ACTIVE MOBILE 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
PRATIMA GORLA

B.Tech, Jawaharlal Nehru Technological University, 2010

Kansas City, Missouri
2013

APPROVAL PAGE

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

Supervisory Committee

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

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

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

ACTIVE MOBILE INTERFACE FOR SMART HEALTH

Pratima Gorla, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2013

ABSTRACT

Computer interfaces are rapidly evolving beyond the traditional keyboard-mouse-monitor triad. The widespread availability of touch screens and voice recognition software has made it possible to execute commands in many different ways, especially on mobile computing devices. While this has made it possible to get work done with the minimum of physical exertion, the consequent sedentary life style increases the risk of health problems like obesity and cardiovascular disease. While the use of accelerometers and camera based sensors in electronic gaming involving physical exercise has grown considerably, such interfaces are yet to find their way into the workplace. This thesis is based on the general idea of making human computer interfaces more energetic for routine use, and not just for gaming. It proposes the idea of a virtual keyboard that requires energetic arm or leg motions to type. Tri-axial accelerometers embedded into a gaming device are employed together with a light-weight motion detection and interpretation algorithm to simulate the act of typing. The thesis includes quantitative and qualitative evaluation of the prototype interface deployed for use on a mobile device.

TABLE OF CONTENTS

ILLUSTRATIONS

Figure Page

TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Technology is advancing day by day. On one hand, it is simplifying man's activities but it has negative impacts on the other. The real world is being progressively substituted by a virtual existence. As technology progressively makes tasks easier and easier, it is also making it harder to achieve the minimum amount of physical activity required for a human being to maintain a healthy body. Increased sedentary life style along with unhealthy food habits are leading to many health problems like obesity, cardiovascular diseases, diabetes and depression.

The negative impact of technology on younger generation is also a problem. The percentage of children watching the television and playing console games has increased. There has been a continual increase in the percentage of obesity in children over the years. The prevalence of obesity in children increased from 7% in 1990 to 18% in 2010 [6]. Adolescent obesity increased from 5% to 18% [6] over the same duration. Current statistics indicate 43% of boys and 40.4% of girls are either overweight or obese [7]. There has been an increasing trend, from 30% in 1990 to 45% in 2010. If the same trend continues, more than half the children will be overweight by 2020. Obesity is seen more in low income and low education families in US. The problem of obesity is also affecting the economy of the nation. If current trends of food habits and life style continue, national health care will cost between $861 and $957 billion by 2030 [7].

The problem of obesity occurs when energy intake is greater than the energy spent by the person. Patterns in diet and physical activity performed by a person affect weight. Attempts to reduce the diet consumed can result in malnutrition and affect growth of the child. An alternate choice is to increase the physical activity performed by a child. Providing body with minimum

physical activity not only improves a child's body composition psychologically but also regulates body weight [6].

Physical activity in children and teenagers can be influenced by various factors like psychological, social-contextual abilities of a person. It requires social support of parents, coaches or peers [5]. Three major factors, which motivate children to perform physical activity, are physical competence, social acceptance and fun derived from participation [5]. The third factor is the driving factor as it decreases negative and increases positive experiences related to physical activity. Enjoyment in physical activity keeps children and teenagers coming back, thus increasing their activity levels and obtaining better long-term healthy-outcomes.

1.2 Problem Statement

Children and adolescents are easily addicted to gaming and watching television. They spend more time on indoor activities that lack physical activity rather than outdoor games like football. Access to open spaces to participate in outdoor activities may be limited, and conventional methods of physical activities like gym may not be affordable for everyone. Therefore, it has become a challenge to parents and society to change existing behavioral patterns.

Smart phones are now cheaper than before. While they provide access to a range of entertainment as well as work related activities, their use long periods of physical inactivity. What if their use required caloric expenditure? The ideal solution should be acceptable, fun to use, and be compatible with productivity.

Technical challenges are the application should perform gesture recognition, be affordable, portable and computationally simple. An application that can recognize physical activity performed by users through gesture recognition can motivate users. The application should not be

computationally intensive so that the processing can be performed by mobile phones, which have limited processing power to carry out complex processing.

1.3 Thesis Outline

This thesis proposes the idea of an Active Mobile Interface (AMI) that requires physical activity while using mobile applications. It replaces the keypad/touchscreen conventional messaging interface with one that requires considerable energy expenditure. Vigorous actions performed by the user are recognized and mapped to a virtual keyboard. Gestures can be performed are either by hand or by leg. The principle is to allow a set of physical actions to be translated to device commands.

Recognition of the gestures with minimum processing time is a key requirement. Simple techniques like Peak Detection, Valid Point Detection algorithms are applied to detect and identify. Exceptional cases are handled by filtering channel leakage and noise. The Nintendo Wiimote [37] is used as an input device. It contains an accelerometer to detect 3-axis acceleration and supports Bluetooth to establish connection with end devices. It is cost effective and simple to use. The system requires no training and can be adapted to a variety of mobile or desktop application settings.

CHAPTER 2

RELATED WORK

2.1 Does Technology Lead to Obesity?

Obesity is a serious problem seen in 40 percent of the children and adolescents [15]. It is one of the world's growing concerns. Obesity is due to unhealthy diet, sedentary lifestyles etc. Obesity seen in the children can increase the chance of them being prone to cardiovascular disease, hypertension, cancer, osteoarthritis and developing diabetes. It costs nearly 190 billions of dollars annually for medical expenses due to obesity [15]. Obesity is causing trouble not only to an individual but also to the health care industry and also the government economy.

Development of overweight can be due to various reasons. However, study has shown children and adolescents spend longer duration on playing digital games, watching television and using computer. This in turn reduced the time spent on the weekly physical activity. It is not just affected physical statistics of a child but also reduced his interaction with his family and surroundings. Increased use of information and communication technology has come one of the significant reason for overweight and obesity in children.

2.2 Applications for Physical Activity

Study show 60 minutes of physical activity for 5 days in a week can help people to overcome problem of obesity. Female ratio can be more prone to overweight than boys can [3]. In addition, family of the child and environment in which he is grown also affects the children physical activity [3]. Many fitness programs like gym, swimming, yoga etc are available but not everyone may be able to take advantages of them.

Many applications are developed to face the effects of obesity in children and adolescents. Applications with a virtual trainer to monitor physical activity of the users are introduced in market.

Second Life [18] is one such application where a virtual life is created and animated images called avatar can be seen. Avatar indicates identity of user. Second Life helps user to watch a trainer to perform activities and replicate them. Elizabeth *et al.* [19] says that appearance of user's avatar may affect their real life appearance as well. Third party can modify environmental factors in Second Life. Modifying the surrounding of an avatar accordingly may influence their behavioral patterns as well [19]. However, these applications are developed for desktop system; it is not possible to carry a desktop.

Kinect [21] is another motion-sensing device developed by Microsoft. It is developed for gaming console. It contains a web-camera, which detects the user standing in front of the gaming console. Users can give commands to the console through gestures and voice. Kinect is used for capturing voice, facial and gestural recognition.

Kinect [21] is fun to use and it has many applications like fruit ninja, Zumba dance etc., which motivate users to do lots of physical activity. However, this system is not portable and it is pricy. It requires lot of space to avoid being hit by surroundings. Also, not all applications in Kinect may contain heavy gestures.

Figure 2.1: Microsoft Kinect [21]

This system requires proper light conditions to capture user images but this may not be possible in all scenarios. Obstacles during gaming between the user and the gaming console may

cause disturbance during the game. Kinect contains infrared laser projector and CMOS sensor for capturing user video data, infrared rays may cause skin irritations [22]. Other games like Mario Fit, Human Pacman etc. may need additional software that may not attract many users.

2.3 Sensors for Recognizing Physical Activity

Human activities can be recognized by wearing sensors all around the body [35]. These are required mostly for recognizing any disorders under medical surveillance. The need for tracking health condition of the patient has increased and it needs to be performed during their daily activities.

Most of the conventional methods of tracking activities of the patient require assistance with the observation or patients reporting by end of the day. However, this can cause errors and degrade the quality of care. With the help of an automated system, personal devices can be used for monitoring physical activities.

However, as per Lee et al [36] not every one may carry smart devices for monitoring physical activity of people. Numbers of smart device users are increasing day by day but still these devices are not available to everyone around the world. Using a single sensor in hand for obtaining physical activity may affect the accuracy obtained. It may not be as efficient as attaching multiple sensors to different parts of the body. Accuracy of using one sensor is reduced by 35% when the system is tested with five sensors in the same environment [36].

In addition, mobile phones with integrated sensors are used for sensing physical activity but many other factors like internal memory of the phone, other applications running on the phone need to be considered. Lack of memory and other services running on the phone may slow down the application. The application for detecting physical activity of the user consumes more resources for processing and sensing, which in turn effects other applications running in the system. This

reduces the efficiency of the whole system. To overcome this we have come with an idea of separating the sensing and processing devices [17]. Applications in mobile devices are used for tracking diet of the user and external trackers are used for monitoring user's activity. External devices for tracking user activity comes with an accelerometer or/and altimeter. We have many devices like Nike's fuel band, Fitbit One, Jawbone's Up, BodyMedia Fit [17]. However, all of these devices are costly and they are only for monitoring physical activity of a user but not for motivating the user. These are used with conventional physical exercise.

Figure 2.2: Mobile Fitness Applications [17]

Table 1: Comparing Different Platforms

	Portable	Motion Recognition	Requires Dedicated Space	Cost	Motivates User
Kinect [21]	No	Yes	Yes	$100 (Kinect) + $300 (Xbox) + $300 (TV)	Yes
Second Life [18]	No	No	No	$0 (Second Life) + $500 (High end Desktop configuration)	Yes
Mobile Fitness Apps [17]	Yes	No	No	$100 (Appending Hardware) + $200 (Smartphone)	No
Active Mobile Interface	Yes	Yes	No	$20 (Wiimote) + $200 (Smartphone)	Yes

2.4 Models for Gesture Detection

Gesture recognition is about interpreting user gestures by using various algorithms. Gestures are generally performed using hand or face but they can be originated by the motion of any body part. Gesture recognition is to simplify interaction between machines and human. Gesture recognition is done by using various techniques like image processing. Gesture recognition is of two types: first is offline gestures where processing takes place after the user interacts with system and second is online gestures where direct manipulation of output takes place. Gestures are movement of hands and arms in different directions. Gestures are different from other body language as they have more association with speech. Gesture recognition is used for automation. Automation of various systems used in our daily tasks does not just simplify our work but lessen the physical activity. Gesture recognition is used various applications like remote control for giving instruction, gaming applications, sign language recognition. Here we discuss some of the algorithms to recognize gestures.

2.4.1 Hidden Markov Model for Gesture Recognition

Hidden Markov Model (HMM) is a statistical model and it considered simplest Bayesian network as per Yang *et al* [2]. In HMM, state is not visible to the users directly but in other Markov models, state is not visible. In HMM output, which is dependent on the state, is visible; it contains probability distribution of the output tokens. Thus, the sequence of output tokens from HMM gives the details of sequence of states. HMM is known for recognizing patterns in various applications like speech, gestures, handwriting and bioinformatics.

HMM makes assumptions that the emission probability and transition probability depend on data of the current state. This assumption may not be satisfied when probability of staying in the state drops down exponentially. In addition, one cannot assume that data is distributed normally, Gaussian mixture assumption fails.

Number of parameters used in HMM increased when the state of HMM increases. Various techniques like parameter typing are taken to reduce the number of parameters which in-turn increases complexity. It also increases the size of data required to train HMM. HMM only maximize the probabilities of observation but it never minimizes observed probabilities of instances. The number of transitions and states in few domains can be determined only by trial and error or guess.

For real world implementation of this system, many additions need to be appended to the existing algorithm. Many adjustments need to be done to the existing system, which include modifying the state duration model, parameter-typing etc. The concept of HMM is difficult to understand and implement. This method of recognition requires training, training may take more time and few implementations may require recognition with no training.

2.4.2 Latent Dynamic Discriminative models

'Latent Dynamic Conditional Random Field' (LDCRF) [26] is a visual based gesture recognition algorithm. It detects very light motions and visual gestures unlike HMM. LDCRF detects

both intrinsic sub-structure and extrinsic dynamics, thus combining strength of Conditional Random Fields (CRFs) and Hidden-state Conditional Random Fields (HCRFs). Extrinsic dynamics is analyzed from a continuous stream where internal sub-structure is learnt from hidden states.

Several authors propose recognition of visual gestures and head gestures. Eye gaze tracking and headshake detection are done based on Hidden Markov Models and related models. CRF model is used to classify various human gesture activities and it is being used for text segmentation, word recognition. This approach is observed to be better than HMM. CRF can consider overlapping features and can model arbitrary features of data sequence. Models like HCRF are advantageous in vision and speech recognition as they can exploit hidden states [26]. This model is employed for recognition of objects from cluttered sequences and recognition of arms and head gestures from segmented data. However, this model is based on training of pre-segmented data [26]. LDCRF model include hidden states with clear partition. Belief propagation is used during training and testing to compute interference. Support Vector Machine (SVM), is another model, which detects head gestures and eye gaze. Here the input is the head velocity or eye gaze and output is a margin per class. Here it does not consider dynamics between frames, so for training frame based data is considered.

During human gesture recognition, the main goal is to detect a gesture label in frame from a sequence of video frames. In LDCRF, visual sequences which contain extrinsic structure and intrinsic structure i.e. transition between labels and dynamics within class labels are considered. When a set of data is tested with CRF and LDCRF, CRF recognizes only 72% of data when LDCRF gives a perfect result. This model shows a better performance when compared to CRF, HCRF, HMM, SVM [26].

Table 2: Comparing Different Gesture Recognition Techniques

	Training Required	**Record Motions**	**Computational Intensity**	**Eye/Speech Recognition**
HMM [2]	Yes	Yes	Complex	Yes
Latent Dynamic Discriminative Model [26]	Yes	Yes	Complex	Yes
PCA-based Gesture Recognition [24]	Yes	Yes	Simple	No
Active Mobile Interface (AMI)	No	No	Simple	No

CHAPTER 3

AMI MODEL FOR GESTURE RECOGNITION

3.1 Overview

In this chapter, we proposed a model for recognizing user gestures. It includes accepting accelerometer data and segmenting it, filtering data, detecting turning points, determining the valid turning points and filtering channel leakage, which are mapped for presentation on various output devices. Acceleration data is obtained through various devices like a Chronos watch [40], Wiimote [37], etc. Data obtained from the accelerometer are recorded, filtered and processed for gesture recognition. A gesture performed by a user is recognized by applying peak detection algorithm, valid peak detection algorithm. Noise and channel leakage is identified and nullified as well. Here we designed acceleration based gesture recognition techniques to detect heavy gestures performed by a user. Figure 3.1 explains the flow of data in the system.

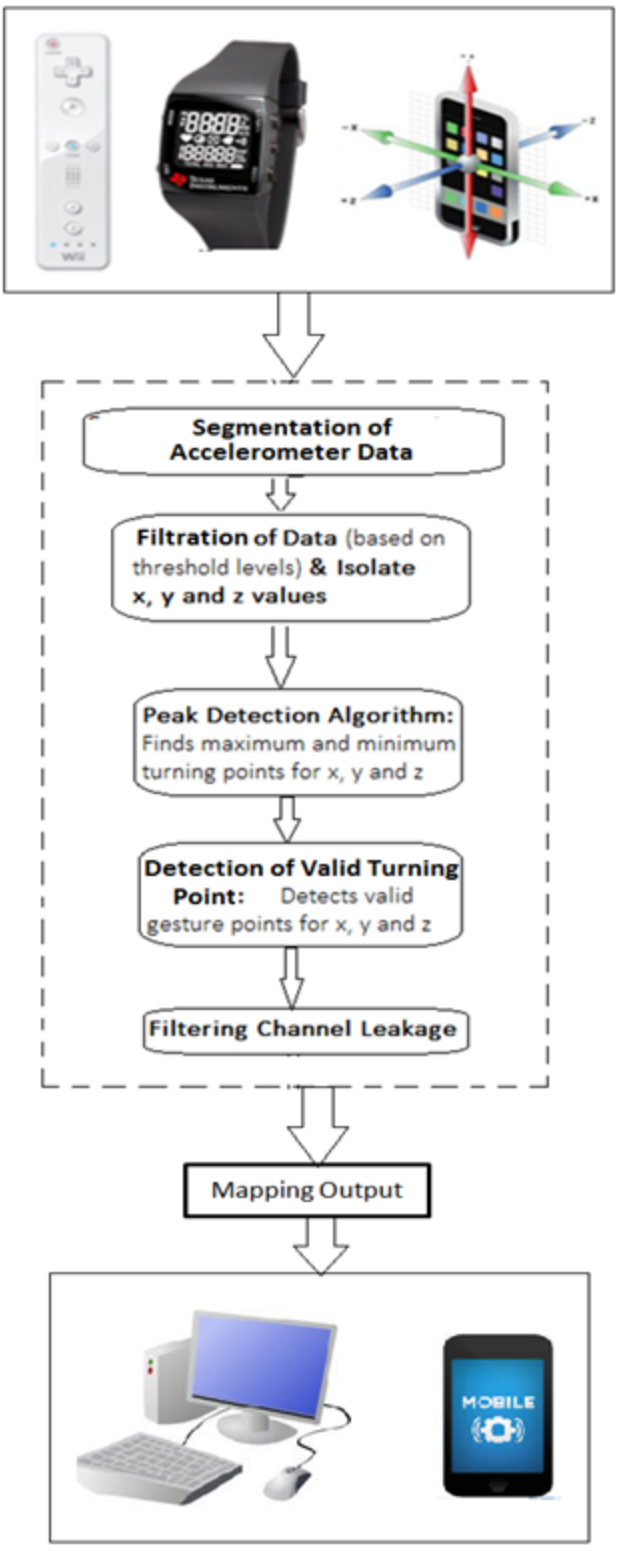


Figure 3.1: Flow Diagram of the Recognition System

Motion Representation

Gestures performed by the user are recorded in the form data obtained from an accelerometer. The accelerometer detects the physical acceleration experienced by the object. Figure 3.2 shows acceleration experienced by an accelerometer in different directions. The accelerometer is attached to either the hand or leg of the user. The data collected contains acceleration of the object in x, y, z directions with respect to gravity. Figure 3.3 shows graphical representation of raw data from an accelerometer. The raw accelerometer data, recorded for a time window, is considered for gesture recognition and then presentation.

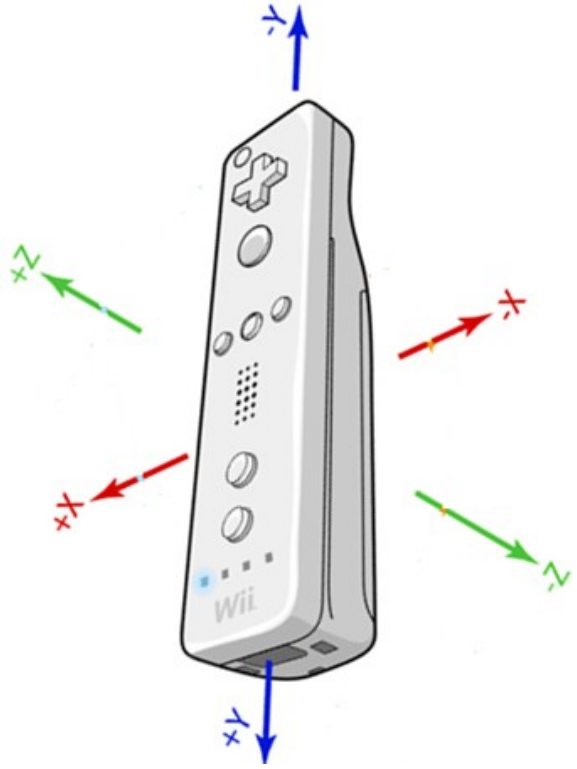


Figure 3.2: Presentation of Acceleration Experienced by an Accelerometer [41]

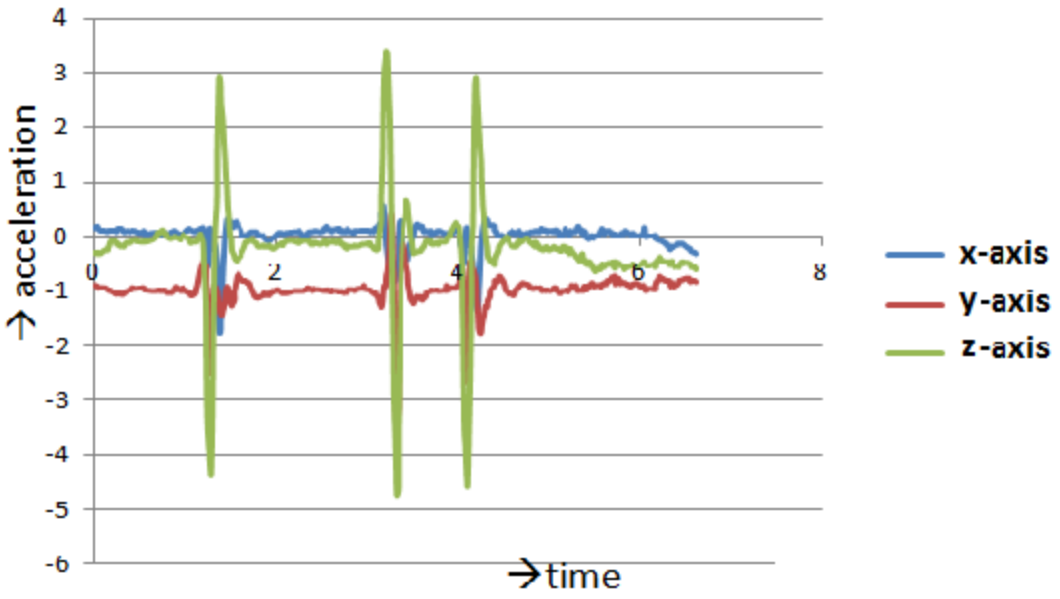


Figure 3.3: Graphical Representation of Raw Data from an Accelerometer

3.2 Motion Patterns

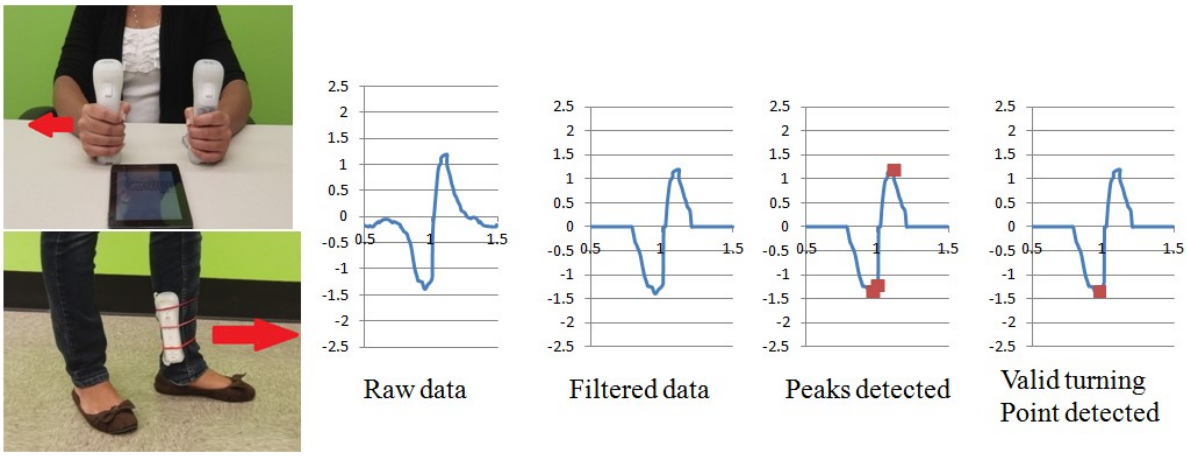


Figure 3.4: Motion Performed by the User

When the user performs a gesture, we get a pair of turning points (TP1, TP2) with opposite sign. The first turning point (TP1) is due to maximum force exerted by the user and the second turning point (TP2) is the deceleration obtained when the user stops. We can see in Figure 3.4, gestures performed by the user and the graphical representation of raw data, filtered data, detected

pair of turning points (TP1, TP2) and determined valid turning point (TP1). So a gesture performed by the user is recognized by determining the first turning point (TP1). If the time taken to perform an action is α seconds, then α/2 seconds is the time difference between the pair of turning points (TP1, TP2).

Determination of Direction/ Magnitude:

Direction of the action performed and the magnitude of the force exerted can be determined from valid turning points detected along the x-axis, y-axis and z-axis. In this section, we are going to discuss about detecting different types of actions.

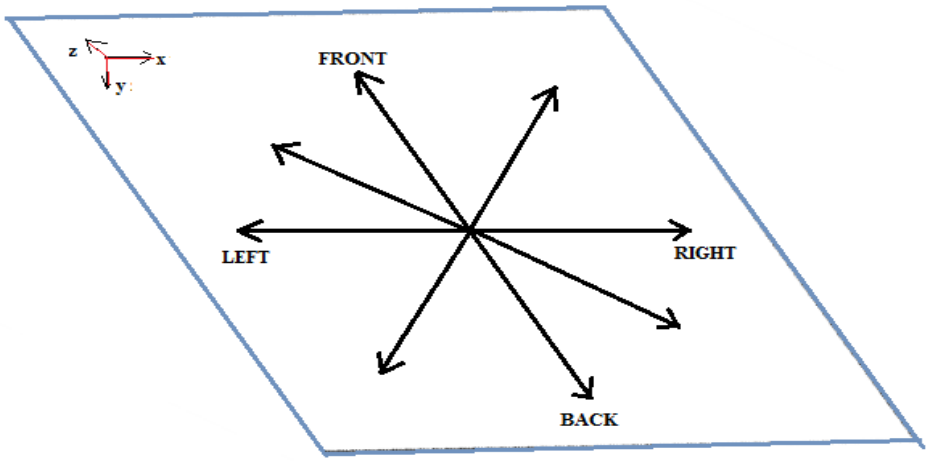
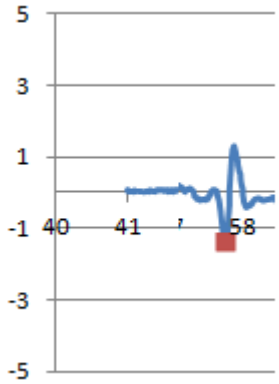
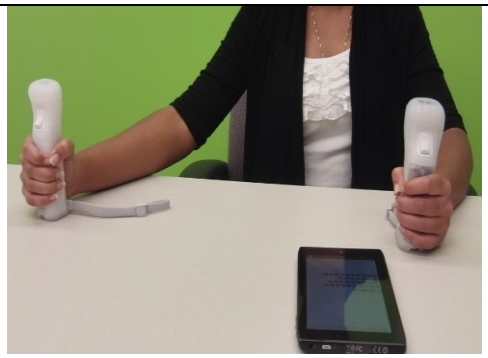
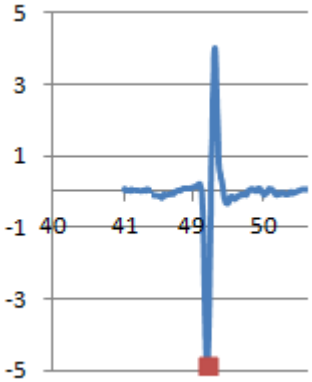


Figure 3.5: Different Gestures Performed by the User

Determination of Magnitude

The absolute value of the valid turning point (TP1) gives the magnitude. The magnitude of the valid turning point is considered as the output value to be presented on mobile or desktop applications. TP1 gives the maximum acceleration exerted by the user while performing an action. Below, Table 3 shows the graphical pattern for different forces exerted by the user.

