6.1	Instructions to Use the Application.....	76
6.2	Evaluation.....	80
7.CONCLUSION AND FUTURE WORK		
7.1	Conclusion	95
7.2	Future Work	95
REFERENCE LIST.....		97
VITA		101

ILLUSTRATIONS

Figure	Page
2.1 eZChronos Watch	7
2.2 Chronos Control Center GUI.....	8
2.3 Using the eZ430-Chronos Watch with a PC.....	8
2.4 Chronos Control Center with Acceleration Data	9
2.5 Screenshot of the Chronos Flying Mouse Application	10
2.6 Chronos Flying Mouse Settings	11
3.1 Graph which Shows PCA Concept	12
3.2 First Principal Component and Second Principal Component for the Sample Data	15
3.3 3D Acceleration Readings Measured from an Accelerometer	16
3.4 Representation of the First Principal Component of the Acceleration Data.....	17
3.5 Acceleration Measure for Gesture G2.....	18
3.6 First Principal Component of the Acceleration Data for Gesture G2	19
3.7 Acceleration for the Test Gesture	20
3.8 First Principal Components of G1, G2 and Test Gesture	21
3.9 Three samples for Gesture G1.....	28
3.10 Three samples for Gesture G2.....	29
4.1 Architecture of Gestural Human Computer Interface for Smart Health	34
4.2 Log File with 3D Acceleration Values.....	37
4.3 Log File that Represents the Set of 3D Acceleration Values when the User is at Rest	38
4.5 Log File that Represents the Set of 3D Acceleration Values when the User Moves his Hands Slowly	39
4.6 3D Acceleration Values when a Heavy Gesture is Performed by the User	40

4.7 Log File Represents the Filtered Data.....	41
4.8 Log File and Graph when Two Gestures are Done Continuously	46
4.9 Segmented File and Graph Represents First Heavy Gesture.....	47
4.10 Segmented File and Graph Represents Second Heavy Gesture	48
5.1 A Matrix Constructed from a Text File	59
5.2 Average Principal Vector for Delete Gesture	61
5.3 Average Principal Vector for Send Gesture	62
5.4 Average Principal Vector for Scroll Down Gesture.....	63
5.5 Average Principal Vector for Scroll Up Gesture.....	64
5.6 Average Principal Vector for Punch Gesture	65
5.7 Average Principal Vector for Launch Gesture	66
5.8 Average Principal Vector for Search Gesture	67
5.9 Average Principal Vector for Open Gesture	68
5.10 Graph Showing First Principal Component of all the Gestures and Test Gesture.....	69
5.11 Application’s Output.....	70
5.12 Matrix for which Average Values are Found	71
6.1 Recognition Rate for the Gestures Done by an User Based upon Predefined Training	81
6.2 Recognition Rate of the Gesyures by a User’s Training.....	82
6.3 Average Recognition Rate for All 15 Volunteers	83
6.4 Percentage of Recognition of Gestures by Volunteers’ Training	84
6.5 Accuracy of All Gestures	85
6.6 Error Bar Chart Representing both Training Dependent and Independent.....	86
6.7 Accuracy of Pairs of Gestures Performed Continuously.....	87
6.8 Gestures Used for Evaluation for HMM Model Using Wii-mote.....	87
6.9 Average Recognition Rate of Gestures Using HMM Model	88

6.10 Graph Shows the Recognition Rate of the Reference Gestures of Our model Versus HMM model.....	89
6.11 Run Times for Different Modules in Application.....	90
6.12 Graph Shows Heart Rate of User at Three Different Events.....	93

TABLES

Table	Page
1.Training Set for Naive Bayes Classification.....	24
2.Calculated Parameters for the Training Set of Classification Example	25
3.Models Used for Gesture Classification	30
4.Description of Gestures and the Corresponding Email Activities	49
5.Description and Images of Gestures	51
6. Comparison of Run Time Models	91
7.Heart Rate of a user while he is at Rest	92
8.Heart Rate while Doing Conventional Actions in an Email Session for 8 min	92
9.Heart Rate whileDooing the Gestures for 10min	93

CHAPTER 1

INTRODUCTION

1.1 Motivation

The usage of personal computers and other computing devices is increasing day by day. There has been a substantial increase in the proportion of the workday that is based on their use. People interact with these computing devices by means of interfaces like keyboard, mouse, voice and touch screen. Interaction with computing devices can potentially result in two kinds of health problems. First, one can suffer from occupational injuries, which can result from repetitive actions and from resting parts of the body in unnatural positions for long periods of time. Second, humans can suffer from the effects of a sedentary lifestyle as the usage of the devices requires very less expenditure of energy. Two out of three US adults are overweight, and today's growing concern is childhood obesity [1]. A life style that combines a lack of physical activity with readily available calorie-dense foods has resulted in several health problems like obesity, diabetes and even cancer.

1.2 Problem Statement

People diet or exercise in order to prevent or overcome the health problems caused by weight gain. Because of time constraints or inability to afford gym memberships, it is very hard for many people to overcome the effects of a sedentary life style. On the other hand, several studies have clearly shown that even a mild exercise regimen for relatively short periods of time provides substantial health benefits [2]. This thesis exploits this observation by proposing a unique solution to overcome the health problems caused our collective obsession with personal computers and smartphones. The central idea is to raise the level of physical activity by replacing traditional interfaces like keyboard, mouse and touchscreen with alternative intuitive interfaces that require

considerable expenditure of energy. Apart from increasing energy expenditure, this idea can also provide entertainment to users, which can alleviate the monotony of repetitive work. Additionally, it can minimize occupational injuries by requiring the body to be mobile and not frozen in unergonomic positions for long periods of time. As the higher level of physical activity is integrated with routine tasks, there is less need to spend time and money on traditional exercise sessions.

This thesis presents a Gestural Human Computer Interface where users map heavy hand gestures to control an application. A heavy hand gesture is defined as movement involving the whole arm (rather than just fingers); it increases energy expenditure by involving the larger muscle groups. This can customize the interface to different user work habits and preferences. The effectiveness of such interfaces can be evaluated in terms of calorie expenditure, impact on productivity and acceptability.

Recognition of these gestures with minimal overhead in computational processing is the main concern, and is the core problem addressed by this thesis work. The Chronos watch from Texas Instruments, which has a 3-axis accelerometer embedded in it, is used as an input device for recording the gestures. This is a cost effective device and easy to use.

The models used for the recognition of these gestures are

1. Principal component Analysis
2. Naïve Bayes Classification.

CHAPTER 2

RELATED WORK

2.1 Different Devices and Software for Gesture Recognition

Several applications based on gesture recognition have been developed. Some of these are used for gaming, and others to facilitate certain activities. A variety of devices are used for gesture recognition, such as wired gloves, cameras and accelerometers. Different models are used to recognize gestures based on the requirements of applications.

Wired Gloves:

Wired Gloves or data gloves are used in many systems for hand gesture recognition [3]. Wired gloves are used in application areas such as sign language education, teaching deaf children or physically challenged people, simulating 3D molecules, proteins and DNA in the field of chemistry and biology, multi media education, music in education and physical education. Wired glove is a 3D input device and consists of sensors to determine 3D positional information. The position and orientation of the data gloves are determined by an electromagnetic transducer and strain gauge sensing devices. The data glove consists of small lightweight sensors. Each sensor is a fiber optic cable with a photo transistor at one end and an LED at the other. Hand movements are recorded by a Polhemus 3 Space 3 dimensional position and orientation sensor [4]. In most cases, a motion tracker such as a magnetic tracking device or inertial tracking device is attached to capture the global position or data rotation of the glove. Accompanying software is used to interpret these hand movements. These gestures can then be categorized into useful information, such as to recognize Sign Language or other symbolic functions. Wired glove is a part of haptic science. Haptic science is the science of applying tactile sensation to human interaction with computers [5]. A data glove can act as a haptic device because it simulates physical contact between the computer and the user, and it can also act as an output device. Wired gloves have only been available at a huge cost, with the

finger bend sensors and the tracking device having to be bought separately. There are a variety of gloves which can be used to capture hand motions. Motion capture gloves range in price from \$2000 per pair for wired and up to \$3500 per pair for wireless.

Video Tracking:

Video Signals are used as input for the recognition of gestures. An edge detection algorithm is usually applied to the digitized picture for the identification of hand, face or body position [6]. These edge detection algorithms are computationally expensive and dedicated hardware is also required. The latest research, Digits by Cambridge, utilizes a small camera for the gesture control offered by Kinect [7]. This camera has to be worn on a wrist strap and it tracks 2D movement rather than 3D. It recognizes finger movements that are used to control the software. This can therefore be used in applications such as translation of a sign language into a written text or controlling the slides of a presentation using simple finger movements. A company Leap Motion makes gesture control with a new type of motion controller [8]. The leap is a simple motion controller that can be plugged into any of the USB ports on a PC. Leap software has to be installed and once the motion controller is plugged in, it turns the 8 cubic feet of air in front of it into 3D interaction space. All the motions within that space are tracked. The camera is about the size of a business-card holder, and can discern all 10 fingers individually. Such detailed scanning allows us to do actions like actions like pinch-to-zoom, or zero in on fine details in a drawing application.

Accelerometer for Gesture Recognition:

Most of the previous work on gesture recognition is based on computer vision techniques. Lighting condition and camera angle/field of vision are important constraints of computer vision based approaches. In cases where there is poor lighting, it is very difficult to recognize gestures using a camera based system. Additionally, it is inconvenient to have to face the camera all the time.

Accelerometer-based gesture recognition is a technique that is compatible with almost all physical and computing environments. These devices are small and wireless, and hence easily wearable for interacting with a wide range of applications. Examples of such kinds of device include Nintendo Wii-mote and Chronos watch from Texas Instruments [9].

The Nintendo Wii-mote contains an integrated 3 axis acceleration sensor, and is connected via the Bluetooth human interface device protocol for transmitting data [10]. A Java library is available for gesture recognition with APIs for Bluetooth Wireless technology. It is a reusable and extensible gesture recognition library for the recognition process. The application uses hidden Markov models for gesture recognition [10].

In our application, heavy gesture based emailing, we have used Chronos watch from Texas Instruments which consists of a 3 axis accelerometer. This watch is easy to wear and interact with a wide range of applications, and this is also less expensive compared to the Wii-mote.

Models Used for Recognition:

Hidden Markov Model:

This is commonly used in many fields, e.g., Speech Recognition, Pattern recognition and Gesture Recognition. HMMs are stochastic models for data that is serial or temporal. The word "hidden" in the HMM refers to the hidden states that are mapped to the data. This model is typically used for modeling sequences of events. This model is particularly useful when the data is noisy and incomplete. It is based on estimating probability distributions and efficient algorithms for learning and recognition such as Baum-Welch and Viterbi algorithms [10].

Support Vector Machine:

This technique can also be used for the recognition of gestures [11]. It is a supervised linear classification method that has a property of maximizing margins between classes. With an appropriate choice of kernel functions, it also has nonlinear extensions. By using this method, 3 x 10

matrix is transformed into a linear vector with basic statistical elements like minimum, mean and length. These values can be used to define the dynamics and direction of a gesture. Z-normalization is used to make all the dimensions equal. In the recognition phase, the incoming movement is also transformed and is normalized in the same way as the training samples before SVM classifies the gesture.

Fuzzy Rule Based Method for Gesture Recognition:

Fuzzy Logic is a multivalued logic that allows intermediate values to be defined between conventional evaluations like yes or no, true or false, 0 or 1, etc. This is used for the recognition of hand gestures acquired from a data glove. An application for the recognition of gestures uses fuzzy rule based method in which sets of angles of finger joints are used for the classification of hand configurations. The set of all lists of segments of a given set of gestures is able to recognize every such gesture [12].

In our application, we have used a light weight process for the recognition of gestures other than Hidden Markov Model. The model which is used in our application is discussed in chapter 3.

2.2 Chronos Watch

Overview of Chronos Watch Used in Our Application:

eZ430-Chronos is a highly integrated, wearable wireless development system [9]. It may be used as a wireless sensor node for remote data collection. It consists of a three axis accelerometer for motion sensitive control. Figure 2.1 shows eZ430-Chronos watch.



Figure 2.1: eZChronos Watch

Acceleration Mode – RF: This mode requires the Chronos Control Center PC software. Figure 2.2 shows the GUI of the Chronos control center. The watch can be placed in the acceleration mode by pressing "#" until "ACC" is shown on the LCD. "Acc" mode provides a continuous transmission of 3D acceleration from the watch using TI's SimpliciTI protocol stack.

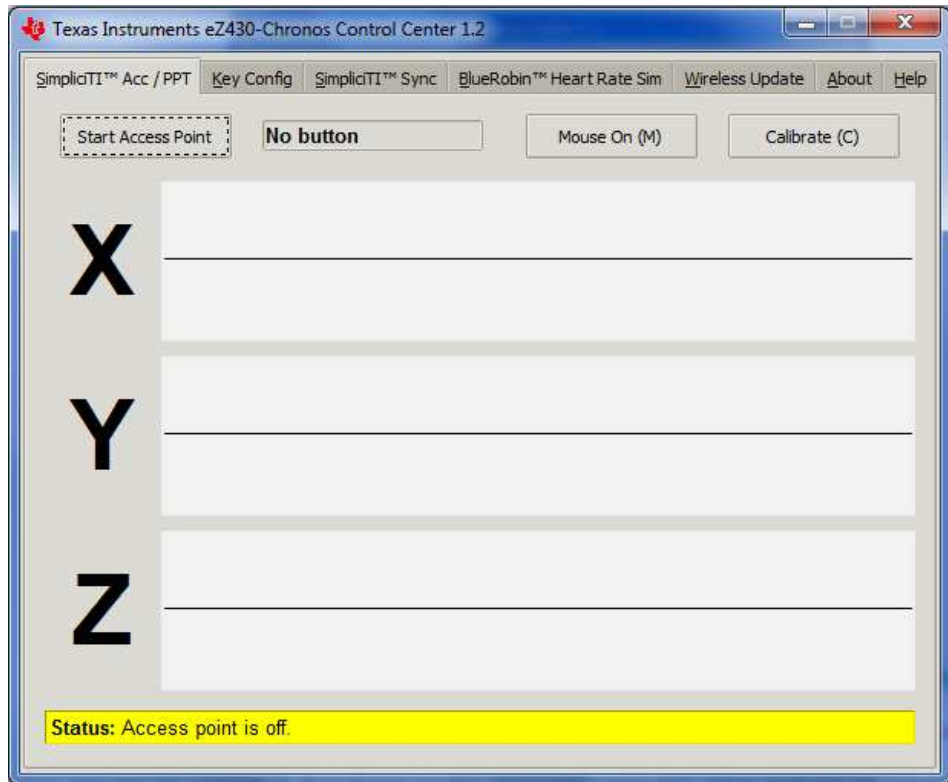


Figure 2.2: Chronos Control Center GUI

eZ430-Chronos RF Access Point: The RF access point is useful to communicate wirelessly with the Chronos directly from a PC so that data can be downloaded, or programs running on a computer can be controlled. It is based on the CC1111F32 controller, which features an integrated USB controller in addition to a 1-GHz radio.



Figure 2.3: Using the eZ430-Chronos Watch With a PC

Transmission of Acceleration Data: Acceleration data that is transmitted from the watch to PC can be seen in the control center in real time. Watch has to be switched to the acceleration mode, and the RF access point has to be connected to the PC. When “start access point button” is clicked in this interface, the watch is connected to the PC and 3D acceleration values are transmitted. Figure 2.4 displays the acceleration data of the watch for each axis.

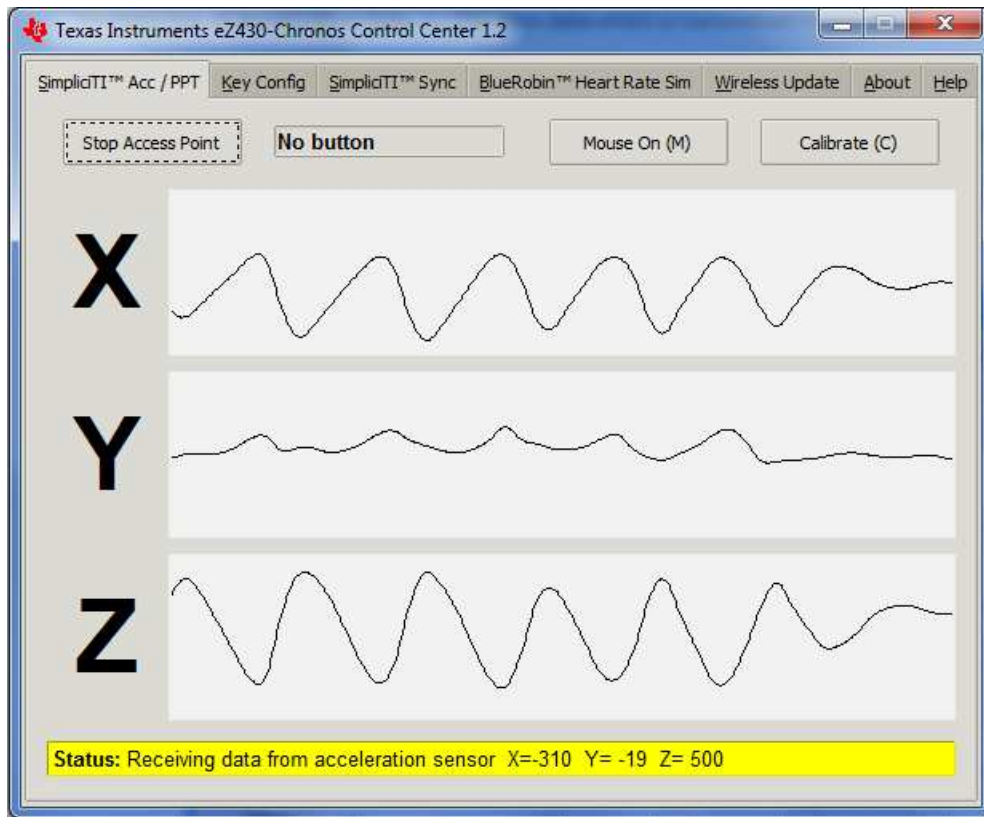


Figure 2.4: Chronos Control Center with Acceleration Data

2.3 Chronos Flying Mouse

The Chronos Flying Mouse is an innovative interface that can act as a high precision mouse or a mouse-based joystick for applications that do not support true joystick input [13]. It is designed to be a highly accurate, intuitive computer-input device. It has many customization options and offers

the abilities of several computer input devices without the position constraints of mounted hardware. The primary mouse mode of the Chronos Flying Mouse enables the user to control a cursor. This application is useful for presentations featuring interactive content, such as program demonstrations or interactive PowerPoint presentations with links. In our application, we have used the Chronos Flying Mouse code for data collection. Figure 2.5 shows the user interface of the Chronos Flying Mouse.



Figure 2.5: Screenshot of the Chronos Flying Mouse Application

This flying mouse has several joystick modes that allow gaming using a wide variety of household objects. This mode allows existing video games to interact with the Chronos without special programming and allows the Chronos to interact with a wide variety of applications.

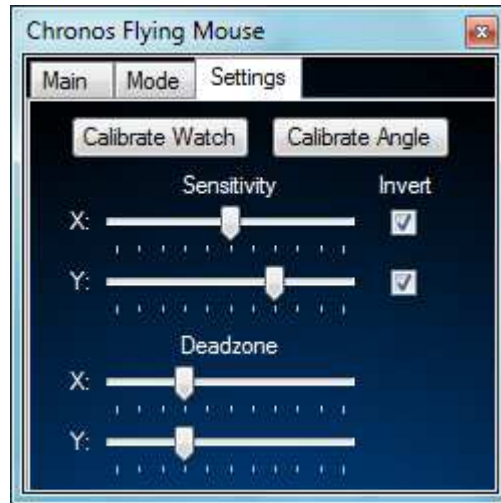


Figure 2.6: Chronos Flying Mouse Settings

Settings of Chronos Flying Mouse: This application offers axis independent sensitivity control and inversion. Because of this, a user can wear the watch on either the right or left hand. User can calibrate the orientation to any desired angle. Figure 2.6 shows the GUI for Chronos Flying Mouse settings. These settings are stored in an editable INI file without any registry modifications. In each run of the application, these settings are saved and loaded; the Chronos Flying Mouse can be therefore be safely deleted or moved to another computer without a complicated installation or uninstallation process.

Functionality of Chronos Flying Mouse: Accelerometer data collected from the chronos watch is sent through an exponential smoothing filter. This is used to remove noise or any unwanted high frequency components. A Chebyshev high-pass filter is used to identify snaps, or other sharp movements, in order to click. Euclidean angles are calculated from the acceleration from free-fall vector returned and various settings are applied to produce the appropriate mouse movement or joystick position. The most recent stable mouse position is calculated constantly, so that when a click from the high frequency data registers, the mouse is rewound to its stable position for the click. This allows for usable mouse clicks despite potential interference from unintentional motion.

CHAPTER 3

MODEL FOR GESTURE RECOGNITION

An accelerometer is embedded in many devices such as Wii-mote, Chronos watch etc. in order to measure triaxial acceleration. This measured acceleration data is used for gesture recognition. In our model, recognition is based on principal component analysis and Naïve Bayes classification.

3.1 Principal Components

The main use of Principal Component Analysis (PCA) is to reduce the dimensionality of a data set while retaining as much information as possible. Orthogonal transformation is used in principal component analysis to convert a set of values of possibly correlated variables into a set of values of linearly uncorrelated variables. The maximum number of principal components is equal to the number of original variables. So for the 3-axis acceleration data, the maximum number of principal components is equal to three.

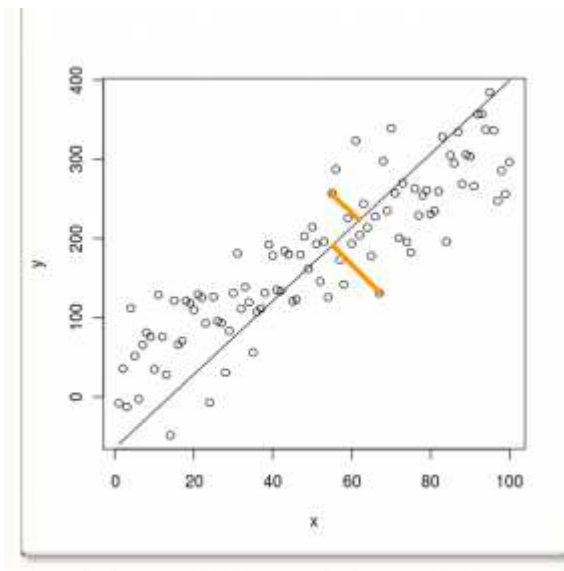


Figure 3.1: Visual depiction of the first Principal component

These principal components are perpendicular to each other, and the first principal component contributes the maximum variability to the data. PCA axis is the line that goes through the centroid of the data points and also minimizes the square of the distance of each data point to that line. See figure 3.1.

Thus, any large set of data can be represented as a single vector, the first principal component, pointing to the direction in which there is maximum variance in the data. The succeeding components have the highest possible variance under the constraint that they must be completely uncorrelated which implies that they are orthogonal to each other. Thus, a set of correlated variables are transformed into a set of uncorrelated variables which are ordered by reducing variability.

Calculation of Principal Component Coefficients

For the given set of data,

1. Covariance matrix has to be calculated.
2. Then the eigenvector vector of the covariance matrix has to be calculated.

Data =

7	4	3
4	1	8
6	3	5
8	6	1
8	5	7
7	2	9
5	3	3
9	5	8
7	4	5
8	2	2

Covariance of the data:

Cov(data) =

```
[ 2.3222222222 1.6111111111 -0.4333333333  
 1.6111111111 2.5000000000 -1.2777777778  
 -0.4333333333 -1.2777777778 7.8777777778]
```

Eigenvectors Vectors of the covariance matrix:

```
[-0.7017274262 -0.6990371198 -0.1375707982  
 0.7074570306 -0.6608891708 -0.2504596851  
 0.0841615661 -0.2730798586 0.9583027818]
```

This matrix contains the coefficients for principal components. Columns represent the coefficients for principal components. Column 1 represents the first principal component and column 3 represents the first principal component. Figure 3.2 shows a graph of data together with the first and second principal components.