Table 3: Presentation of Actions with Different Force

Action Performed	Final Position	Graphical Representation
Moving hand toward right, less force is applied.		
Moving hand toward right, more force is applied.		

Gesture towards right/ towards left:

The sign of a valid turning point along the x, y or z direction determines the direction of the gesture. If a gesture is performed by moving an arm from left to right or right to left, variation of the accelerometer is noticed in the x direction. Ideally, no variation should be noticed in y and z directions. However, due to channel leakage we may also notice slight variation in y and z directions, which will be discussed later. In case of action towards right, acceleration values along x-axis initially decrease and then increase. When turning points are determined, we get a negative turning point (TP1) and positive turning point (TP2). Both the peaks occur within the time duration of $\alpha/2$ seconds.

Similarly, when the gesture is performed by moving an arm/leg towards left, we notice variation of acceleration in the x-axis exactly opposite to variation of acceleration in the x-axis of the gesture towards right. The motion of the arm in this case is occurring in the same plane as the previous case but the direction is exactly opposite. Therefore, in both the cases ideally we notice no variation along y and z directions, but a pair of turning points with opposite sign in the x direction. Table 4 shows the gesture from towards right and towards left. In Figure 3.6, Figure 3.7 we can notice a pair of turning points in the x-axis occurring in time duration of $\alpha/2$ seconds.

Table 4: Gestures Performed in x-axis (towards Right and towards Left)

Action Performed	**Initial Position**	**Final Position**
Moving hand toward right, less force is applied.		
Moving hand toward left: cursor moves towards left		

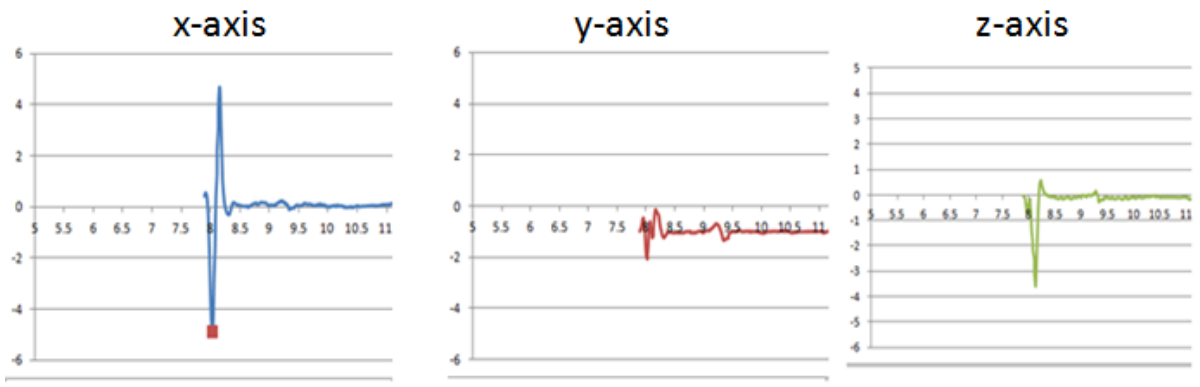


Figure 3.6: Graphical Representation of Gesture towards Right

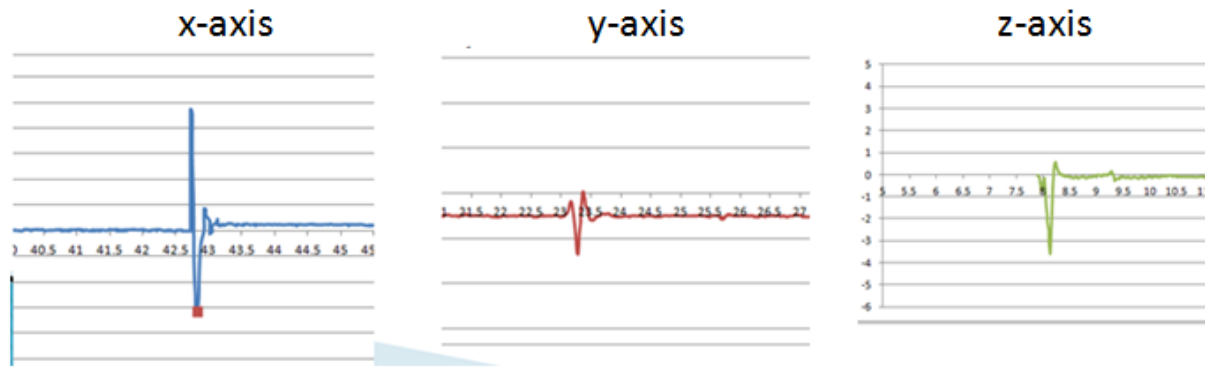


Figure 3.7: Graphical Representation of Gesture towards Left

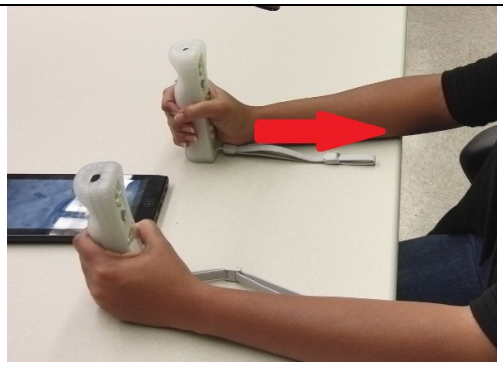
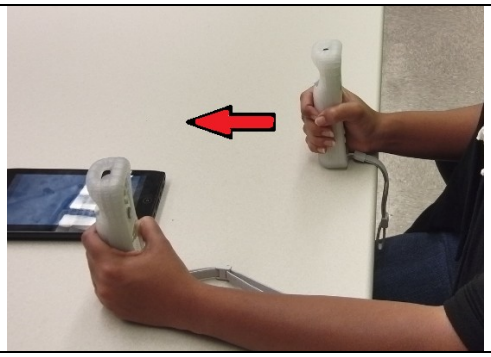
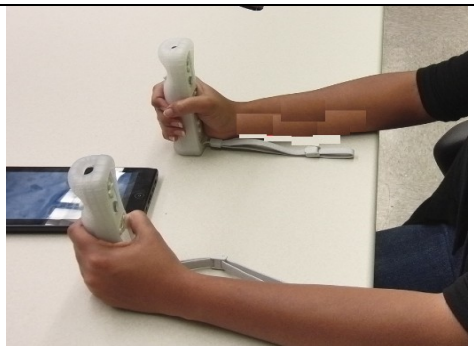
Gesture towards back/ towards front:

When a gesture is performed by moving an arm towards front and towards back, we notice variation in the z direction. Here, the Wiimote [37] is held vertically. If the Wiimote is held horizontally and moved towards front and towards back, variation will be noticed in the y-axis. No variation should be noticed in the x-axis and the y-axis ideally.

If the user performs an action by moving arm/leg towards back, the acceleration along the z-axis initially decreases and then increases. The first turning point (TP1) obtained is a negative turning point. Similarly, when an action is performed by moving arm/leg towards front, the acceleration along the z-axis varies exactly opposite. Here we get a positive turning point (TP1) as a valid turning

point. Difference in the sign of valid turning points determines the direction of actions. Below, Table 5 shows the gestures performed by the user. Figure 3.8, Figure 3.9 shows the variation of acceleration in the x, y and z axes.

Table 5: Gestures Performed in y-axis (towards Back and towards Front)

Action Performed	Initial Position	Final Position
Moving arm towards back		
Moving arm towards front		

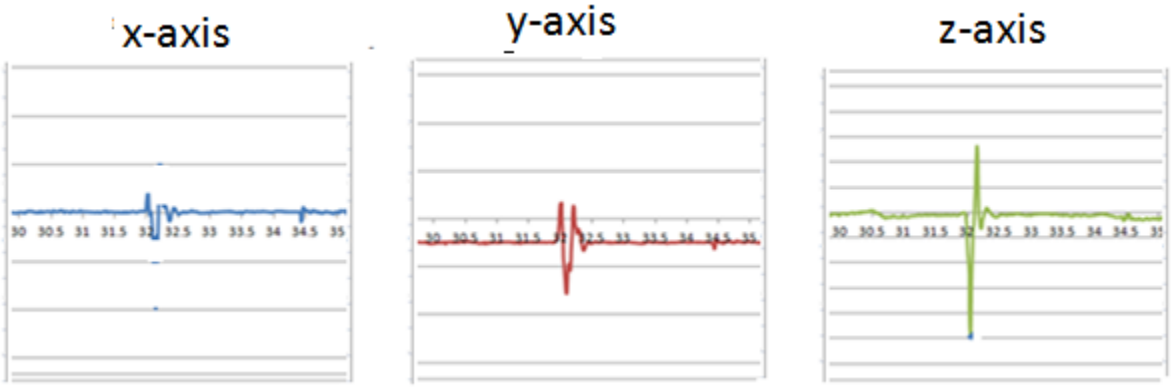


Figure 3.8: Graphical Representation of Gesture towards Front

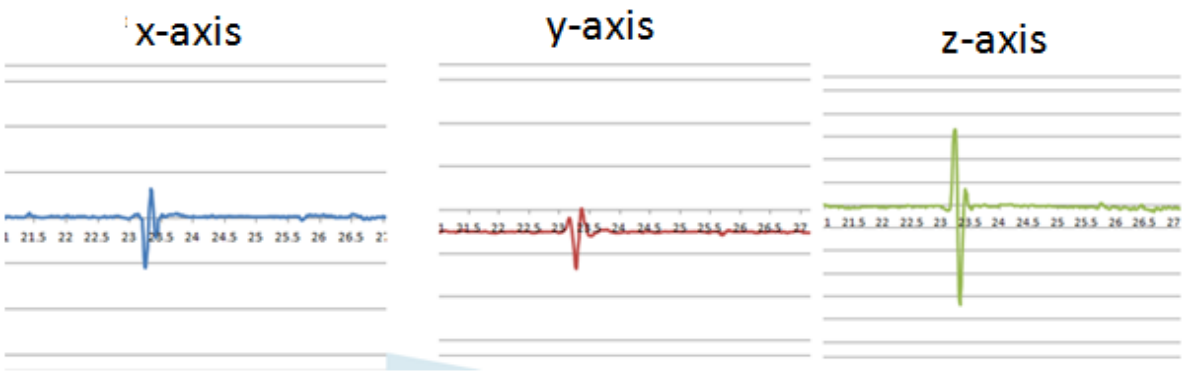
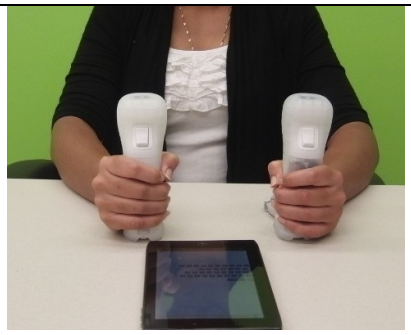


Figure 3.9: Graphical Representation of Gesture towards Back

Moving up and down:

Letters on the screen are selected by moving an arm up and down. We notice the variation of acceleration values in the y direction, since the displacement of the accelerometer is in y plane. Here we should notice no significant change in x and z directions. Since we are using variation in the y plane for selecting letters in the application, this case can be considered as 1,0 situation (i.e., selecting or not selecting a letter). This case is simple when compared to the first two cases i.e. variation in x and z directions as we do not consider the magnitude and just consider the sign of the first turning point. In addition, we set high threshold levels to detect only heavy gestures along the y-axis.

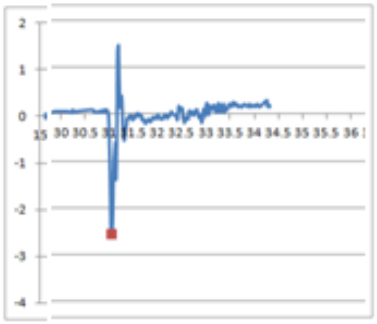
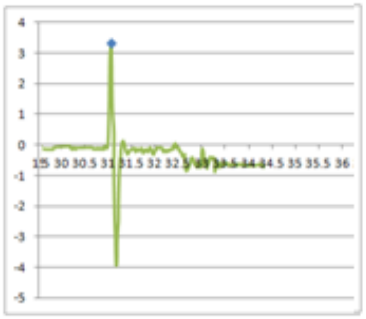
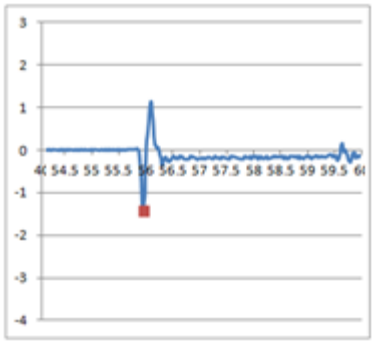
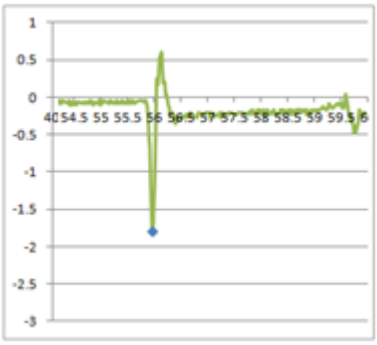
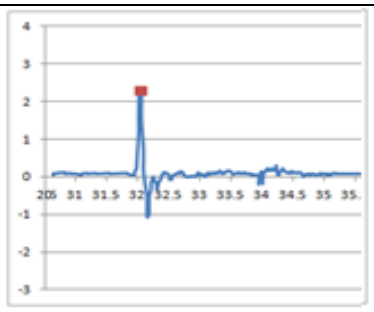
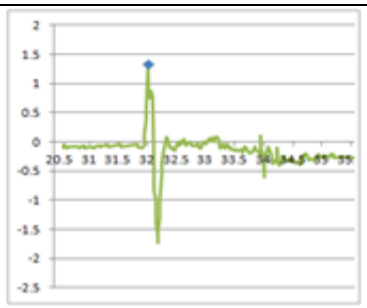
Table 6: Gestures Performed along z-axis (Up and Down)

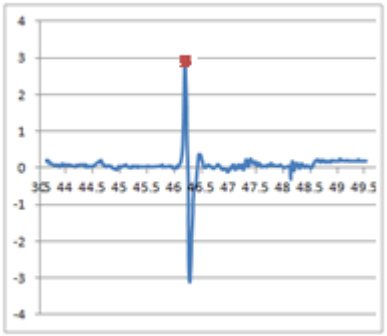
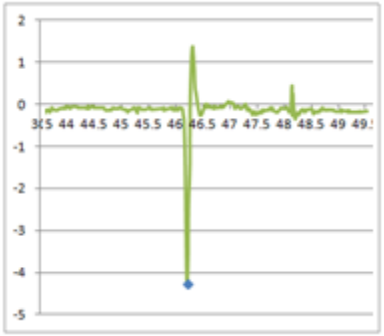
Action Performed	Initial Position	Final Position
Moving hand up and down		

Diagonal Gestures

Active Mobile Interface can recognize diagonal gestures in addition to left/right, front/ back gestures. It determines all four types of diagonal gestures, which are discussed further in this section. Since the gestures towards front, back, right and left affect variations along the x-axis and z-axis, we study only variations along the x-axis and z-axis. Table 7 shows the graphical presentation of diagonal motion.

Table 7: Diagonal Gestures

Action	x-axis	z-axis
Moved Front Towards Right		
Moved Back Towards Right		
Moved Front Towards Left		

Moved Back Towards Left		

Gesture performed by moving arm/leg front towards right:

Table 8 shows the diagonal gesture 'front towards right' is a combination of gestures towards front and towards right. As we see variation along x-axis similar to the variation when gesture is towards right and variation along z-axis is similar to the variation when gesture is towards front.

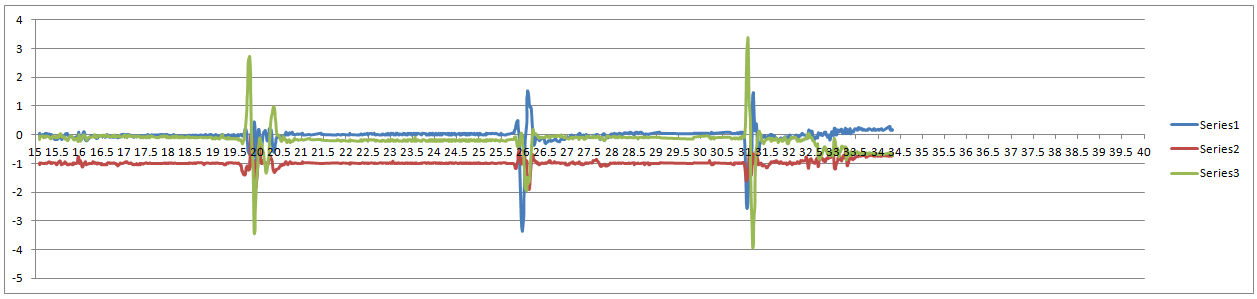
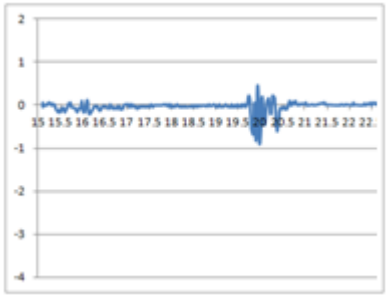
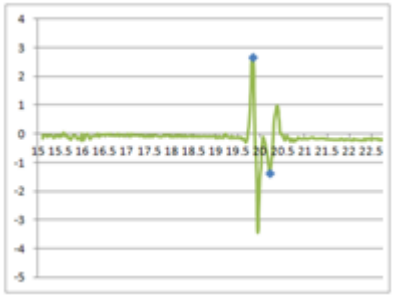
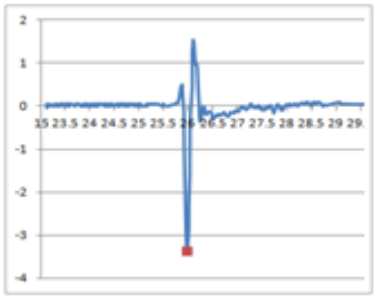
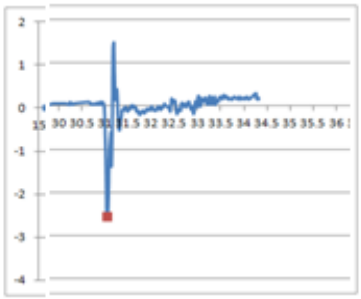
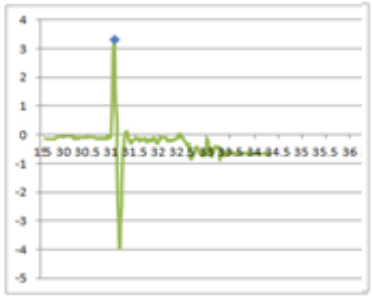


Figure 3.10: Graphical Representation of Gestures towards Front, Right and Front towards Right

Table 8: Gestures towards Front, towards Right and Front towards Right

Action	x-axis	z-axis
Towards Front		
Towards Right		
Moved front towards right		

Gesture moved back towards right:

Table 9 shows the diagonal gesture 'back towards right' is a combination of gestures towards back and towards right. As we see variation along x-axis similar to the variation when gesture is towards right and variation along z-axis is similar to the variation when gesture is towards back.

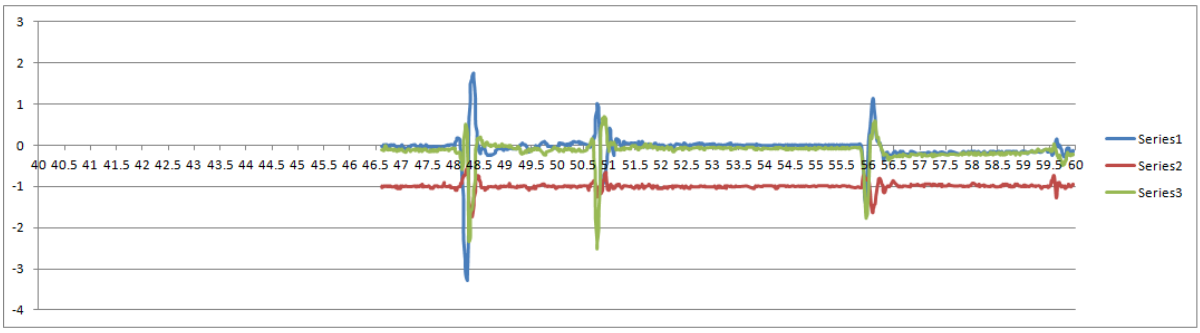
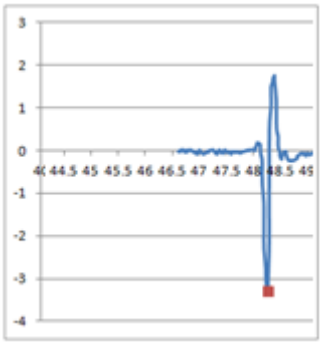
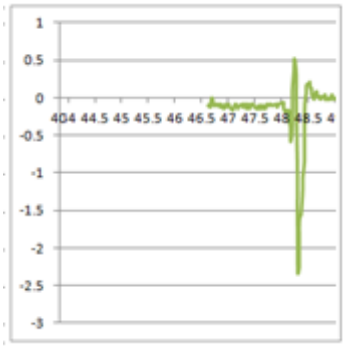
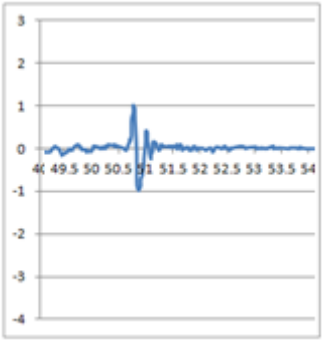
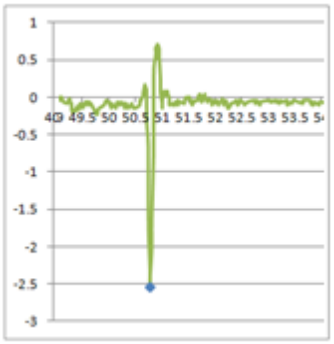
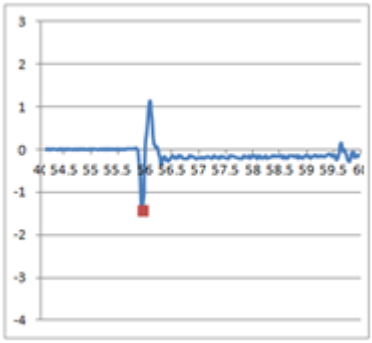
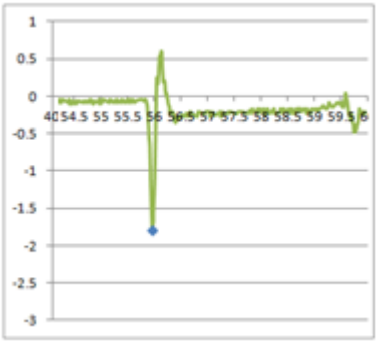


Figure 3.11: Graphical Representation of Gestures towards Right, Back and Back towards Right

Table 9: Diagonal Gestures towards Right, towards Back and Back towards Right

Action	x-axis	z-axis
Towards Right		
Moved Back		
Moved Back towards Right		

Gesture front-towards left:

Table 10 shows that the diagonal gesture 'front towards left' is a combination of gestures towards front and towards left. As we see variation along x-axis similar to the variation when gesture is towards left and variation along z-axis is similar to the variation when gesture is towards front.

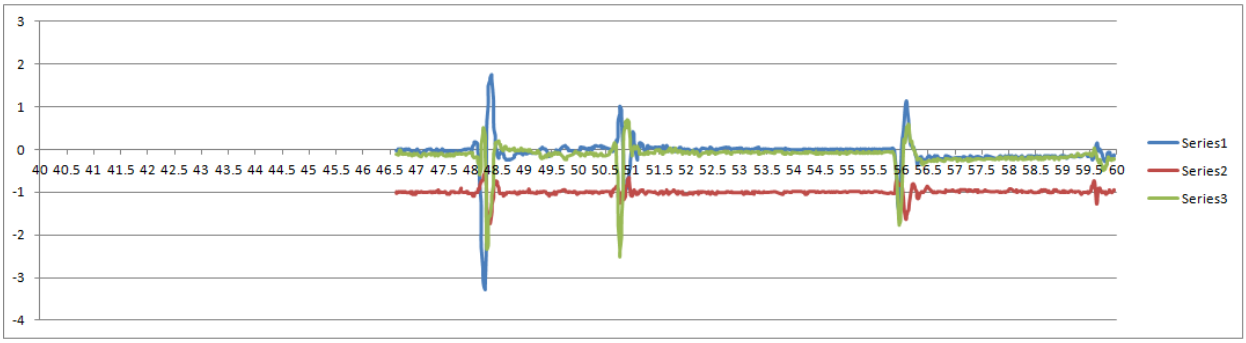
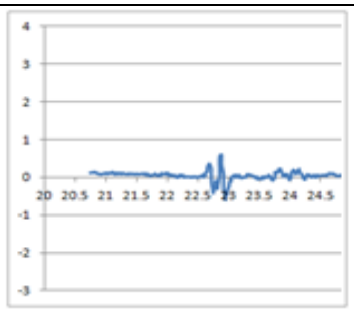
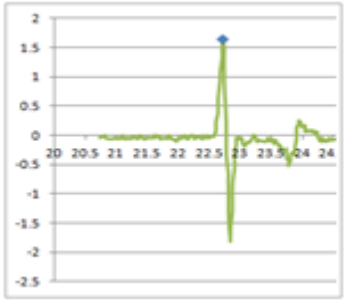
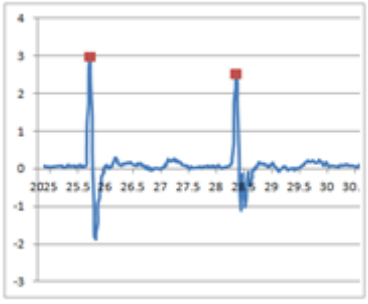
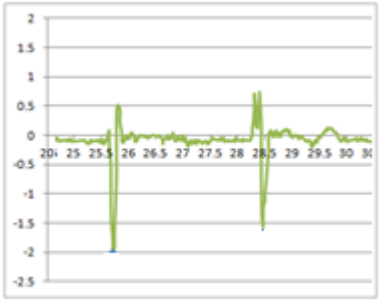
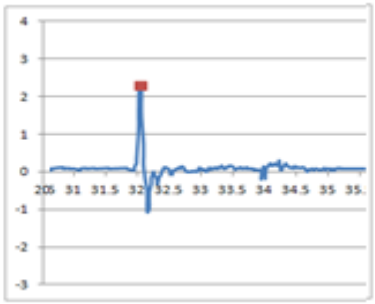
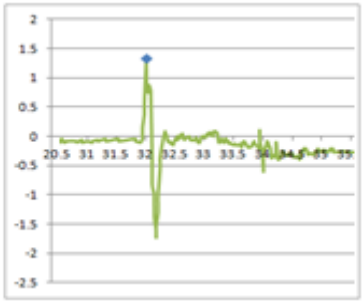


Figure 3.12: Graphical Representation of Gestures towards Front, Left and Front towards Left

Table 10: Diagonal Gestures towards Front, towards Left and Front towards Left

Action	x-axis	z-axis
Moved Front		
Towards Left		

Moved Front towards Left		

Gesture back-towards left:

Table 11 shows that the diagonal gesture, 'back towards left' is a combination of gestures towards back and towards left. As we see variation along x-axis similar to the variation when gesture is towards left and variation along z-axis is similar to the variation when gesture is towards back.

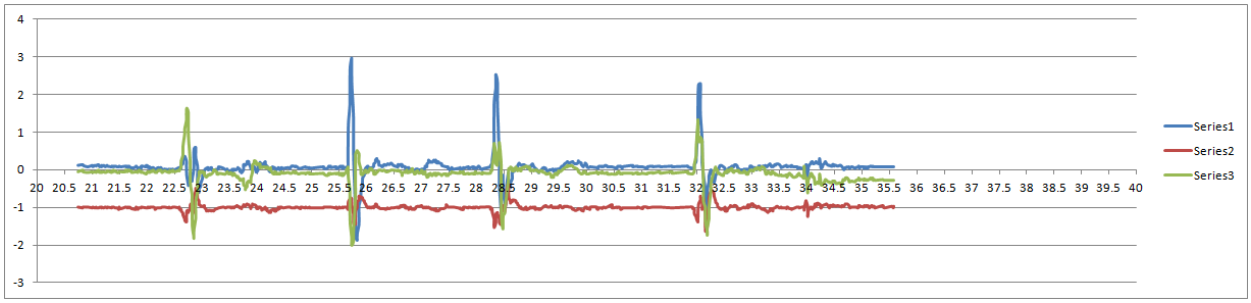
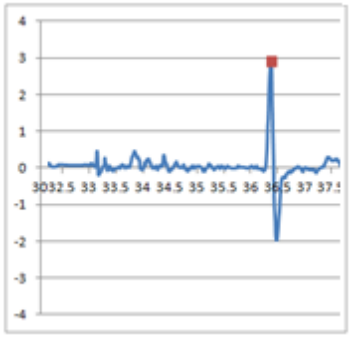
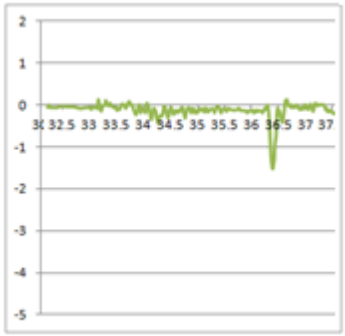
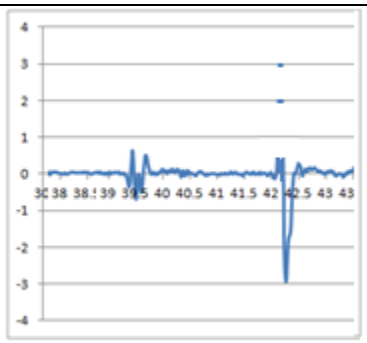
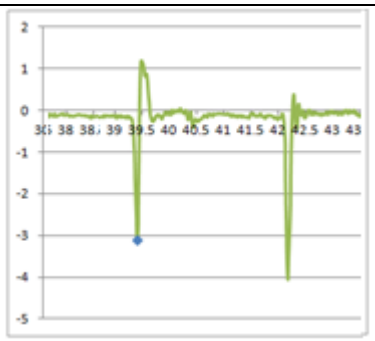
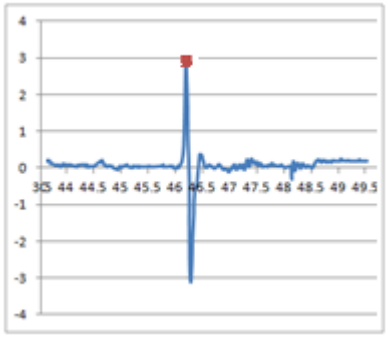
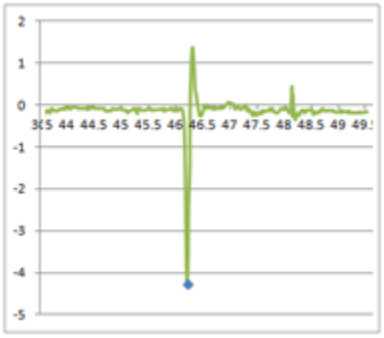


Figure 3.13: Graphical Representation of Gestures towards Left, Back and Back towards Left

Table 11: Diagonal Gestures towards Left, towards Back and Back towards Left

Action	x-axis	z-axis
Towards Left		
Moved Back		
Moved Back towards Left		

3.3 Segmentation of Data

The accelerometer data is collected for a time window of β seconds. The data collected is considered for recognizing gestures, which are performed in the time window. Collected data contains variations of acceleration in x, y, and z directions along time instance. The window size 'β' can be varied to any value. Increasing the time window size 'β' increases the time delay, as the time taken for recording gestures increases. Decreasing the time window size 'β' below 'α' decreases the

accuracy. However, we can resolve it by increasing the overhead, but this increase the processing time. More details about overhead will be discussed in next sections. Therefore, the best value of the time window is the maximum value of 'α', where 'α' is the maximum time taken to perform a gesture. So considering delay and accuracy, time window size 'β' is fixed to 'α' seconds.

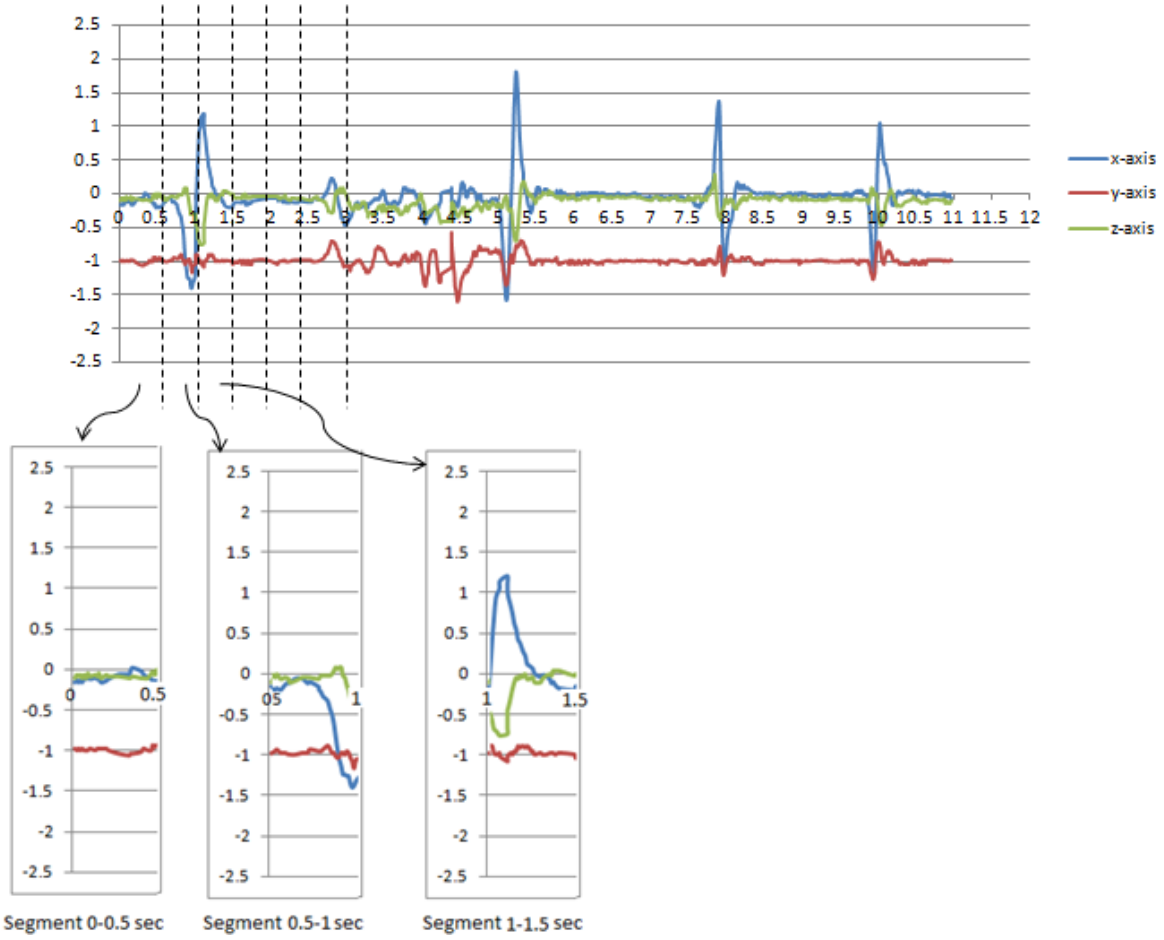


Figure 3.14: Segmentation of Data

In Figure 3.14, the first graph represents the accelerometer data and the second graph represents data segments. Accelerometer data is segmented for 'β' i.e. data is recorded for 'β' seconds. Each segment is processed individually for gesture recognition.

Algorithm:

```
Segment_acceleromterData(type Raw_accelerometer_data)
{
        Window_size=0.5seconds;
        If(current_time<=time_data_read+0.5)
                Wait(time_data_read+0.5- current_time)
        Raw_accelerometer_data_window=Read_data(Raw_acceleromer_data, Window_size);
        Data_set[]=Split_raw_data(Raw_accelerometer_data_window);
        Time_data_read=current_time;
}
```

3.4 Filtration of Data

Collected segments of data are processed for gesture recognition. Data is isolated and filtered to remove noise and reduce computational intensity. First acceleration values along x, y and z directions are isolated. Later isolated values along x, y and z are filtered based on threshold levels. Threshold values are dependent on the static value 'φ' of the accelerometer and error state 'Δ'. Static value 'φ' is obtained when the accelerometer is in rest position. The accelerometer in a Wiimote that is used for gesture recognition is very sensitive and it captures very small actions when compared to the accelerometer in a Chronos watch [40]. However, the intention of this project is to perform heavy gestures. So, by setting higher threshold levels, only heavy motions are captured, ignoring light gestures.

Threshold level contains maximum and minimum value. The coordinates above maximum value and below minimum value are considered for the next step of recognition, while the rest of the values are approximated to the static value. After isolating and filtering x, y and z values based on the threshold level, we have three sets of coordinates, which are considered individually for next steps. Each set (i.e. x, y and z) are processed simultaneously to reduce the delay time and maintain synchronization. Figure 3.16 show the processing a segment of data. Figure 3.18 and Figure 3.19 shows the isolated and filtered data collected for 10 seconds respectively.

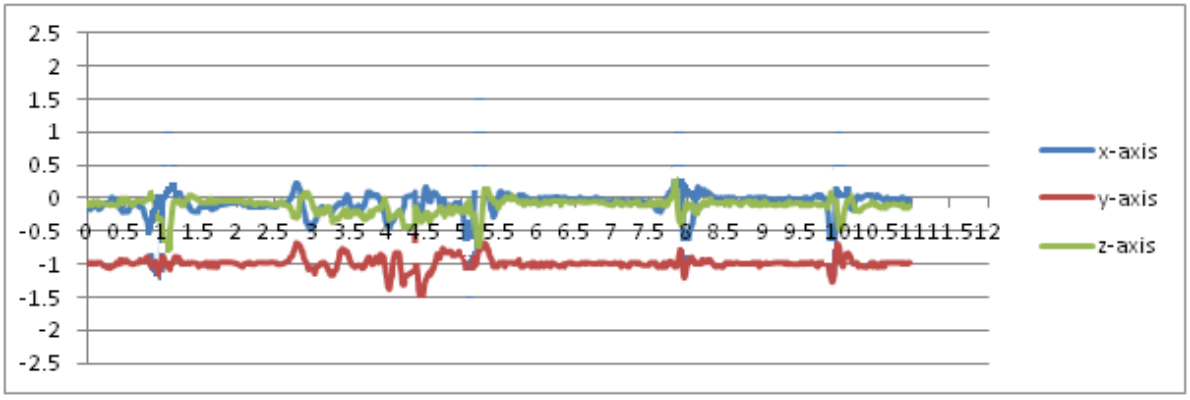


Figure 3.15: Data from the Accelerometer in Static Position

Figure 3.16: Isolating and Filtering Segment of Data

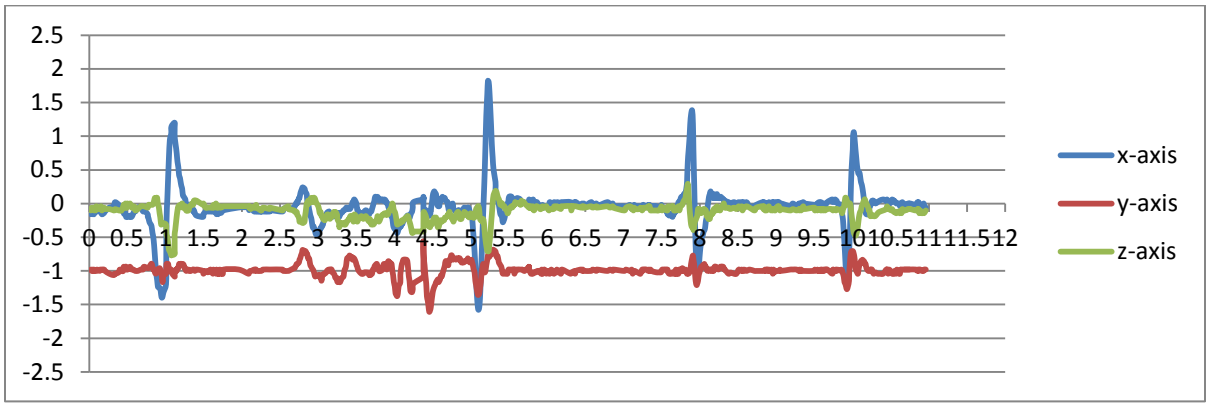


Figure 3.17: Graphical Representation of Raw Accelerometer Data

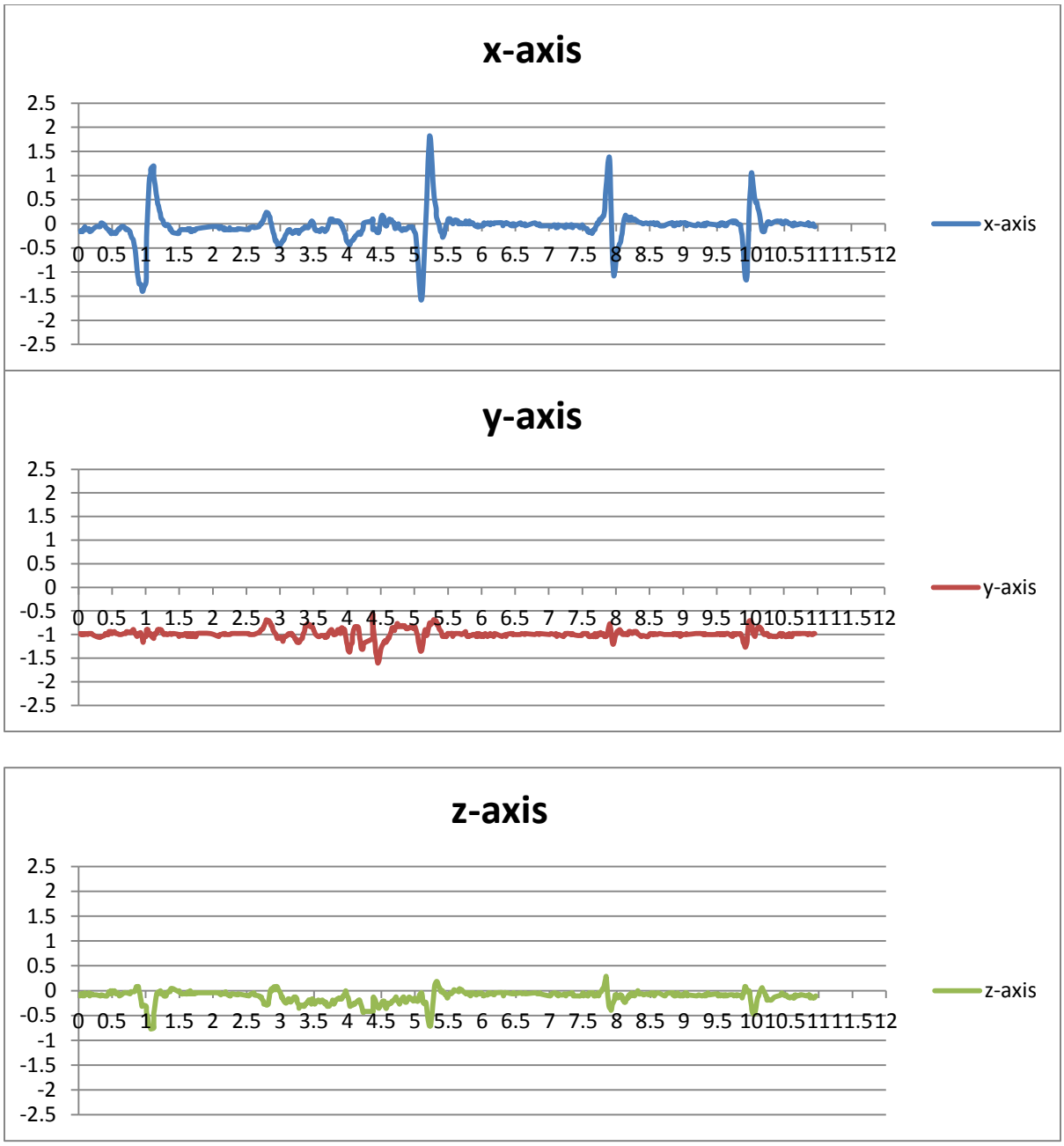


Figure 3.18: Graphical Representation of Data after Isolation

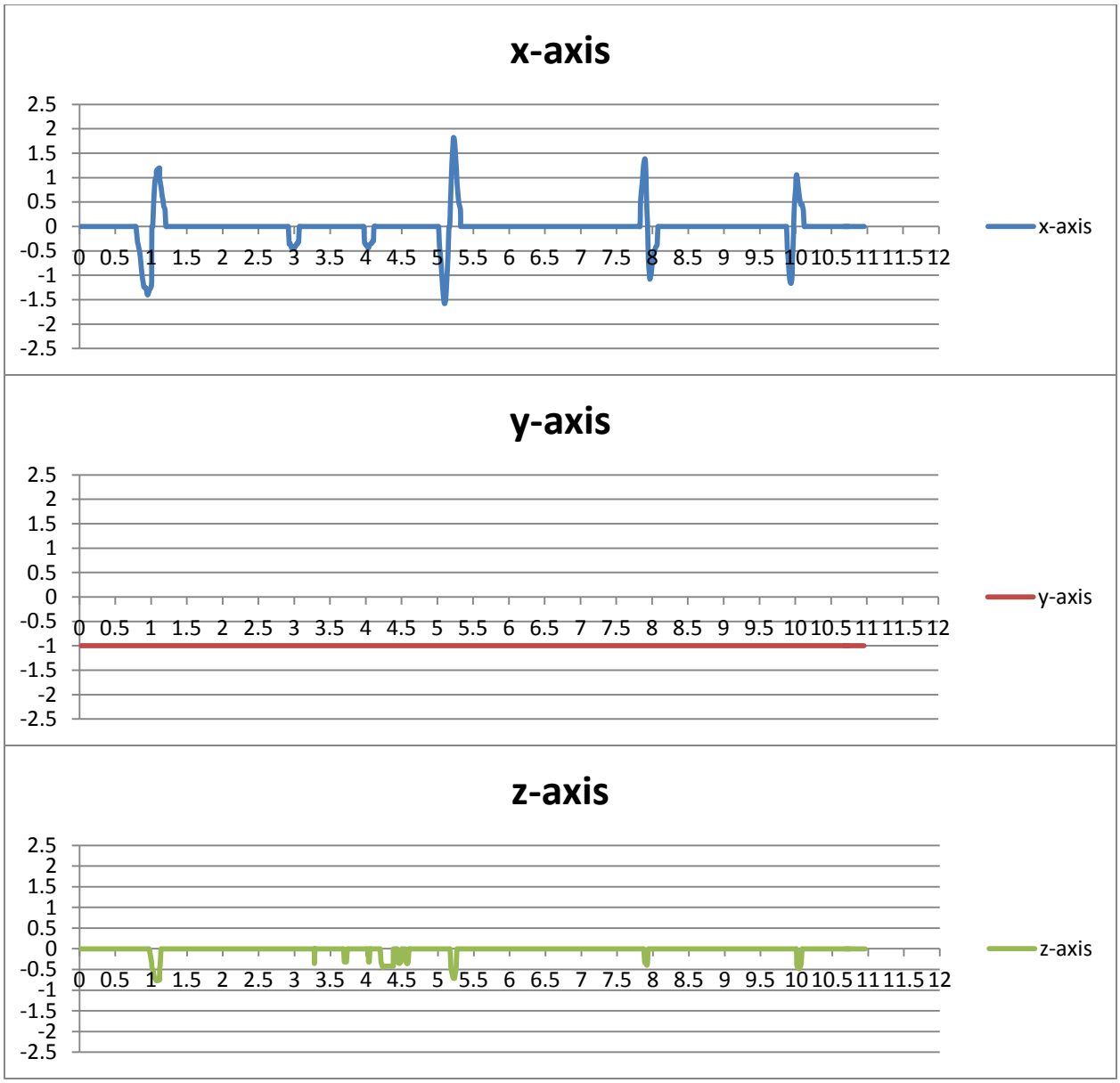


Figure 3.19: Graphical Representation of Data after Filtration

Threshold level = φ ± Δ

Minimum value= φ- Δ

Maximum value=φ+ Δ

where φ is a static value,

Δ is an error rate

From Figure 3.15, we say that the static value 'φ' of the accelerometer is '0' in x-axis, '-1' in y-axis and '0' in z-axis. The best value of the error state 'Δ' is determined from the f-measure in chapter 5.

Algorithm:

```
Filter_accelerometerData(type isolated_accelerometerData)
{
        Minimum_value=Static_value- Δ;
        Maximum_value=Static_value+ Δ;
        While((Line_read=isolated_acceleromterData.readLine)!=null)
        {
                If(Minimum_value < Line_read < Maximum_value)
                {
                        Line_read=Static_value;
                        Line_read=isolated_acceleromterData.readLine;
                }
                Filtered_acceleromterData.writeLine(Line_read);
        }
}
```

In order to improve performance and decrease delay of the system, processing of raw acceleration coordinates is skipped based on certain parameters. Processing of acceleration values in this step includes isolating and approximating to static value based on threshold levels. If the co-ordinate value is within threshold level (i.e. if the value is between range of maximum and minimum value), the next value read by the accelerometer is skipped and the alternate value is considered for recognition. This is done to improve the performance of the system in terms of speed.

3.5 Peak Detection Algorithm

Peak detection algorithm (PDA) is designed to detect maximum and minimum turning points occurring in the filtered accelerometer data. Output of PDA is considered for valid turning point

detection. Peak is also called turning point. Turning point detected is dependent on the previous value and the next value.

Here x-axis, y-axis and z-axis are considering individually in determining turning points. Output of the Peak Detection Algorithm returns three sets of turning points along x, y and z directions. Turning points detected can be seen in Figure 3.20 in red. Decision of the maximum and minimum peaks is dependent on other values occurring in the segment of window size 'β' seconds. Maximum turning point occurs above the static value and minimum turning point occurs below the static value.

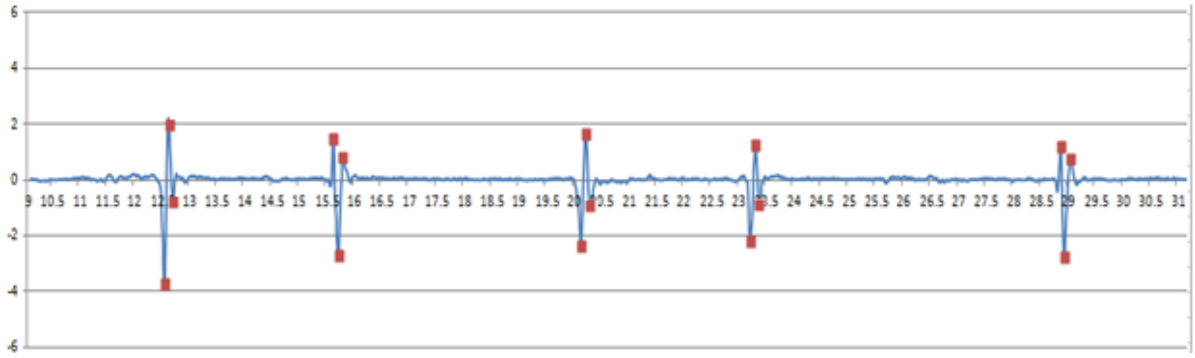


Figure 3.20: Acceleration Data with Turning Points Detected

Determination of Turning points:

Refer Figure 3.21 to understand determination of turning points. To find a maximum peak (p5), the current value (p1) is taken as the maximum value and it is compared with the next occurring value (p2). If the next occurring value (p2) is greater than the maximum value (p1), then it (p2) is taken as the maximum value and considered for the next comparison. This process of comparison is repeated until the next occurring value (p6) is less than the maximum value (p5). If the next occurring value (p6) is less than the maximum value (p5), then the maximum value (p5) is considered a turning point. The process of comparison is repeated again until the next occurring

value is less than or equal to the static value. Once the value is less than or equal to the static value, the detected turning point is returned.