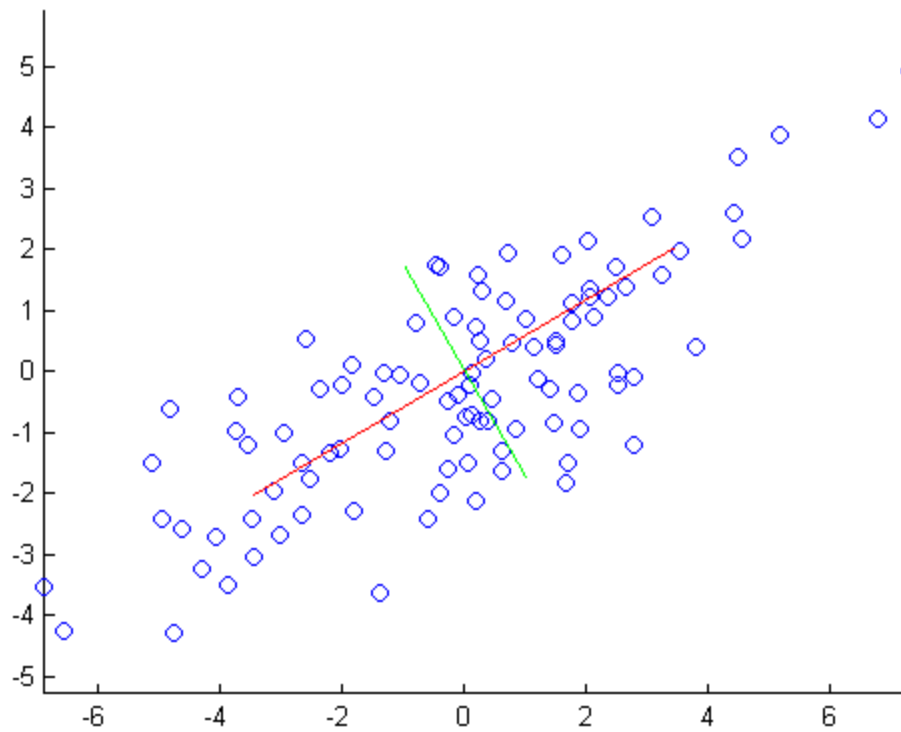


Figure 3.2: First Principal Component and Second Principal Component for the Sample Data

Angles Between Principal Components: From the above calculations, the first principal component vector is represented as,

$$f_c = -0.1375707982i - 0.2504596851j + 0.9583027818k$$

the second principal component vector is given by

$$s_c = -0.6990371198i - 0.6608891708j - 0.2730798586k$$

And the third principal component vector is represented as

$$t_c = -0.7017274262i + 0.7074570306j + 0.0841615661k$$

Let v_1 and v_2 be two vectors. By considering the dot product for the vectors, the angle between any two vectors can be calculated by the formula:

$$\cos \theta = (v_1 \cdot v_2) / (|v_1| |v_2|)$$

Based on the above formula,

Angle between first principal component and second principal component is 90 degrees

Angle between second principal component and third principal component is 90 degrees

Angle between first principal component and third principal component is 90 degrees.

3.2 Principal Components for Gesture Recognition

An accelerometer provides a set of 3D data that represents the acceleration of a limb performing the gesture. Figure 3.3 represents 3D acceleration for a gesture G1 measured by using an accelerometer:

X axis	Y axis	Z axis
-104	17	13
-82	28	-28
-45	-14	4
-8	-35	48
55	-42	100
126	-44	120
127	-50	107
126	-42	86
89	-50	70
23	-45	56
-102	-2	40
-128	4	26
-128	17	6
-90	19	5
-44	25	9
-22	28	9
-19	30	9

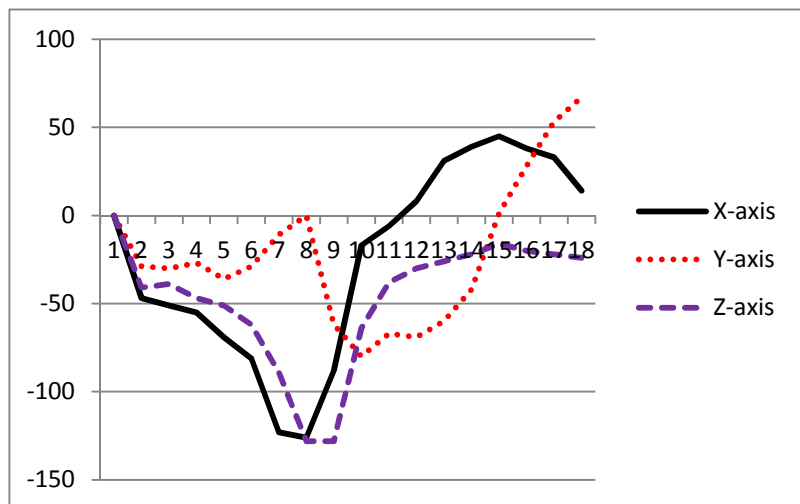


Figure 3.3: 3D Acceleration Readings Measured from an Accelerometer

By looking at the acceleration data we can say that the gesture is performed mainly in the x-z plane as there is a low magnitude of accelerations along the y-axis. We can calculate the first principal component for this data and then represent this whole data as a single vector.

First Principal component vector is $0.8825i - 0.2702j + 0.3850k$

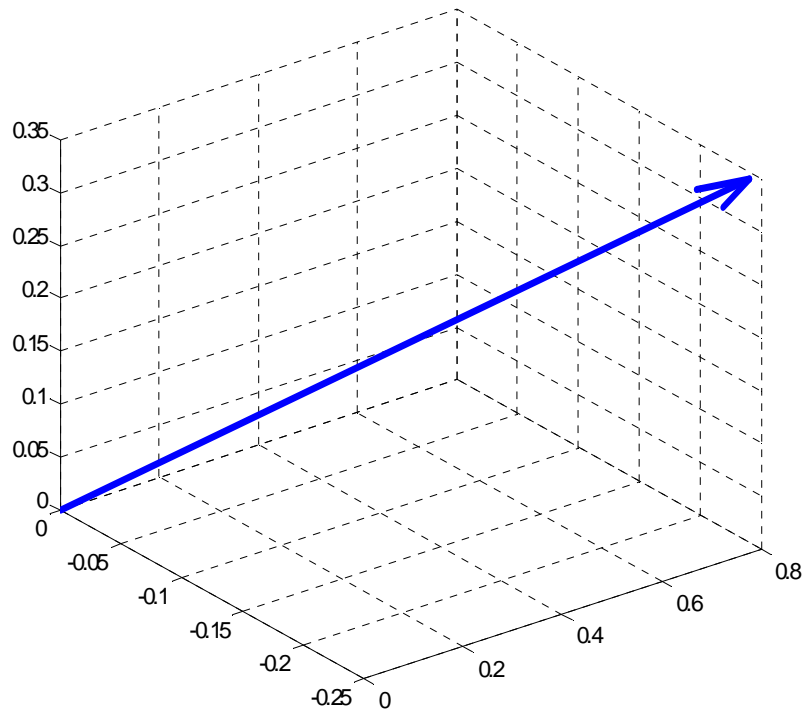


Figure 3.4: Representation of the First Principal Component of the Acceleration Data

Figure 3.5 represents 3D acceleration for another gesture G2 measured by using an accelerometer:

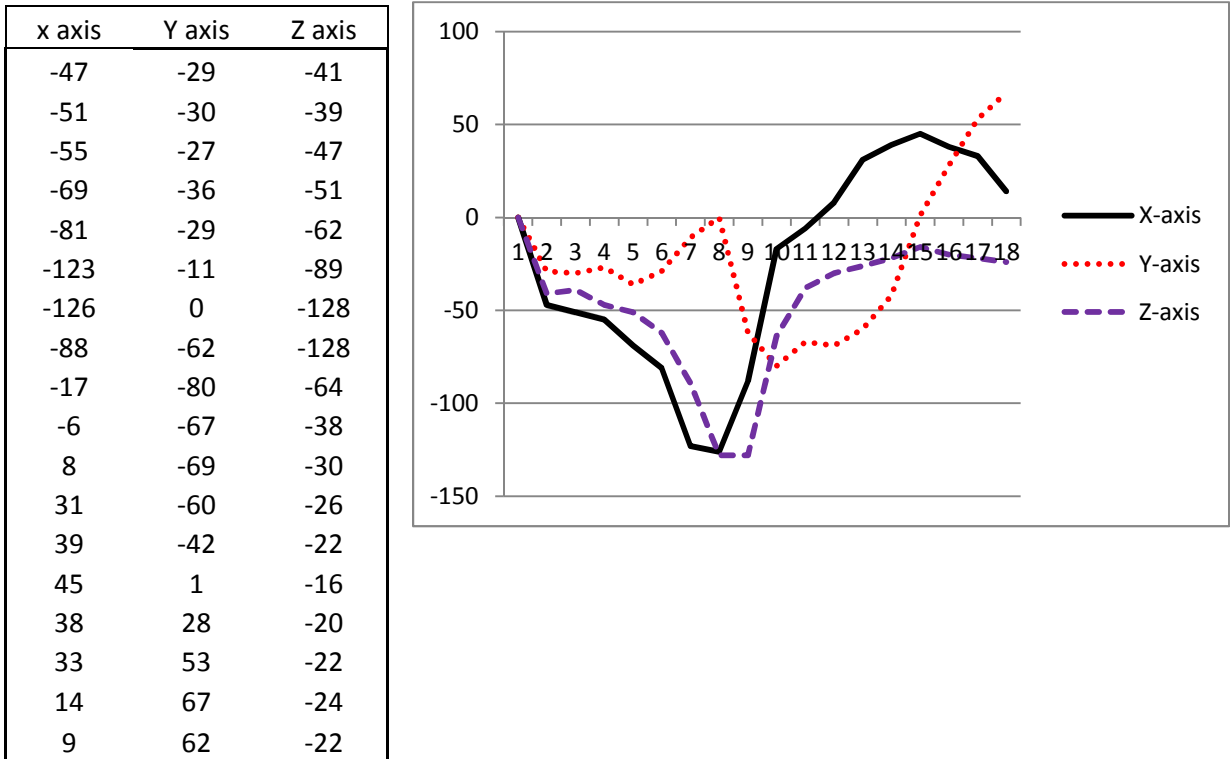


Figure 3.5: Acceleration Measure for Gesture G2

The first Principal component vector is $0.8256i + 0.3017j + 0.4768k$. Figure 3.6 represents the first principal component of the acceleration data for gesture G2

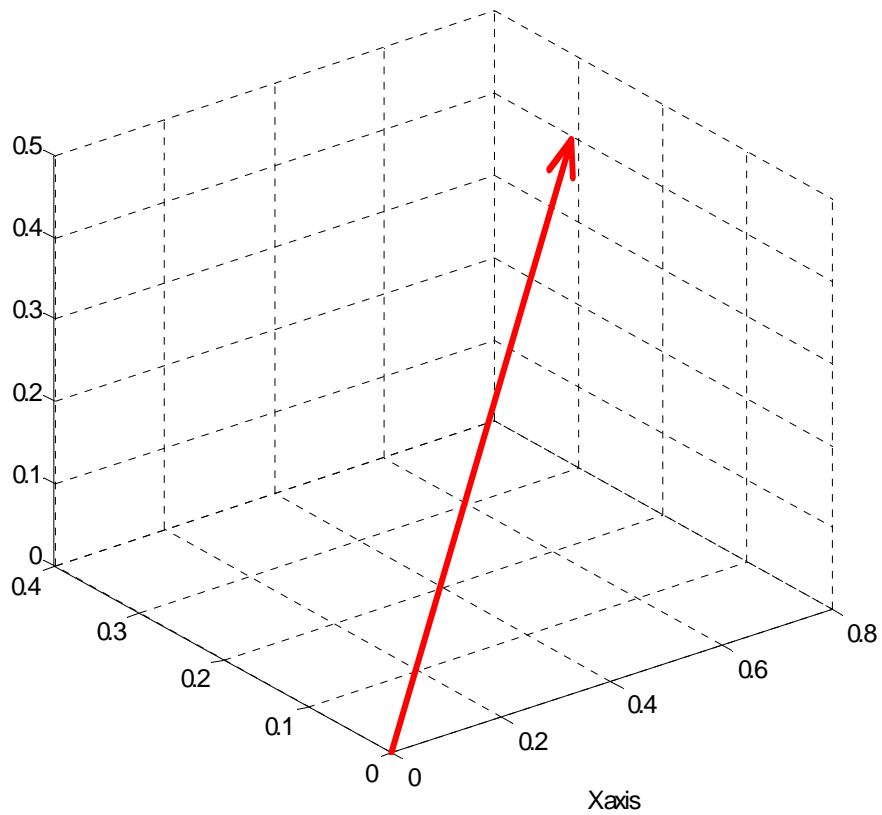


Figure 3.6: First Principal Component of the Acceleration Data for Gesture G2

Gesture Recognition by Considering Angles: When a gesture G1 or G2 is repeated as a test, we will get another set of acceleration data. For example if gesture G2 is performed again, the acceleration data is similar to the acceleration data we already obtained for G2 gesture. Figure 3.7 represents the acceleration for the test gesture:

X axis	Y axis	Z axis
8	29	11
8	22	9
11	9	10
10	-17	7
15	-48	2
8	-60	11
-5	-40	16
-6	-12	34
-21	52	62
-64	77	89
-68	78	94
60	47	97
112	57	124
51	34	111
-59	-37	8
-61	-67	-32
-42	-57	-39
-28	-10	-32
-19	10	-18
-10	13	2
-17	29	7
-14	42	6
-8	47	7
-16	47	11

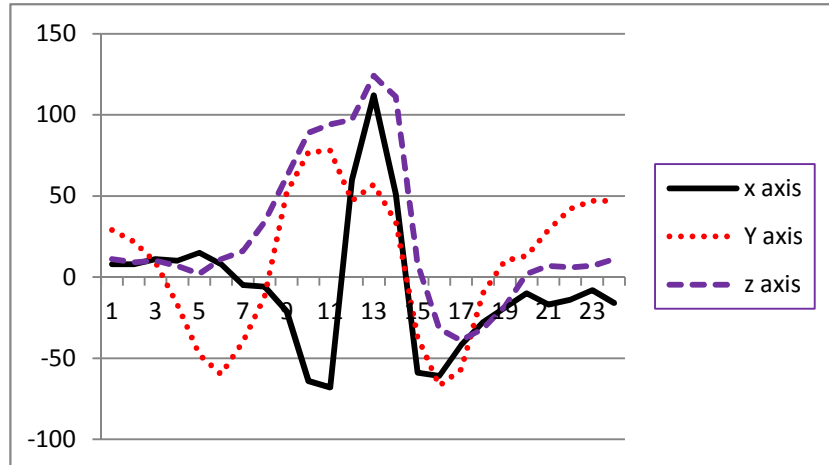


Figure 3.7: Acceleration for the Test Gesture

The first principal component vector for this test gesture is given by

$$0.3853i + 0.5948j + 0.7055k$$

Figure 3.8 represents the first principal components of G1, G2 and the test gesture superimposed in the same graph. The red line represents the test gesture. In order to classify this as G1 or G2 we can simply find the angle between

1. Test gesture and G1
2. Test gesture and G2

We can classify the test gesture as G1 or G2 based on the smaller of the two angles

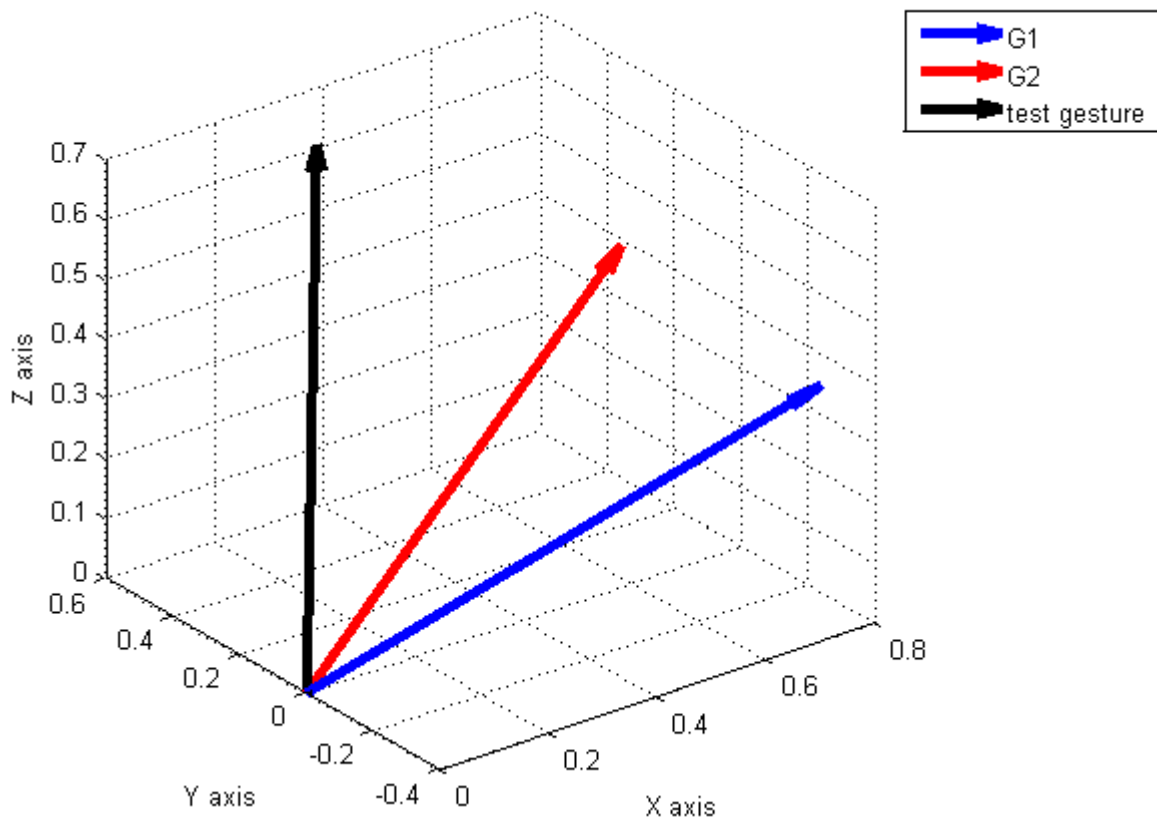


Figure 3.8: First Principal Components of G1, G2 and Test Gesture

1. Angle between Test gesture and G1 is 63.1968
2. Angle between Test gesture and G2 is 33.4925

AS the minimum angle is 33.4925, which is between test gesture and G2 we can say that the test gesture is classified as G2.

This is how we can recognize a gesture using principal components.

Limitation of Principal Component Analysis Method for Gesture Recognition: If two or more gestures are done in the same plane, in some cases there is a high probability of principal component vectors pointing in almost same direction as the maximum variance of the data may be in the same direction.

In such cases, we can consider another parameter along with principal components to classify the gesture. We consider magnitudes of the acceleration data.

3.3 Naïve Bayes Classifier and Multivariate Gaussian Distribution Probability Densities

The Naïve Bayes classifier is a statistical classifier. It can predict probabilities such as the probability that a given sample belongs to a particular class. This is based on the Bayes theorem formulation of conditional probabilities. In Bayesian interpretation, probability measures a degree of belief. The Bayes theorem links the degree of belief in a proposition before and after accounting for data or evidence. Let A is the proposition and B is the evidence.

Bayes theorem gives the relation between the probabilities of A and B , $P(A)$ and $P(B)$ and the conditional probabilities of A given B and B given A , $P(A|B)$ and $P(B|A)$ respectively.

For proposition A and evidence B ,

- $P(A)$, the *prior* is the initial degree of belief in A .
- $P(A | B)$, the *posterior* is the degree of belief having accounted for B .
- $P(B | A) / P(B)$ represents the support B provides for A .

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

Example: Suppose a customer visits a shop and leaves. Generally speaking, the probability that a customer is female is 50%. Someone tells the shop owner that the customer had long hair.

Intuitively, there is now a higher chance that the customer was female.

To calculate the probability that the customer is female, assume that

F represents the event that the person who came to the shop was female

L represents the event that the customer has long hair.

$P(F)$, the *prior*, the initial degree of belief in $F = 0.5$ (assume 50% of persons are females)

Let's say that 60% of the females have long hair.

$P(L|F)$, the probability of L given F is 0.6

Let's say that 30% of the males have long hair.

$P(L|M)$, the probability of L given M is 0.3.

$$\text{Then } P(F|L) = \frac{P(L|F)P(F)}{P(M)}$$

$$= \frac{P(L|F)P(F)}{P(L|F)P(F) + P(L|M)P(M)}$$

So the probability that the customer is female is obtained as 66.66%

The Bayesian Classifier is capable of calculating the most probable output depending on the input. It is possible to add new raw data at runtime and have a better probabilistic classifier. A naive Bayes classifier assumes that the presence of a particular feature of a class is unrelated to the presence of any other feature, given the class variable. Even if the features depend on each other or upon the existence of other features, a Naive Bayes classifier considers all of these properties to independently contribute to the probability. For features that are real numbers (rather than discrete values), probability densities can be used instead of probability mass functions. The Gaussian multivariate probability density is given by the formula below:

Probability density =

$$\frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

where k = number of dimensions.

Σ - covariance matrix of the multivariate normal distribution

$|\Sigma|$ - Determinant of the covariance matrix

$\boldsymbol{\mu}$ - Mean

X- Test data

Example 2: The problem is to classify whether a given person is male or female based on features Body Mass Index and foot size.

We need training data for the classification of persons. Table 1 shows a training set for person classification.

Table 1: Training Set for Naive Bayes Classification

Sex	Body Mass Index (BMI)	Foot Size
Male	21	11
Male	23	12
Male	20	10
Male	26	12
Female	18	6
Female	20	8
Female	21	9
Female	19	7

Based on the probability density formula, we have to calculate mean and variance for Body Mass Index and foot size data. Table 2 shows the mean and variance for Body Mass Index and foot size data for male and female.

Table 2: Calculated Parameters for the Training Set of Classification Example

	Male	Female
Mean (BMI)	22.5	19.5
Variance (BMI)	7	1.6667
Mean (foot size)	11.25	7.5
Variance (foot size)	9.1667e-01	1.6667e+00

Consider a test sample to be classified as male or female.

Sex	Body Mass Index	Foot Size (inches)
Sample	23	8

Based on the probability density formula we have to calculate,

$P(\text{BMI} | \text{male})$, Probability density of height in males

$P(\text{Foot size} | \text{male})$, Probability density of male foot size

$P(\text{BMI} | \text{female})$, Probability density of female when height is considered

$P(\text{Foot size} | \text{female})$, Probability density of female when foot size is considered.

Probability of male,

$$P(\text{male}) = P(\text{BMI} | \text{male}) * P(\text{Foot size} | \text{male})$$

Probability of female,

$$P(\text{female}) = P(\text{height} | \text{female}) * P(\text{Foot size} | \text{female})$$

$p(\text{BMI} \mid \text{male})$ is calculated by substituting the parameters in the probability density equation. In this case μ is 22.5 and σ^2 is 7. See Table 2. Here, we get a value which is greater than 1. As this is a probability density it can have a value greater than 1.

Similarly, the remaining parameters are also calculated by using the training set and mean and variance values shown in Table 2.

$$p(\text{BMI} \mid \text{male}) = 0.14811736$$

$$p(\text{foot size} \mid \text{male}) = 0.00131124$$

$$P(\text{male}) = 0.14811736 * 0.00131124 = 0.000194218$$

Similarly, $P(\text{female})$ can also be calculated.

$$p(\text{BMI} \mid \text{female}) = 0.00783395$$

$$p(\text{foot size} \mid \text{female}) = 0.28668838$$

$$P(\text{female}) = 0.00783395 * 0.28668838 = 0.002246$$

As in this case $P(\text{female}) > P(\text{male})$, we can say that the given sample is female

3.4 Naïve Bayes Classification for Gesture Recognition

The Naive Bayes classifier can be used for the recognition of gestures. With the help of an accelerometer, we can get a set of 3D data which represents the acceleration of our hands performing the gesture. In section 3.2, we have seen that Principal components can be used for gesture classification. We can therefore calculate the Probability density of a gesture by considering Principal components, P_{pca} , as one of the features.