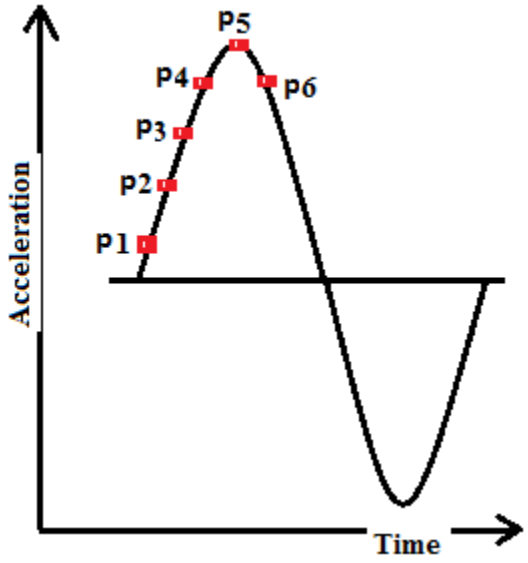


Figure 3.21: Detecting Turning Point

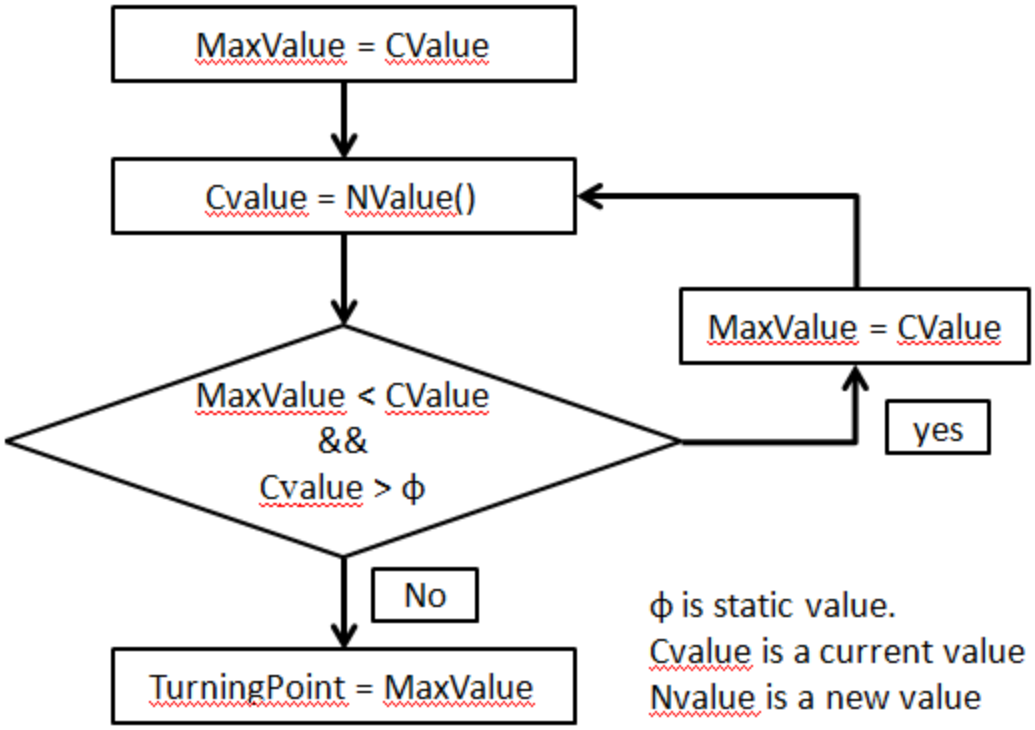


Figure 3.22: Flow Chat of Peak Detection Algorithm

A similar process is repeated to determine minimum turning points. Here the process of comparison is repeated until the next occurring value is greater than the minimum value. If the next occurring value is greater than the minimum value, then it is considered a turning point. Once the next occurring value is equal to or greater than the static value, the detected turning point is returned.

Algorithm:

```
Detecting_Turning_Points(String Filtered_Segmented_Data)
{
   CurrentLine=Filtered_Segmented_data.readLine;
   Maximum_value=Static_value_x;
   Manimum_value=Static_value_x;
   While(CurrentLine!=null)
  {
    //Detecting Maximum Turning Points
```

```
  if(currentLine_x>Maximum_value)
  {
    While(CurrentLine_x>=StaticValue)
    {
    if(currentLine_x>Maximum_value)//Comparing previous value with current value
      Maximum_value=currentLine;//If is current value is the maximum value
    else
    {
      Write_TurningPoint(CurrentLine); //Maximum turning point is obtained and recorded
      Maximum_value=Static_value_x;
    }
    currentLine=Filtered_Segemented_data.readNextLine;
    }
  }
  //Detecting Minimum Turning Points
  if(currentLine_x<Minimum_value)
  {
    While(CurrentLine_x<=StaticValue)
    {
      if(currentLine_x<Minimum__value)//Comparing previous value with current value
      {
        Minimum_value=currentLine;//If is current value is the min value
      }
      else
      {
        Write_TurningPoint(CurrentLine);//Minimum turning point is obtained and recorded
        Minimum_value=Static_value_x;
      }
    currentLine=Filtered_Segemented_data.readLine;
    }
  }
}
```

Peak Detection Algorithm: Challenges

Case1:

When processing a segment of data, we may get the last value in the segment as a turning point. This occurs due to segmentation of data. It is handled by returning the maximum value obtained as the turning point, when the processing of a segment is completed.

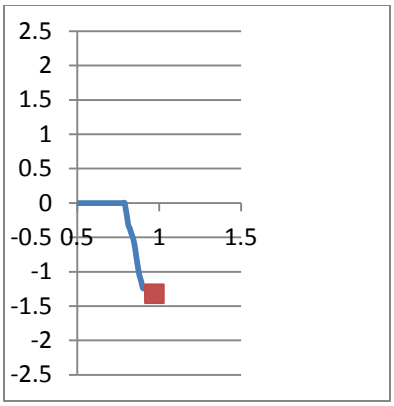


Figure 3.23: Last Value in the Segment is Turning Point

Case2:

When processing a segment of data, we may get the first value in the segment as the turning point. This case is occurs due to segmentation of data. It is handled by comparing the current value with the next value and returning it if the next value is less than the current value and the current value is the first value in the segment.

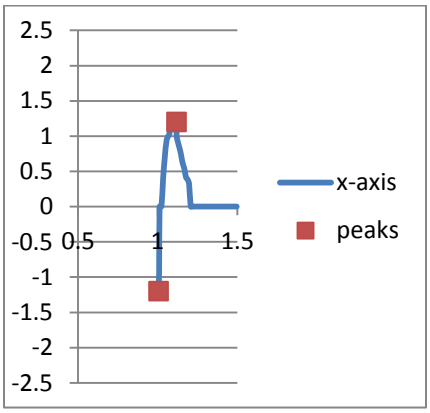


Figure 3.24: First Value in the Segment is Turning Point

Case 3:

When the user performs two actions with ignorable pause or slight jerk, we notice variation in acceleration as seen in Figure 3.25. Here we get two turning points but only one turning point is the actual turning point. This case is handled by comparing the detected turning point with previous

turning point obtained, if the both the turning points are above/ below the static level and both of them are within a time duration of 'α' seconds.

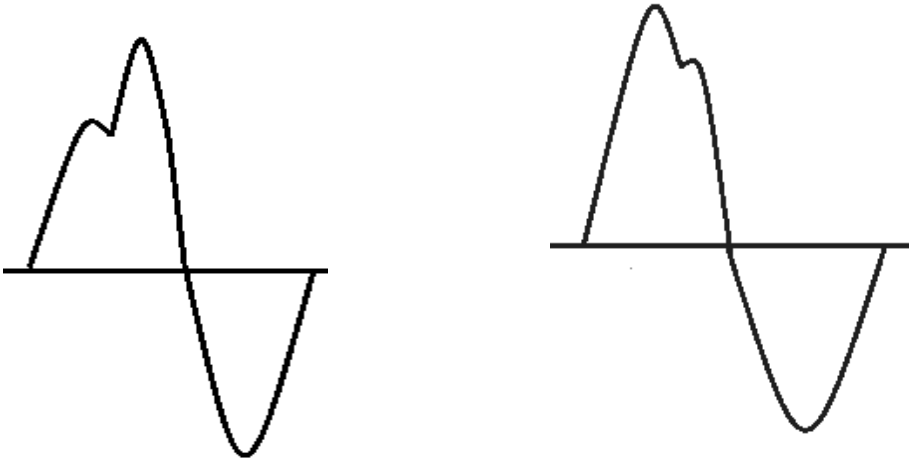


Figure 3.25: Variation in Acceleration due to Ignorable Pause

3.6 Detection of Valid Turning Points

In this step, we detect valid turning points from the set of peaks determined. A valid turning point determined represents the direction and force of the gesture performed by the user. Peak Detection Algorithm (PDA) returns a set of maximum and minimum peaks for the given time window of 'β'seconds. Valid turning points occur when the user changes position and orientation of the accelerometer voluntarily.

When the user performs a gesture voluntarily, the following graphical patterns are seen in Figure 3.26 is noticed. A valid gesture contains a pair of turning points with the opposite sign. Two turning points occur in a time stamp of half the time taken for performing a gesture. If 'α' is the time taken for a gesture, then the pair of turning points occur in a time duration of 'α/2' seconds.

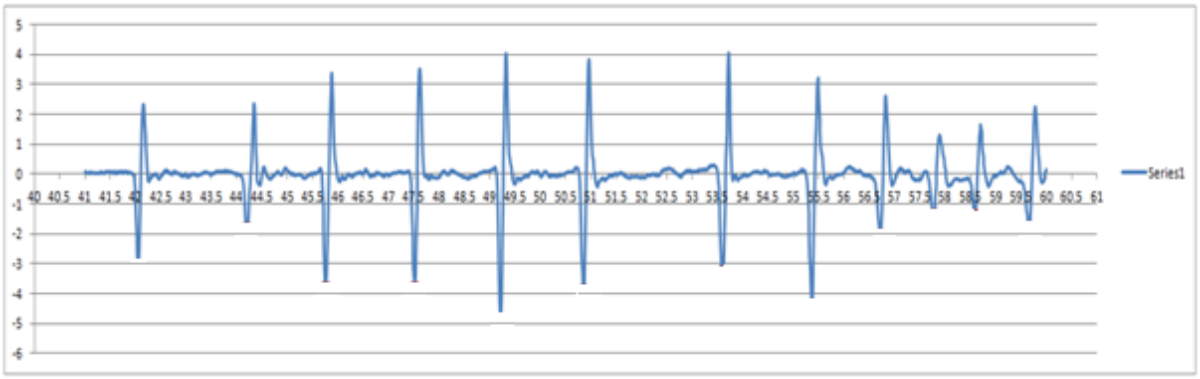


Figure 3.26: Graphical Representation of Accelerometer Data

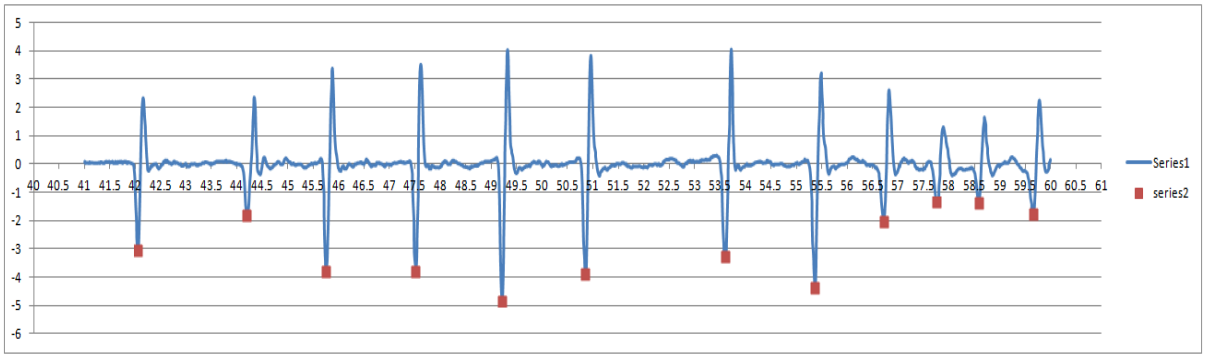


Figure 3.27: Graphical Representation of the Data with Valid Turning Points Detected

Turning points with opposite signs occurring in the time stamp 'α/2' are considered for gesture recognition. The first occurring turning point is due to the action performed by the user voluntarily. The second occurring turning point is the reaction as the user suddenly stops the accelerometer. So the first turning point is considered and the second turning point is ignored. The magnitude of two turning points peaks is approximately equal. The direction of the action is based on the sign of the first turning point obtained. Three sets of valid turning points are determined along all three x-axis, y-axis and z-axis individually.

Determining the Validity of Turning Point:

Here is the process to determine valid turning points. The time of occurrence and the sign of the turning points are considered for determining their validity. The validity of a turning point is

dependent on its time of occurrence and the time of occurrence of the next turning point. If the first turning point and the second turning point occur within a period of 'α/2' seconds and if both the turning points are of opposite signs, then the first turning point is considered valid. Similarly, to determine validity of the second turning point, time of occurrence and sign of the third turning point is considered.

Algorithm:

```
Determining_valid_turning_points(type Detected_turning_points)
{
        current_line_read= Detected_turning_points.readLine();
        next_line_read= Detected_turning_points.readLine();
        do{
                If(0.03 <= (timeOf_next_line_read - timeOf_line_read) <= 0.20)
                {
                        If(sign(current_line_read)== - (sign(next_line_read)))
                                Valid_turning_point.writeLine(current_Line_read);
                }
        Current_line_read=next_line_read;
        While((next_line_read=Detected_turning_points.readLine())!=null);
        Valid_turning_point.writeLine(current_Line_read);
}
```

Exceptional Cases

Case 1:

Several exceptional cases need to be considered while detecting valid gesture points. In this case, a gesture may contain three turning points. Thus, we may get three turning points in time duration of 'α' seconds. Both the first turning point and the second turning point will be detected as valid turning points, which is undesirable as only first turning point is valid. This case occurs due to human error (e.g. a slight undesirable jerk is noticed when the motion of hand/leg is towards left). This error does not occur if the motion is towards right.

The error is handled by checking for the third turning point along with first two turning points within the time duration of 'α' seconds. The third turning point and second turning point are of opposite signs. If we get three turning points in the mentioned time duration, then the first turning point is considered a valid turning point and other two points are ignored. Below, Figure 3.28 shows the graph with an error and Figure 3.29 shows the graph after the error is rectified.

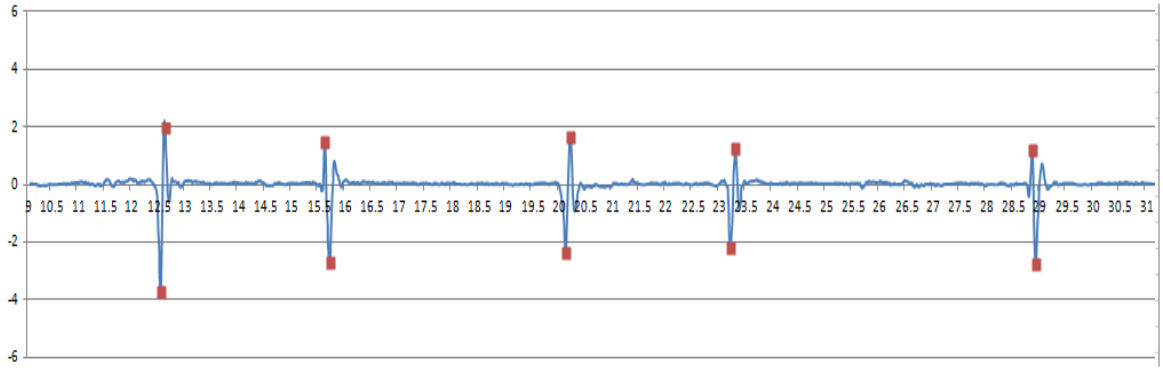


Figure 3.28: Graphical Representation of the Data along x-axis with Undesired Turning Points

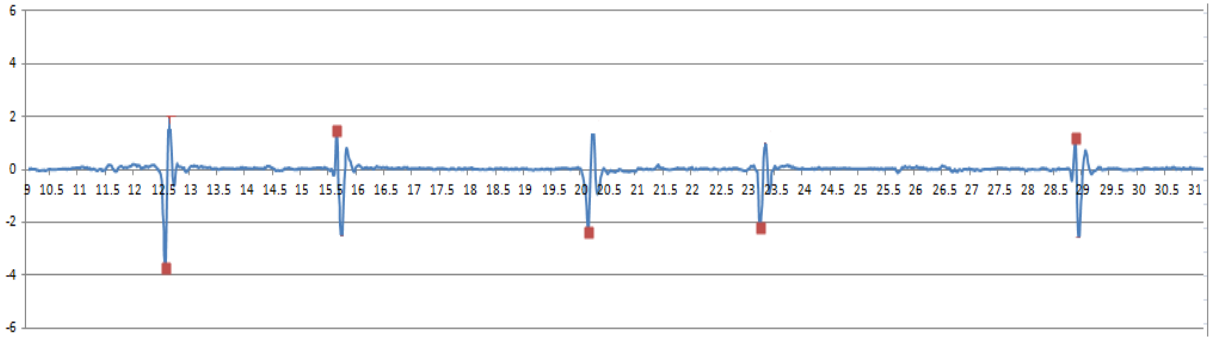


Figure 3.29: Representation of the Data along x-axis with Undesired Turning Points Filtered

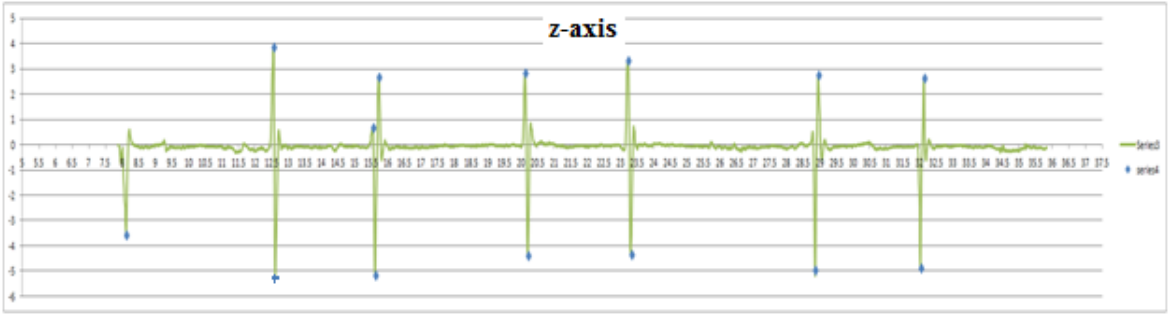


Figure 3.30: Graphical Representation of Data along z-axis with Undesired Turning Points

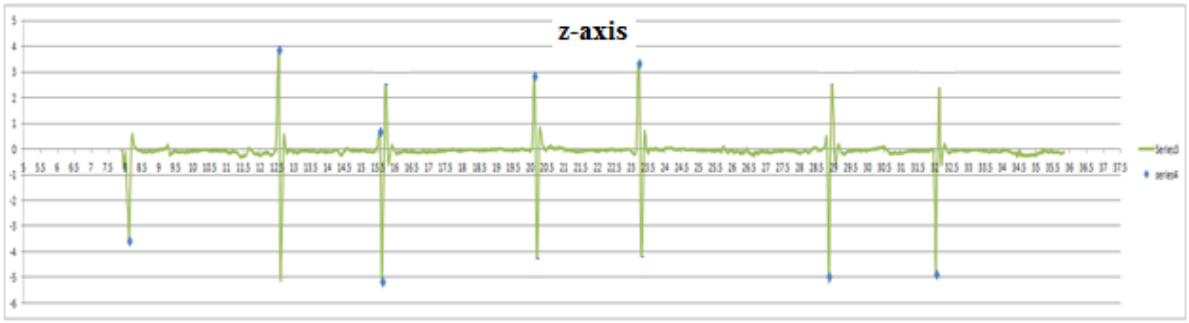


Figure 3.31: Representation of Data along z-axis with Undesired Turning Points Filtered

Case 2:

The Second case occurs due to the segmentation of raw data read from the accelerometer. Data is segmented based on the time window 'β' seconds, and then turning points are detected for a single time window. Consider the case shown in Figure 3.32. The first two turning points are above static level and the third turning point is below static level. Since each time window is processed individually for the gesture recognition, the first time window t1 returns peak1. By processing the second time window t2, we get a turning point above static level (i.e. peak2) and another turning point below static level (i.e. peak3). Time window t1 returns no valid turning point. Time window t2 compares peak2 and peak3 and returns peak2 as a valid gesture point. This is undesirable as peak1 is a valid gesture point.

This error is handled by adding an overhead. The overhead contains the last detected turning point from the previous time window. So when processing time window t2, it considers peak1 from overhead along with peak2 and peak3 for determining the valid turning point. Thus, we get peak1 after rectifying the error. Figure 3.33 is another scenario where the error is rectified by adding the overhead.

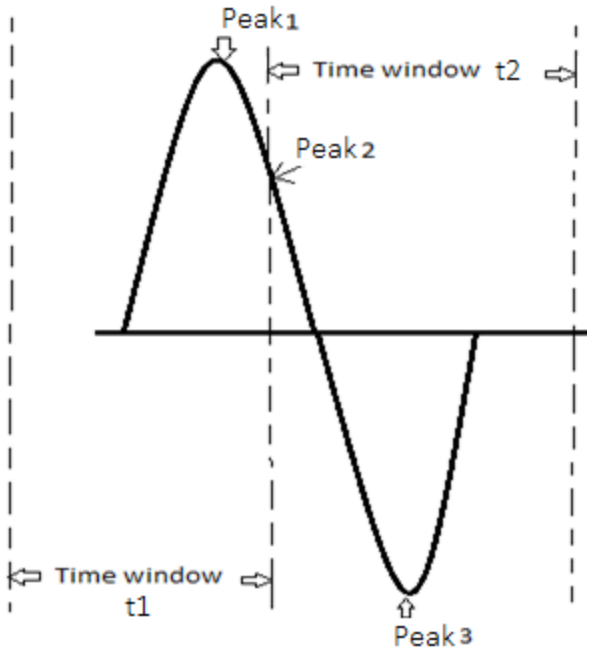


Figure 3.32: Graphical Representation of Turning Points based on Time Window - 1

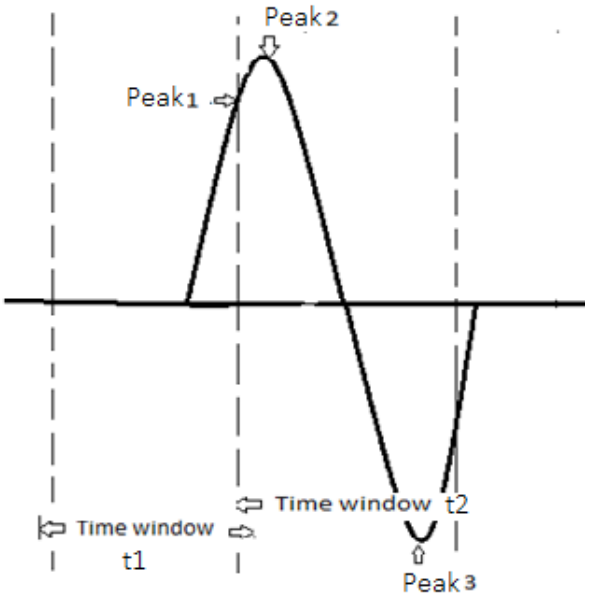


Figure 3.33: Graphical Representation of Turning Points based on Time Window - 2

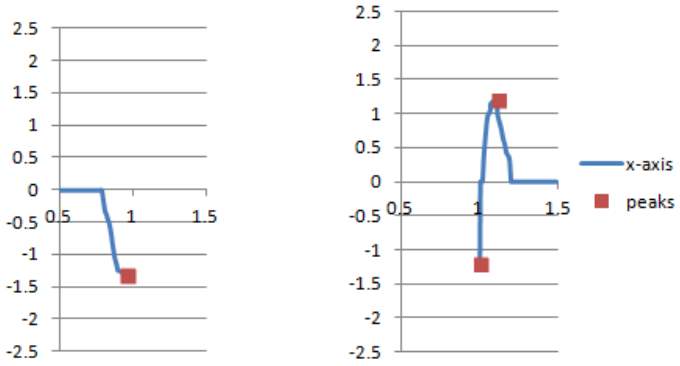


Figure 3.34: Processing Individual Segments

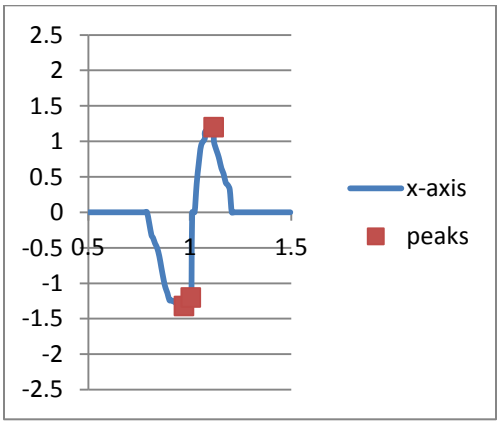


Figure 3.35: Data Segments after Adding Overhead

Case 3:

Figure 3.36 shows the graphical representation of accelerometer data along the z-axis. Blue points in the graph represent the detected valid turning points. In Figure 3.36, we can see multiple undesirable points detected (marked in blue). By considering the first exceptional case, checking for the third turning point, we filtered few undesirable turning points. Output is shown in Figure 3.37.

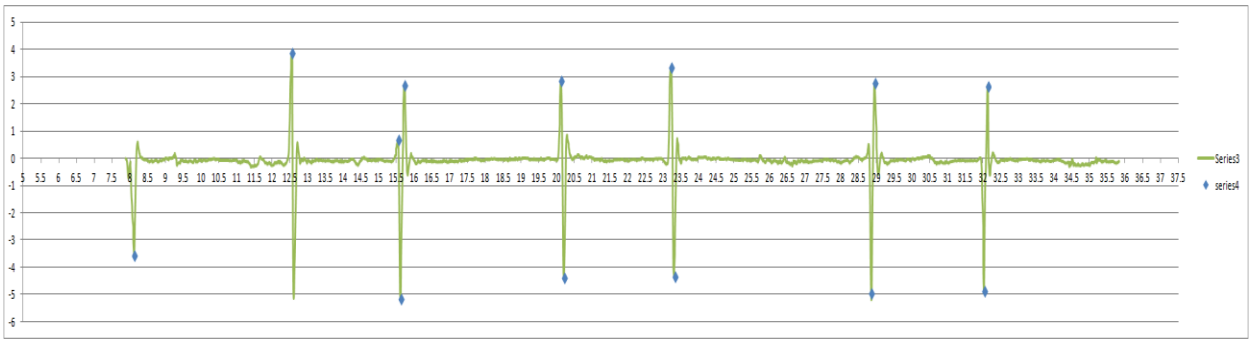


Figure 3.36: Graphical Representation of Data with Undesirable Turning Points Detected

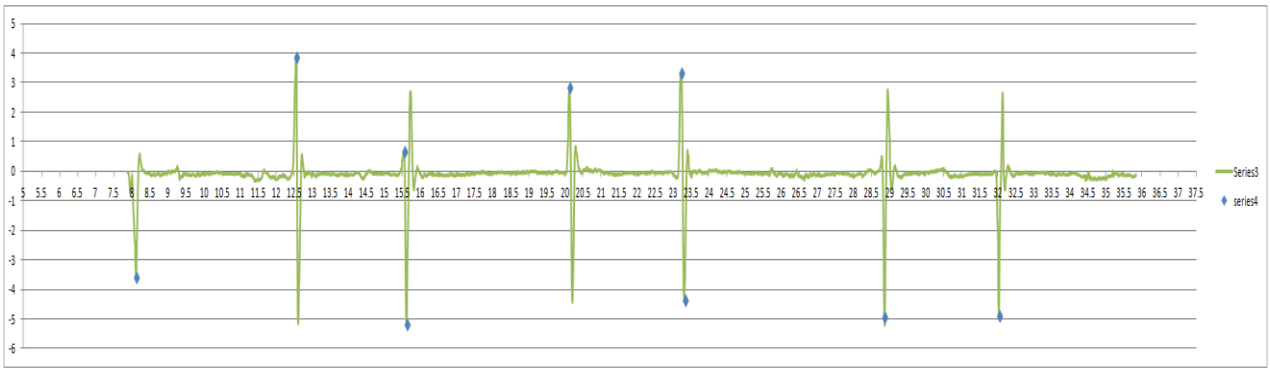


Figure 3.37: Data after Removing Undesirable Turning Points by Checking Exception Case 1

After checking for the first exceptional case 1, we notice undesirable points at time stamp 15 and 23 seconds. The case of the undesirable turning point occurring at the time stamp 23 is due to the exception case 2, by adding overhead and validating turning points based on magnitude it can be resolved. However, in case of the undesired turning point occurring at the time stamp 15 can be fixed while mapping it with pixel representation. Table 12 shows an example of exception case 3 and how it is solved. By taking summation of both the points (i.e. a1+a2), before presenting it on the output screen we get ‘a3’. ‘a3’ is approximately equal to ‘a2’ and it is in the same direction as ‘a2’.

Table 12: Resolving Error due to Exception Case 3

	Time Instance(sec)	Magnitude
1.	t1	a1
2.	t2	a2 (mag(a2)>mag(a1))

3.7 Filtration of Channel Leakage

When a user performs a gesture, we notice variation of acceleration in all three co-ordinates. When these values are plotted against time of occurrence, a horizontally compressed sinusoidal wave is noticed along the x-axis, y-axis or z-axis. When the user performs a gesture towards left/right or front/back, we should notice variation only along the x-axis or z-axis respectively. However, due to channel leakage we notice variation in all three axes. When valid turning points are detected, we get them in the desirable direction along with turning points in undesirable directions.

Rectifying Channel Leakage

Channel leakage is rectified by considering magnitude. Magnitude of turning points in undesirable directions is less when compared to magnitude of the valid gesture points detected in the desirable direction. The channel leakage is determined by comparing the magnitude of detected valid turning points in x and z directions. In order to nullify the effect of channel leakage, difference between magnitudes is considered. If the difference of the magnitudes is less than or equal to 1.5, valid turning points with the least magnitude are made null.

In Figure 3.38, accelerometer is moved towards left/right and the action is performed along the x-axis. Valid turning points should be noticed only in the x direction. However, due to channel leakage, valid turning points are recognized on the z-axis also. Here the magnitude of points

obtained in the y-axis and z-axis is less than the magnitude of points in the x-axis. By considering the magnitude and the time occurrence of turning points, channel leakage effect is nullified in Figure 3.39. Figure 3.40 is another example where an action is performed along the z-axis, but channel leakage is noticed along the x-axis and y-axis.

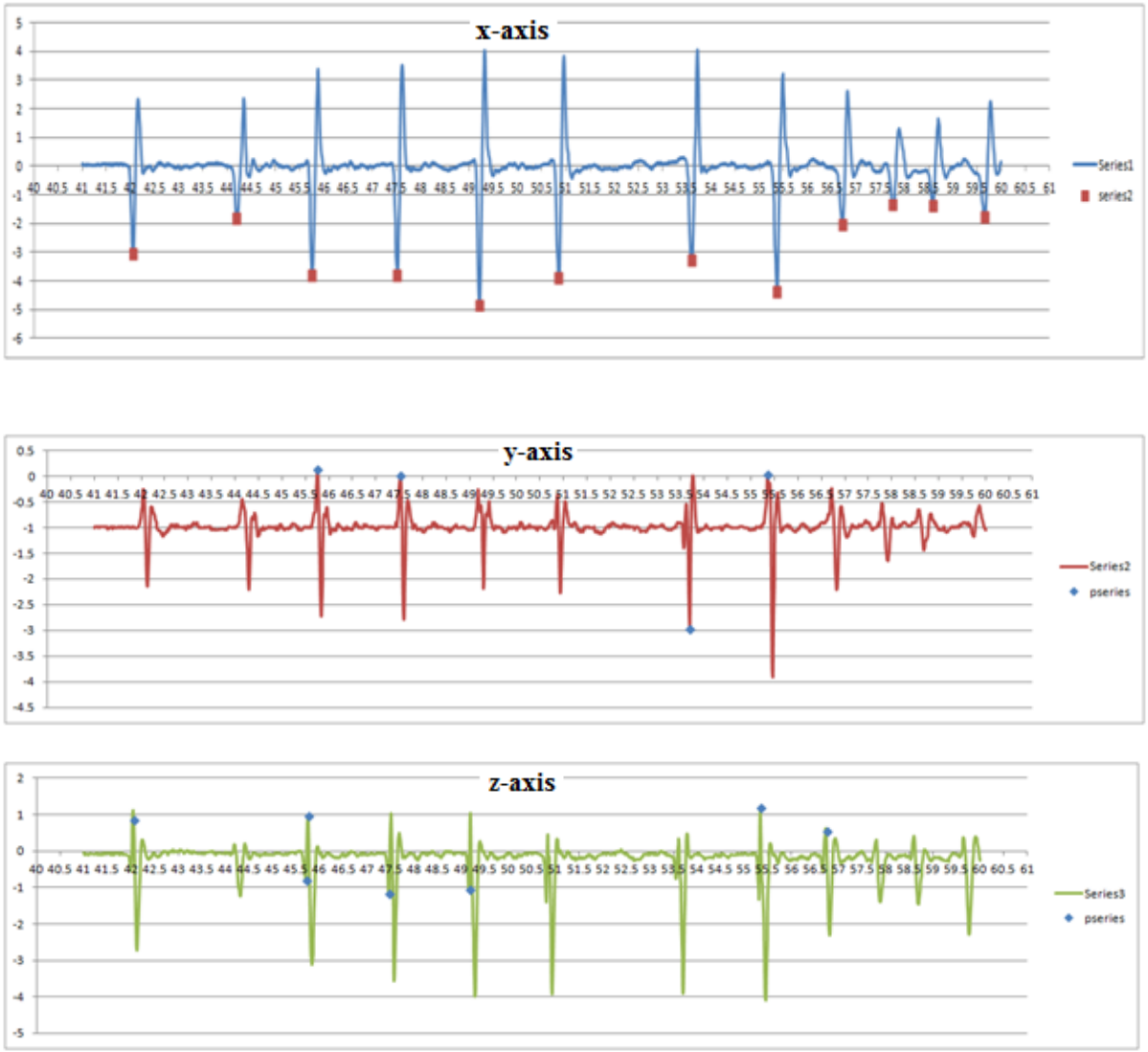


Figure 3.38: Representation of Data for Gestures along x-axis with Channel Leakage

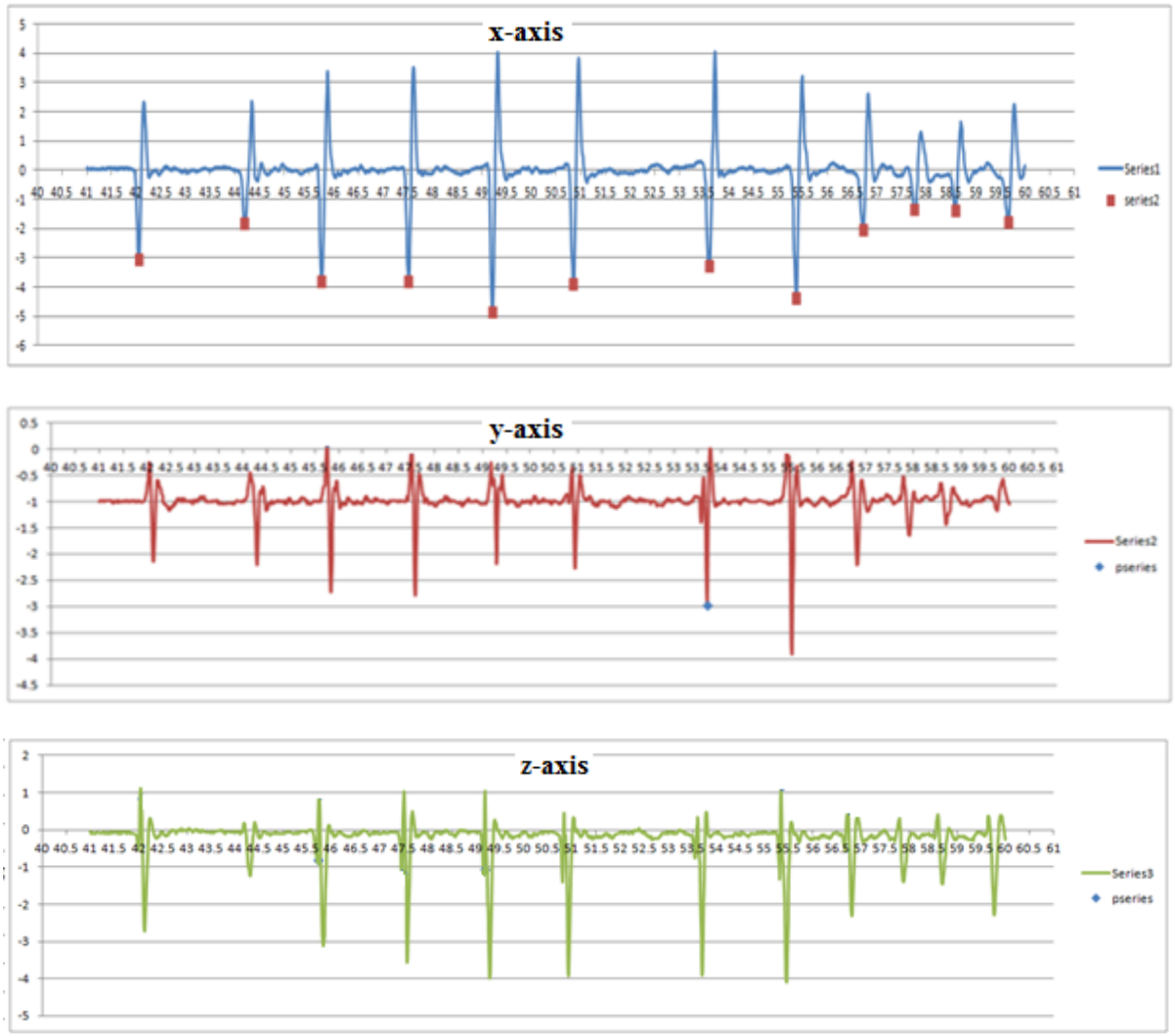


Figure 3.39: Accelerometer Data for Gestures along x-axis after Filtering Channel Leakage

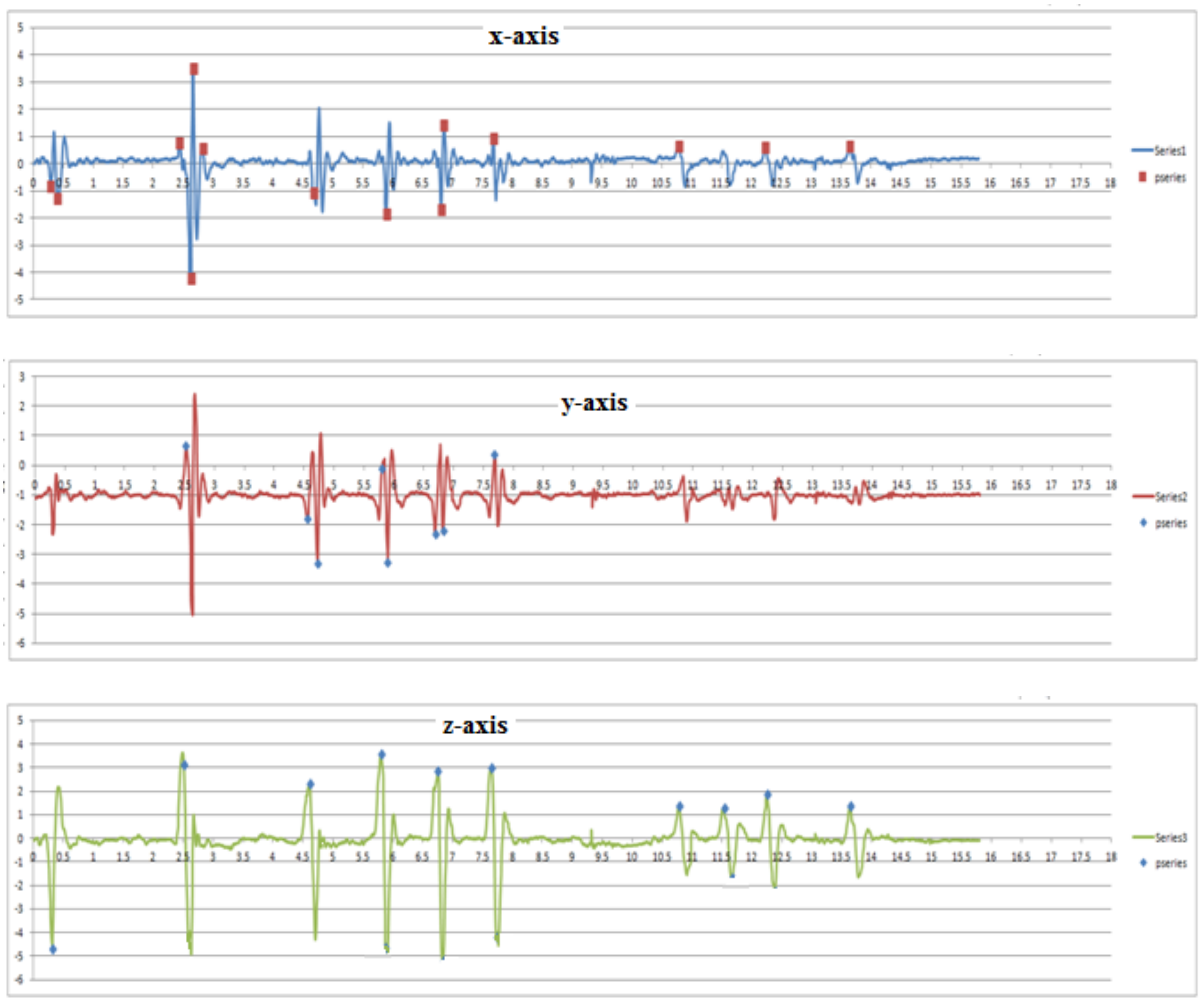


Figure 3.40: Representation of Data for Gestures along z-axis with Channel Leakage

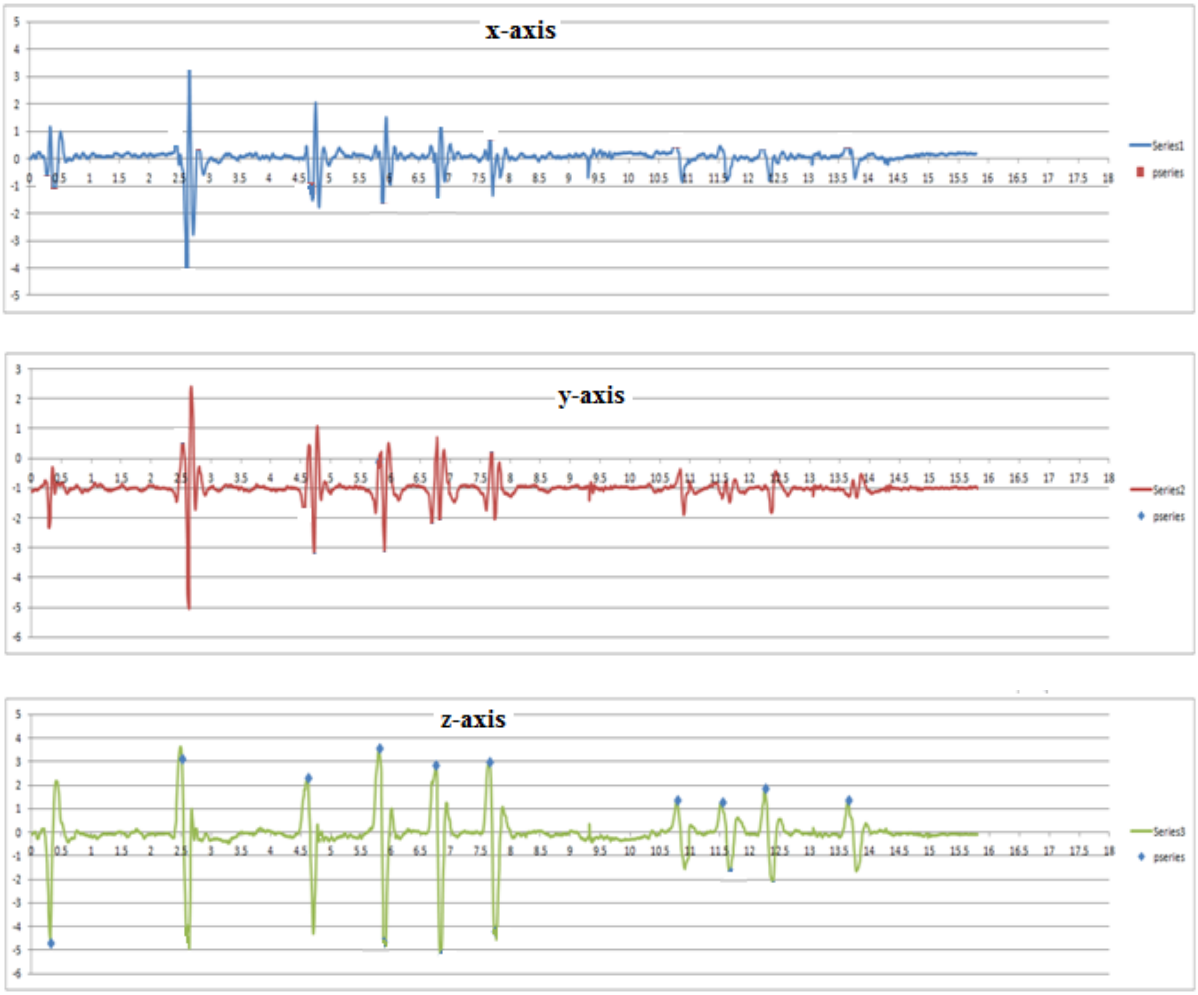


Figure 3.41: Accelerometer Data for Gestures along z-axis with Filtered Channel Leakage

3.8 Calorie Consumption Matrix

Energy expended by the user through AMI application is updated after the user performs a gesture. In this section, we discuss in detail the calculation of the calories burnt by the user. The energy expended by the user is based on force exerted and distance moved. The distance moved and force exerted are calculated based on his weight, the time taken to perform a gesture and magnitude of valid turning points. To be more precise, we consider the weight of the body part performing the gesture.

W—Work Done by user i.e. the calories spent

F—Force exerted by arm

D – Distance moved by arm

m= mass of arm/leg

Σa- Average RMS value of magnitude of valid turning points

Σn- Number of valid turning points

t – mean value of the time taken by gesture

We have work done by user, $W=F * D$

Here force exerted and distance moved by user can be obtained from below formula

$$F=m*a$$

$$D=1/2 * a * t^2$$

From above formula we can say that the work done as:

$$W= ½* m * a^2 * t^2$$

CHAPTER 4

AMI IMPLEMENTATION

4.1 AMI Architecture

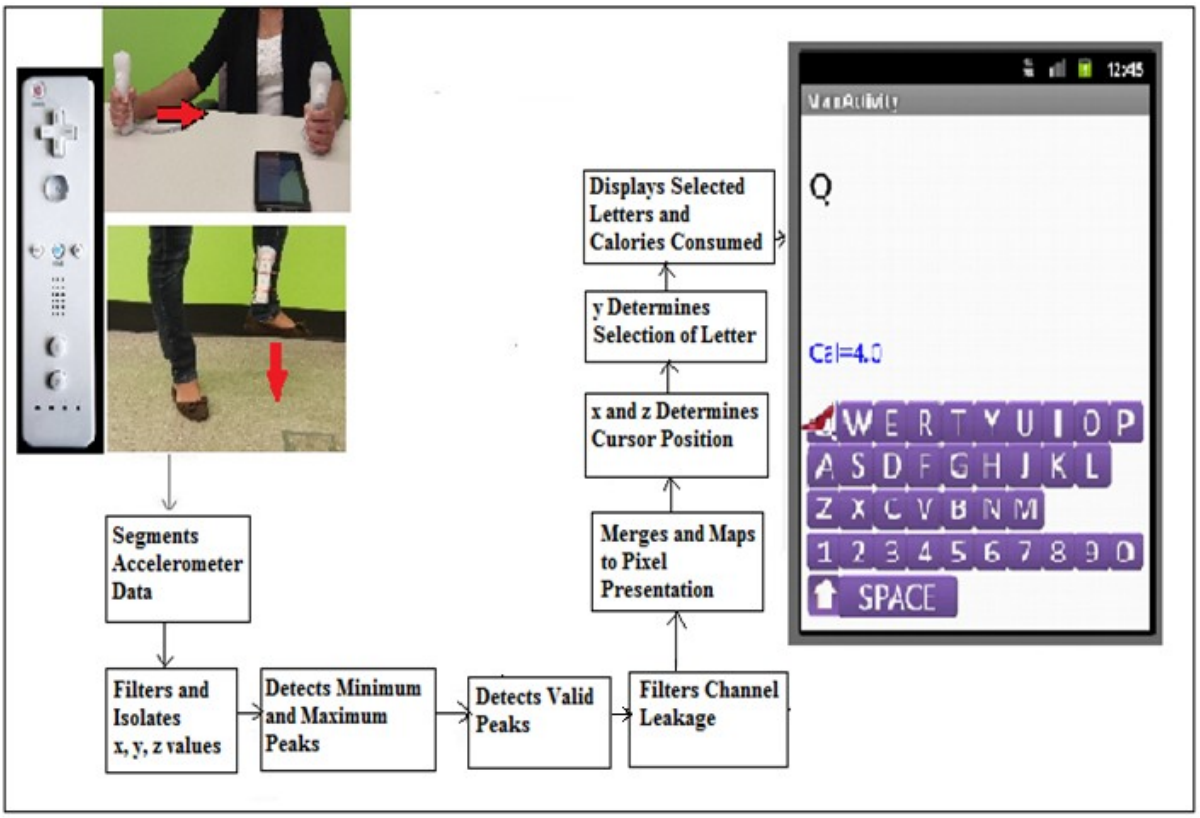


Figure 4.1: Architecture of Active Mobile Interface

We are using the accelerometer in a Nintendo Wii Remote [37] to obtain acceleration values. The Wii Remote with its motion sensing capability returns acceleration values in x, y and z directions. The orientation of the hand and leg is detected by attaching the Wii Remote to them. Acceleration values from the Wii Remote are collected in the client-side system for further processing.

Acceleration values read from the accelerometer are written into a file. Acceleration values recorded for a time window of 'β' seconds are read from the file and considered for gesture

recognition. Values occurring in the time window are filtered based on the threshold level and static value. The static value is obtained by placing Wiimote in the static position for some time. The static value of the Wiimote can be seen in Figure 4.2. X, y and z values are isolated to simplify further processing.

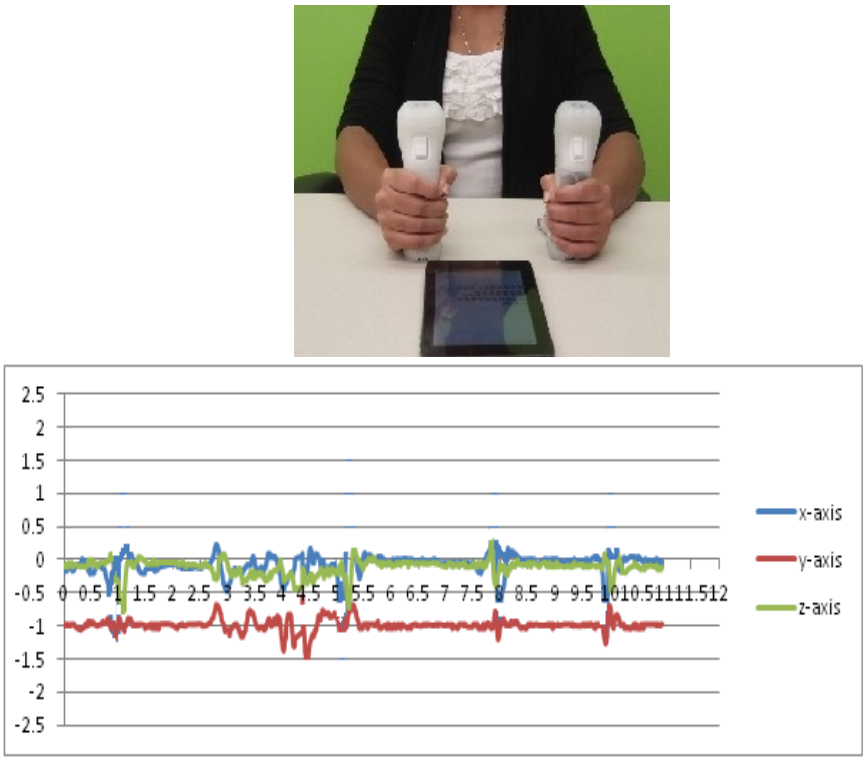


Figure 4.2: Determining Static Value of Wiimote

In the next step, values pass through a differentiator where the peak detection algorithm is applied on filtered data to find maximum and minimum turning points. Next, data pass through a classifier. The classifier detects valid gesture points using the Valid Turning Point Detection Algorithm. The classifier determines valid turning points from set of turning points based on the time taken for a gesture. Next step is filtering channel leakage. Effect of Channel leakage is nullified in this step. To nullify channel leakage magnitude of valid points in all three directions are considered.

Valid points obtained after channel leakage are mapped to pixels and merged as a single point. Mapping to pixels are done by multiplying acceleration values with sensitivity. Sensitivity can be varied as per the requirement of the user. X and z values are used to determine the cursor position and y values are used for selecting letters on the Android application.

Pixel coordinates are appended to the cursor position (i.e. pixel value of the cursor) and approximated to the nearest key value. These values are updated on the screen. Key positions on the screen are predetermined and they are static. Obtained approximated pixel coordinates are compared with pre determined mid values of keys to determine value of the keys. When the key is selected, the letter on key is printed on the screen.

When a gesture is performed by the user, approximate calories consumed for a gesture are calculated and printed on the screen. We can see total calories spent by the user. It is updated dynamically whenever a gesture is performed by the user. Statistics required to calculate calorie consumption are gathered from the user through the mobile interface. The mobile application developed is user friendly. It gathers user data like sensitivity, weight of the user and gender before starting the gesture recognition.

4.2 Wii Remote

The Wii Remote [37] also known as, Wiimote is primarily used as a controller for a Nintendo Wii console. Its motion sensing capability is used for providing gaming experiences. It allows a user to interact with a game or any other application through gestures. It is a one handed device unlike many other traditional devices. It contains an inbuilt accelerometer, IR camera and Bluetooth.

It is used in many gaming applications like ping-pong, racing games, tennis, etc. It can be used like a sword, drum, bat or steering wheel in many gaming applications. We can get various

supporting devices like Wii Zapper or Wii Wheel for holding the Wiimote for various gaming applications like light gun shooters or Racing, etc.

The Wiimote contains a start/stop button to start and stop the Wii controller [37]. It contains four buttons in a '+' shape directional pad. The four buttons (in the clockwise direction) are used for going up, moving right, falling down, and moving toward left in various animated applications. The 'A' button acts like a mouse left click, Pause/ resume or recording a gesture etc. The 'Home' button acts like a menu button. The '+' and '–' buttons are used for increasing/ reducing volume or zooming in/out of the image. The button in the back of the Wiimote is used for the right click of mouse or recognizing a gesture etc. The '1' and '2' buttons may not be used in many applications. However, in AMI both '1' and '2' buttons are held together to initiate the Bluetooth connection. Functionality of the keys varies as per the applications.