Suppose if we have two gestures performed in the same plane and a major part of those gestures are performed in the same direction, then we will have the greatest variance in the same direction. Thus, we may have the first principal components of the two gestures in the same direction. In this case, Principal components alone may not be useful for the classification of gestures. So we have to consider another factor along with the principal components for classification. Magnitude of the acceleration values can be considered as another factor. The probability density of a gesture can be calculated by considering the average magnitude of x, y, z components of acceleration P_{mag} as follows:

$$\text{Probability density of a gesture} = P_{pca} * P_{mag}$$

Training Set: In order to make a training set for each gesture, a gesture has to be repeated a few times. If a gesture is repeated 3 times then we will get three samples of the gesture. With the help of an accelerometer, we will get several 3D acceleration datapoints for each sample.

Let's say a gesture G1 is repeated three times. We have to find the principal components and average magnitudes of x y z acceleration values. A training set is developed on these two features and based on that gesture classification will be done. Figure 3.9 shows three samples of gesture G1. First principal components and average magnitudes for x y and z axis are calculated for each sample. "Fc" represents the first principal component, and "Mag" represents the vector of average magnitude of x, y, z components of acceleration in our explanation.

Mag is given by,

$$\text{Mag}=[\text{Avg}(\text{Xaxis}), \text{Avg}(\text{yaxis}), \text{Avg}(\text{zaxis})]$$

X axis	Y axis	Z axis
-50	-26	1
-60	-25	10
-97	-10	4
-128	2	-33
-128	-5	-36
-128	1	-67
-128	-19	-78
-114	-35	-80
-79	-44	-99
-51	-51	-102
-39	-45	-104
-53	-75	-82
-8	-75	-84
-25	-80	-63
-22	-82	-41
-19	-81	-36
-26	-76	-4
-6	-61	-2

X axis	Y axis	Z axis
-50	-30	1
-77	-28	11
-128	-9	-10
-128	-28	-31
-128	-4	-91
-128	-39	-99
-128	-57	-110
-109	-65	-122
-74	-70	-121
-52	-82	-107
-21	-81	-109
-19	-83	-94
-3	-88	-85
-21	-84	-51
-24	-70	-23
3	-61	-25
-37	-69	11
-18	-61	49

X axis	Y axis	Z axis
-46	-27	-16
-61	-25	-12
-79	-19	-5
-128	-15	-16
-128	-13	-24
-128	11	-40
-128	-15	-61
-128	-37	-75
-124	-45	-85
-92	-48	-93
-65	-68	-102
-58	-92	-108
-38	-114	-113
-43	-103	-88
-40	-97	-49
8	-80	-44
-38	-70	-7
-1	-68	26

$$F_c = 0.593i - 0.095j + 0.799k$$

$$\text{Mag} = [-16.88 \quad -12.16 \quad 6.6]$$

$$F_c = 0.623i - 0.138j + 0.769k$$

$$\text{Mag} = [0.4 \quad -24.95 \quad 12.5]$$

$$F_c = 0.560i - 0.1075j + 0.821k$$

$$\text{Mag} = [-13.48 \quad -16.70 \quad 20.48]$$

Figure 3.9: Three Samples for Gesture G1

Now let's say gesture G2 is also performed three times to form the training set. Figure 3.10 represents three samples of gesture G2, their first principal components and vector of average magnitudes of x, y and z components.

X axis	Y axis	Z axis
-73	14	-30
-38	29	-33
-44	23	-23
-42	18	-16
-54	7	12
-58	-3	5
-59	9	-27
-53	1	-23
-64	-6	-25
-83	5	-61
-15	56	-25
-33	21	44
-37	39	40
-105	111	61
-26	118	84
-17	79	83
-14	62	27
10	30	-3
-21	-57	-74
-51	-128	-128
-128	-97	-31
32	0	-9
17	14	-8

X axis	Y axis	Z axis
-63	25	-21
-50	24	-61
-55	20	-11
-43	5	-28
-87	-23	7
-27	19	13
-51	70	8
-70	87	64
-26	111	87
-29	103	74
1	78	46
4	34	2
-12	-32	-93
-96	-125	-128
-128	-125	-85
58	-47	21
-35	31	-22
38	-41	14
-11	-13	-18
-93	6	-41
-76	-8	-24
-50	-11	-23
-52	1	-18

X axis	Y axis	Z axis
-48	18	-37
-43	17	-15
-71	-4	16
-59	-1	-3
-45	8	-15
-48	9	-11
-74	11	-76
-63	29	15
7	3	48
-31	62	69
-67	52	60
3	86	42
-8	30	6
30	-14	-37
-128	-128	-101
25	22	-23
-86	64	-62
-20	-7	-10
10	-21	-10
-66	1	-21
-87	0	-24
-66	0	-7
-56	10	-7

$$F_c = 0.152i + 0.723j + 0.673k$$

$$\text{Mag} = [-48.75 \quad -4.931 \quad -15.56]$$

$$F_c = 0.202i + 0.706j + 0.678k$$

$$\text{Mag} = [-48.11 \quad 6.67 \quad 12.22]$$

$$F_c = 0.104i + 0.680j + 0.725k$$

$$\text{Mag} = [-48.5 \quad 7.44 \quad 12.88]$$

Figure 3.10: Three samples for Gesture G2

A set of classification models can be formed. Table 3 shows the models for gesture classification.

Table 3: Models used for Gesture Classification

Gesture	Principal component matrix	Magnitudes matrix
G1	[0.593 -0.095 0.799	[-16.88 -12.16 6.6]
	0.623 -0.138 0.769	0.4 -24.95 12.5
	0.560 -0.1075 0.821]	-48.5 7.44 12.88]
G2	[0.152 0.723 0.673	[-48.75 -4.931 -15.56
	0.202 0.706 0.678k	-48.11 6.67 12.22
	0.104 0.680 0.725k]	-48.5 7.44 12.88]

Consider a test gesture G_T . It has to be classified as G1 or G2.

Probability density of a gesture for a given class = $P_{pca} * P_{mag}$

Calculation of Probability Density of a Gesture G1:

$$\text{Probability density} = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

Where $K=3$ (3 Dimension)

Σ - covariance matrix of the multivariate normal distribution

$|\Sigma|$ - Determinant of the covariance matrix

μ - Mean

x - Test data

For the calculation of P_{pca} , the principal component matrix for G1 is taken from the training set

$$\text{Principal component matrix} = \begin{bmatrix} 0.593 & -0.095 & 0.799 \\ 0.623 & -0.138 & 0.769 \\ 0.560 & -0.1075 & 0.821 \end{bmatrix}$$

$$\Sigma = 1.0e-03 * \begin{bmatrix} 0.9930 & -0.4665 & -0.8170 \\ -0.4665 & 0.4893 & 0.4335 \\ -0.8170 & 0.4335 & 0.6813 \end{bmatrix}$$

$$|\Sigma| = -1.2722e-26$$

$$\Sigma^{-1} = 1.0e+19 * \begin{bmatrix} -1.3916 & 0.3476 & -1.8898 \\ 0.3476 & -0.0868 & 0.4721 \\ -1.8898 & 0.4721 & -2.5665 \end{bmatrix}$$

$$\mu = [(0.593+0.623+0.560)/3 \quad (-0.095-0.138-0.1075)/3 \quad (0.799+0.769+0.821)/3]$$

$$\mu = [0.592 \quad -0.1135 \quad 0.796]$$

For the test gesture G_T , the principal component is [0.170 0.747 0.6416]

$$\text{So } x = [0.170 \ 0.747 \ 0.6416]$$

By substituting all these values in the probability density formula we get,

$$P_{pca} = 0$$

Similarly, P_{mag} is also calculated by taking magnitude matrix for G1 from the training set.

$$P_{mag} = 3.080328002979498E-49$$

$$\text{Probability density of G1} = P_{pca} * P_{mag}$$

$$= 0 * 3.080328002979498E-49$$

$$= 0$$

Calculation of Probability of Gesture G2:

P_{pca} is calculated by taking the principal component matrix from the training set, and is obtained as

$$P_{pca} = 225.03729944998364$$

P_{mag} is calculated by taking magnitude matrix from the training set, and it is obtained as

$$P_{mag} = 9.524782203457256E-6$$

Probability density of gesture G2 = $P_{pca} * P_{mag}$

$$= 225.03729944998364 * 9.524782203457256E-6$$

$$= 0.0021434312649152854$$

The test gesture is classified as the gesture that has the highest class-dependent probability density.

As the probability density of gesture G2 is greater than the probability density of gesture G1, we can say that the test gesture is G2.

CHAPTER 4

APPLICATION

4.1 Application Architecture

In Gestural Human Computer Interface for Smart Health, the user wears the Chronos Watch and performs a set of predefined gestures for common activities in e-mail. Activities include composing an email, sending an email, deleting an email, forwarding an email, searching for emails in the Inbox, opening an attachment, scrolling up and scrolling down. The Chronos watch has an ultra-small size, low power 3D accelerometer that measures acceleration relative to free-fall. The RF access point allows it to wirelessly communicate with the Chronos directly from the PC. Figure 4.1 shows the application architecture. The acceleration values are written to a text file by using the Chronos Flying Mouse application. As the Chronos watch continuously senses the acceleration of the hands, some undesired values are also written into the file, corresponding to accidental hand movements. The file is parsed to filter undesired acceleration values. The filtering is based upon the difference between two consecutive 3D acceleration values, which are referred to as delta values. As the idea of the application is to perform heavy gestures, only heavy movements of the hands should be considered. When the user performs a heavy gesture, the magnitudes of the acceleration values will be high but will be low and almost constant if he moves hands slowly. Thus only the acceleration values that correspond to the actual heavy gesture will be considered, with the remaining unwanted values being discarded. The filtered data corresponding to one heavy gesture is considered as one segment. The log file into which the sensed acceleration values are written is monitored continuously and divided into segments. Each segment is sent to the Recognition algorithm. Recognition is based on the magnitudes of the 3D acceleration values and the first principal component of the segment. The matching algorithm interprets the gesture performed by the user and maps it to the corresponding activity in the email.

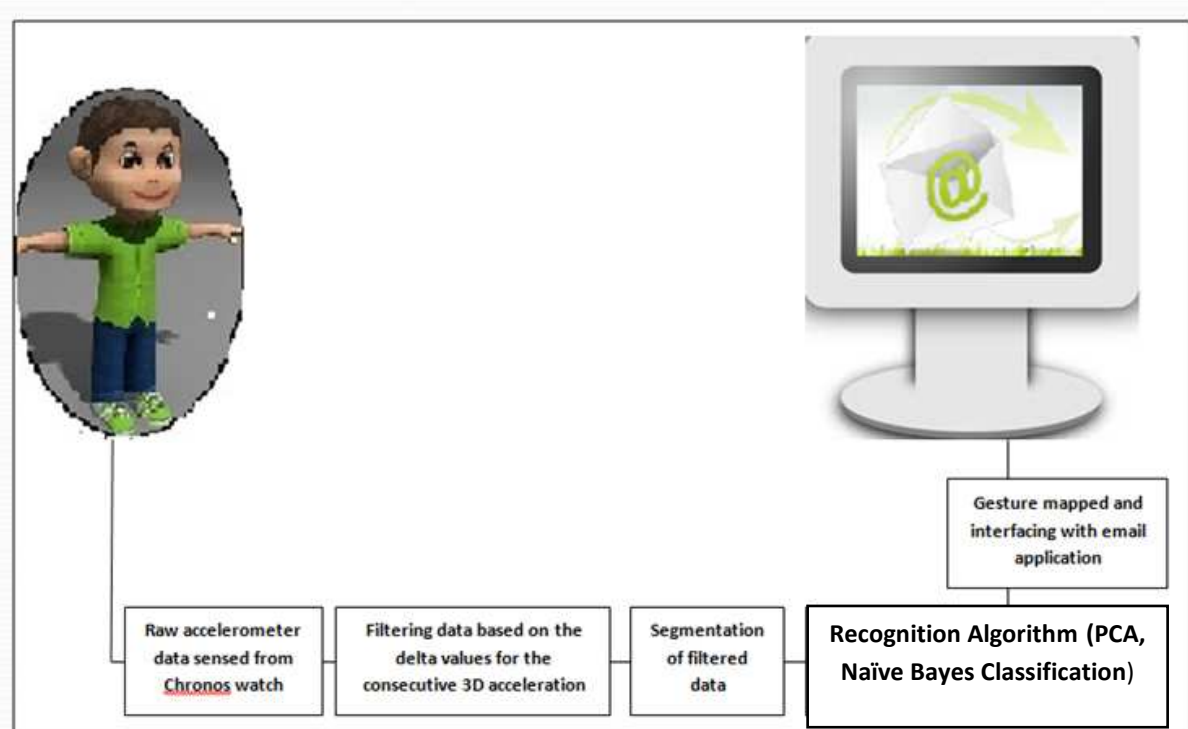


Figure 4.1: Architecture of Gestural Human Computer Interface for Smart Health

4.2 Acceleration Data Collection from Chronos Watch

In our application, we have used Chronos Flying Mouse application source code in order to collect three dimensional acceleration data measured by Chronos watch. User has to tie the watch and switch it to acceleration mode. The action is started by clicking on the start button on the Chronos Flying Mouse window. The RF access point allows one to wirelessly communicate with the Chronos directly from the PC. The code snippet that is used to check whether the RF access point is connected correctly to the PC is reproduced below.

```

int start_com()
{
    if (is_com) return 0;
}
  
```

```

is_com = BM_GetCOM(com_port);
is_com = BM_OpenCOM(com_port, 115200, 30, false, false);
if (!is_com)
{
    set_status("Plug in RF point");
    return -1;
}

if (is_ppjoy)
    set_status("RF point and PPJoy found");
else
    set_status("RF point found; no PPJoy found");

return 0;
}

```

During that communication, the raw accelerometer data is passed through an exponential smoothing filter for the smooth cursor movement in the flying mouse application. While passing the raw accelerometer data to its own filter we capture the acceleration data and write it to a text file called "log.txt." The data from the file is then passed to a filter to discard unwanted acceleration values which correspond to involuntary small hand movements that do not constitute the heavy gesture. The code used to collect 3D acceleration into our "log.txt" file is given below and the number of values recorded per second is 30.

Code Snippet for Acceleration Data Collection from Chronos Flying Mouse Application:

Void transform(DWORD spldata, of stream raw data) **// function used to collect 3D acceleration values**

```

{
    vector3 vdata = tovector3(spldata);           // Our Edited code . vector3 is a class
                                                    //contains x y z public data members

    ifstream fin("path\\filename.txt");

    while(!fin.eof())

    fin>>d1>>x1>> y1>> z1;
}

```

```
    fin.close();  
    fopen_s(&mydatalog1,"path /filename.txt","a");  
    count++;  
    sprintf(mycount, "%d ", count);  
    sprintf(myx, "%f ", vdata.x );  
    sprintf(myy, "%f ", vdata.y);  
    sprintf(myz, "%f\n ", vdata.z);  
    fprintf(mydatalog1,mycount);    //writing 3D acceleration values to a file  
    fprintf(mydatalog1,myx);  
    fprintf(mydatalog1,myy);  
    fprintf(mydatalog1,myz);  
}
```

This code writes the stream of 3D acceleration values into the log file. Figure 4.2 shows these values

X axis	Y axis	Z axis
-65	30	5
-91	15	-53
-88	2	-52
-62	-11	-22
-8	-18	42
55	-15	96
110	-20	116
102	-28	103
121	-35	87
127	-30	64
96	-31	53
43	-35	67
-56	-33	78
-122	-3	50
-128	12	7
-94	30	-7
-45	40	-8
-12	39	-1
-8	38	4
-15	37	6
-15	38	7
-22	31	7
-20	33	5
-20	35	6
-18	36	7
-12	36	11
-3	32	16
-16	48	13
-11	26	20
-8	30	20
-10	38	16
-7	39	14
-6	38	13
-13	37	11
-6	36	13
-7	34	13

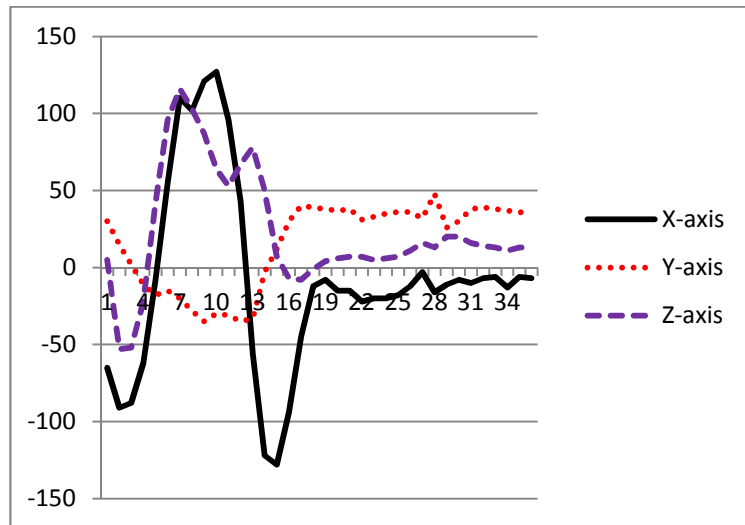


Figure 4.2: Log File with 3D Acceleration Values

4.3 Filtering of the Acceleration Data

Filtering: Acceleration data is passed through our own filter in addition to the exponential smoothing filter of the Chronos Flying Mouse application. This filter is used in order to distinguish the actual heavy gesture from the normal hand movements of the user and incomplete gestures. Filtering is done based on the absolute difference between consecutive 3D acceleration values which are referred to as delta values. As the user has to perform a heavy gesture to perform an email activity, the 3D acceleration values have to be large by definition. When user is at rest, the acceleration will almost be zero in x and y axis, and a value of 1g in the z axis. This is because the accelerometer of the Chronos watch measures acceleration in units of gravity. Figure 4.3 shows an excerpt of 3D acceleration values from the log file when the user is at rest:

X axis	Y axis	Z axis
-12	39	-1
-8	38	4
-15	37	6
-15	38	7
-22	31	7
-20	33	5
-20	35	6
-18	36	7
-12	36	11
-3	32	16
-16	48	13
-11	26	20
-8	30	20
-10	38	16

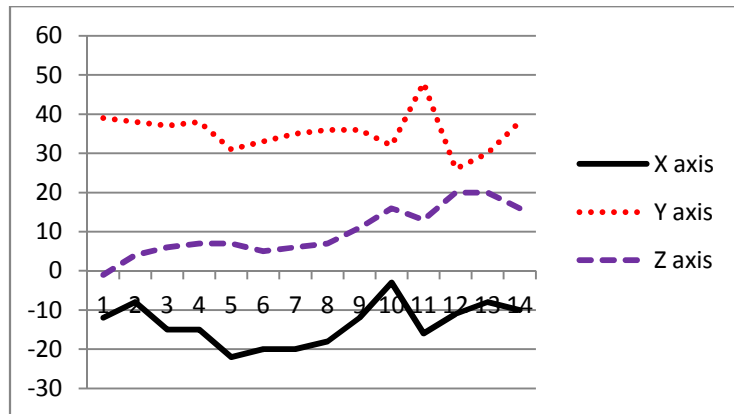


Figure 4.3: Log File of 3D Acceleration Values when User is at Rest

When the user is moving hands normally, the velocity will almost be constant. Hence the acceleration is very low or almost zero in x and y axes and 2g in the z axis. Figure 4.5 shows the log file of 3D acceleration values when the user moves hands slowly:

X axis	Y axis	Z axis
-24	18	29
-10	18	24
-27	12	26
-32	10	29
-30	11	23
-38	16	7
-40	19	-1
-31	18	4
-25	16	14
-23	19	19
-21	20	21
-19	15	27
-31	19	27
-35	14	37
-43	12	36
-39	8	36
-25	8	33
-21	9	22
-30	6	19
-34	3	20
-34	4	23
-43	9	29
-32	4	30
-22	5	29
-23	7	29
-28	6	30
-30	5	27

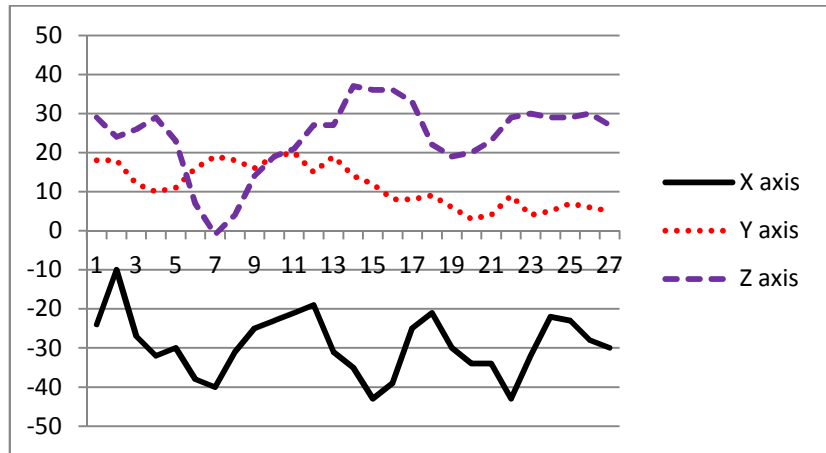


Figure 4.5: Log file of 3D Acceleration Values When the User Moves Hands Slowly

When the user performs a heavy gesture the 3D acceleration values will have larger magnitudes and, depending upon the plane in which the user is performing the gesture, the acceleration values

will be high in those planes. Figure 4.6 shows the log file of 3D acceleration values when the user performs a heavy gesture.

X axis	Y axis	Z axis
-15	-2	-12
-128	38	-18
-128	31	-61
-128	-1	-74
-101	-49	-68
-64	-42	-74
-54	-53	-67
-35	-50	-57
-35	-43	-29
-34	-46	-17
-50	-44	13
-30	-70	101
10	-49	101
-25	-25	127
21	18	25
-5	-14	36
-9	-8	43
-24	0	33
-27	-1	27
-21	-1	27
-18	-1	28
-10	-5	32
-9	-9	33
-14	-7	31
-13	-6	32
-14	-6	33
-19	-4	34
-21	-4	30
-23	-3	35
-21	-3	35

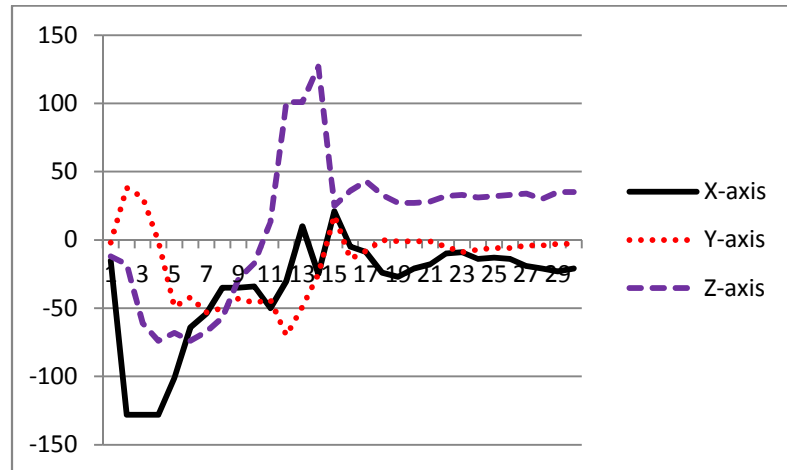


Figure 4.6: 3D Acceleration Values when a Heavy Gesture is performed by the User.

Based on the above three log files, it is clearly observed that the acceleration values are constant when the user is at rest or moving hands slowly. So the delta values in such cases will be very low.