It contains four LED lights that blink when the controller is ready to establish the Bluetooth connection. When the Bluetooth connection is established, the first led light glows. A Wiimote comes with an IR camera with inbuilt image processing. It is capable of tracking four objects in motion. It contains two IR LED clusters. A Wiimote returns processed data but not raw image and processed data is used for tracking points in the output. It contains red button near the batteries for resetting the device and establishing Bluetooth connection with various other devices [37].

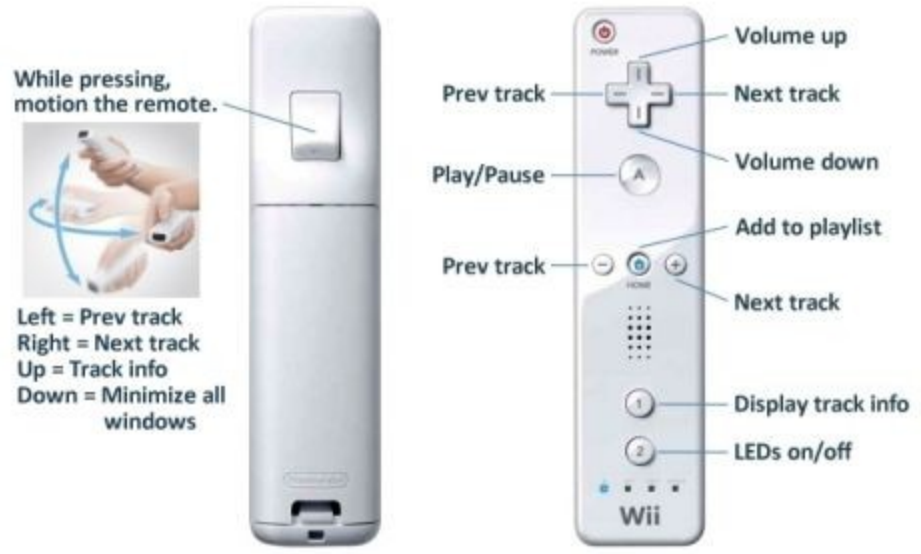


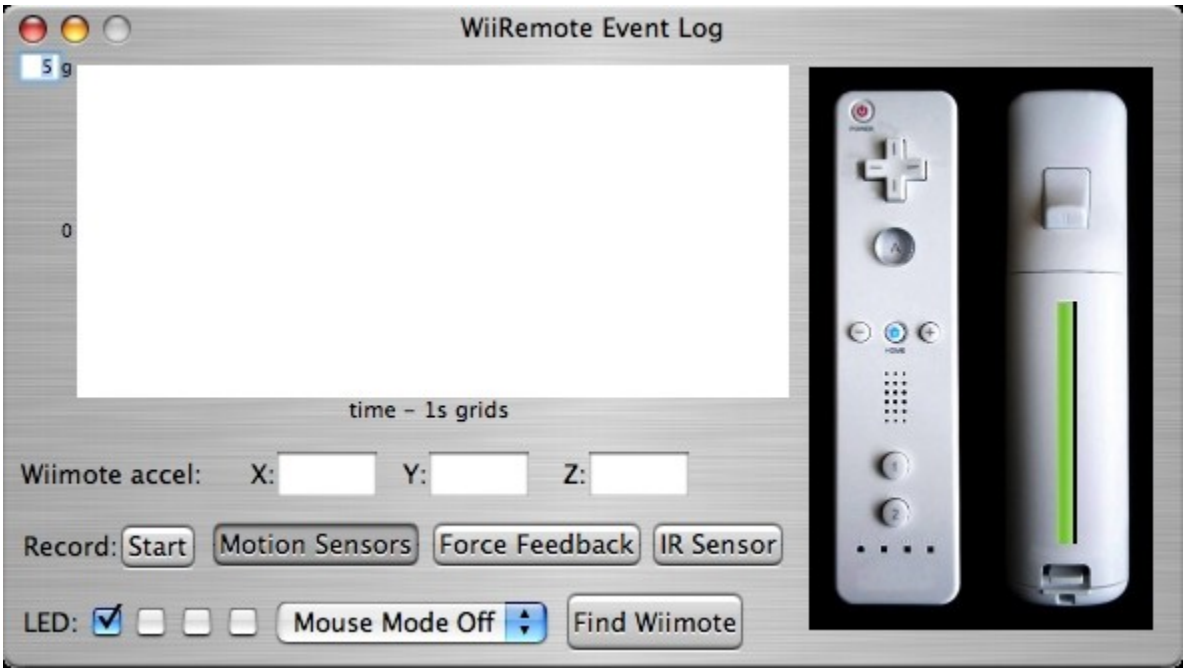
Figure 4.3: Wii Remote Device [41]

Wii Remote contains EEPROM chip where data of 6KB can be freely read and written. It uses batteries as a source of power. We require two AA batteries to power a Wiimote for 25 hours. The device comes with a wrist strap that is attached to bottom of the device. Some gaming applications come with a warning screen to remind player to use strap before starting a game. Strap is for ensuring safety of the controller, as some applications like tennis or ping-pong require heavy gestures. The device also comes with a jacket that wraps the controllers and provides better grip to users. It also protects the controller when it is accidentally dropped.

A Wii Remote is initially connected to the client side system through the inbuilt Bluetooth. Acceleration values detected by the Wii Remote are sent through Bluetooth to the client side system. We are using Darwiin Remote application [42] in a Mac system to establish Bluetooth connection between a Wii Remote and a Mac system. Once the Wiimote is connected to the system, we can record values into a text file. Text file can be saved to any required location. Here it is saved in to the project file that processes the data.

4.3 DarwiinRemote

DarwiinRemote [42] is an application developed for Mac OS. It is used for allowing a Wiimote to control various applications on the Mac system. It has a user-friendly interface. DarwiinRemote interface can be seen in the Figure 4.3.



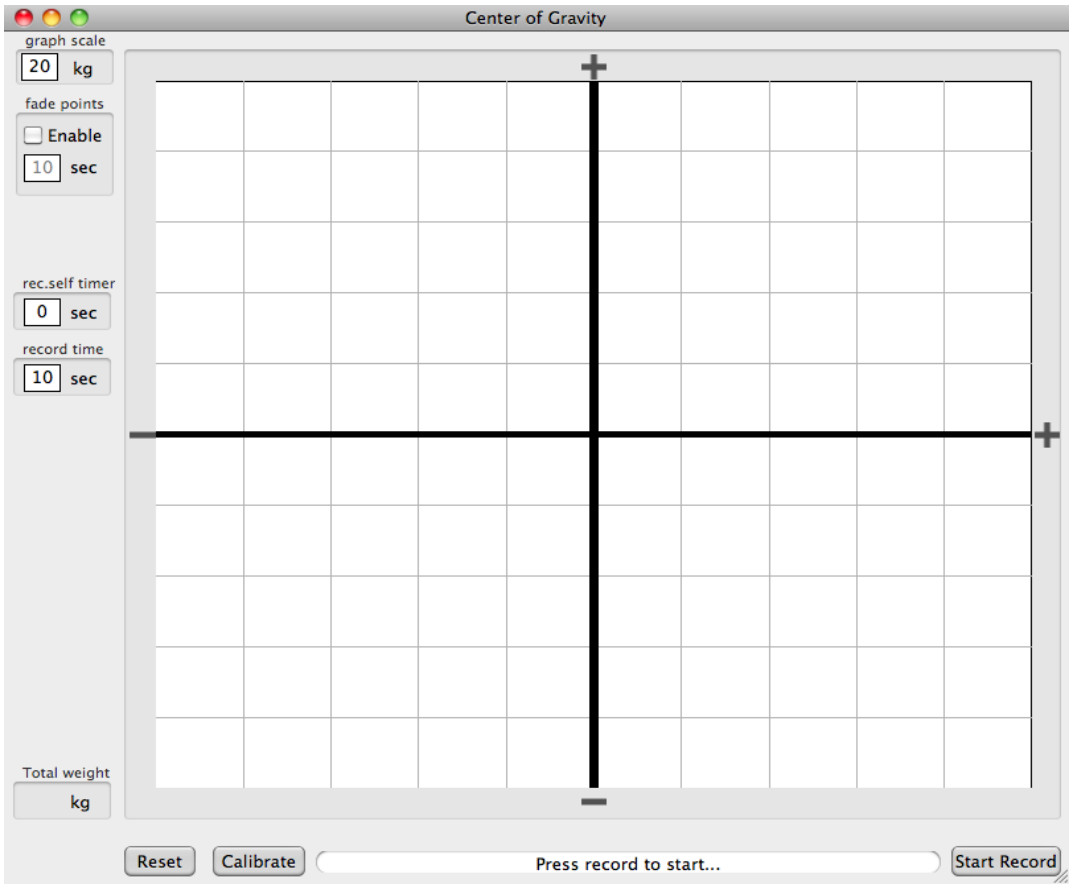


Figure 4.4: DarwiinRemote Interface [42]

DarwiinRemote [42] displays the control stick on the right side of the interface, which indicates Wiimote position, buttons pressed and the LED lights glowing on the Wiimote. It also displays life of the battery in the Wiimote. The button 'Find Wiimote' in the interface detects the Wiimote with the range of Mac inbuilt Bluetooth. Once connection is established between the Mac system and Wiimote, DarwiinRemote displays the Wiimote acceleration values graphically on the screen. We can see x, y and z values updated dynamically. By using the 'Record' option, we can record the accelerometer values into a .txt file, .rft file or any required format. Using the 'Mouse Mode on/off' option, we can control the cursor on the Mac system.

DarwiinRemote [42] not only records accelerometer values but also can accept infrared signals from the Wii sensor bar. DarwiinRemote also has interface to record pressure based on center of gravity**.** DarwiinRemote allows the user to control cursor on Mac system using Wii Remote. DarwiinRemote works on all Mac OS.

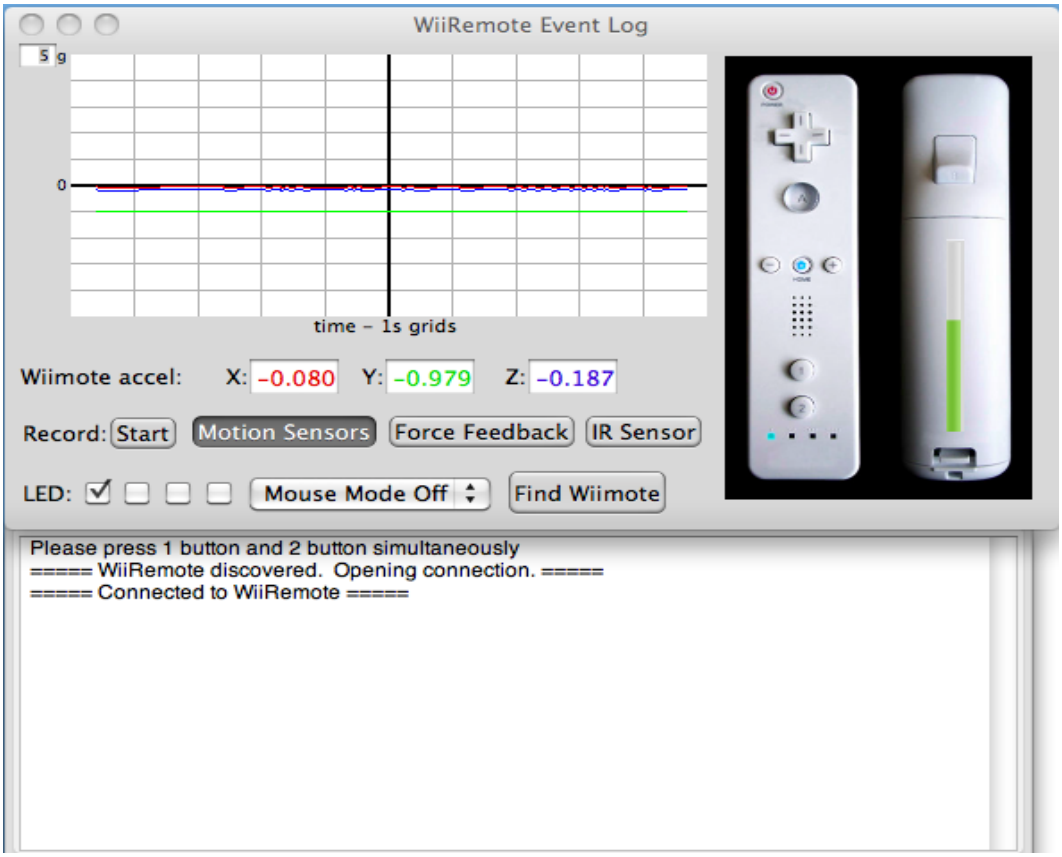


Figure 4.5: DarwiinRemote Recording Accelerometer Values

4.4 Mobile Application

4.4.1 Android Application

Overview

Android [44] is an operating system based on Linux, used for smart-phone and tablets. It is an open source released by Google, which can be freely modified and distributed. It has a large

group of developers writing various applications. Applications are developed using programming language Java. It is currently world's widely used platform for smart-phones. Android applications can be developed using various software and software development tools at no cost. Android applications can also be deployed into real time devices at no cost. Applications improve functionality of smart phones. Few applications help us control various devices like television and other electronic devices by acting as remote controller. Due to its low cost, it encourages large community of developers to implement their ideas at no cost.

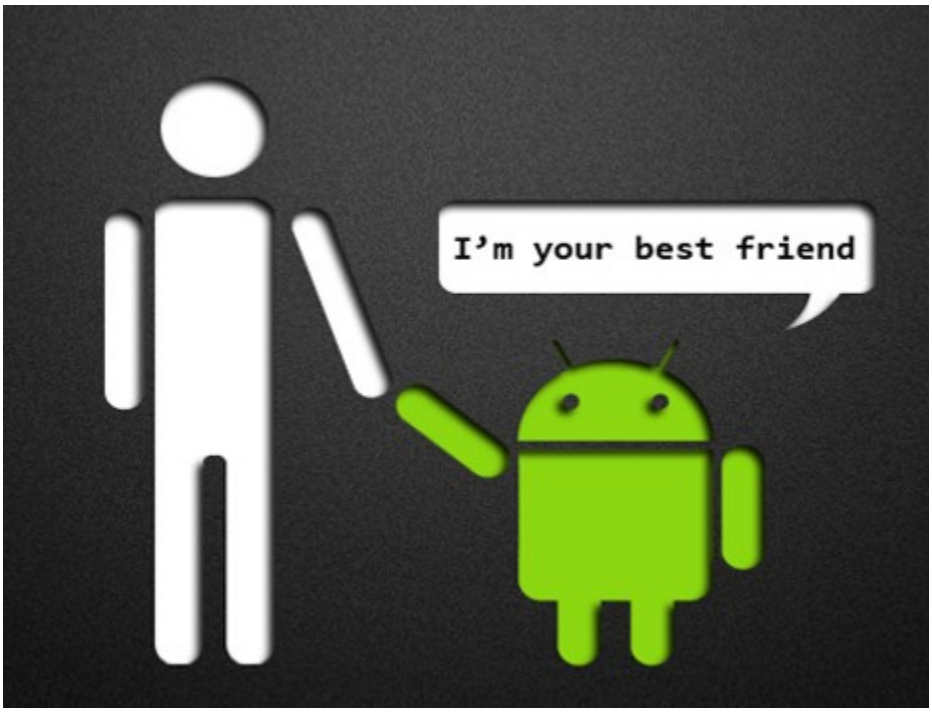


Figure 4.6: Android Smartphone [43]

Android OS [44] is designed to manage RAM for reducing power consumed. When an application is not used in an android mobile, it enters into the inactive state, which means operating system suspends the application. The inactive applications require no processing and battery power. When memory goes down, operating system kills the inactive apps and processes. Android applications run in an isolated area called sandbox. It is isolated from other resources in the system.

Android OS is secured, if the user wants to use system resources through the application, access permissions needs to be provided in the program. For example, in order to connect the application to internet or Wifi, permissions are required and it is provided while developing the application. This provides security to the system keeping it free from malware. Thus, Android applications are very secured and consume less power.

4.4.2 Mapping Accelerometer Values on to Pixels

Valid turning points obtained after processing accelerometer data are mapped with pixels for presentation on the mobile platform. Valid turning points obtained individually along x, y and z directions are mapped to pixels, for determining cursor position and selecting letters. Valid turning points occurring in the same time window are merged to a single point. Distance moved by the cursor is directly proportional to the value of the valid turning point detected. Distance moved by the cursor can be increased by performing a gesture with more force.

Merging to a single point:

Valid turning points occurring in the same time window are considered. Points along the same direction are added to get a single point. This is done to remove human errors occurring when a gesture is performed and to remove undesirable turning points detected.

Valid turning points along x and z directions are considered for controlling cursor motion on the screen. Valid turning points along x and z directions are merged and mapped to pixels to obtain coordinates of the cursor to be displaced on the android screen. Valid turning point along the y direction is for selecting the letter.

Figure 4.7: Motion from Left to Right (along x-axis)

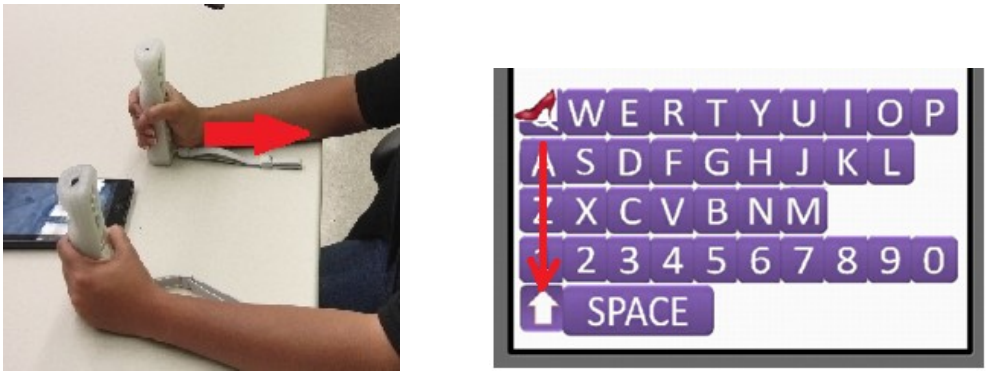


Figure 4.8: Motion from Front to Back (along y-axis)

Sensitivity

Mapping acceleration values with pixel presentation is done by multiplying valid turning points with 'sensitivity'. Sensitivity can be varied from values -10 to -350. Sensitivity is proportional to the distance moved by the cursor on the screen. With less force, distance moved by the cursor on the screen can be increased by increasing the sensitivity value. The user selects sensitivity and it is varied according to the motion performed by the user. The sensitivity value is lowered when the application is used for the leg model and higher sensitivity value is chosen when the application is used for the hand model.

Pixel=Acceleration*sensitivity

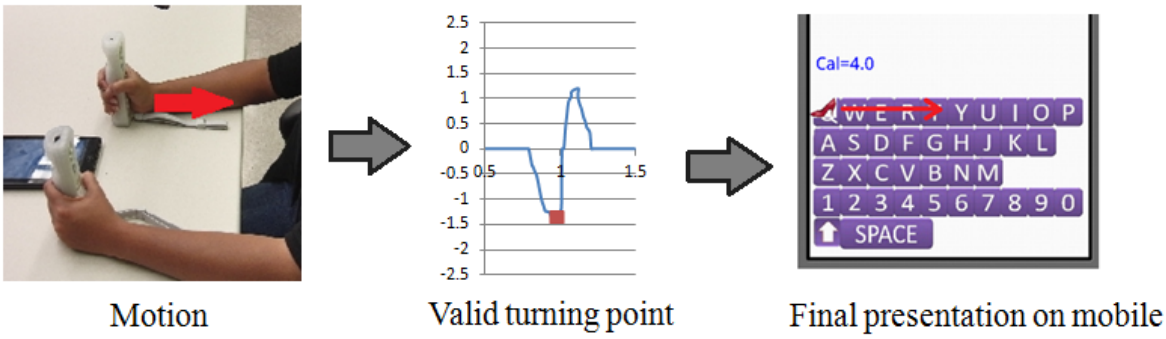


Figure 4.9: Presenting Motion on the Mobile Screen

4.4.3 Presentation on Mobile Platform

Valid turning points are multiplied with the sensitivity to obtain pixel coordinates. Pixel coordinates obtained are approximated to the nearest key value before they are used for controlling the cursor position. The key value is the midpoint of the key present on the android screen and it is represented in terms of pixels. The middle point of the key is obtained by averaging all the pixel coordinates allotted for a key on the android screen. The region of a key on the screen i.e. pixel coordinates of the key are obtained based on position of the key, key size and distance between keys.

Middle point of the nth key along x-axis= (position of the nth key in x-axis+ length of key)/2
Middle point of the nth key along y-axis= (position of the nth key in y-axis+ height of the key)/2

All pixels on the AMI keyboard are approximated to 39 key values i.e. 26 key values for letters, 10 key values for numbers and 3 key values for ‘delete’, ‘space’ and ‘shift’ key. Each of these 39 key values represents a character.

Presentation of a recognized action horizontally:

When the user performs an action, the pixel coordinates of the valid turning point are appended to the existing cursor value. The final cursor value obtained is approximated to the nearest key value on the key pad. This is done to make sure that the cursor moves exactly from one key to another key. Table 13 shows the minimum and maximum value of valid-gesture-pixel-coordinates that needs to be obtained to move the cursor from one key to another key horizontally.

Table 13: Minimum and Maximum Values to Displace Cursor Horizontally

	Minimum value	**Maximum value**
To reach next key Eg: Q to W	(distance between keys)/2+ (size of key)/2	size of key/2+(distance between keys)+size of key +(distance between keys)/2 =(distance between keys)*3/2+size of key*3/2
To reach 2nd key Eg: Q to E	(distance between keys)*3/2 + size of key*3/2	Size of key/2+distance between keys+ size of key+ distance between keys+ size of Key+ distance between keys/2 =size of keys*5/2+distance between keys*5/2
To reach Nth key Eg: Q to any other key	size of keys*(N+1)/2+distance between keys*(N+1)/2	size of keys*(2*N+1)/2+distance between keys*(2*N+1)/2

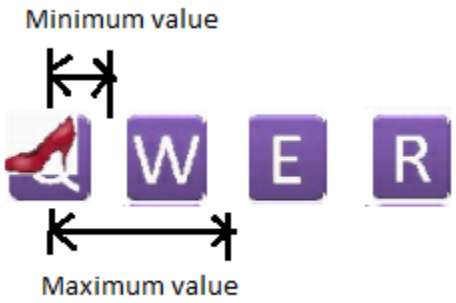


Figure 4.10: Minimum and Maximum Values to Reach Next Key

The values in the range of the minimum value of the key to the maximum value of the key are approximated to the mid value of the key. From Figure 4.10 value of x coordinate (in pixels) of the key is equal to x coordinate of the key 'W', this value is greater than the minimum value and less than the maximum value. Table 13 can be generalized as following:

To move from 1st key to nth key:

Minimum value=size of keys*(2n-1)/2+distance between keys*(2n-1)/2

Maximum value=size of keys*(2n+1)/2+distance between keys*(2n+1)/2

Value to reach nth key:

$$ks*(2n-1)/2+dk*(2n-1)/2 < rv <= ks*(2n+1)/2+dk*(2n+1)/2$$

Where ks: size of keys,

dk: distance between keys,

rv: value to reach nth key

Presentation of a recognized action vertically:

For the vertical cursor movement, five rows needs to be considered. Here pixels obtained after multiplying valid turning point along the z-axis with sensitivity, are approximated of a key value in the vertical direction. This is to make sure that the cursor is present in either of the row. Parameters that need to be considered are size of the key and distance between rows. These vary according to the screen size. Table 14 represents the maximum and minimum value that is required for the cursor to move from one row to another row. Minimum and maximum values can be understood from Figure 4.11.

Table 14: Minimum and Maximum Values to Move Cursor Vertically

	Minimum value	**Maximum value**
To move one rows down	size of key/2+distance between rows/2	size of key/2+distance between rows+ size of key+ distance between rows/2 = size of key*3/2+distance between rows * 3/2
To move two rows down	size of key*3/2+distance between rows*3/2	size of key/2+ distance between keys+ size of key+ distance between keys+ size of key+ distance between keys/2 = size of keys*5/2+ distance between keys*5/2
To move N rows down	size of key* (2*N-3)/2+distance between rows*(2*N-3)/2	size of keys*(2*N-1)/2+ distance between keys* (2*N-1)/2

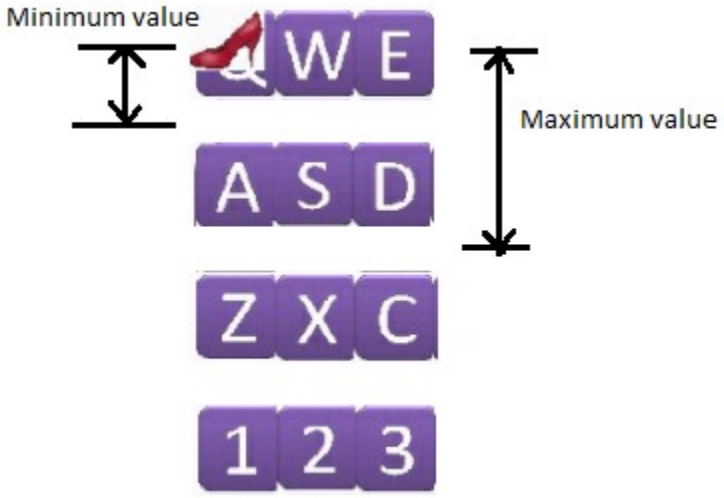


Figure 4.11: Minimum and Maximum Values to Reach Next Row

To move from 1st row to nth row:

Minimum value=size of keys*(2n-3)/2+distance between rows*(2n-3)/2

Maximum value=size of keys*(2n-1)/2+distance between keys*(2n-1)/2

Value to reach nth key:

$$ks*(2n-3)/2+dk*(2n-3)/2 < rv <= ks*(2n-1)/2+dk*(2n-1)/2$$

Where ks: size of keys,

dk: distance between keys,

rv: value to reach nth key

4.4.4 AMI Mobile Application

The mobile interface of AMI is developed to have effective interaction between the user and the system. The mobile application contains two activities. The first activity contains multiple fields to collect the user information required to calculate calorie consumption and obtain the sensitivity value. The sensitivity value taken from the user is used for mapping acceleration values with pixels on the screen. This value is varied from -10 to -350 and can be chosen as per the user requirements. In order to calculate calorie consumption, we need weight of the user, gender, age, and hand/leg model. Weight, gender and the model are used to calculate the mass of the body part to which accelerometer is attached.

Once the user enters the required parameters and presses the button 'start text', the second activity is displayed. The second activity contains the 'QWERTY' keypad, number, shift key, space and send button. The letters selected will be displaced on the screen. The screen also displays the calorie consumed and updates energy spent after every action.

When the user initially opens the second activity, we need to start the communication between the accelerometer, processing system and display on the android screen. In order to establish the connection we just tap on the screen. Once the connection is established, user

gestures are recognized and presented on the screen through the motion of the cursor and letter selection.

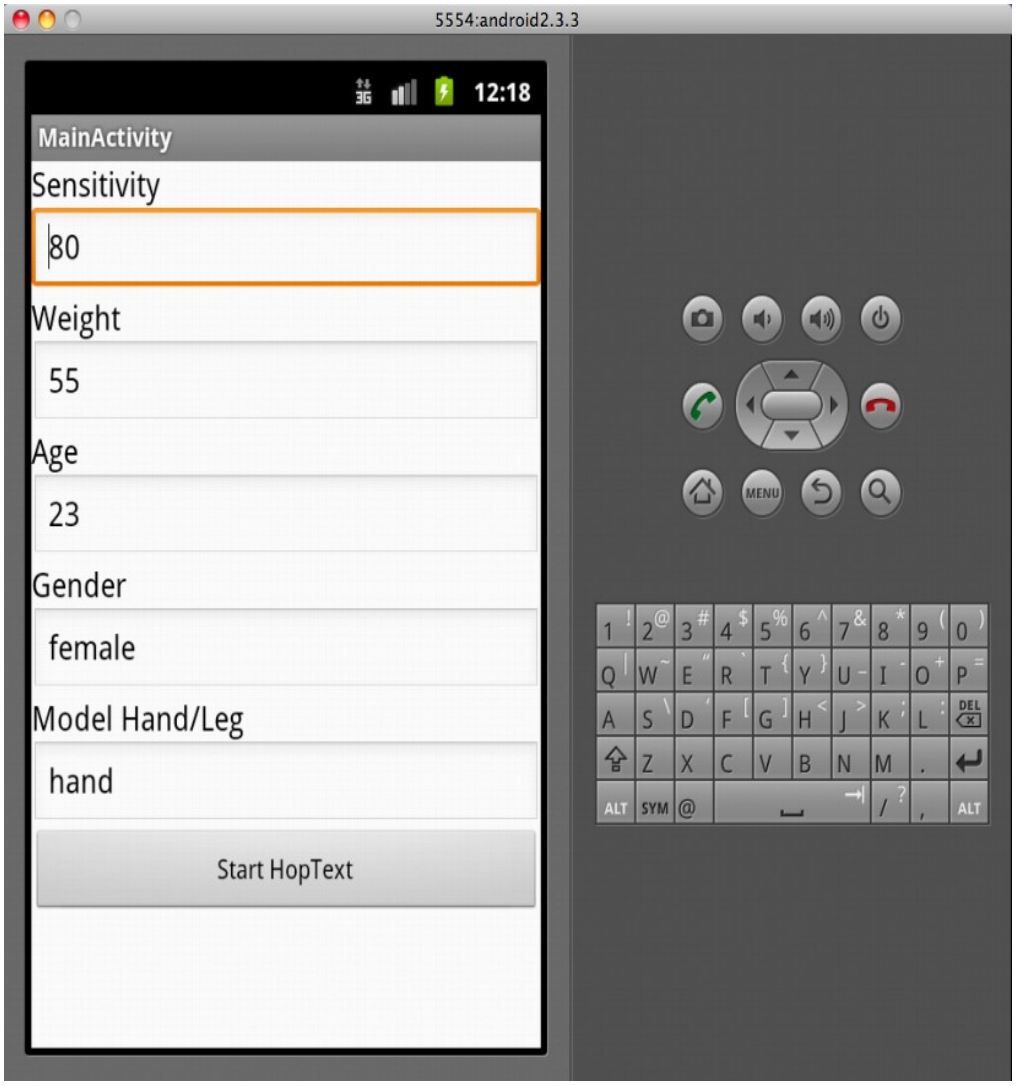


Figure 4.12: Mobile Interface to Gather User Requirements

Figure 4.13: Mobile Interface with QWERTY Keypad and Active Interface

CHAPTER 5

RESULTS AND EVALUATION

5.1 Performance Results

Accuracy Results:

In this section, we are calculating accuracy of the AMI system by calculating accuracy of the Peak Detection Algorithm (PDA). The accuracy of detecting valid turning points and filtering channel leakage are dependent on accuracy of PDA, as the output of PDA is considered for determining valid turning points and filtering channel leakage. Valid turning points are determined accurately if PDA detects all the turning points. Therefore, accuracy of the AMI system is dependent on the accuracy of PDA. Figure 5.1 shows graphical presentation of the accelerometer data collected for one minute, green marks indicate turning points and red marks indicate the valid turning points.

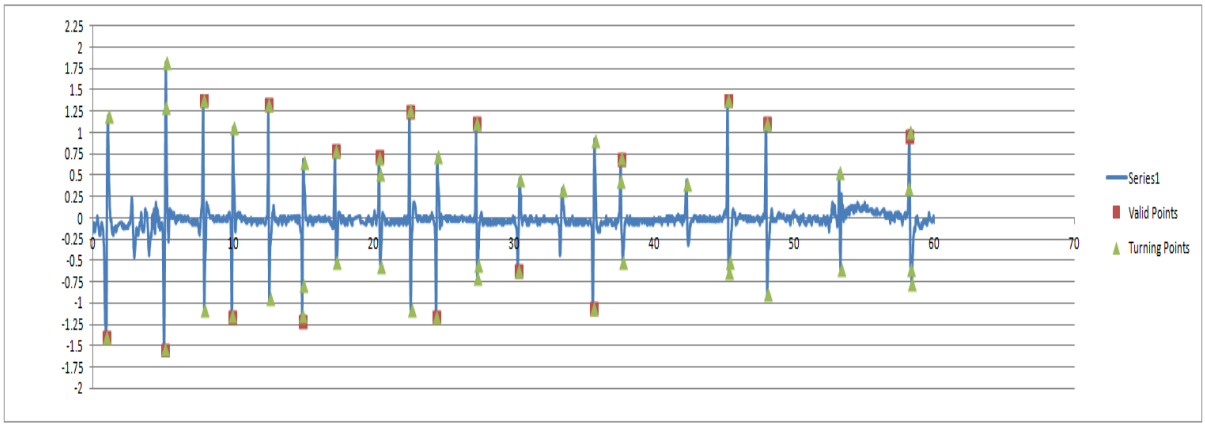


Figure 5.1: Graphical Representation of Data with Turning Points and Valid Turning Points

Here, accuracy of PDA is calculated based on precision and recall of a set of points. Considering three different cases, F-measure is calculated for each case containing a set of points. Each case contains ten sets of data readings. Each set of data reading contains gestures performed for one minute. Each case is evaluated by setting different threshold level for filtering raw accelerometer data. Table 15 shows threshold levels set.

Table 15: Cases of Different Threshold Level

	x-axis Minimum Threshold level	**x-axis Maximum Threshold level**	**y-axis Minimum Threshold level (Δ=0.75)**	**y-axis Maximum Threshold level (Δ=0.75)**	**z-axis Minimum Threshold level**	**z-axis Maximum Threshold level**
Static Level 'ϕ'	0	0	-1	-1	0	0
Case1	-0.6	0.6	-1.75	-0.25	-0.6	0.6
Case2	-0.3	0.3	-1.75	-0.25	-0.3	0.3
Case3	-0.005	0.005	-1.75	-0.25	-0.005	0.005

Static level 'ϕ' in Table 15 indicates readings of accelerometer in rest or static position. The static accelerometer value of the Wiimote is '0' in x-axis, '-1' in y-axis, '0' in z-axis. The magnitude of the threshold level for filtering raw data is set to 0.6, 0.3 or 0.005 along x-axis and z-axis. Here the magnitude of threshold levels is varied only along the x-axis and z-axis, as they decide the position of the cursor. Displacement of the cursor is dependent on valid turning points detected along the x-axis and z-axis. Values along y-axis decide if a letter on keypad is selected or not. Threshold value 'Δ' along y-axis is set to a high value, in order to avoid noise and channel leakage. 'Δ' value (i.e. 0.75) along y-axis is chosen greater than 'Δ' value considered in case 1 (i.e. 0.6). Minimum threshold value along y-axis is -1.75 (i.e. ϕ + Δ) and the maximum threshold value along y-axis is -0.25 (i.e. ϕ + Δ), where 'ϕ' value along y-axis is '-1' and 'Δ' value along y-axis is '0.75'. We focus on two axes (i.e. x-axis and z-axis) as we are detecting different gestures occurring in 2-d plane.

$$\text{Threshold level} = \phi \pm \Delta$$

$$\text{Maximum value} = \phi + \Delta$$

$$\text{Minimum value} = \phi - \Delta$$

F-measure of the system is dependent on the true set of turning points retrieved, true set of turning points not retrieved and false set of turning points retrieved. We are detecting three sets of turning points from the output returned by PDA. Figure 5.2 explains the true set of turning points and false set of turning points.

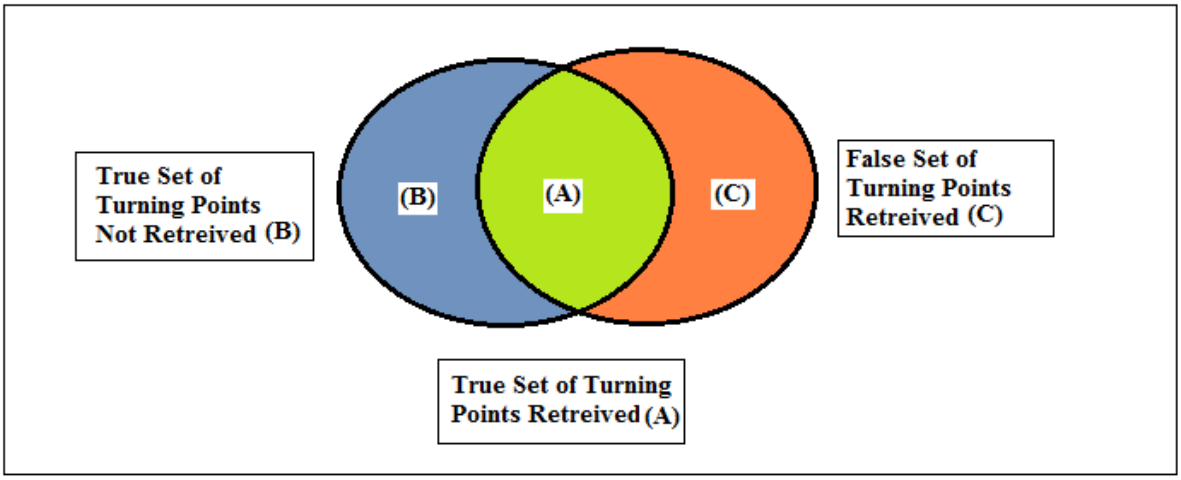


Figure 5.2: Distinguishing False and True Set of Turning Points

Case 1: Threshold value (-0.005, 0.005)

Table 16: Precision and Recall for Threshold Level (-0.005, 0.005)

S. No	Relevant Set of Turning Points Retrieved (A)	Relevant Set of Turning Points Not Retrieved (B)	False Set of Turning Points Retrieved (C)	Recall= A/(A+B)*100	Precision= A/(A+C)*100
1	11	8	2	0.58	0.85
2	2	1	4	0.67	0.33
3	10	5	2	0.67	0.83
4	7	8	6	0.47	0.54
5	10	6	8	0.63	0.56
6	5	5	4	0.5	0.56
7	7	5	4	0.58	0.64
8	8	3	0	0.73	1
9	10	4	2	0.71	0.83
10	9	5	1	0.64	0.9
	Σ=79	Σ=50	Σ=33	0.61	0.71

Case 2: Threshold value (-0.3, 0.3)

Table 17: Precision and Recall for Threshold Level (-0.3, 0.3)

S. No	Relevant Set of Turning Points Retrieved (A)	Relevant Set of Turning Points Not Retrieved (B)	False Set of Turning Points Retrieved (C)	Recall= A/(A+B)*100	Precision= A/(A+C)*100
1	17	2	0	0.9	1
2	3	0	6	1	0.33
3	13	2	1	0.87	0.93
4	15	0	2	1	0.88
5	15	1	2	0.94	0.88
6	9	1	2	0.9	0.82
7	10	2	3	0.83	0.77
8	10	1	0	0.91	1
9	14	0	1	1	0.93
10	14	0	0	1	1
	Σ=120	Σ=9	Σ=17	0.93	0.88

Case 3: Threshold Level (-0.6, 0.6)

Table 18: Precision and Recall for Threshold Level (-0.6, 06)

S. No	Relevant Set of Turning Points Retrieved (A)	Relevant Set of Turning Points Not Retrieved (B)	False Set of Turning Points Retrieved (C)	Recall= A/(A+B)*100	Precision= A/(A+C)*100
1	12	7	0	0.63	1
2	3	0	0	1	1
3	11	4	1	0.73	0.92
4	12	3	0	0.8	1
5	13	3	0	0.81	1
6	6	4	0	0.6	1
7	7	5	0	0.58	1
8	7	4	0	0.64	1
9	11	3	0	0.79	1
10	8	6	0	0.57	1
	Σ=90	Σ=39	Σ=1	0. 7	0.99

Table 19: Comparing Precision and Recall for Different Threshold Levels

	Threshold Level	**Recall**	**Precision**	**F-measure**
Case1	(-0.005,0.005)	0.61	0.71	0.66
Case2	(-0.3,0.3)	0.93	0.88	0.90
Case3	(-0.6,0.6)	0.70	0.99	0.82

In Table 16, Table 17 and Table 18, precision and recall are calculated for each set of data readings by varying the threshold level. In Table 19, we are comparing precision and recall for different threshold levels. We notice that the precision value increases with increase in the threshold level, which indicates that the false set of relevant points retrieved decreases with increase in the threshold level. Increase in the threshold level decreases noise thus decreasing the false set of irrelevant points detected.

Ideally, recall is inversely proportional to precision and it should decrease with increase in the threshold level. Case 1, less threshold level should detect all the turning points thus giving highest recall. However, recall is high at the threshold level (-0.3, 0.3). In the first case, the threshold level is (-0.005, 0.005), chance of retrieving irrelevant points and relevant points increases. However, when detected valid turning points are considered it reports less recall. In the third case, where threshold level is (-0.6, 0.6), small gestures are within the threshold level, thus they are filtered. Therefore, we notice low recall. Figure 5.3, we can see the F-measure for different threshold levels.

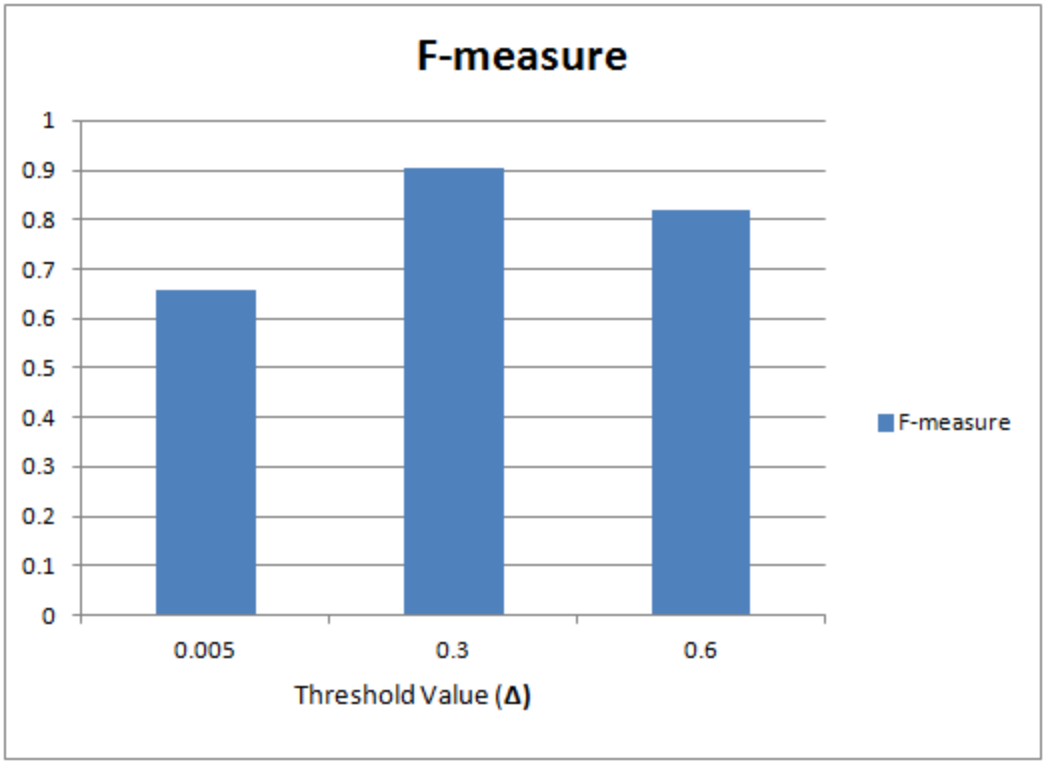


Figure 5.3: Comparing F-measure for different Threshold Level

Determination of Time Duration (α)

Time duration 'α' is the time taken to perform a gesture. Whenever a gesture is performed we get a pair of turning points, the first turning point in the pair is considered a valid turning point. Valid turning points are determined based on the time taken for a gesture 'α'. Two turning points due to a gesture occur in the time duration of 'α/2'. Valid turning points are determined from a set of minimum and maximum turning points and the time duration 'α/2'. Here the time duration 'α/2' is used to determine a valid turning point from pair turning points. In few exceptional cases, we may get three turning points for a valid gesture. However, all the three turning points occur in a time duration 'α'. In this section, we determine the time difference between first two turning points in a valid gesture. Table 20 shows the time difference between first two turning points in a gesture and number of gestures with the time difference 'α/2' between first two turning points.

Table 20: Calculation of Time Difference ‘α/2’

Time Difference between first two turning points in a Gesture (seconds) ‘α/2’	Number of Gestures
0.19	0
0.18	0
0.17	4
0.16	4
0.15	16
0.14	7
0.13	9
0.12	13
0.11	15
0.1	17
0.09	22
0.08	20
0.07	15
0.06	10
0.05	6
0.04	2
0.03	0
0.02	2
0.01	0

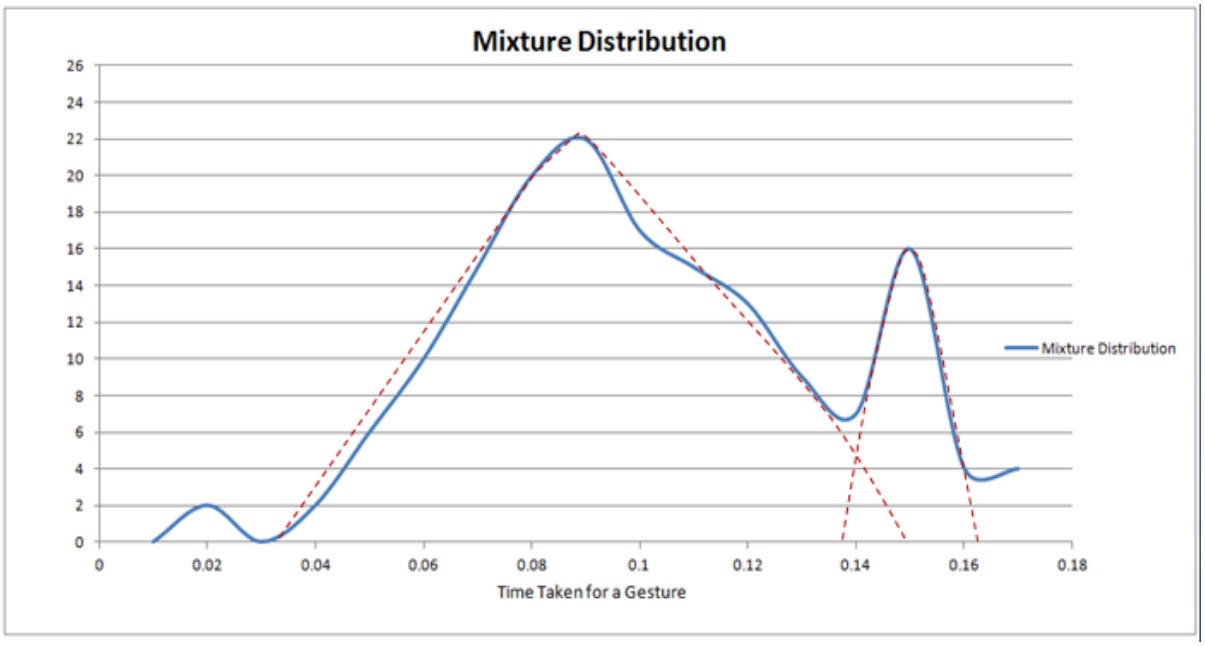


Figure 5.4: Gaussian Representation of Valid Timestamp

Figure 5.4 show two Gaussian curves. First Gaussian curve is due to gestures with three turning points and gestures with two turning points. Here we are considering the time difference between first two turning points of all gestures. In case of gesture with three turning points, time difference between first two turning points is less than 0.14 seconds. We check for third turning point if the time stamp is less than 0.14 seconds. If three turning points are detected in the time duration 'α', the first turning point is considered and other two turning points are ignored. Second Gaussian curve consists of gestures with two turning points.

Gaussian curve 1:

Mean =0.09 seconds

Standard Deviation = Mean±2(0.02)= (0.05,0.13)

Gaussian curve 2:

Mean=0.15

Standard Deviation = Mean±2(0.02)=(0.11,0.19)

From the mixture distribution of two Gaussian curves, we can say that the time difference between first two turning points of all gestures is (0.02, 0.18). From the above calculation of the standard deviation of two Gaussian curves, we get that in case of 95% of the gestures, first two turning points occurs in a time duration of (0.05, 0.19). If 'α/2' is the time difference between first two turning points, then 'α' is the time taken for a gesture. The time taken by a gesture is (0.10, 0.38). However, time difference of first two turning points in gestures with three turning points is (0.05, 0.13). Considering three turning points, the total time taken for a gesture with three turning points is 0.52 (i.e. 0.13*4=0.52). So the total time taken for a gesture is (0.10, 0.52). Time range is tested considering x-axis but results remain same for all the three co-ordinates. All gestures captured below 0.05 seconds are considered as noise.

Calorie Calculation Matrix

AMI application determines calories expended by the user. It updates the mobile screen about the energy expenditure after every gesture performed by the user. Energy spent by the user is dependent on the mass of the body part performing a gesture, maximum acceleration exerted by the user and time taken to perform a gesture.

First activity on the AMI mobile application collects user parameters like sensitivity, weight of the user, gender of the user and body part to which he wants to attach accelerometer (i.e. model he wants to use). Weight, gender and hand/leg model are used to calculate the energy spent by the user. The system reports different calorie statistics depending on the user performance and data entered by the user.

Here is the formula to calculate energy expended for a gesture: $W = F * D = ½ * m * a^2 * t^2$

Acceleration 'a' in the above formula is calculated by considering average of RMS value of the acceleration exerted by the user. Since we are calculating energy spent by the user for hand

model, ‘a’ is the acceleration exerted by the hand. Average weight ‘M’ of a woman is assumed to be 55 kilograms. Weight of the hand ‘m’ is 5.35% of the body weight in case of women [4]. ‘t’ is the time taken to perform a gesture. Here we will calculate energy expended by the user for hand model.

$a = \Sigma a/\Sigma n = 77.68/30 * 0.63 = 1.63128\ m/sec^2$

$m = M * 5.35/100$

$t = 0.12$ seconds

$W = \frac{1}{2} * M * 5.35 * (1.63128)^2 * (0.12)^2$

If the average weight of the Woman is 55 Kg

$W = 3.46\ J = 0.6\ cal$

Approximately work done by the user for a gesture comes to 0.6 calories. For one minute, a user can perform 20 motions continuously, if she performs actions without pausing for significant amount of time. Then the energy expended by hand model for one minute is 12 calories. For five minutes, if she performs gestures continuously she can spend around 60 calories.

5.2 Comparing with Other Models

In this section AMI model is compared with Hidden Markov Model (HMM) and Principal Component Analysis (PCA) based recognition. Performance is evaluated in terms of accuracy and speed. Accuracy of motions towards left, towards right, towards back and towards front are tested. Here we tested each motion 20 times and the percentage of accuracy is calculated based on number of times a model detects an action. Table 21 shows the percentage of accuracy of each model.

Table 21: Comparing Accuracy of Different Models

Motions	HMM Model [2]	PCA Based Recognition [24]	Active Mobile Interface
Left to Right	90%	64.29%	96.67%
Front to Back	84%	50%	91.43%
Right to Left	81.5%	42.86%	90.33%
Back to Front	81.5%	51.14%	100%

In case of HMM and PCA based recognition, gesture towards right, towards back, towards left and towards front are initially trained and then recognized. These models require training of gestures before recognition. In case of HMM, each gesture is trained for around 10 times and in case of PCA each gesture is trained for 8 times. After training all the four gestures, they are tested 20 times and accuracy is calculated. Figure 5.5 shows the graphical representation of accuracy comparison of the three models.

Accuracy of each gesture =

(Number of times a gesture is recognized/ Total number of times a gesture is tested * 100)

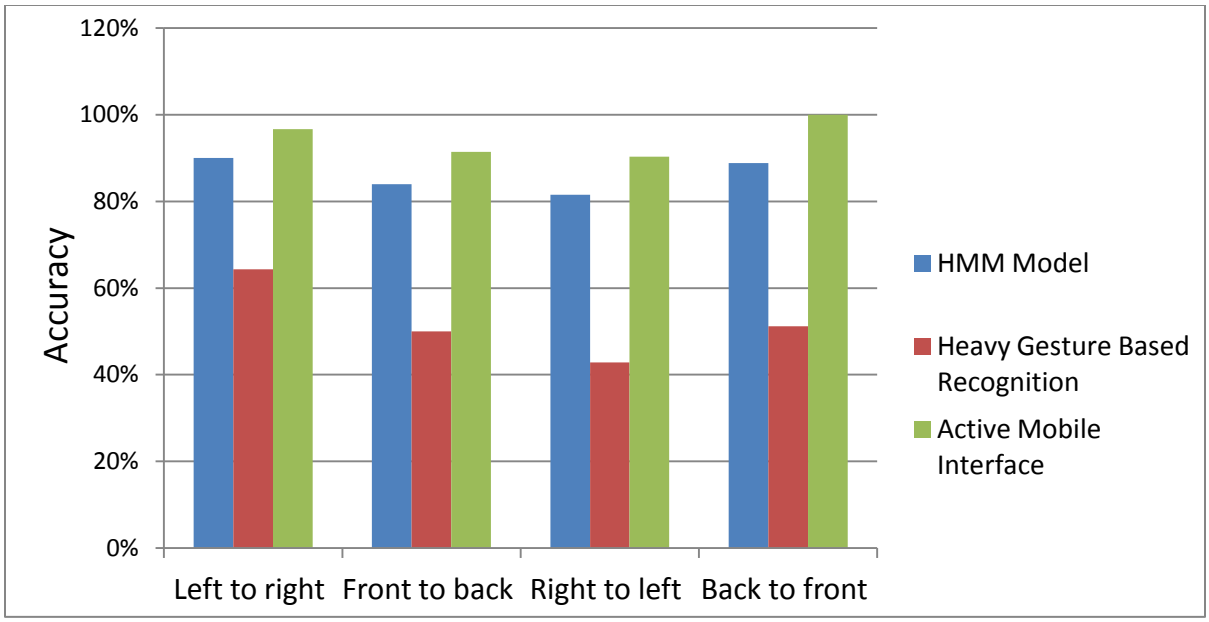


Figure 5.5: Comparing AMI with Other Models in Terms of Accuracy

Accuracy detected by PCA is low as it cannot accurately differentiate the gestures (in opposite directions) performed in same axis. PCA fails to recognize symmetric gestures along same axis but in opposite direction accurately. PCA cannot differentiate gestures performed with different force, it detects all the heavy gestures above certain threshold as a single gesture. PCA cannot detect diagonal gestures, it approximates the diagonal gesture towards left moved front as a gesture towards left or a gesture towards front. In case of HMM, it can recognize the gestures in different directions but cannot differentiate gestures based on force applied. In order to recognize diagonal gestures with HMM, training and performance of gesture should be more precise; otherwise, it recognizes diagonal gesture towards front moved right as a gesture towards front or a gesture towards right. Limitations of HMM and PCA approach are diagonal motions cannot be recognized and gestures cannot differentiated based on force applied.