All the acceleration values with high delta values are retained while values with low delta values are discarded with the help of the filter. Figure 4.7 shows the log file of the filtered data:

X axis	Y axis	Z axis
-15	-2	-12
-128	38	-18
-128	31	-61
-128	-1	-74
-101	-49	-68
-64	-42	-74
-54	-53	-67
-35	-50	-57
-35	-43	-29
-34	-46	-17
-50	-44	13
-30	-70	101
10	-49	101
-25	-25	127
21	18	25
-5	-14	36
-9	-8	43
-24	0	33
-27	-1	27
-21	-1	27
-18	-1	28
-10	-5	32

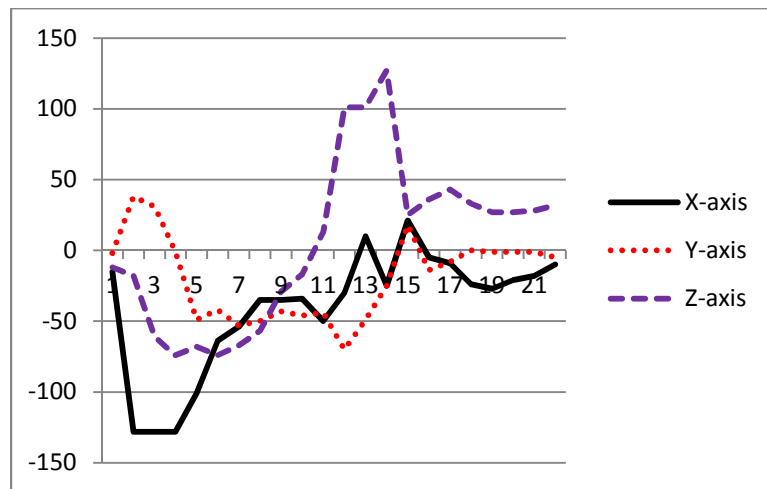


Figure 4.7: Log File Representing Filtered Data

4.4 Segmentation of Acceleration Data

The user typically sets the watch in acceleration mode and clicks on the start button only once, followed by a sequence of gestures. The accelerometer continuously senses the acceleration of the hand and 3D acceleration values are continuously written into the log file. This log file contains data corresponding to heavy gestures, as well as rest and normal hand movements. This large file ideally needs to be divided into segments, where each segment corresponds to one heavy gesture; data corresponding to unwanted activities must be discarded. We keep polling the log file, and whenever certain consecutive constant delta values are observed, these are discarded. Each set of values delimited by (relatively) constant values is sent to a text file named "current time stamp.txt". Each such file represents a gesture. We continue polling the data in the log file, and if we observe consecutive delta values with greater magnitude, we again send those values to another file named "current time stamp.txt." The data in each file forwarded to the recognition algorithm.

The code snippet below is used for filtering and segmentation of the data:

Pseudo Code:

```
Void Segment(DWORD spldata, ofstream& rawdata)
{
    if(abs(x1-vdata.x)<9 && abs(y1-vdata.y)<9 && abs(z1-vdata.z)<9)           //finding the
    difference                                                                //between consecutive acceleration
    values
    {
        if(f_count1!=0 && abs(count-f_count1)==1 )
        {
            filtercount++;
        }
        f_count1=count;
    }

    if(filtercount<3)                                                         // This condition indicates valid data to be sent for
    recognition
    {
```

```

writeit.open("path/filt.txt",ios::app);
writeit<<count<<<<" "<<vdata.x<<" "<<vdata.y<<" "<<vdata.z<<"\n";
writeit.close();
}
if(filtercount>=3) // This condition is to discard unnecessary values
{
    if(count>13)
    {
        the_date[0] = '\0'; // After discarding unnecessary values send them to
                            // timestamp.txt

        now = time(NULL);

        if (now != -1)
        {

            strftime(the_date, MAX_DATE, "%Y-%m-%d+%H-%M-%S.txt", gmtime(&now))
        }
        string datetime = "C:/Users/sowmya/Downloads/";
        datetime = datetime+ the_date;
        ifstream myReadFile;
        ofstream mywritefile1;
        mywritefile1.open(datetime,ios::app);
        myReadFile.open("C:/Users/sowmya/Downloads/msp430_005/Debug/filt.txt");
        if (myReadFile.is_open()) {
            while (!myReadFile.eof()) { //Placing the necessary values in
timestamp.txt
                myReadFile>>output1>>output2>>output3>>output4;
                mywritefile1<<output2<<" "<<output3<<" "<<output4;
                mywritefile1<<"\n";
            }
        }
        myReadFile.close();
        mywritefile1.close();
    }
    ct++; //Resetting all the variables for the next
segment
    count=0;
    f_count1=0;
    f_count2=0;
    filtercount=0;
    x1=0,y1=0,z1=0,d1=0;
    DeleteFile("C:/Users/sowmya/Downloads/msp430_005/Debug/filt.txt");

}
}

```

Consider the following hypothetical sequence of interactions with an e-mail program. The user searches for a particular email message in his Inbox and then deletes it. During the process, every movement is recorded by the watch. The log file contains the data corresponding to the gesture for search, data corresponding to normal hand movement or when the hand is at rest, and data corresponding to the action for delete. Filtering and segmentation code is used to filter the data corresponding to normal hand movements and dividing the data into two segments in this case. Figure 4.8 shows the corresponding log file. It contains data corresponding to the first gesture, data corresponding to normal hand movement during a delay between the first gesture and second gesture, and data corresponding to the second gesture. The file is filtered and divided into two segments by filtering and segmentation algorithms. In the log file, each heavy gesture which can be done in a second, is expected to have at least 10 lines of 3D acceleration values. Apart from the lines corresponding gestures remaining all lines will be filtered. Thus, filtering and segmentation is performed.

Line Number	X axis	Y axis	Z axis
1	-55	-16	-38
2	-50	-16	-40
3	-48	-18	-38
4	-67	-12	-32
5	-80	0	-21
6	-44	18	1
7	-106	48	46
8	-77	58	15
9	-30	-14	-56
10	-127	-128	-78
11	-31	-94	-20
12	-24	-76	-34
13	-7	-17	-16
14	-37	-16	-6
15	-58	-9	-11
16	-59	-16	-14
17	-69	-10	-8
18	-75	-6	-3
19	-65	-9	-6
20	-61	-11	-12
21	-54	-14	-13
22	-51	-13	-12
1	-58	-9	-10
2	-55	-10	-7
3	-56	-10	-12
4	-49	-10	-4
5	-53	-11	-4
1	-53	-9	-3
2	-51	-5	-2
3	-47	-6	1
4	-48	-6	1
1	-36	-14	18
2	-38	-12	20
3	-37	-14	22
4	-37	-13	21
1	-39	-14	20
2	-39	-15	20
3	-35	-17	20
4	-37	-19	19
1	-38	-23	16
2	-38	-23	15
3	-38	-23	16
4	-35	-22	16
1	-35	-24	17
2	-35	-23	17
3	-36	-24	16
4	-36	-24	16
1	-36	-24	17
2	-34	-23	17
3	-35	-23	18
4	-35	-23	18
1	-34	-23	18
2	-34	-23	18
3	-35	-22	18
4	-35	-23	18
1	-35	-22	18
2	-34	-22	18
3	-35	-23	17
4	-34	-22	18

Line Number	X axis	Y axis	Z axis
1	-34	-23	18
2	-34	-23	17
3	-34	-22	19
4	-34	-23	18
1	-33	-23	22
2	-33	-27	31
3	-33	-37	37
4	-38	-35	21
5	-40	-18	3
6	-41	-20	0
7	-50	-17	-9
8	-38	-15	-25
9	-42	-15	-26
10	-42	-19	-29
11	-47	-13	-39
12	-52	-13	-36
13	-50	-12	-35
14	-92	-13	-32
15	-89	-23	-38
16	-39	-10	-11
17	-60	-23	-14
18	-63	-17	-28
19	-128	28	-51
20	-128	11	-60
21	-128	-9	-87
22	-107	-22	-93
23	-87	-37	-102
24	-25	-82	-55
25	-54	-99	-25
26	-18	-94	24
27	-8	-94	123
28	-13	-11	97
29	1	-17	41
30	-15	-45	47
31	-41	-25	27
32	-11	-16	30
33	-21	-17	27
34	-26	-20	23
35	-26	-21	24
1	-20	-18	23
2	-27	-21	26
3	-17	-18	30
4	-22	-22	25
5	-21	-15	25
6	-22	-21	31

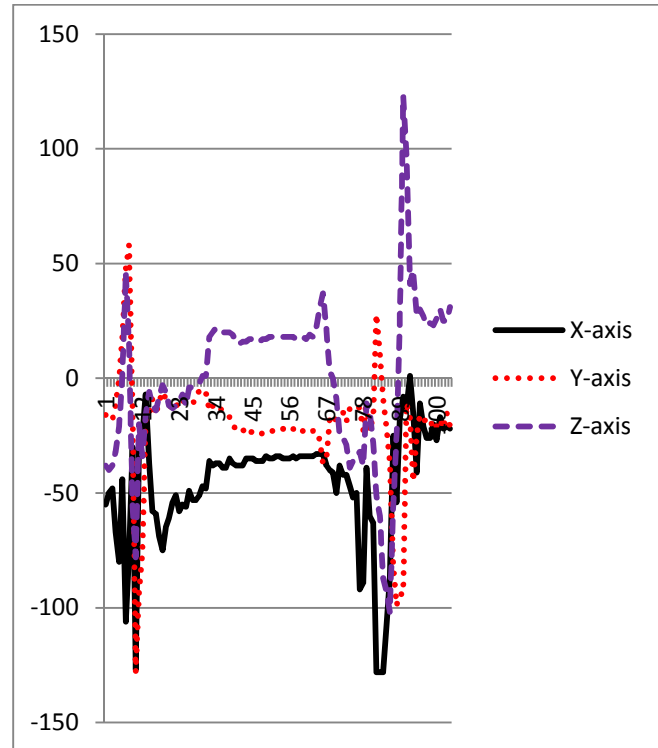


Figure 4.8: Log File and Graph for a sequence of Two Gestures performed continuously

Figures 4.9 and 4.10 show two files and graphs after segmentation of the log file shown in figure 4.8.

Line Number	X axis	Y axis	Z axis
1	-55	-16	-38
2	-50	-16	-40
3	-48	-18	-38
4	-67	-12	-32
5	-80	0	-21
6	-44	18	1
7	-106	48	46
8	-77	58	15
9	-30	-14	-56
10	-127	-128	-78
11	-31	-94	-20
12	-24	-76	-34
13	-7	-17	-16
14	-37	-16	-6
15	-58	-9	-11
16	-59	-16	-14
17	-69	-10	-8
18	-75	-6	-3
19	-65	-9	-6
20	-61	-11	-12
21	-54	-14	-13
22	-51	-13	-12

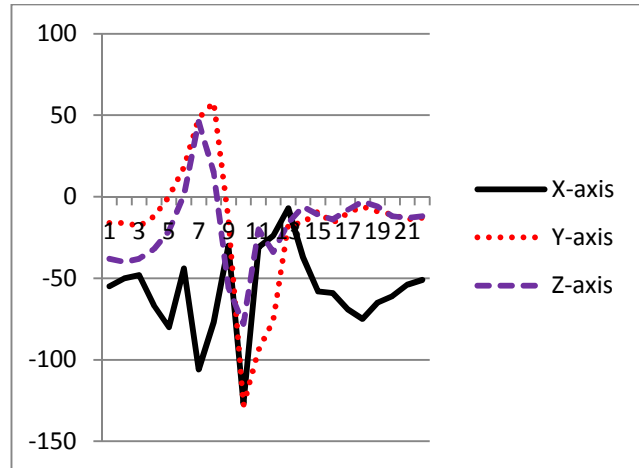


Figure 4.9: Segmented File and Graph for First Heavy Gesture

Line Number	X axis	Y axis	Z axis
1	-33	-23	22
2	-33	-27	31
3	-33	-37	37
4	-38	-35	21
5	-40	-18	3
6	-41	-20	0
7	-50	-17	-9
8	-38	-15	-25
9	-42	-15	-26
10	-42	-19	-29
11	-47	-13	-39
12	-52	-13	-36
13	-50	-12	-35
14	-92	-13	-32
15	-89	-23	-38
16	-39	-10	-11
17	-60	-23	-14
18	-63	-17	-28
19	-128	28	-51
20	-128	11	-60
21	-128	-9	-87
22	-107	-22	-93
23	-87	-37	-102
24	-25	-82	-55
25	-54	-99	-25
26	-18	-94	24
27	-8	-94	123
28	-13	-11	97
29	1	-17	41
30	-15	-45	47
31	-41	-25	27
32	-11	-16	30
33	-21	-17	27
34	-26	-20	23
35	-26	-21	24

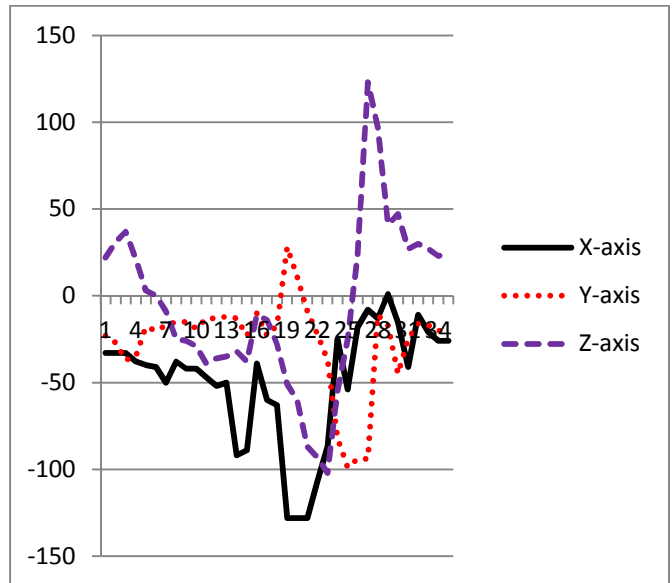


Figure 4.10: Segmented File and Graph for Second Heavy Gesture

4.5 Description of Gestures

Table 4: Description of Gestures and Corresponding Email Activities

SNO	Email Action	Conventional method	Gesture	Description of Gesture
1	Deleting an email	Clicking the delete option	Delete	Imagine that you have a blackboard in front of you (parallel to the screen of your computer). Draw a big 'Z' on that board.(This is just like you are rubbing the blackboard with a duster)
2	Sending an email	Clicking the send Button	Send	Imagine that you have a blackboard in front of you (parallel to the screen of your computer). Draw a big 'Tick mark' on that board.
3	See recent mails in the current page of Inbox	Scroll up the scroll bar to see the recent mails	Turn Right	Imagine that you have a big book on your keyboard. Turn a page of that book to the right to see the previous page. Place your hand on the bottom left corner of the keyboard and turn the page to right making a big arc.
4	See older mails in the current page of Inbox	Scroll down the scroll bar to see older mails	Turn Left	Imagine that you have a big book on your keyboard. Turn a page of that book to the left to see the next page. Place your hand on the bottom right corner of the keyboard and turn the page to left making a big arc.
5	Forwarding an email	Clicking the 'Forward to' button	Forward	Join your palms with the fingers pointing away from you and lock your fingers. Move your locked hand away from you

Table 4: -- Description of gestures and the corresponding email activities

SNO	Email Action	Conventional method	Gesture	Description of Gesture
6	Compose an email	Clicking compose or New button	Compose	Imagine that some nuts are scattered on the table over a small area. Gather all of them at once and toss upwards.
7	Searching for a mail in Inbox	Clicking the Search Button	Search	Imagine that there is a book on the keyboard and a magnifying glass in your hand. Place your hand at the bottom center of the keyboard and rotate your hand clockwise. Make a circle and semicircle with your hand.
8	Opening an attachment	Clicking the open Button in the pop-up window after downloading	Open	Place your hands at your shoulders with palms facing the screen of the computer. Then make fists and slowly raise your hands up as if you are opening the shutter of a store.

Table 5: Description and Images of Gestures


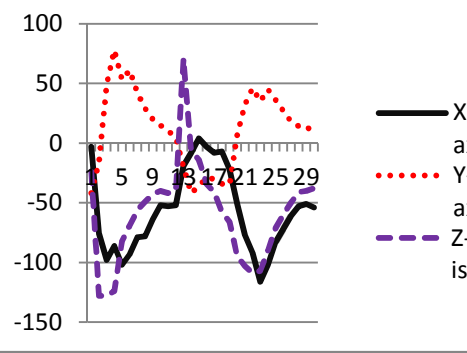

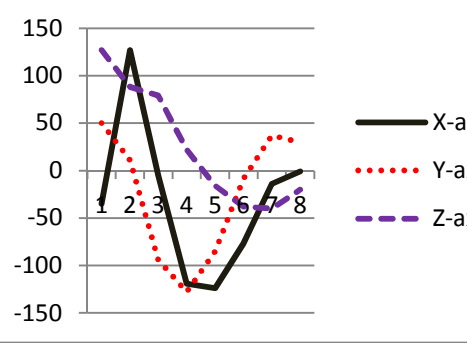
Description of Gesture	Image Showing How to do the Gesture	Graph Showing Acceleration Values of a Sample
<p>Delete:</p> <p>Imagine that you have a blackboard in front of you (parallel to the screen of your computer). Draw a big 'Z' on that board.(This is just like you are rubbing the blackboard with a duster)</p>		
<p>Send:</p> <p>Imagine that you have a blackboard in front of you (parallel to the screen of your computer). Draw a big 'Tick mark' on that board.</p>		

Table 5: -- Description and Images of Gestures


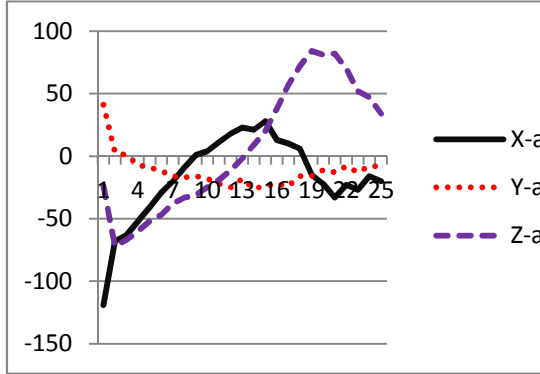
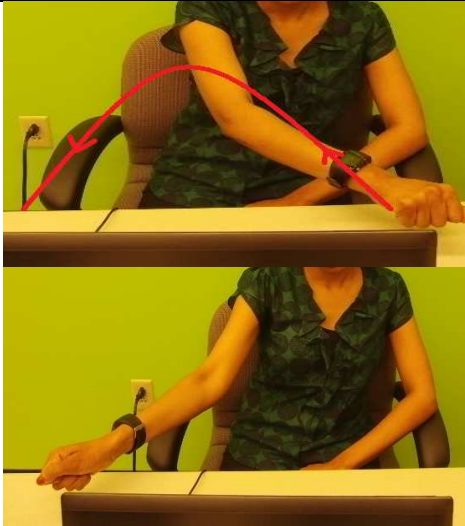
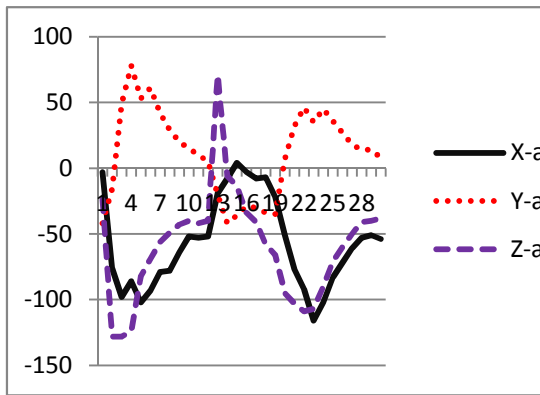
Description of Gesture	Image Showing How to do the Gesture	Graph Showing Acceleration Values of a Sample
<p>Turn:</p> <p>Imagine that you have a big book on your keyboard. Turn a page of that book to the left to see the next page. Place your hand on the bottom right corner of the keyboard and turn the page to left making a big arc.</p>		
<p>Turn from left to right:</p> <p>Imagine that you have a big book on your keyboard. Turn a page of that book to the right to see the previous page. Place your hand on the bottom left corner of the keyboard and turn the page to right making a big arc.</p>		

Table 5: -- Description and images of gestures


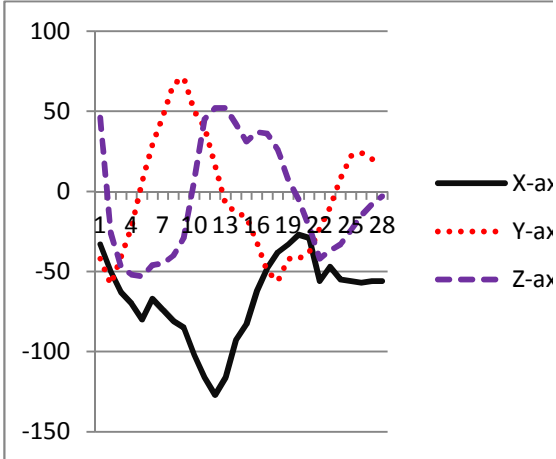

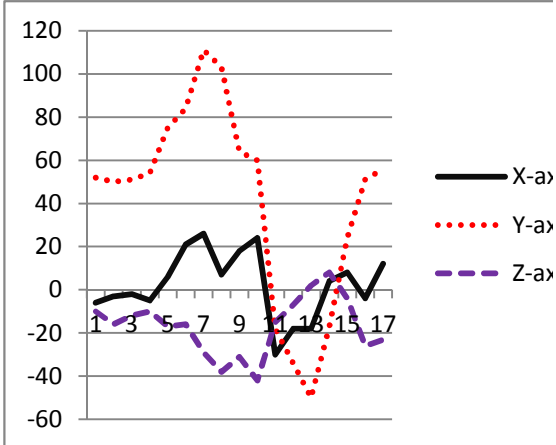

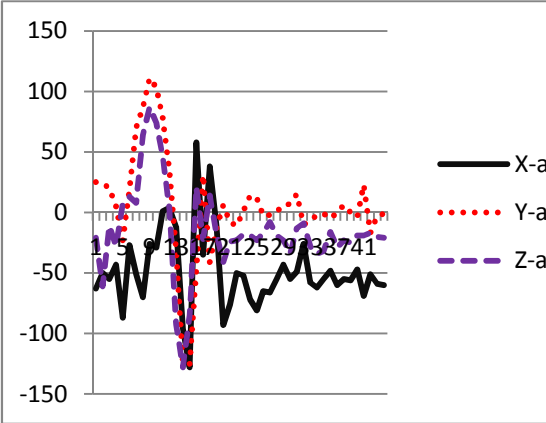
Description of Gesture	Image Showing How to do the Gesture	Graph Showing Acceleration Values of a Sample																																												
<p>Search:</p> <p>Imagine that there is a book on the keyboard and a magnifying glass in your hand. Place your hand at the bottom center of the keyboard and rotate your hand clockwise. Make a circle and semicircle with your hand.</p>		 <table border="1"> <caption>Approximate Acceleration Values for Search Gesture</caption> <thead> <tr> <th>Sample</th> <th>X-axis (solid black)</th> <th>Y-axis (dotted red)</th> <th>Z-axis (dashed purple)</th> </tr> </thead> <tbody> <tr><td>1</td><td>-20</td><td>-50</td><td>50</td></tr> <tr><td>4</td><td>-70</td><td>0</td><td>-50</td></tr> <tr><td>7</td><td>-100</td><td>70</td><td>-50</td></tr> <tr><td>10</td><td>-130</td><td>0</td><td>50</td></tr> <tr><td>13</td><td>-100</td><td>0</td><td>50</td></tr> <tr><td>16</td><td>-50</td><td>-50</td><td>50</td></tr> <tr><td>19</td><td>-20</td><td>-50</td><td>50</td></tr> <tr><td>22</td><td>-50</td><td>0</td><td>50</td></tr> <tr><td>25</td><td>-50</td><td>20</td><td>50</td></tr> <tr><td>28</td><td>-50</td><td>0</td><td>50</td></tr> </tbody> </table>	Sample	X-axis (solid black)	Y-axis (dotted red)	Z-axis (dashed purple)	1	-20	-50	50	4	-70	0	-50	7	-100	70	-50	10	-130	0	50	13	-100	0	50	16	-50	-50	50	19	-20	-50	50	22	-50	0	50	25	-50	20	50	28	-50	0	50
Sample	X-axis (solid black)	Y-axis (dotted red)	Z-axis (dashed purple)																																											
1	-20	-50	50																																											
4	-70	0	-50																																											
7	-100	70	-50																																											
10	-130	0	50																																											
13	-100	0	50																																											
16	-50	-50	50																																											
19	-20	-50	50																																											
22	-50	0	50																																											
25	-50	20	50																																											
28	-50	0	50																																											
<p>Open:</p> <p>Place your hands at your shoulders with palms facing the screen of the computer. Then make fists and slowly raise your hands up as if you are opening the shutter of a store.</p>		 <table border="1"> <caption>Approximate Acceleration Values for Open Gesture</caption> <thead> <tr> <th>Sample</th> <th>X-axis (solid black)</th> <th>Y-axis (dotted red)</th> <th>Z-axis (dashed purple)</th> </tr> </thead> <tbody> <tr><td>1</td><td>-5</td><td>50</td><td>-10</td></tr> <tr><td>3</td><td>-5</td><td>50</td><td>-10</td></tr> <tr><td>5</td><td>5</td><td>80</td><td>-10</td></tr> <tr><td>7</td><td>25</td><td>110</td><td>-10</td></tr> <tr><td>9</td><td>5</td><td>70</td><td>-10</td></tr> <tr><td>11</td><td>-10</td><td>0</td><td>-10</td></tr> <tr><td>13</td><td>-10</td><td>-50</td><td>-10</td></tr> <tr><td>15</td><td>5</td><td>20</td><td>-10</td></tr> <tr><td>17</td><td>10</td><td>50</td><td>-10</td></tr> </tbody> </table>	Sample	X-axis (solid black)	Y-axis (dotted red)	Z-axis (dashed purple)	1	-5	50	-10	3	-5	50	-10	5	5	80	-10	7	25	110	-10	9	5	70	-10	11	-10	0	-10	13	-10	-50	-10	15	5	20	-10	17	10	50	-10				
Sample	X-axis (solid black)	Y-axis (dotted red)	Z-axis (dashed purple)																																											
1	-5	50	-10																																											
3	-5	50	-10																																											
5	5	80	-10																																											
7	25	110	-10																																											
9	5	70	-10																																											
11	-10	0	-10																																											
13	-10	-50	-10																																											
15	5	20	-10																																											
17	10	50	-10																																											

Table 5: -- Description and images of gestures

Description of Gesture	Image Showing How to do the Gesture	Graph Showing Acceleration Values of a Sample
<p>Punch: Join your palms with the fingers pointing away from you and lock your fingers. Quickly Move your locked hand away from you</p>		

CHAPTER 5

IMPLEMENTATION

5.1 Overview of Implementation

The previous chapters focused on the methodology that makes it possible to recognize a set of gestures based on acceleration values. This chapter discusses the implementation of training and recognition. Users can train the application with as many gestures they want from the pool of the gestures defined to map to email activities. Each gesture needs to be repeated eight times to create a training set. Once all gestures are presented, training is completed by mapping to a multivariate Gaussian model. The application is then ready to recognize the gestures.

In the implementation of this application, we use the modified .exe file of the Chronos Flying Mouse application which is recompiled after adding our source code. This .exe file is invoked from the training and recognition module.

Code Snippet Corresponding to Training and Recognition:

```
class NewThread extends MatlabSyntax implements Runnable
{
    Thread t;
    // Training and Recognition code
}

public class trainreco
{
    public static void main(String[] args)
    {
        new NewThread();

        Process p = rt.exec("Filepath/Chronosflyingmouse.exe");
    }
}
```

In our application, we run the Chronos Flying Mouse .exe file in the main thread and the training and recognition code in the child thread. Since both the threads run simultaneously, training and recognition can occur in real time.

5.2 Training of Gestures

As discussed in section 3.4, we have a set of eight gestures, each used to perform an email activity. Each gesture has to be trained eight times and the user can train as many gestures as he wants. When the user does a gesture eight times, eight .txt files are generated, each corresponding to one sample of the gesture. These text files contain the acceleration data that are written by our data collection algorithm in the Chronos Flying Mouse application code. As discussed in section 5.1, the Chronos Flying Mouse application and code corresponding to the training and recognition run simultaneously. The acceleration data is continuously written into .txt Files and the training code reads the files generated to do calculations for PCA and probability densities in real time for practical purposes.

For proper concurrency control, the main thread locks each file while writing it and unlocks it as soon as it is done. The child thread opens the files sequentially for training and recognition.

Code Snippet Showing How Files are Searched in the Target Directory

```
class myfilesearch {
    public File[] finder(String dirName){

        File dir = new File(dirName);

        return dir.listFiles(new FilenameFilter() {
            public boolean accept(File dir, String filename)
            { return filename.endsWith(".txt") ; }
        });
    }
}
```

The files are indexed in a file array that is sorted by modification time, such that the first element in the file array is the oldest file created and the last element in the array is the newest file.

Code Snippet Showing How Files are Sorted According to Modified Time.

```
class sortfiles
{
    public File[] getallfiles(){
        File temp;
        Scanner sc2 = null;
        myfilesearch sh = new myfilesearch();
        File[] f=sh.finder("C:/Users/sowmya/Downloads");

        //System.out.println(f.length);
        for(int j=0;j<(f.length)-2;j++)
        {
            for(int i=0; i<(f.length)-1 ; i++)
            {
                if( f[i].lastModified() > f[i+1].lastModified())
                {
                    temp = f[i];
                    f[i] = f[i+1];
                    f[i+1] = temp;
                }
            }
        }

        return f;
    }
}
```

Ex:

```
C:\Users\sowmya\Downloads\2012-11-09+04-12-20.txt
C:\Users\sowmya\Downloads\2012-11-09+04-14-25.txt
C:\Users\sowmya\Downloads\2012-11-09+04-15-25.txt
C:\Users\sowmya\Downloads\2012-11-09+04-16-39.txt
C:\Users\sowmya\Downloads\2012-11-09+04-16-41.txt
C:\Users\sowmya\Downloads\2012-11-09+04-16-43.txt
C:\Users\sowmya\Downloads\2012-11-09+04-17-02.txt
C:\Users\sowmya\Downloads\2012-11-09+04-17-59.txt
C:\Users\sowmya\Downloads\2012-11-09+04-19-15.txt
C:\Users\sowmya\Downloads\2012-11-09+04-20-38.txt
C:\Users\sowmya\Downloads\2012-11-09+04-20-41.txt
C:\Users\sowmya\Downloads\2012-11-09+04-20-48.txt
C:\Users\sowmya\Downloads\2012-11-09+04-21-13.txt
C:\Users\sowmya\Downloads\2012-11-09+04-21-17.txt
C:\Users\sowmya\Downloads\2012-11-09+04-21-28.txt
```

Each file is read, and a matrix is constructed from the file.

Code Snippet Shows How the Matrix is constructed from a File.

```

FileInputStream fstream = new FileInputStream(f[i]);
String sr = f[i].toString();
sc2 = new Scanner(new File(sr));
DataInputStream in = new DataInputStream(fstream);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
String str;

while ((str = br.readLine()) != null) {
    System.out.println("str is"+ str);
count++;

}
in.close();
//count = count-2;
m=count;
System.out.println("count is"+ m);
Matrix B = normRnd(0,1,m,3) ;
while (sc2.hasNextLine()) {
    Scanner s2 = new Scanner(sc2.nextLine());

    boolean b;

    while (b = s2.hasNext()) {
        int threecount=1;
        while(threecount<4)
        {
            String s = s2.next();
            double cond = Double.parseDouble(s);
            int convertint = (int) cond;
            set(B,matrixindex,threecount,convertint);

                threecount++;
        }

        matrixindex++;
    }
}

```

In Figure 5.1, the file on the left contains the acceleration data, while the one on the right side shows the matrix that is constructed from the data.

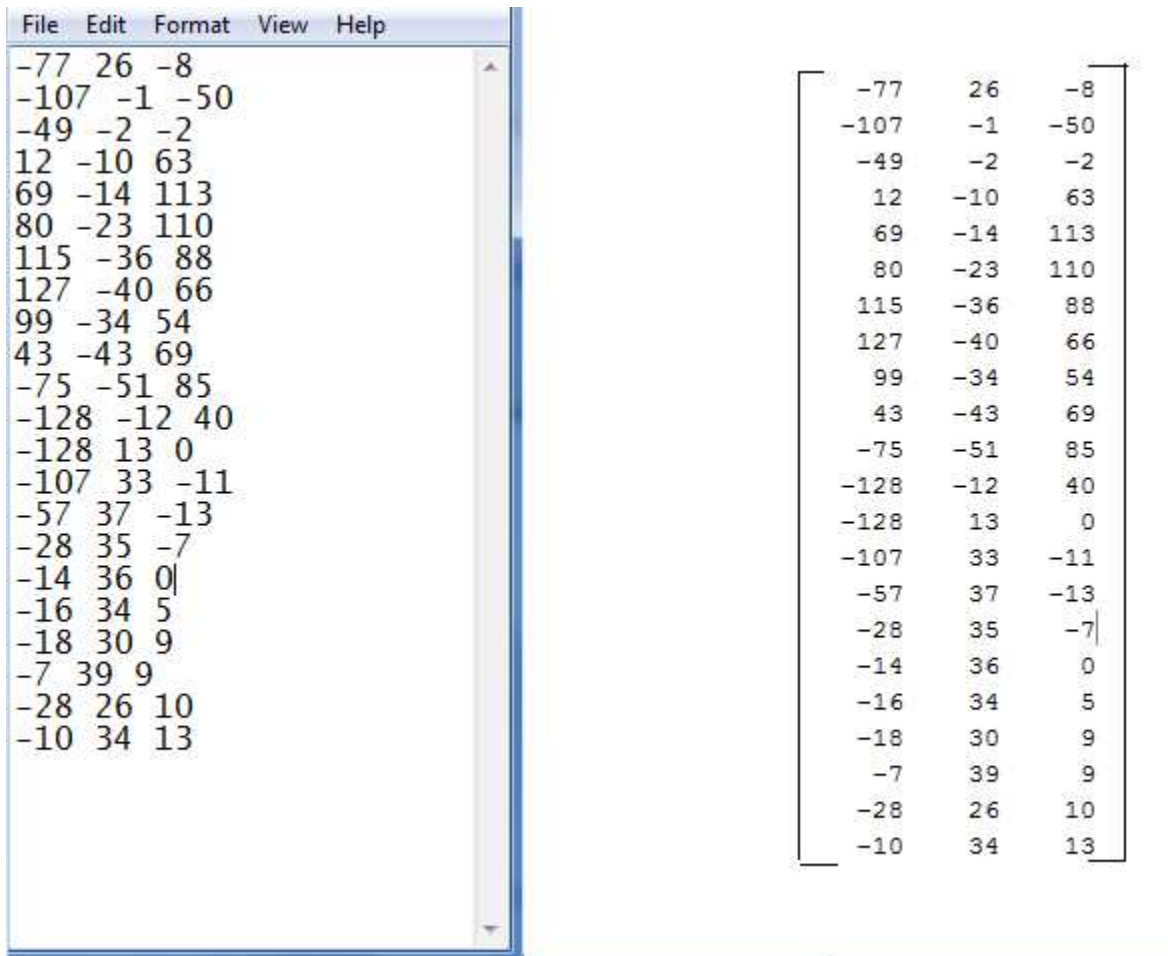


Figure 5.1: A Matrix Constructed from a Text File

Based on the matrix, variances and covariances are calculated, and Eigenvectors are calculated in turn for the transformation corresponding to the covariance matrix. As discussed in section 3.1., the principal component corresponds to the Eigenvector with the highest variance, which is used for gesture training and recognition.

Code Snippet Showing How the Principal Component is Calculated from the Constructed Matrix

```
Matrix covariance = cov(B);
Matrix eigenVectors = eig_V(covariance);
Matrix eigenValues = eig_D(covariance);
System.out.println("eigen Vectors are");
disp(eigenVectors);
```

```
Vector<Double> v = new Vector<Double>();  
v.add(get(eigenVectors, 1, 3));  
v.add(get(eigenVectors, 2, 3));  
v.add(get(eigenVectors, 3, 3));
```

Eigen vectors for the covariance of the matrix shown in fig: by using the above code

```
[0.8741 -0.4760 -0.0965  
-0.2283 -0.5781 0.7834  
0.4287 0.6628 0.6140]
```

The first principal component calculated based on the code and matrix above is

```
[0.8741 -0.2283 0.4287]
```

For each sample, the principal component and average magnitude is calculated as discussed above.

Thus we get eight principal components for the eight samples, and then find the average principal component.

The average principal component vector of the eight samples for the gesture delete is shown in Figure 5.2 and represented as $0.8899i + -0.2730j + 0.3611k$

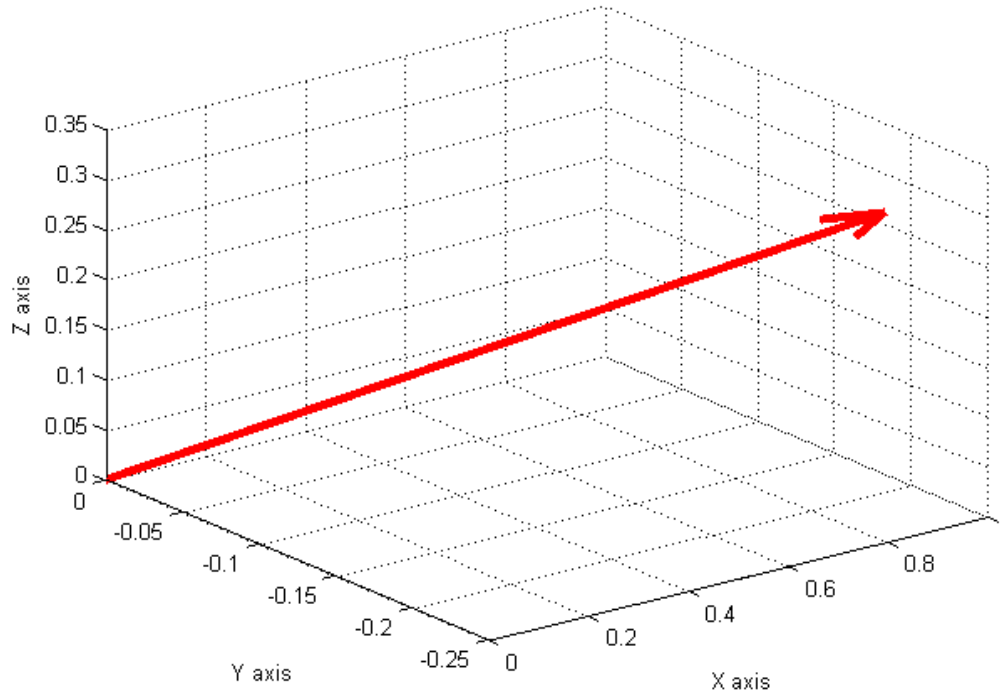


Figure 5.2: Average Principal Vector for Delete Gesture

Similarly, the average principal component vector for eight samples of the send gesture is shown in figure 5.3 and represented as $0.2373 i + 0.715j + 0.6372125k$.

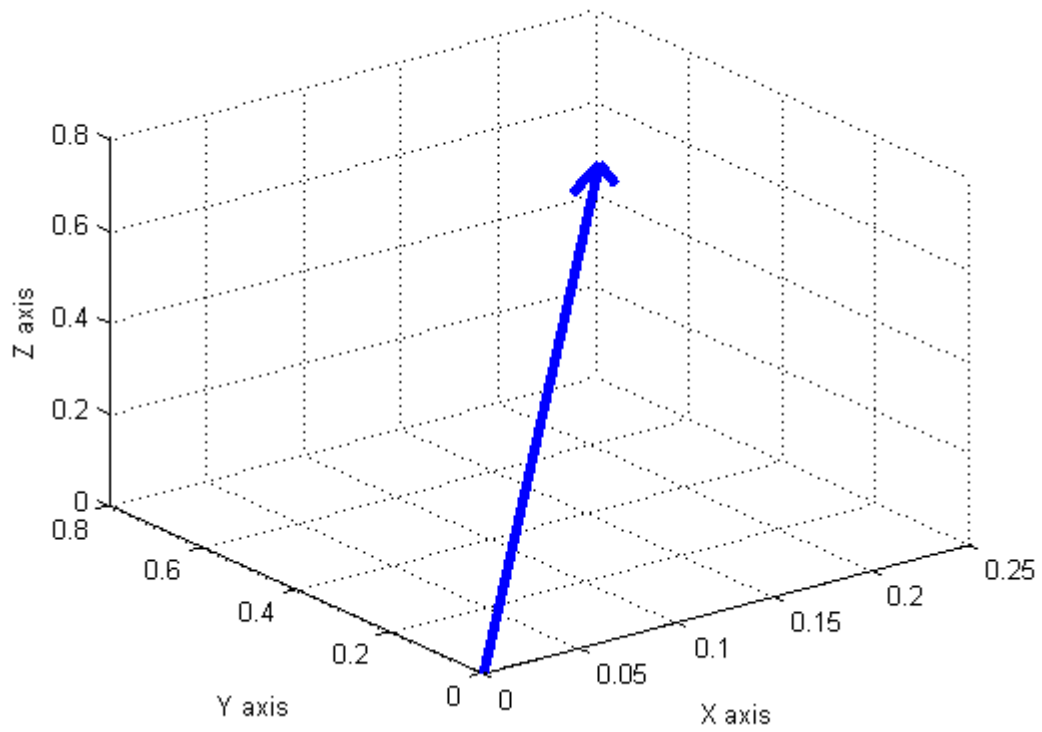


Figure 5.3: Average Principal Vector for Send Gesture

The average principal component vector of the eight samples of the scroll down gesture is shown in Figure 5.4, represented as $-0.4821375i - 0.0086125j + 0.87165k$.

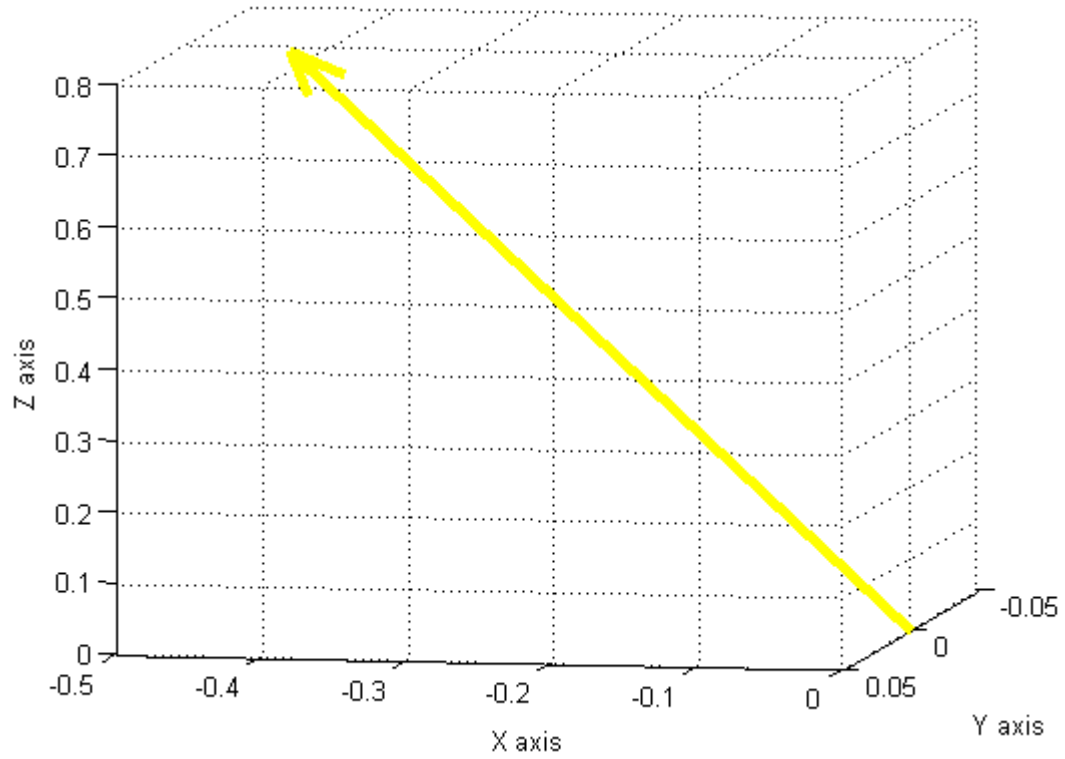


Figure 5.4: Average Principal Vector for Scroll Down Gesture

The average principal component vector of the eight samples of the scroll up gesture is shown in Figure 5.5, represented as $0.1351375i - 0.0643875j + 0.9763k$.

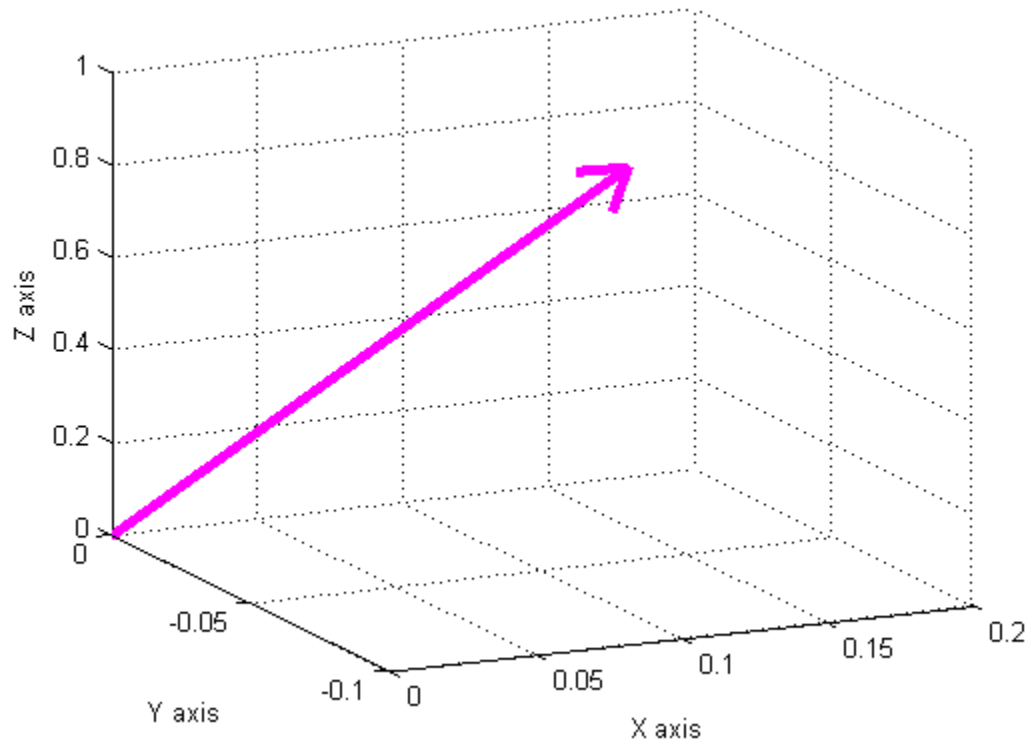


Figure 5.5: Average Principal Vector for Scroll Up Gesture

The average principal component vector of the eight samples of the punch gesture is shown in Figure 5.6 and is represented as $0.20200388i + 0.71321853j + 0.654672013k$.

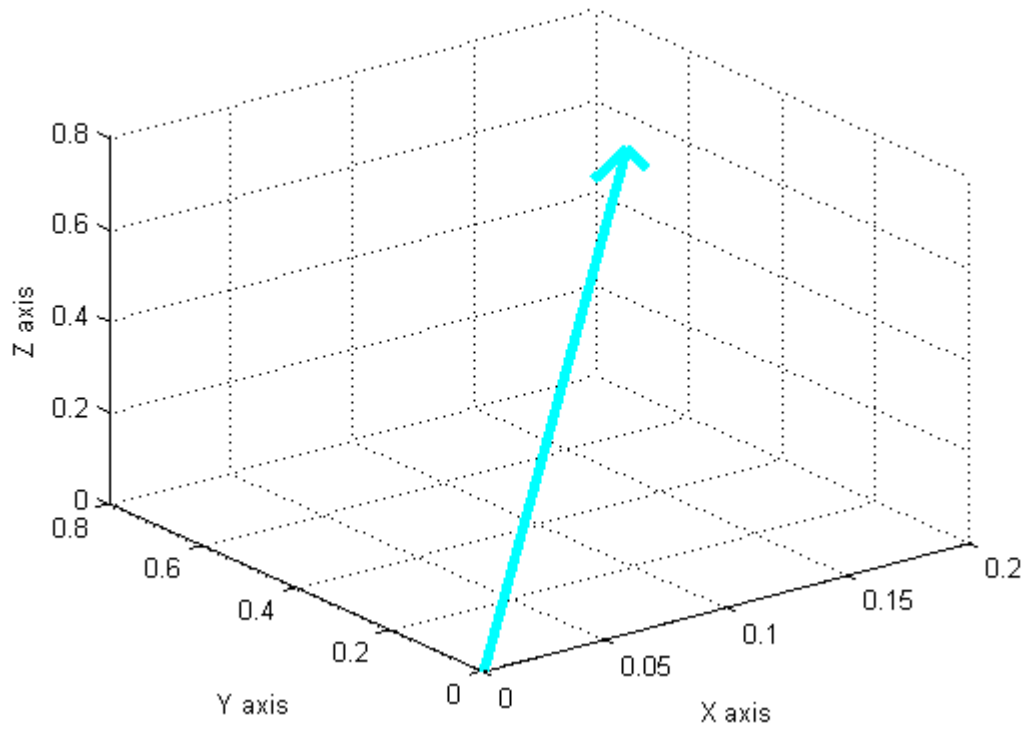


Figure 5.6: Average Principal Vector for Punch Gesture

The average principal component vector of the eight samples of the launch gesture is shown in Figure 5.7 and is represented as $0.7915i + 0.3174875j + 0.50325k$.

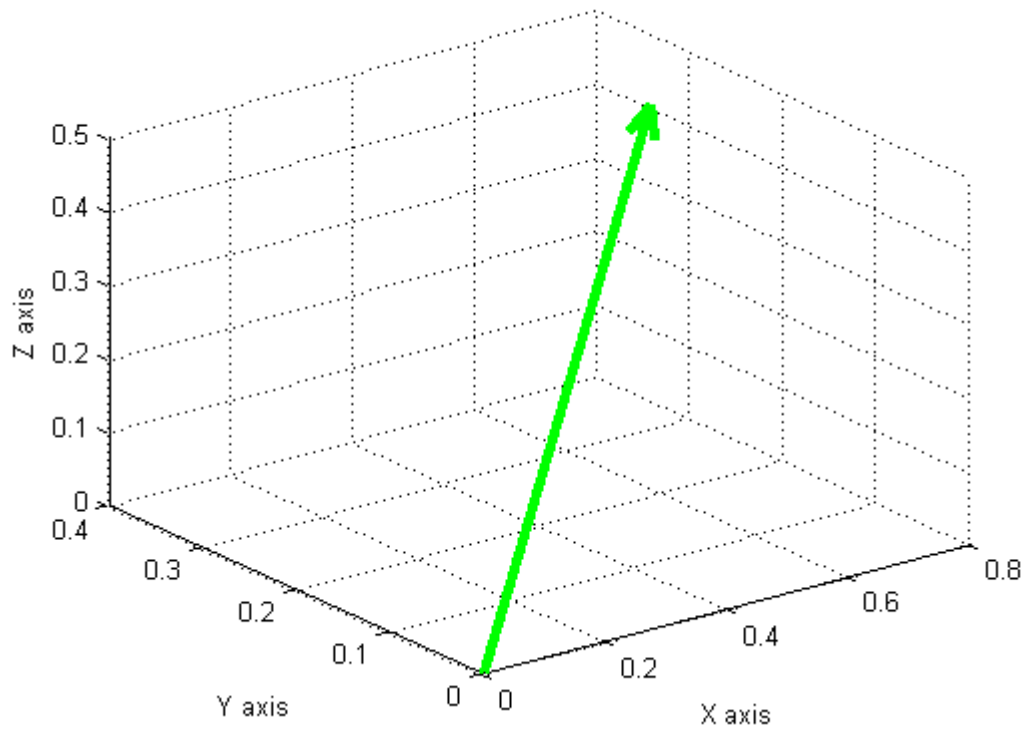


Figure 5.7: Average Principal Vector for Launch Gesture

The average principal component vector of the eight samples of the search gesture is shown in Figure 5.8 and is represented as $-0.4482i + 0.8723j - 0.1212k$.

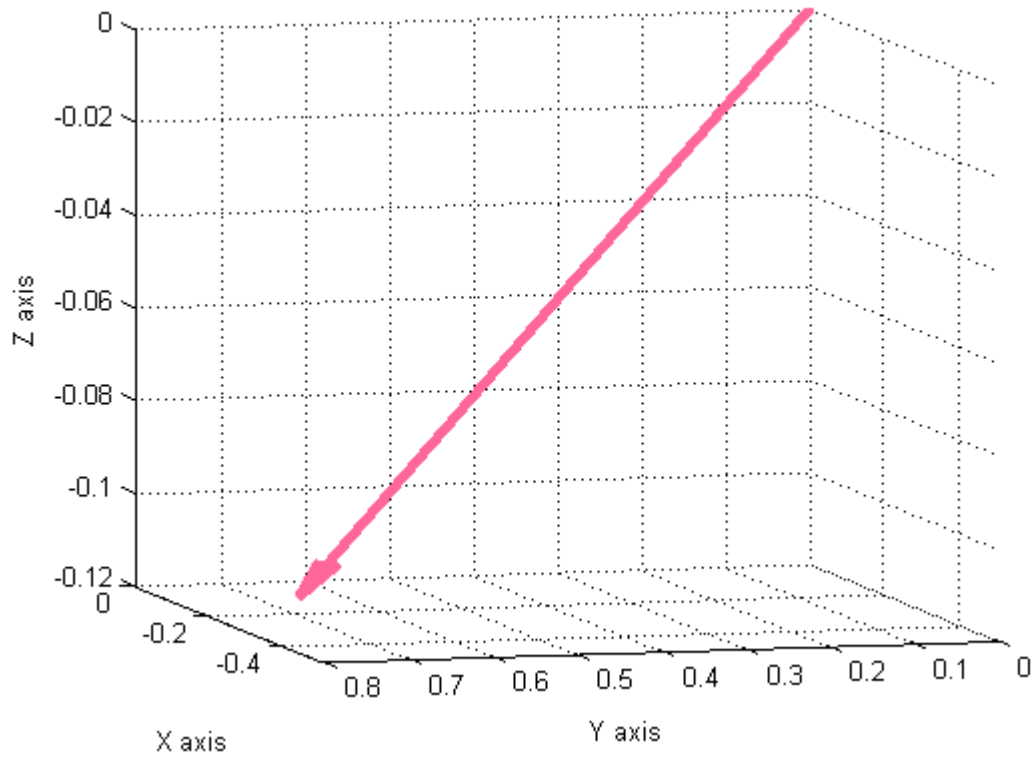


Figure 5.8: Average Principal Vector for Search Gesture

The average principal component vector of the eight samples of the open gesture is shown in Figure 5.9 and is represented as $-0.192809669i - 0.951398035j + 0.213508419k$.

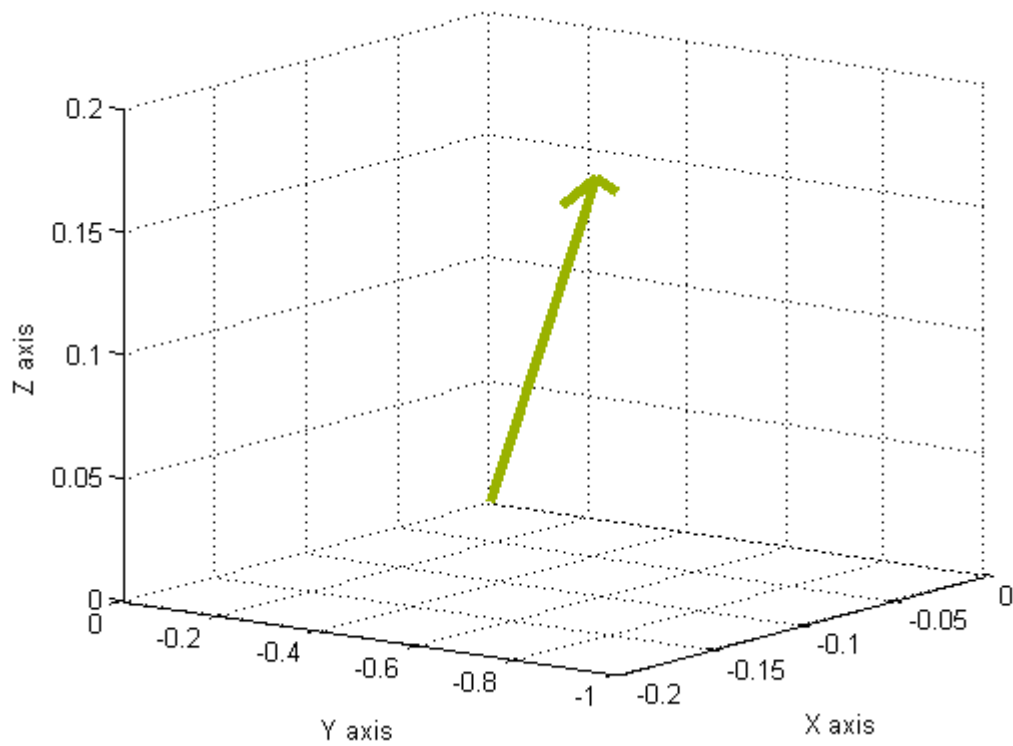


Figure 5.9: Average Principal Vector for Open Gesture

5.3 Recognition of Gestures

Consider a test gesture and let's say its principal component vector is $0.87i - 0.22j + 0.42k$

Once the principal component is calculated for the test gesture, angles between the test gesture and the average principal component of each gesture are calculated. Figure 5.10 shows the first principal component of all the predefined gestures and the test gesture.

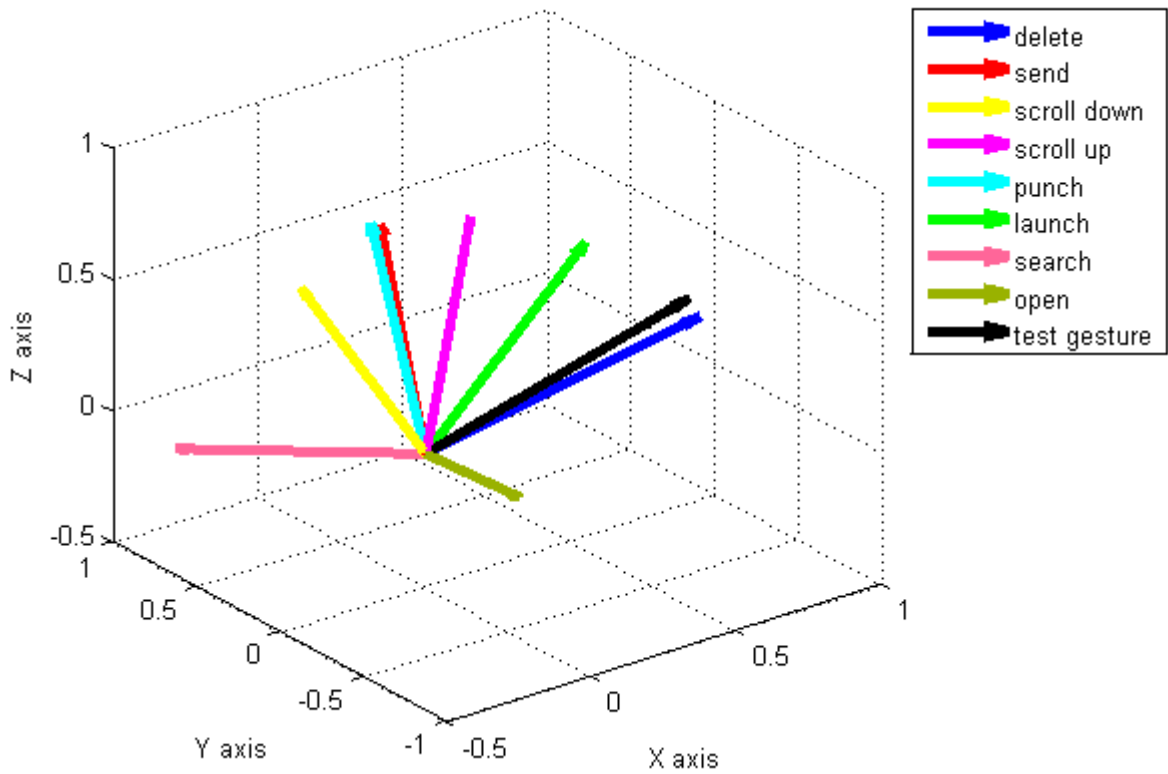


Figure 5.10: Graph Showing First Principal Component of all Gesture Models with Test Gesture.

Angles are calculated by using the dot product of two vectors as discussed earlier.

Angle between test gesture and delete gesture stereotype principal components = 4.68

Angle between test gesture and send gesture stereotype principal components = 71.30

Angle between test gesture and scroll down gesture stereotype principal components = 92.74

Angle between test gesture and scroll up gesture stereotype principal components = 56.15

Angle between test gesture and punch gesture stereotype principal components = 72.75

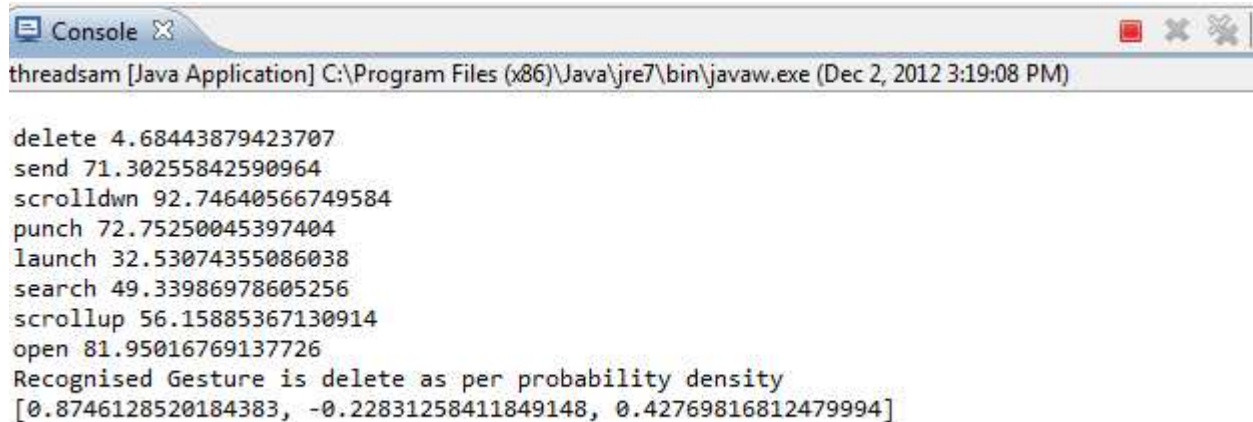
Angle between test gesture and launch gesture stereotype principal components = 32.53

Angle between test gesture and search gesture stereotype principal components = 49.33

Angle between test gesture and open gesture stereotype principal components = 81.95

The smallest angle (1.74 degrees) is between the test gesture and the delete gesture model. The test gesture is therefore recognized as delete.

Figure 5.11 is a screen shot of the application's output for gesture recognition.

A screenshot of a Java application console window. The title bar reads "Console" with a close button. The main text in the console shows a list of gestures with numerical values: delete 4.68443879423707, send 71.30255842590964, scrollldwn 92.74640566749584, punch 72.75250045397404, launch 32.53074355086038, search 49.33986978605256, scrollup 56.15885367130914, and open 81.95016769137726. Below this list, it states "Recognised Gesture is delete as per probability density" followed by a vector of three numbers in square brackets: [0.8746128520184383, -0.22831258411849148, 0.42769816812479994].

```
delete 4.68443879423707
send 71.30255842590964
scrollldwn 92.74640566749584
punch 72.75250045397404
launch 32.53074355086038
search 49.33986978605256
scrollup 56.15885367130914
open 81.95016769137726
Recognised Gesture is delete as per probability density
[0.8746128520184383, -0.22831258411849148, 0.42769816812479994]
```

Figure 5.11: Application Output for Gesture Recognition

For the second method of gesture recognition, we take into account the probability densities of both the principal component and the magnitude; the average x, y, and z accelerations are also taken into account. The joint probability density of a gesture is given by the product of these two probability densities.

The average magnitudes are calculated as follows. For the matrix shown in Figure 5.12, parsed from the stream file, we take the averages of the first, second and third columns, which represent the acceleration values in x, y and z dimensions.

Code Snippet Showing How to Calculate the Average Values of the Matrix:

```
double sum1=0;
double sum2=0;
double sum3=0;
```

```

for(int row=1; row<matrixindex;row++)
{
    sum1 = sum1+ get(B,row,1);
}
for(int row=1; row< matrixindex;row++)
{
    sum2 = sum2+ get(B,row,2);
}
for(int row=1; row< matrixindex; row++)
{
    sum3 = sum3+ get(B,row,3);
}
double avg1 = sum1/(m-1);
double avg2=sum2/(m-1);
double avg3= sum3/(m-1);

```

-77	26	-8
-107	-1	-50
-49	-2	-2
12	-10	63
69	-14	113
80	-23	110
115	-36	88
127	-40	66
99	-34	54
43	-43	69
-75	-51	85
-128	-12	40
-128	13	0
-107	33	-11
-57	37	-13
-28	35	-7
-14	36	0
-16	34	5
-18	30	9
-7	39	9
-28	26	10
-10	34	13

Figure 5.12 : Matrix used to Compute Average Values

Avg(x) = -13.8182

Avg(y)=3.5

Avg(z) = 29.22727

The first principal component and average x y z accelerations are thus calculated for a file representing one gesture sample. As discussed in section 3.4, a training set for gesture classification is created based on these two features.

Code Snippet Showing How Principal Component Matrix and Magnitude Matrix are constructed

from the Training Set:

```
public void gettrainsets(int p, Matrix traineigenset, Matrix trainmagset, Matrix
eigenVectors, Matrix tmag)
{
    int q=1;
    System.out.println("p is " +p + "q is "+q);
    set(traineigenset,p,q,get(eigenVectors, 1, 3));
    set(trainmagset,p,q,get(tmag, 1, 1));
    q++;
    System.out.println("p is " +p + "q is "+q);
    set(traineigenset,p,q,get(eigenVectors, 2, 3));
    set(trainmagset,p,q,get(tmag, 2, 1));
    q++;
    System.out.println("p is " +p + "q is "+q);
    set(traineigenset,p,q,get(eigenVectors, 3, 3));
    set(trainmagset,p,q,get(tmag, 3, 1));

    disp(traineigenset);
}
```

Once the training set is created with the help of the above code, the likelihood for each gesture model is calculated as follows.

$$\text{Probability density} = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

Where

Σ - covariance matrix of the multivariate normal distribution

$|\Sigma|$ - Determinant of the covariance matrix,

Σ^{-1} - Inverse of the covariance matrix

μ - Mean

Code Snippet for the Calculation of the Required Parameters

```
double sum1=0;
double sum2=0;
double sum3=0;

for(int row=1; row<9;row++)
{
    sum1 = sum1+ get(traineigenset,row,1);
}
for(int row=1; row< 9;row++)
{
    sum2 = sum2+ get(traineigenset,row,2);
}
for(int row=1; row< 9; row++)
{
    sum3 = sum3+ get(traineigenset,row,3);
}
double ag1 = sum1/(8);
double ag2=sum2/(8);
double ag3= sum3/(8);

Matrix meaneigenset = normRnd(0,1,3,1);
set(meaneigenset,1,1,ag1);
set(meaneigenset,2,1,ag2);
set(meaneigenset,3,1,ag3);

double s1=0;
double s2=0;
double s3=0;

for(int row=1; row<9;row++)
{
    s1 = s1+ get(trainmagset,row,1);
}
for(int row=1; row< 9;row++)
{
    s2 = s2+ get(trainmagset,row,2);
}
for(int row=1; row< 9; row++)
{
    s3 = s3+ get(trainmagset,row,3);
}
```

```

double a1 = s1/(8);
double a2=s2/(8);
double a3= s3/(8);

Matrix meanmagset = normRnd(0,1,3,1);
set(meanmagset,1,1,a1);
set(meanmagset,2,1,a2);
set(meanmagset,3,1,a3);

meaneigenmatrices.add(meaneigenset);
Matrix coveigenset= cov(traineigenset);
Matrix invcoveigenset = inv(coveigenset);
inveigenmatrices.add(invcoveigenset);
double detcoveigenset = det(coveigenset);
eigendets.add(detcoveigenset);

meanmagmatrices.add(meanmagset);
Matrix covmagset = cov(trainmagset);
Matrix invcovmagset = inv(covmagset);
invmagmatrices.add(invcovmagset);
double detcovmagset = det(covmagset);
magdets.add(detcovmagset);

```

The probability of a gesture when Principal component is taken into account is calculated as follows:

```

for( probcnt =0; probcnt<meaneigenmatrices.size(); probcnt++)
{
Matrix mdiff = minus(tpca,(Matrix)meaneigenmatrices.get(probcnt));
Matrix mtrans = t(mdiff);
Matrix interpca =
times(mtrans,(Matrix)inveigenmatrices.get(probcnt));
Matrix finalpca = times(interpca,mdiff);
double expower=-get(finalpca,1,1)/2;
double probpca =
(1/(Math.sqrt(Math.pow(2*3.14159,3)*(double)eigendets.get(probcnt)))
)*Math.exp(expower);
System.out.println("prob pca is");
System.out.println(probpca);
}

```

The probability density of a gesture when average magnitudes of x, y, z components of acceleration are considered is calculated as follows:

```

for( probcnt =0; probcnt<meaneigenmatrices.size(); probcnt++)
{

Matrix magdiff = minus(tmag,(Matrix)meanmagmatrices.get(probcnt));
Matrix magtrans = t(magdiff);
Matrix intermag = times(magtrans,(Matrix)invmagmatrices.get(probcnt));
Matrix finalmag = times(intermag,magdiff);
double magexpower=-get(finalmag,1,1)/2;
double probmag =

```

```

        (1/(Math.sqrt(Math.pow(2*3.14159,3)*(double)magdets.get(probcnt))))*
        Math.exp(magexpower);

    System.out.println("prob mag is");
    System.out.println(probmag);

}

```

The joint probability density of a gesture is the product of the two probability densities.

```

probdensity=(probpca*probmag);
totprob.add(probpca*probmag);

```

The recognized gesture is the gesture with highest likelihood or probability density.

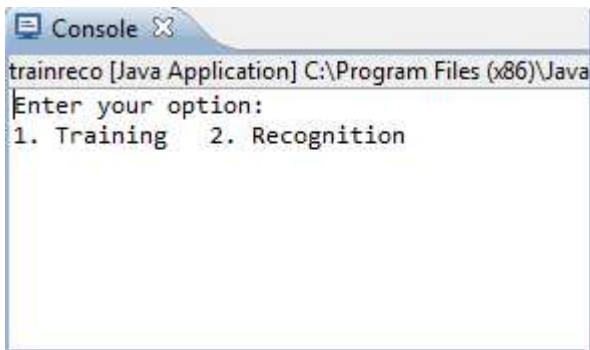
CHAPTER 6

EVALUATION OF GESTURAL HUMAN COMPUTER INTERFACE FOR SMART HEALTH

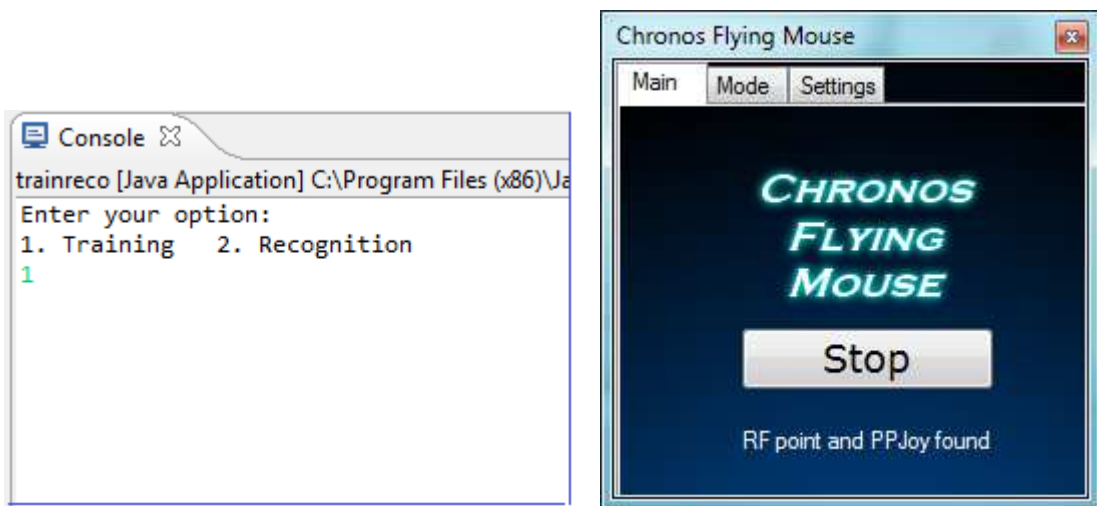
6.1 Instructions to Use the Application

Tie the Chronos watch to your right hand

1. Run the application
2. You will see “Enter your option” message in the console. Enter “1” to train your gesture.



3. When you press 1, the Chronos Flying Mouse window will pop up.

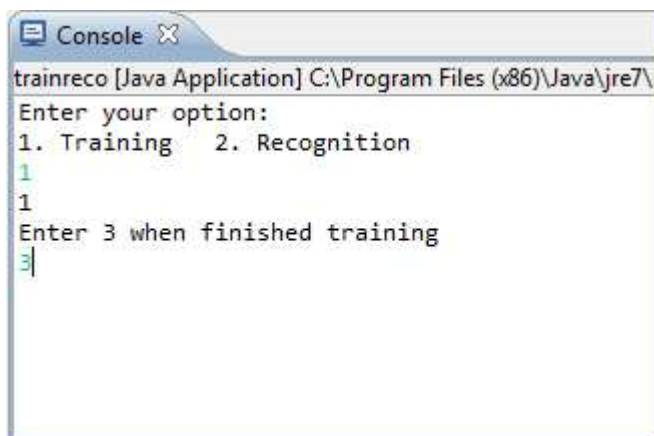


4. Click on “start/stop” button on the Chronos Flying Mouse window
5. Press the button on the watch that is on the bottom right of the watch. If everything is performed correctly, the cursor will move on the screen if you slowly move your hand.

6. Perform the gesture you want to train. After completing it go to the C:\\ drive to see if text file with the current time stamp is generated. That file will be generated only if you do heavy gesture with reasonable force. Gestures of very short duration will not be recorded.

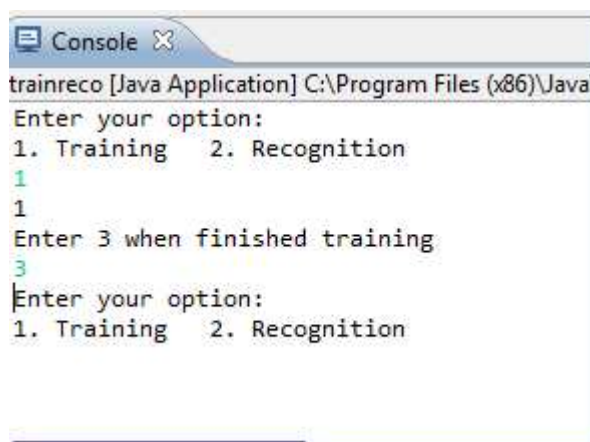
7. Repeat the same gesture seven more times, leaving a gap of 3-4 seconds between gestures. Check the C:// drive to confirm that a total of eight .txt files are generated corresponding to each gesture.

8. Once eight txt files are generated, close the Chronos Flying Mouse window and press 3 in the console where you see "Enter 3 when finished training".



```
trainreco [Java Application] C:\\Program Files (x86)\\Java\\jre7\\bin
Enter your option:
1. Training 2. Recognition
1
1
Enter 3 when finished training
3|
```

9. Once you have done that you will see "Enter your option" in the console.



```
trainreco [Java Application] C:\\Program Files (x86)\\Java\\bin
Enter your option:
1. Training 2. Recognition
1
1
Enter 3 when finished training
3
Enter your option:
1. Training 2. Recognition
```

If you wish to train another gesture then repeat steps 3 to 8.

10. Once you are done with training different gestures, enter "2" in the console.


```
trainreco [Java Application] C:\Program Files (x86)\Java\jre7\bin\
Enter your option:
1. Training  2. Recognition
1
1
Enter 3 when finished training
3
Enter your option:
1. Training  2. Recognition
2|
```

11. The Chronos Flying Mouse Window will pop up again. Click on the start button and perform any gesture from the set of gestures you have trained. Confirm that a new file corresponding to the gesture has appeared in C://.

12. Once you see the new .txt file that corresponds to the test gesture, enter " 1" in the console.

```
trainreco [Java Application] C:\Program Files (x86)\Java\jre7\bin\
Enter your option:
1. Training  2. Recognition
1
1
Enter 3 when finished training
3
Enter your option:
1. Training  2. Recognition
2
2
Training done
Enter your option:
1. Continue Recognition  2. Stop
1|
```

13. The application will display the recognized gesture (See Figure 5.12).

14. Repeat steps 10 to 13 to test the recognition as many times as you want.

15. Once you are done, enter "2" in the console to exit the application.

```
Console X
trainreco [Java Application] C:\Program Files (x86)\Java\jre7\bin\
Enter your option:
1. Training  2. Recognition
1
1
Enter 3 when finished training
3
Enter your option:
1. Training  2. Recognition
2
2
Training done
Enter your option:
1. Continue Recognition  2. Stop
2|
```

6.2 Evaluation

We have eight predefined gestures that map to email activities. In order to track the performance, we conducted an evaluation with a volunteer group consisting of two men and 13 women aged between 22 and 30 years. We performed two kinds of evaluation – volunteer independent and volunteer dependent. In volunteer independent evaluation, we pretrained all the eight gestures to be recognized and asked each volunteer to randomly choose and execute any five gestures out of the eight. In volunteer training dependent evaluation, each volunteer trained two to three gestures to the application and then tested the ability to recognize the same gestures. If misrecognition of gestures occurred during volunteer training independent evaluation, then the volunteer was asked to create a volunteer dependent model of the misrecognized gestures and test the accuracy of recognition.

Evaluation Setup: Each volunteer was requested to wear the watch and connect the eZ430-Chronos RF Access Point to the PC for receiving acceleration data. An overview of the application and its uses was given. We then explained all the eight gestures, how to do perform them and their purpose. The operating instructions for the watch were explained to the volunteers. The volunteers then used our application to train and/or execute the gestures.

Volunteer Training Independent Evaluation: For this evaluation, we used gesture models based on a pre-existing training set for all the fifteen volunteers to test gesture recognition. Each volunteer was asked to choose any five gestures out of the eight predefined gestures and repeat each 5 times. Thus, each volunteer executed $5*5 = 25$ gestures, the recognition of which was measured.

Example of Volunteer Training Independent Evaluation: Since a pre-existing set of gesture models is used, we test the recognition of gestures performed by a volunteer. A volunteer chooses send, open, punch, launch and search gestures to test recognition. She repeats each gesture 5 times and

tests its recognition, i.e., send is tested five times; open is tested five times and so on. Figure 6.1 shows the recognition accuracy for each gesture performed by that volunteer.

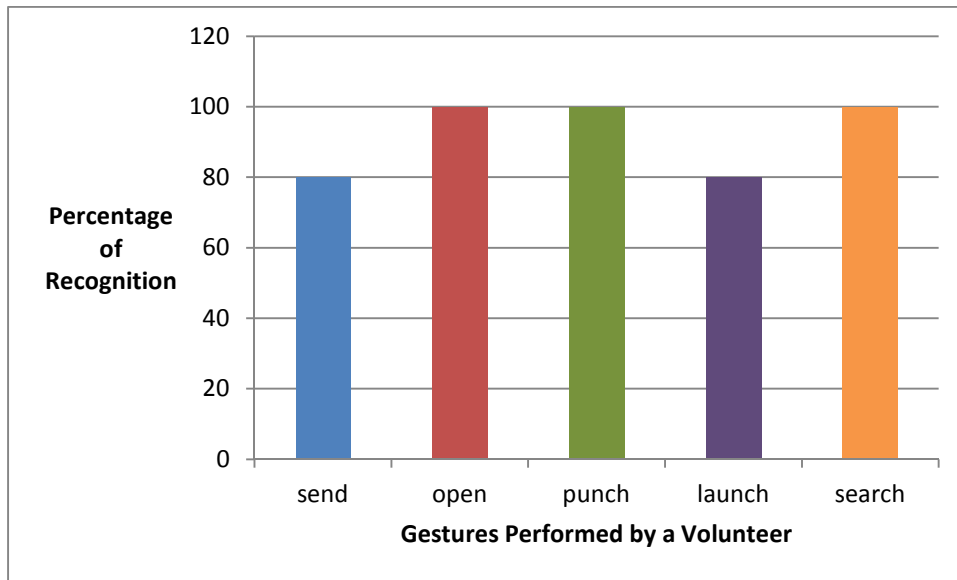


Figure 6.1: Recognition of User Gestures Based upon Predefined Training

Open, punch and search are recognized 100% of the time. Send and launch are recognized four times out of five.

Volunteer Training Dependent Evaluation: The rationale for this type of evaluation is to accommodate user preferences and allow for the fact that different users may perform the same gesture in a slightly different way. For this evaluation, each volunteer chooses two to three gestures from the set of eight gestures and performs training. For training a gesture, volunteer has to repeat it eight times. Once the training is completed for the selected gestures, the same gestures are performed once again, and prediction accuracy is noted.

Example of Volunteer Training Dependent Evaluation: In this evaluation, a volunteer chooses delete, scroll down and scroll up gestures. She repeats the delete gesture eight times for training.

Similarly, she trains the system with scroll down and scroll up gestures. When training is completed, she tests the recognition of each gesture. Figure 6.2 shows the recognition rate for the gestures she selected and trained.

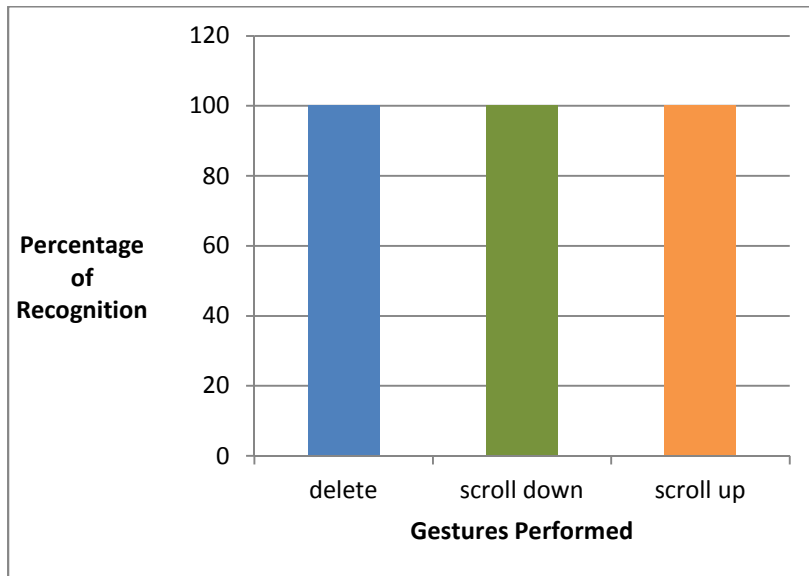


Figure 6.2: Recognition of Gestures After User Training

We observed that the accuracy increased when the volunteer has trained their gestures.

On the whole, for user independent training evaluation, 278 gestures were performed by 15 volunteers of which 240 of them were recognized correctly. Figure 6.3 shows the average recognition accuracy for all 15 volunteers.

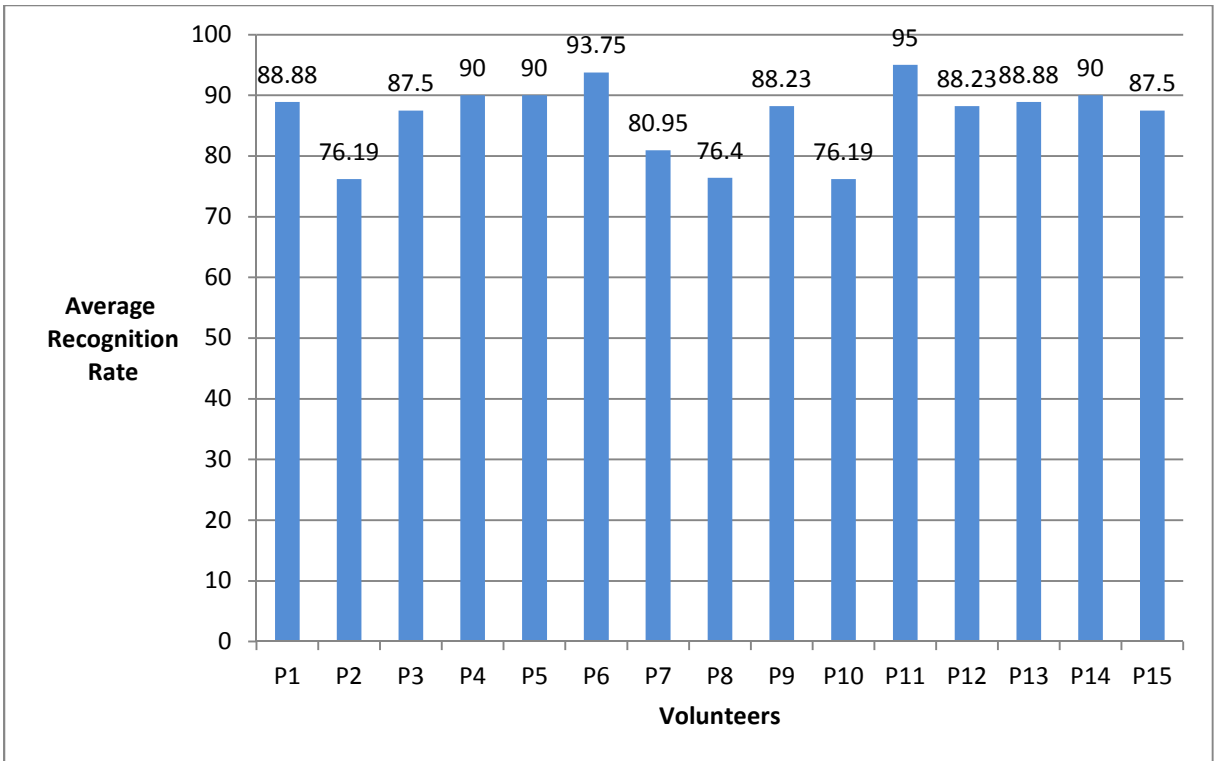


Figure 6.3: Average Recognition Rate for All 15 Volunteers on Pretrained Models

Overall accuracy of the gestures is given by 240/278 (86.33%).

When the volunteers trained their own gestures, essentially all the gestures were recognized correctly. Figure 6.4 shows the percentage of recognition when each volunteer trained selected gestures prior to testing recognition.

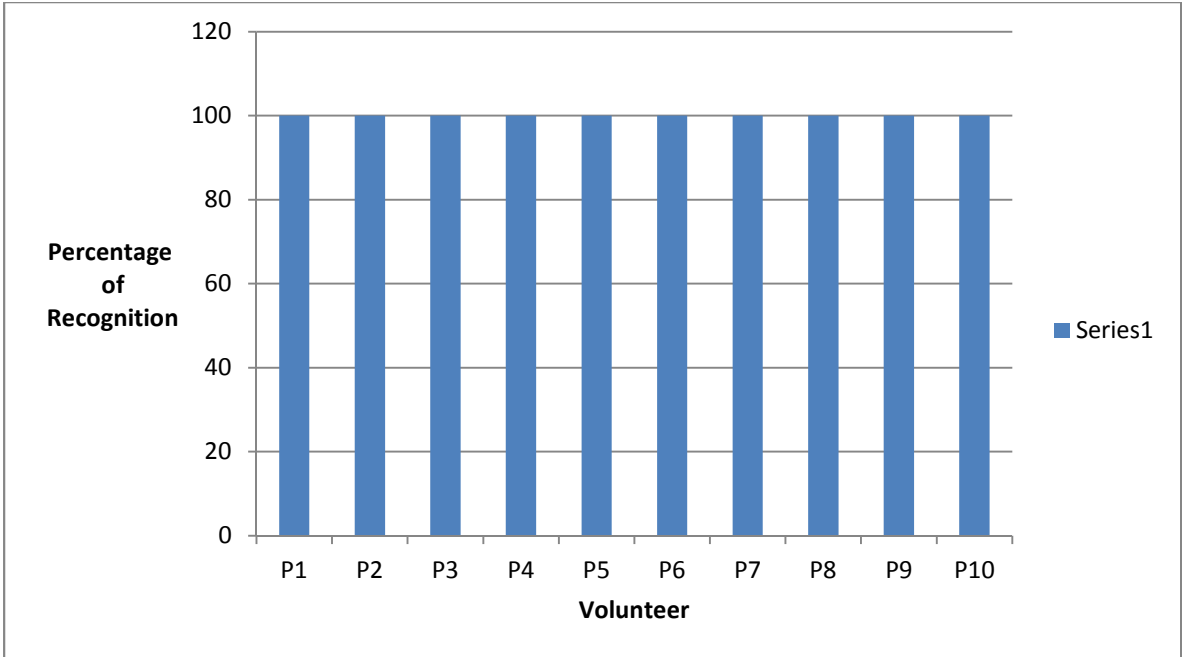


Figure 6.4: Recognition of Gestures by Volunteer Dependent Training

Figure 6.5 shows the average accuracy of each gesture when 15 volunteers tested the recognition of gestures in volunteer training independent evaluation.

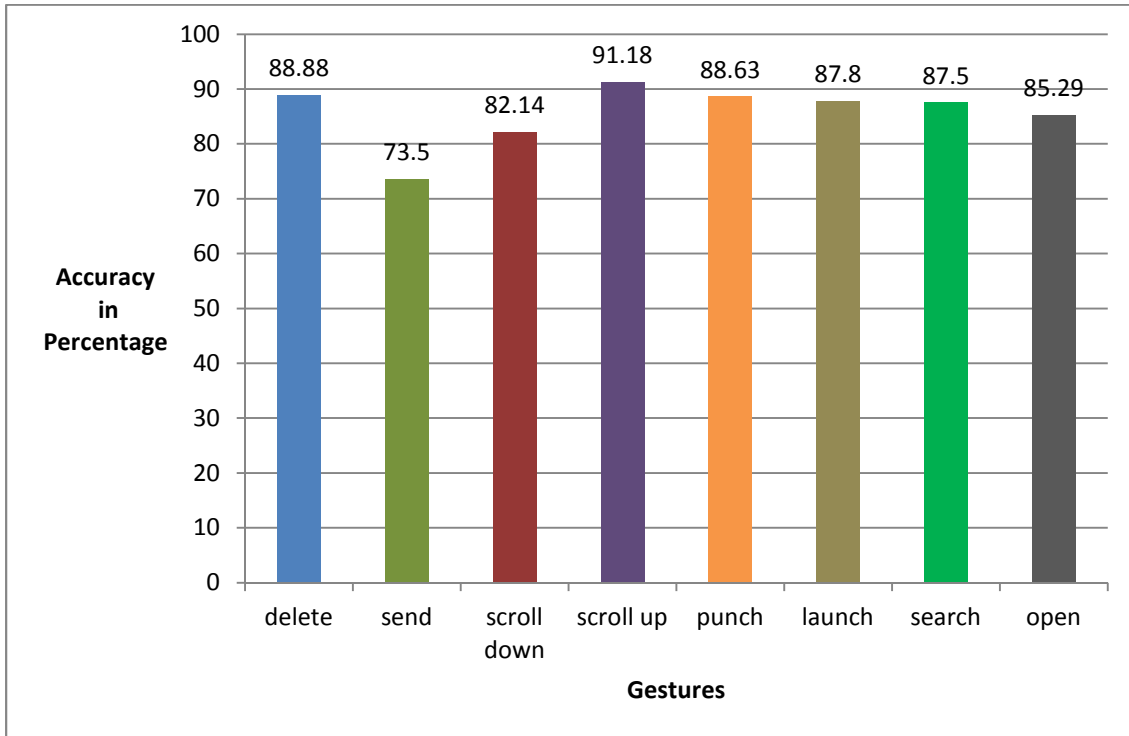


Figure 6.5: Accuracy of All Gestures

The overall accuracy of the gestures is 86.33%.

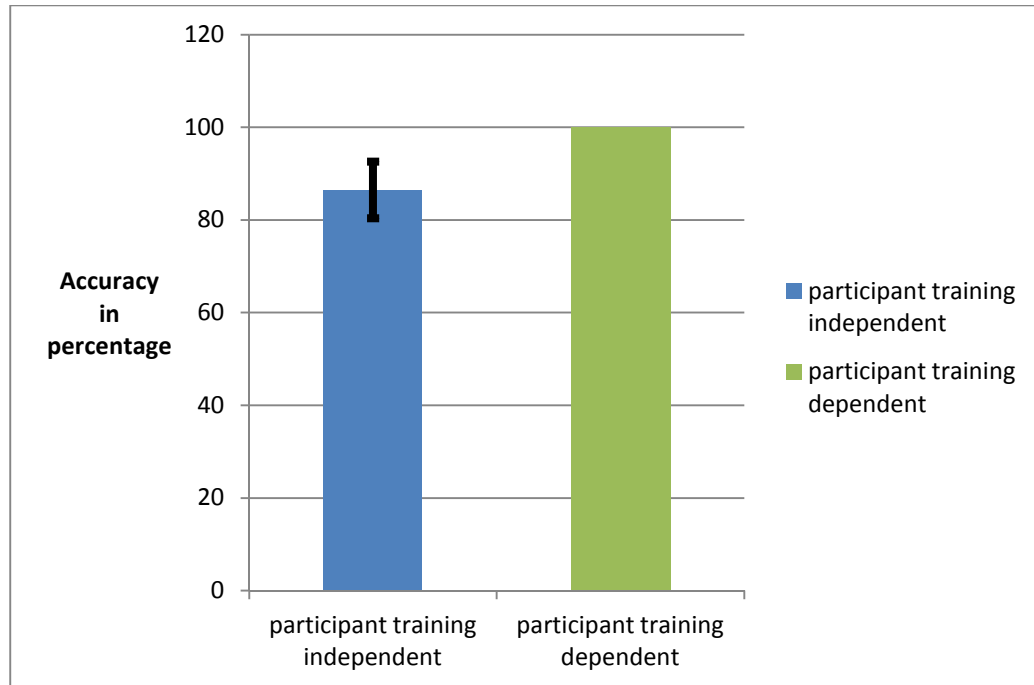


Figure 6.6: Error Bar Chart Representing both Training Dependent and Independent Evaluation

Continuous Evaluation:

We also evaluated the Gestural Human Computer Interface for Smart Health while performing continuous gestures to simulate real world use. We picked pairs of gestures and performed them continuously. We chose pairs of gestures that make sense in an email session. The pairs of gestures we chose for this type of evaluation are: launch-send, scroll up-scroll down, forward-send and search-delete. We thus evaluate filtering and segmentation modules along with the accuracy of gesture recognition. Figure 6.7 shows the accuracy of pairs of gestures performed continuously. Scroll Up and scroll down gestures when performed together got less accuracy percentage. This is because every time when the user performs scroll up gesture and after finishing it, one has to move hand downwards immediately in order to perform scroll down gesture. So that unwanted movement may not be segmented properly as, it is also done with enough force.

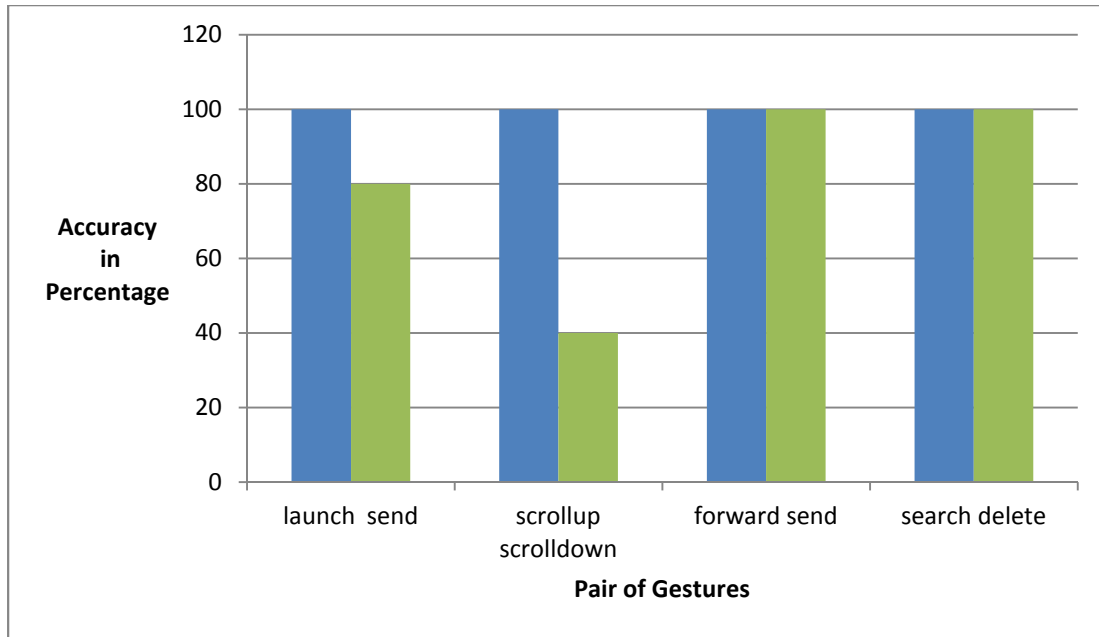


Figure 6.7: Accuracy of Pairs of Gestures Performed Continuously

Our Model Versus HMM:

We also compared our performance against published HMM models for gesture recognition with a Wii controller [10]. Gestures used by the group for tests and evaluations are shown in figure 6.8.



Figure 6.8: Gestures Used for Evaluation of HMM Model using Wii-mote

Figure 6.9 shows the average recognition rate for each of the five gestures by using HMM model [10].

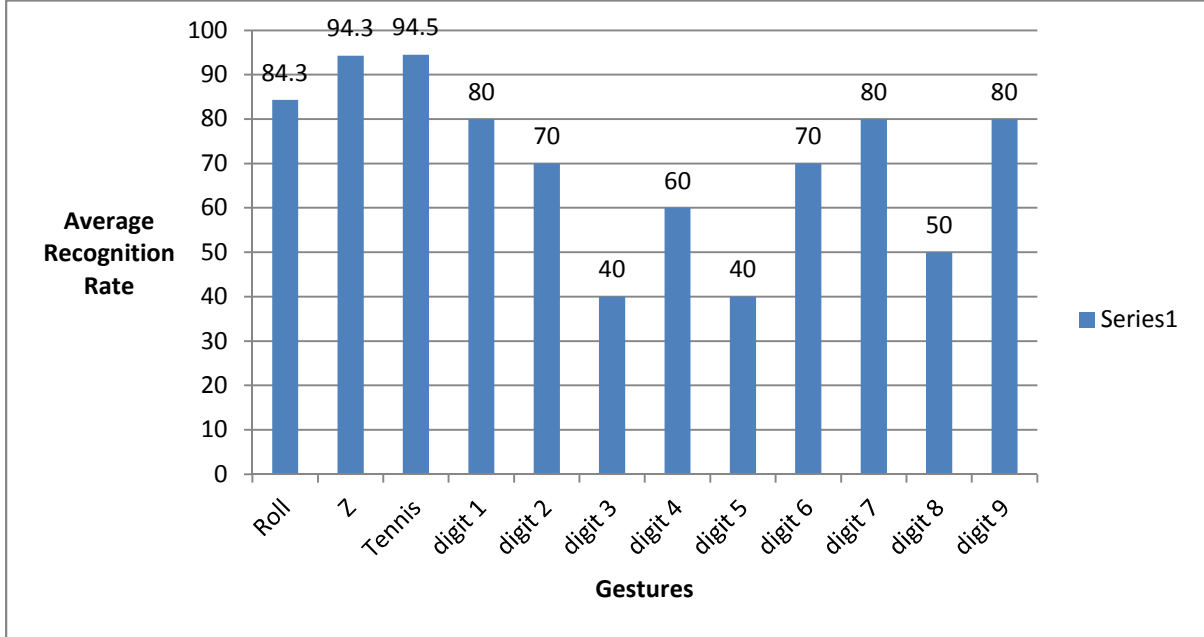


Figure 6.9: Average Recognition Accuracy of Gestures Using HMM Model

To compare the PCA/magnitude model versus HMM, we repeated the same gestures shown in Figure 6.8 and measured the accuracy. Note that principal components by definition can't distinguish between symmetrical closed loop gestures like square and circle as the variance is symmetric. So we restricted the comparative evaluation to the remaining three gestures from the reference gestures. Figure 6.10 shows the average recognition accuracy for the three gestures using our model.

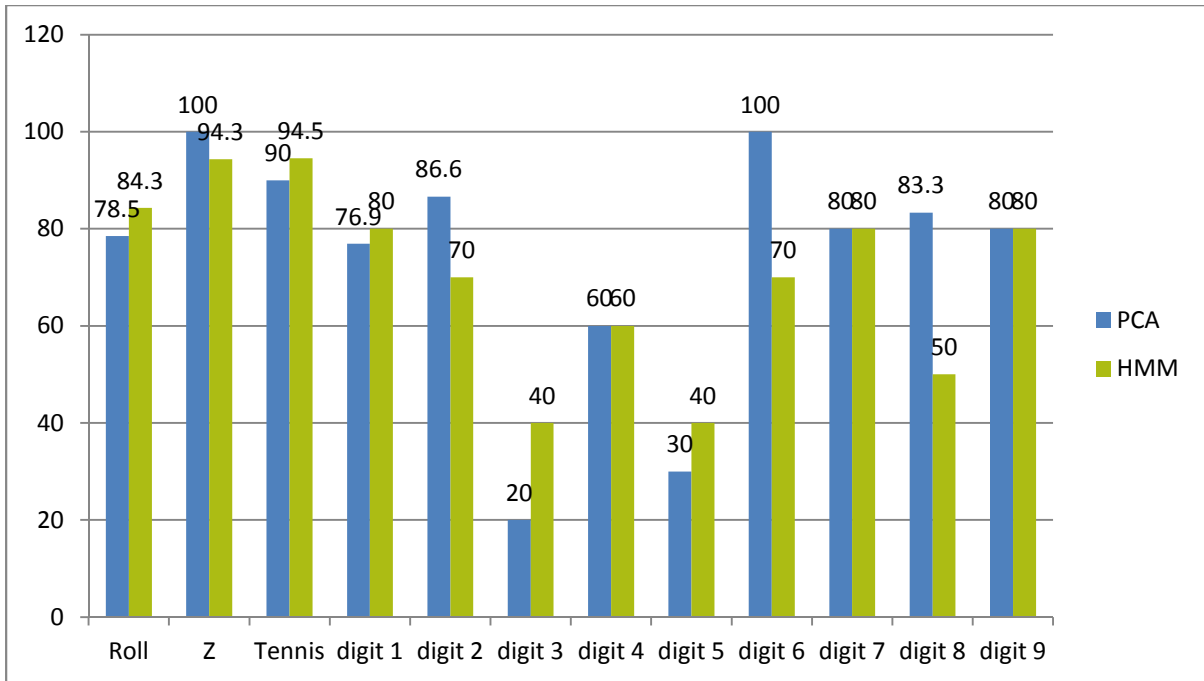


Figure 6.10: Graph Comparing Gesture Recognition Accuracy of our Model Versus HMM

Based on the above comparison, our model yields an average accuracy of 73.78% while the HMM model has an average accuracy of 70.26%.

Run Time of the Application:

Run time was tested on a PC with Intel Core i3 2.40 GHz processor and 4 GB memory. To calculate the run time of the application, a gesture was repeated eight times to train the application. This was followed by a test gesture.

Run Time of File Searching Algorithm:

The time taken by the program to search for eight training samples in a specific folder is 30 ms.

Run Time for Training:

The time taken to build the training set for a gesture based on eight training samples is 142.5 ms.

Run Time for Recognition:

The time taken for the recognition of a gesture when the application is trained by five gestures, each repeated eight times, is 22 ms.

Thus, the total runtime of the program is approximately 194.5 ms. This time doesn't include the time taken by the user to perform a gesture.

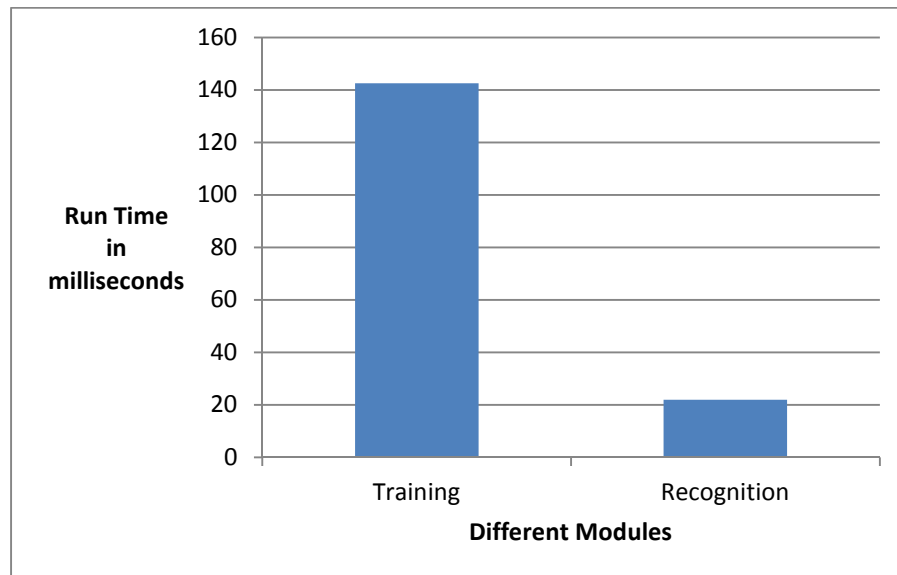


Figure 6.11: Run Times for Different Modules in Application

Run Time of Gesture recognition using Wii-mote:

In order to calculate run time for this application, a gesture is trained and then tested for recognition. We have calculated total run time of Gesture recognition using Wii-mote and it is 19415 ms. This includes the time taken by the user to train a simple gesture "moving hand left to right" once and testing it for recognition.

In the paper on HMM based recognition [11], run time analysis is calculated for their application. They trained 5 gestures, with the time taken for the recognition of each gesture being 52 ms (Table 6)..

Table 6: Comparison of Run Times of Models

	Our Model	HMM Model
# of gestures trained	5	5
Recognition time	22 ms	52 ms

Evaluation of Application in Terms of Energy Expenditure:

We also evaluated “Gestural Human Computer Interface for Smart Health” in terms of energy expenditure. For this, we used “Beurer Heart Rate Monitor.” [14] With the help of this device, we can record the heart rate for a desired duration. We recorded the heart rate of a user in three situations - at rest, during an email session using a traditional interface, and while performing the same set of activities by performing heavy gestures. We conducted t-tests to compare the three sets of heart rate readings. Table 7 shows the heart rate of a user at rest. Mean of that data is 74 and standard deviation is 5.11. Table 8 shows the heart rate of a user during a conventional email session. Mean of that data is 70.63 and standard deviation is 3.32. Table 9 shows the heart rate of a user performing heavy gestures during an email session. Mean of that data is 86.81 and standard deviation is 8.02

Table 7: Heart Rate of User while at Rest

Minutes	Heart Rate
1	73
2	79
3	78
4	70
5	63
6	69
7	78
8	79
9	77
10	76
11	72

Table 8: Heart Rate during Conventional Email Session for 8 min

Minutes	Heart Rate
1	73
2	74
3	71
4	68
5	62
6	70
7	72
8	72
9	72
10	70
11	73

Table 9: Heart Rate during Heavy Gestures for 10 min

Minutes	Heart Rate
1	74
2	74
3	81
4	84
5	88
6	90
7	88
8	90
9	93
10	100
11	93

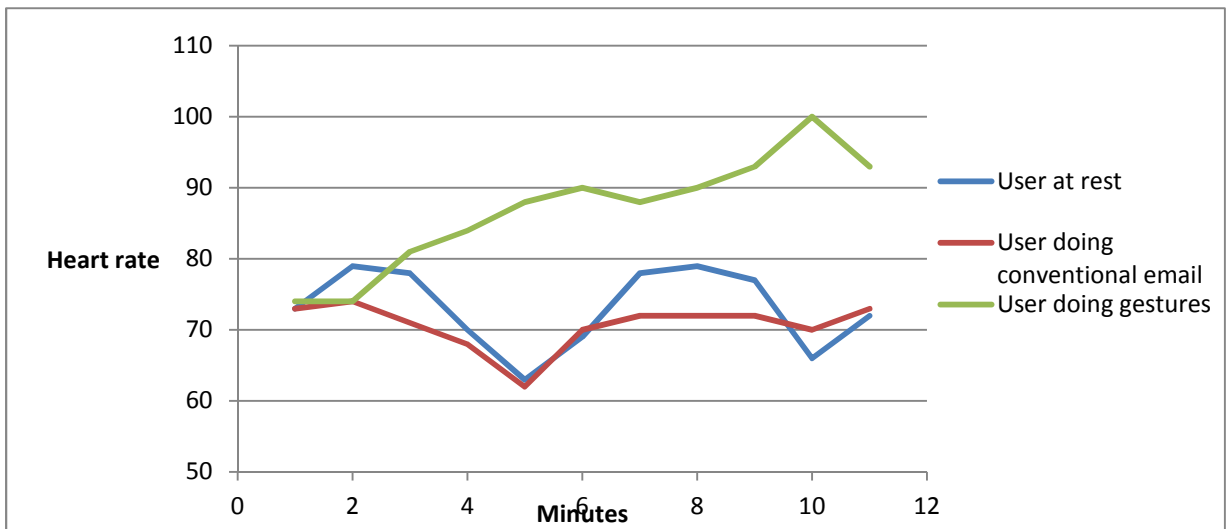


Figure 6.12: Graph Showing User Heart Rate in Three Different Situations

Results of t-test conducted for 'Heart rate while doing gestures' data versus 'Heart rate during Rest' data:

For One-tailed distribution and Two sample unequal variance:

p-value : 0.0001. There is a significant difference between two groups.

Results of t-test conducted for 'Heart rate while doing gestures' data versus 'Heart rate while doing conventional actions' data:

For One-tailed distribution and Two sample unequal variance:

p-value : 0.000058 which is less than 0.05. So there is a significant difference between two groups.

t-test conducted for 'Heart rate while doing conventional actions' data and 'Heart rate during rest' data:

For One-tailed distribution and Two sample unequal variance:

p-value: 0.11 which is greater than 0.05. So there is no significant difference between two groups.

The comparative results show that the heart rate increased significantly when the user performed heavy gestures compared with a conventional email session. The results demonstrate that energy expenditure is higher when "Gestural Human Computer Interface for Smart Health" is used for routine activities.

CHAPTER 7

CONCUSION AND FUTURE WORK

In this section, the contribution of this thesis is summarized. Some thoughts on refining the project and possible directions for future work are presented.

7.1 Conclusion

We proposed the principle of working out while working to achieve potential health benefits without dedicated exercise sessions. We implemented this idea in the form of a Gestural Human Computer Interface for Smart Health where users can control computer applications by performing heavy gestures. We used cost effective off the shelf hardware, the Chronos watch by Texas Instruments, which measures the acceleration generated by heavy gestures. We developed an application for training and recognition of gestures performed by a user. For the recognition, lightweight algorithms based on principal component analysis and Naïve Bayes Classification are used. We showed that the gestures can be successfully customized to accommodate user variation. We observed a significant increase in the heart rate when using the Gestural Human Computer Interface for Smart Health. The successful implementation of the Gestural Human Computer Interface for Smart Health confirms that the application is feasible and effective in raising the heart rate.

7.2 Future Work

We discuss some of the limitations of the project and scope for enhancement in this final section. Since the acceleration data is reduced to the principal component, it cannot distinguish between temporal profiles that are different but with identical time independent distributions. For example, an action consisting of acceleration followed by deceleration cannot be distinguished from

deceleration followed by acceleration if they contain the same 3D acceleration values, although in a different order. For example, the current approach can't recognize the distinction between turning a page from left to right and turning a page from right to left. This could potentially be resolved by breaking down each action in multiple temporal windows, with a principal component for each sub segment.

Since the recognition algorithm is essentially a maximum likelihood classification approach, any gesture (not previously used in training) will be classified to a gesture. This can be resolved by a secondary algorithm that takes the temporal pattern into account, especially in those cases where there is ambiguity with regard to classification.

Though our approach compares favorably with HMM, the platforms used for comparison were not identical – TI Chronos and Wii-mote respectively. This can affect both accuracy and processing time.

The ideal comparison should use identical gestures on the same hardware.

The interface can potentially be highly personalized in allowing users to create and use any gestures of choice to control any aspect of human computer interaction. Using such active interfaces for even short periods of time during the workday could alleviate boredom and counteract the effects of a sedentary lifestyle.

REFERENCES

1. Gesundheit, N. Filling the treatment gap in the weight management of overweight and obese patients. *International Journal of Obesity Supplements*, 2(1). S39-S42.
2. Blair, SN. and Connelly JC. How much physical activity we do? The case for moderate amounts and intensities of physical activity. *Research Quarterly for Exercise and Sport* , 67(2). 193-205.
3. Ortega-Carrillo, H. and Martinez-Miron, E. Wired gloves for everyone. *VRST '08 Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, (Bordeaux, France, 2008), ACM, 305-306.
4. Sethi, A. *Multimedia education theory and practice*, International Scientific Publishing Academy, NewDelhi, 2005, 187.
5. Alhalabi, M., Daniulaitis, V., Kawasaki, H., Tetsuya, M. and Ohtuka, Y. Future Haptic Science Encyclopedia: An Experimental Implementation of Networked Multi-Threaded Haptic Virtual Environment. in *HAPTICS '06 Proceedings of the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, (USA, 2006), IEEE Computer Society, 76.
6. Chen, F., Fu, C., Huang, C. Hand gesture recognition using a real time tracking method and hidden Markov models. *Image and Vision Computing*, 21(3), 745-758.
7. Kim, D., Hilliges O., Izadi, O., Butler, A., Chen, J., Oikonomidis, I. and Oliver, P. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. in proceedings of the *25th annual ACM symposium on User interface software and technology*, (MA, USA, 2012), ACM, 171-176.
8. Leap Motion , <https://www.leapmotion.com>.
9. Texas Instruments, <http://www.ti.com>.
10. Schlomer, T., Poppinga, B., Henze, N. and Boll, S. Gesture recognition with a Wii controller. in proceedings of the *2nd international conference on Tangible and embedded interaction*, (Barcelona, Spain, 2008), ACM, 11-14.

11. Joselli, M. and Clua, E. gRmobile: A framework for touch and accelerometer gesture recognition for mobile games. *SBGAMES '09 Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment*, (Brazil, 2009), IEEE Computer Society, 141-150.
12. Bedregal, B., Costa, O. and Dimuro, G. Fuzzy rule based hand gesture recognition. *Artificial Intelligence in Theory and Practice*, 217(6). 285-294.
13. Texas Instruments, http://processors.wiki.ti.com/index.php/Chronos_Flying_Mouse
14. Beurer, http://www.beurer.com/web/en/product/heart_rate_monitors/heart_rate_monitors
15. Ren, Z., Meng J. and Yuan, J. Depth camera based hand recognition and its applications in human computer interaction. in *8th International Conference Information Communications and Signal Processing*, (Singapore, 2011), Conference Publications, 1-5.
16. Wang, L., Gu, T., Tao X. and Lu, J. A hierarchical approach to real-time activity recognition in body sensor networks. *Pervasive and Mobile Computing*, 8(1). 115-130.
17. Bachlin, M. and Troster G. Swimming performance and technique evaluation with wearable acceleration sensors. *Pervasive and Mobile Computing*, 8(1). 68-81.
18. Kong, F. and Tan, J. DietCam: Automatic dietary assessment with mobile camera phones. *Pervasive and Mobile Computing*, 8(1). 147-163.
19. Freitag, G., Trankner, M. and Wacker, M. Enhanced feed-forward for a user aware multi-touch device. in *7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, (Denmark, 2012), ACM, 578-586.
20. Ewerling, P., Kulik, A. and Froehlich, B. Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions. in *ACM international conference on Interactive tabletops and surfaces*, (Cambridge/Boston, MA, USA, 2012), ACM, 412.
21. Oh, K., Jeong, Y., Kim, S. and Choi, H. Gesture recognition application with parametric hidden markov model for activity based personalized services in APRiME. in *2011 IEEE International Multi-*

- Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, (Miami Beach, FL, 2011), Conference Publications, 189-193.
22. Yang, Z., Li, Y., Zheng, Y., Chen, W. and Zheng, X. An interaction system using mixed hand gestures. in 10th asia pacific conference on Computer human interaction, (Matsue, Japan, 2012). 125-132.
 23. Liu, X. and Fujimura, K. Hand gesture recognition using depth data. in *Sixth IEEE international conference on Automatic face and gesture recognition*, (Washington, DC, USA, 2008). IEEE Computer Society. 529-534.
 24. Nickel, K., Seemann, E. and Stiefelhagen, R. 3D-tracking of head and hands for pointing gesture recognition in a human-robot interaction scenario. in *Sixth IEEE international conference on Automatic face and gesture recognition*, (Washington, DC, USA, 2004). IEEE Computer Society. 565-570.
 25. Schmidt, G. and House, D. Towards Model-Based Gesture Recognition. in *Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, (Washington, DC, USA, 2000). IEEE Computer Society. 416.
 26. Hongo, H., Yasumoto, M., Niwa, Y., Ohya, M. and Yamamoto, K. Focus of Attention for Face and Hand Gesture Recognition Using Multiple Cameras. in *Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, (Washington, DC, USA, 2000). IEEE Computer Society. 156.
 27. Marcel, S., Bernier, O., Viallet, J. and Collobert, D. Hand Gesture Recognition Using Input-Output Hidden Markov Models. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, (Washington, DC, USA, 2000). IEEE Computer Society. 456.
 28. Reilly, R. Applications of face and gesture recognition for human-computer interaction. in *sixth ACM international conference on Multimedia: Face/gesture recognition and their applications*, (Bristol, England, 1998). ACM. 20-27.

29. Kim, D., Song, J. and Kim, D. Simultaneous gesture segmentation and recognition based on forward spotting accumulative HMMs. *Pattern Recognition*. 40 (11), 3012-3026.
30. Liu, J., Zhong, L., Wickramasuriya, J. and Vasudevan, V. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* , 5(6), 657-675.
31. Wright, M., Lin, C., O'Neill, E., Cosker, D. and Johnson, P. 3D gesture recognition: an evaluation of user and system performance. in *9th international conference on Pervasive computing*, (San Francisco, USA, 2011). Springer-Verlag. 294-313.
32. Licsár, A. and Szirányi, T. User-adaptive hand gesture recognition system with interactive training. *Image and Vision Computing*, 23 (12). 1102-1114.
33. Suk, H., Sin, B. and Lee, S. Hand gesture recognition based on dynamic Bayesian network framework. *Pattern Recognition*, 43(9). 3059-3072.

VITA

Sowmya Ginjupalli was born on August 18, 1988, in Guntur, Andhra Pradesh, India. She was educated in a local public school and graduated from high school in 2005. She then completed her Bachelor's degree in Electrical and Electronics Engineering from Acharya Nagarjuna University in Guntur in 2009. Upon the completion of her Bachelor's, she worked as an Assistant Systems Engineer at Tata Consultancy services until November 2010.

In December 2010, Ms.Sowmya came to the United States to study Computer Science at the University of Missouri-Kansas City (UMKC), specializing in Software Engineering. During summer 2012, Ms. Sowmya worked as an intern at Children's Mercy Hospitals and Clinics. She was offered a full time position at Sprint Nextel Corporation. Upon completion of her requirements for the Master's Program, Ms.Sowmya plans to work for Sprint Nextel Corporation.