Runtime Comparison

Table 22: Comparing Run Time of Different Models

Model	HMM Model [2]	PCA Based Recognition [24]	Active Mobile Interface (AMI)
Recognition Time	52 ms	22 ms	8.30 ms

From Table 22, we can say that AMI is the fastest approach when compared to HMM and PCA based recognition. First two models, HMM and PCA reads data and recognizes the gesture based on data read and training data, where as AMI reads accelerometer data and recognizes the gesture based on data read. Therefore, AMI can processes data faster than first two approaches.

AMI requires no training and delay occurs only due to recording of the accelerometer data for time duration of 'β' seconds. The recording time 'β' can be decreased further by increasing overhead, which will increase the processing time.

5.3 User Evaluation

Objective

In this section, we presented results obtained by testing AMI application on users. We tested various aspects which include the maximum number of characters can be typed by users using hand/leg model, improvement in performance on training, heart rate of the users etc. To replace conventional typing mechanism in smart phones, AMI application should type maximum number of letters on the screen. Conventional messaging task is carried out through keypad or touch interface. Messaging application is for exchanging information between people through texting. Now days it has become one of the entertaining application. Especially young people and children send messages for fun rather than using it for emergency or exchanging only important information.

We have come up with the idea of Active Mobile Interface (AMI), where gestures are used to generating text message. AMI application is to do messaging task through physical activity. It is tested on various users of different weight and gender. This application is tested with age group 21-25. We came with interesting results.

User Training:

Conventional messaging applications (i.e. texting on QWERTY key pad or touch interface) is in implementation from many years, so users are trained to use these applications. In order to message on Active Mobile Interface, users are trained for few minutes. Users are given instructions to type their name. Instructions include different directions to move his arm/leg to control the cursor position on the screen and to select the letter.

Table 23: User Productivity for Normal Mode Texting, Hand Mode Texting and Leg Mode Texting

User	Gender	Weight	Normal Mode Texting for 10 Minutes	Hand Model Texting for 10 Minutes	Leg Model Texting for 10 Minutes
User1	Male	63	430	27	5
User2	Male	70	460	33	6
User3	Male	75	455	27	5
User4	Male	60	485	35	8
User5	Male	75	440	28	6
User6	Female	43	465	32	5
User7	Female	50	445	34	7
User8	Female	50	445	30	4
User9	Female	55	450	28	6
User10	Female	53	455	35	6

Table 23 shows performance of ten different users that contains five males and five females. It shows gender of the user, weight of the user and number of characters typed by the user using normal mode texting, hand mode texting and leg mode texting. Average performance of the user for normal mode texting is 453, for hand mode texting is 32 and for leg mode texting is 6. Results show

the productivity is not dependant on the gender. Users who perform above average in normal mode texting can also perform above average in both hand mode texting and leg mode texting.

Table 24: User Performance on Training

User	Attempt 1	Attempt 2	Attempt3	Attempt4	Attempt5	Gender	Notes
User3	13	19	24	27	27	Male	Don't play games
User5	20	27	29	31	30	Male	Play games
User7	19	25	29	32	32	Female	Play games
User9	14	20	25	30	28	Female	Don't play games

We also discuss about improvement in performance and productivity of users up on training. We tested hand model on four different users multiple times. Performance evaluation is carried out for hand model. Here we considered two male users and two female users. We also considered two users (male and female) who play games frequently and other two users (male and female) who do not play games. They were given set of instructions and basic training initially. In first attempt, there performance was poor but gamers performed better than non-gamers did. They are asked to practice for sufficient time before every attempt. Their performance was improved after every attempt. For fourth and fifth attempt, it was not improved much. Thus, we can say that the performance and productivity of the system are improved up on training. Figure 5.7 shows the graphical representation of the improvement in performance of the users on training.

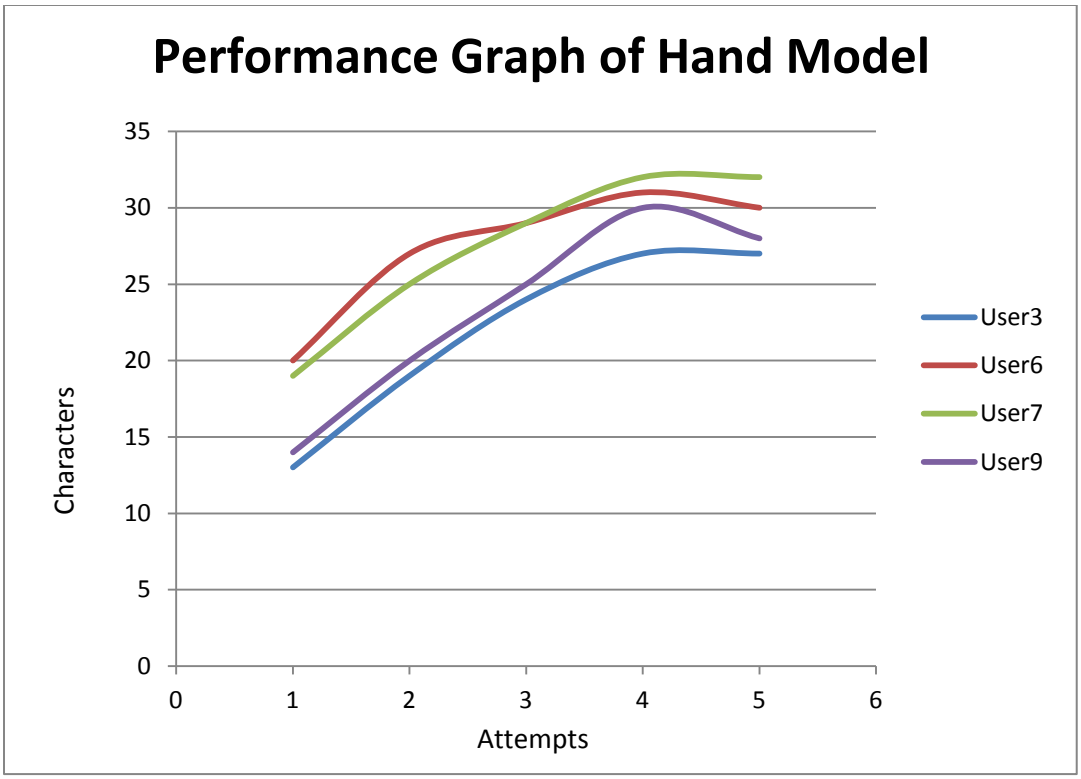


Figure 5.7: Performance Graph of Users after Multiple Attempts

Evaluation of Heart Rate:

We evaluated "Active Mobile Interface" in terms of heart rate using "Beurer Heart Rate Monitor" [45]. Using this device, we can record the heart rate of a user during various activities. We recorded the heart rate for three different activities performed by the user - at normal mode texting (i.e. conventional texting), during hand mode texting using AMI, and while performing leg mode texting using AMI. We calculated t-tests and compared three sets of heart rate recordings. Table 25 shows the heart rate for all three modes of texting.

Table 25: Heart Rates for Normal Mode Texting, Hand Mode Texting and Leg Mode Texting

Time Instance (Minute)	Normal Mode (Heart Rate Per Minute)	Hand Model (Heart Rate Per Minute)	Leg Model (Heart Rate Per Minute)
1	71	71	74
2	71	71	86
3	71	71	110
4	72	71	103
5	72	71	108
6	72	75	103
7	72	90	104
8	71	90	107
9	7 1	90	108
10	71	90	105
11	72	90	108

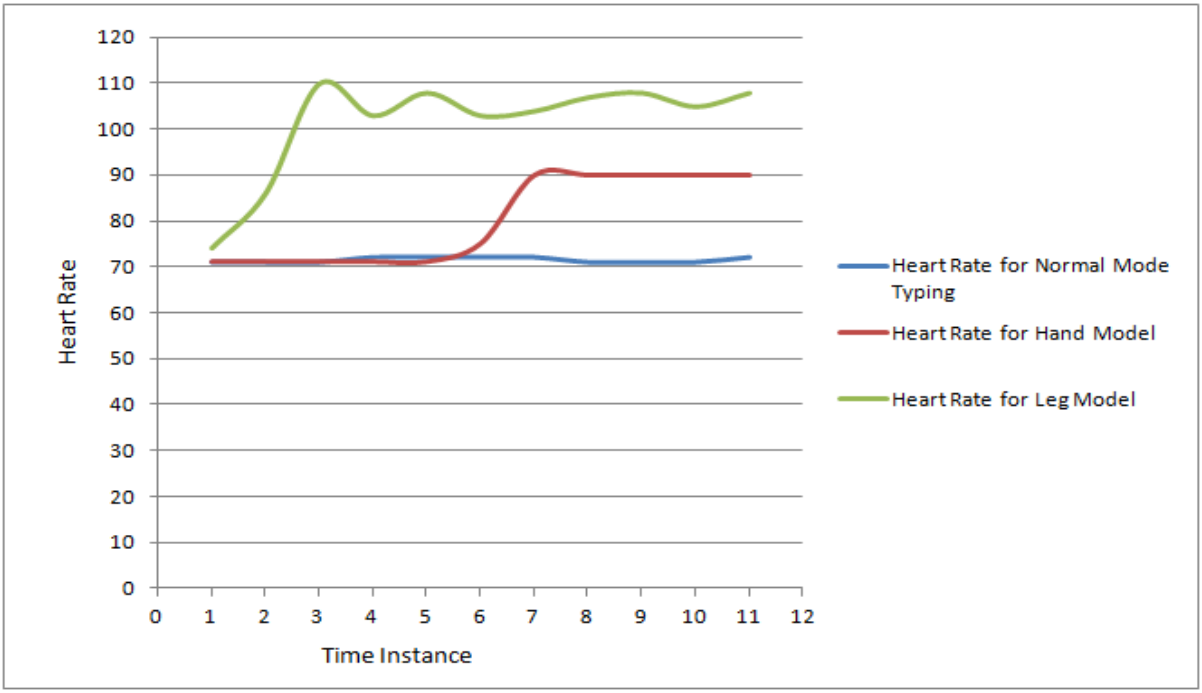


Figure 5.8: Heart Rates for Normal Mode Texting, Hand Mode Texting and Leg Mode Texting

Figure 5.8 shows graphical representation of heart rate at different time instances while performing texting using three different modes. We can see that the heart rate of the user is very high in the case of leg mode texting. In case of normal mode texting (i.e. conventional mode texting) heart rate is same as the user in the rest position. In case of hand mode texting, heart rate is in between normal mode and leg mode texting. Heart rate in the last minute represents the heart rate of the user after all three modes of texting. Higher heart rate indicates higher expenditure, so we can say that the leg mode texting consumes more energy when compared to hand mode texting and hand mode texting consumes more energy compared to normal mode texting.

Table 26: p-value Comparison

	Normal Mode Texting	**Hand Model Texting**	**Leg Model Texting**
Normal Mode Texting	0.5	0.007417	2.2323E-06
Hand Model Texting	0.007417	0.5	5.49E-05
Leg Model Texting	2.2323E-06	5.49E-05	0.5

't-test' results conducted for 'Heart rate while normal mode texting' data versus 'Heart rate during hand mode texting' data:

For 'One-tailed distribution' & 'Two sample unequal variance':

p-value : 0.007417. Significant difference is observed between two groups.

't-test' results conducted for 'Heart rate while normal mode texting' data versus 'Heart rate while leg mode texting' data:

For 'One-tailed distribution' & 'Two sample unequal variance':

p-value : 2.2323E-06 which is less than 0.05. Significant difference is noticed between two groups.

't-test' results conducted for 'Heart rate while hand mode texting' data and 'Heart rate while leg mode texting' data:

For 'One-tailed distribution' & 'Two sample unequal variance':

p-value: 5.49E-05 which is greater than 0.05. Therefore, there is significant difference between two groups.

The results show that the user's heart rate increased significantly when he/she performs heavy gestures than conventional texting. The results prove that energy expended is higher when 'Active Mobile Interface' is used for texting.

Table 27: Heart Rate vs. Productivity (10 min)

	Productivity (Character)	Pulse Rate (RPM)
Normal Mode	470	72
Hand Model	35	90
Leg Model	8	108

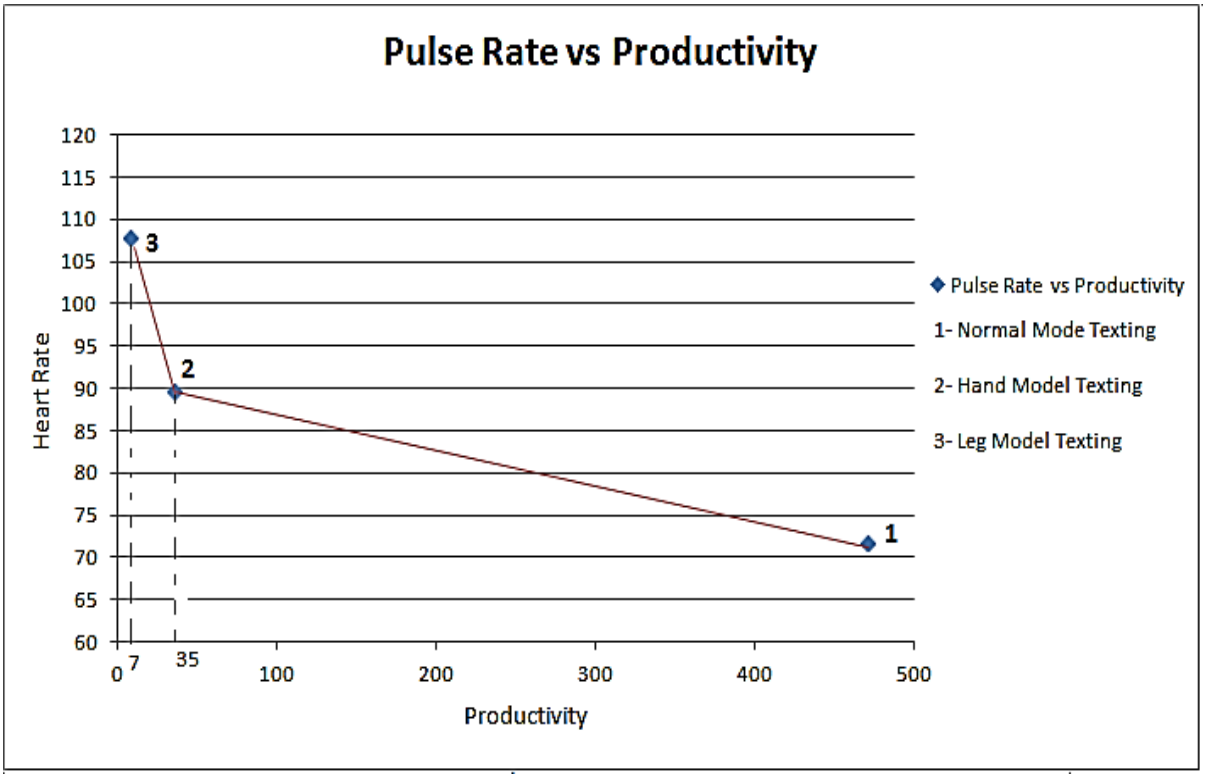


Figure 5.9: Pulse Rate and Productivity of Different Models

In Figure 5.9, we are plotting pulse rate against productivity for three modes. Here we notice that productivity and pulse rates are inversely proportional. Here heart rate recorded in the

last minute is considered for plotting the graph. In case of normal mode texting, heart rate is same as person in rest mode (i.e. 72 rpm). Productivity in the normal mode texting is very high, as the users are trained by default for a very long time. In case of leg model texting, heart rate is very high as the weight, with which the action is performed is high. Though productivity using hand model texting and leg model texting is low compared to normal mode texting, heart rate is very high which means there is a considerable amount of energy is expended using AMI mobile texting.

Heart Rate for Texting (Gender)

Table 28: Comparing Heart Rate of Man and Woman

Gender	Weight (kg)	Normal Mode (Heart Rate Per Minute)	Hand Model (Heart Rate Per Minute)	Leg Model (Heart Rate Per Minute)
Man	73	78	100	120
Woman	53	72	90	108

Here we are comparing heart rate of man and woman after performing three different modes of texting. Each mode of texting is performed for ten minutes and average of last five minutes is presented in Table 28. Weight of the man is noticed to greater than the woman, so his heart rate increases more than the woman while performing texting using hand mode and leg mode. Figure 5.10 shows graphical representation of heart rate of the man and woman for all three modes of texting.

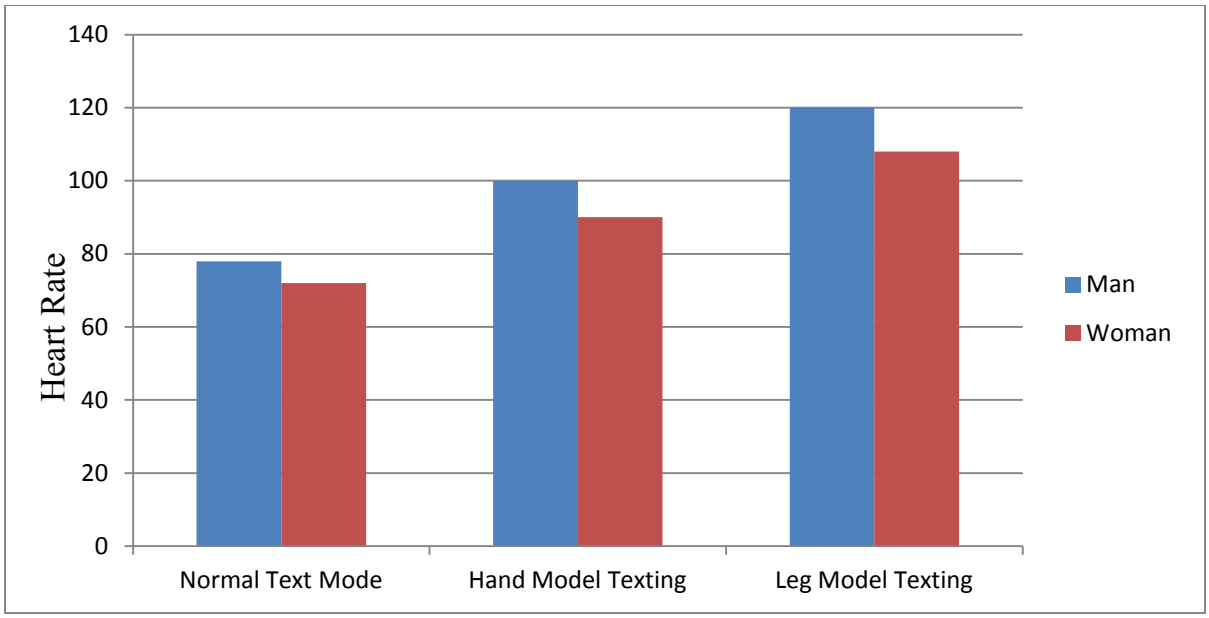


Figure 5.10: Graphical Representation of Heart Rate of Man and Woman

User Comments

AMI application is tested on ten different users. Their opinion and comments are gathered. Most of the users liked the application and they are interested in choosing it over conventional texting as it is different from conventional methods. Most of them think it is confusing and it needs to be improved. Most of the actions to control cursor should to be performed in only one plane (i.e. x-z plane), some of the users tend to perform activities in x-y plane i.e. instead of moving Wiimote towards front (i.e. z-axis) to move the cursor up the row users tend to Wiimote up (i.e. y-axis). Few users tend to tilt Wiimote unintentionally.

Table 29: Users Opinion

	Do you like the AMI application?	Do you want to use AMI again?	Do you think AMI needs to be improved?	Do you think AMI can be used like a gaming application?	Do you choose AMI over conventional texting?	Is it confusing?
User1	Very good	Yes	May be	Yes	Yes	Partially
User2	Fair	No	Yes	I don't know	No	Yes
User3	Good	Yes	May be	Yes	May be	Yes
User4	Very good	Yes	May be	Yes	May be	No
User5	Good	Yes	Yes	Yes	May be	No
User6	Good	Yes	May be	Yes	Yes	Partially
User7	Good	Yes	Yes	Yes	May be	Partially
User8	Good	May be	Yes	Yes	May be	Partially
User9	Good	Yes	Yes	May be	May be	Yes
User10	Good	Yes	Yes	I don't know	No	Yes

Hand model and leg model are tested on five men and five women. Men have more strength and can perform heavy gestures easily and repeatedly. They are willing to put effort required for the application compared to women. Therefore, men showed more interest when compared to women and most of them were willing to retake the test.

Application is tested on gamers and non-gamers. Users who are into gaming are able to perform better than non-gamers in the first attempt. Gamers are able to follow the rules of application more easily when compared to non-gamers. They felt it more like a gaming application rather than normal texting application. Productivity of the application is improved for both the sets of users after training. Table 30 shows the comments by users on AMI mobile application.

Table 30: User Comments

User	Comment
User1	'I get bored by conventional method of typing, but this is more like a game and I'm excited to Hop-Text'
User2	'It is entertaining but I wish I could type more letters in less time.'
User3	'It is good. I need more practice to type more letters and to avoid confusion between the gestures.'
User4	'It is more like a game. It is good to see I can burn calories without actually noticing'
User5	'It is like a joystick game but I'm moving my arm instead of moving just my fingers, wish I can use both my arms instead of one'
User6	'It is fun, I can use it when I'm bored but not in emergency. Also I need more practice to use it a conventional texting application'
User7	'Cursor movement is precise and it is difficult to reach adjacent letter. Otherwise it is fun'
User 8	'It is interesting and more like a video game. But if I can correct errors during Active Mobile Interface automatically it will be great'
User 9	'Burning calories while texting is fun. But I got confused about the gestures especially while moving the cursor up and selecting the letter.'
User 10	'It is good something new other than conventional texting. I need to concentrate while performing gestures, about orientation of Wiimote.'

CHAPTER 6

CONCUSION AND FUTURE WORK

6.1 Conclusion

In this section, the contribution of this thesis is summarized. We presented a novel model for active mobile interface for texting on mobile devices. We successfully implemented portable, affordable, computationally less intensive motion recognition method. We showed potential to motivate people for physical activities while maintaining productivity.

We evaluated AMI application on different set of users. Through a user's study we showed that the AMI application is fun and acceptable. Study also showed that considerable amount of energy is expended using AMI application. Thus, we can spend energy by performing a productive task.

6.2 Future Work

There are some of the limitations of the project and scope for enhancement. AMI application can be extended for diverse apps like gaming and education. Different gaming applications can be implemented with multiple mobile platforms. We can also develop applications with multiple devices like Wiimote on the same platform. The productivity of the mobile application can be improved by adding ontology for word recognition.

Messaging application can also be improved by controlling the action of the cursor dynamically. Rolling the cursor on the QWERTY keypad, controlling its direction, stopping the cursor and selecting the letter whenever a cursor is on the required letter can improve productivity of the application. This might also reduce chance of errors.

REFERENCES

1. Lester J., Choudhury T., and Borriello G. A practical approach to recognizing physical activities. *Pervasive Computing* (2006): 1-16.
2. Yang J., and Xu Y. *Hidden markov model for gesture recognition*. No. Cmu-Ri-Tr-94-10. Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst (1994).
3. Arteaga S. M., Kudek, M., and Woodworth A. Combating obesity trends in teenagers through persuasive mobile technology. *ACM SIGACCESS Accessibility and Computing* 94 (2009): 17-25.
4. Dempster, W. T., and Gaughran, G. R. Properties of body segments based on size and weight. *American Journal of Anatomy* 120.1 (2005): 33-54.
5. Physical Activity: A Journal about the Physical Activity and Factors Motivating Physical Activity. URL: https://www.presidentschallenge.org/informed/digest/docs/200009digest.pdf
6. Goran, M. I., Reynolds, K. D., and Lindquist, C. H. Role of physical activity in the prevention of obesity in children. *International Journal of Obesity* 23 (1999): S18-S33.
7. Schlömer, 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*. ACM (2008): 11-14.
8. Wang, S. B., Quattoni, A., Morency, L. P., Demirdjian, D., and Darrell, T. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 2). IEEE (2006): 1521-1527
9. Gavrila, D. M. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73.1 (1999): 82-98.

10. Ehreumann, M., Lutticke, T., and Dillmann, R. Dynamic gestures as an input device for directing a mobile platform. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on.* Vol. 3. IEEE (2001): 2596-2601.

11. Mantyla, V. M., Mantyjarvi, J., Seppanen, T., and Tuulari, E. Hand gesture recognition of a mobile device user. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*. Vol. 1. IEEE (2000): 281-284.

12. Liu, J., Zhong, L., Wickramasuriya, J., and Vasudevan, V. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, *5*.6 (2009): 657-675.

13. Pylvänäinen, T. Accelerometer based gesture recognition using continuous HMMs. In *Pattern Recognition and Image Analysis*. Springer Berlin Heidelberg (2005): 639-646.

14. Obesity Statistics. URL: http://www.heart.org/idc/groups/heartpublic/wcm/sop/smd/documents/downloadable/ucm_319588.pdf

15. Statistics about Obesity in US: http://www.cdc.gov/healthyyouth/obesity/facts.htm

16. Motivating Kids to do Physical Activity: URL: https://www.presidentschallenge.org/informed/digest/docs/200009digest.pdf

17. Mobile Fitness Applications: http://www.nytimes.com/2012/12/27/garden/devices-to-monitor-physical-activity-and-food-intake.html?_r=1&

18. Second Life. URL:http://secondlife.com/

19. Dean, E., Cook, S., Keating, M., and Murphy, J. Does this avatar make me look fat? Obesity and interviewing in Second Life. *Journal of Virtual Worlds Research* 2.2 (2009): 1-11.

20. Wu, Y., and Huang, T. S. Vision-based gesture recognition: A review.*Urbana 51* (1999): 103-115.

21. Microsoft Kinect. URL: http://www.xbox.com/en-US/kinect

22. Microsoft Kinect. A Microsoft Product for Gaming Console. URL: http://en.wikipedia.org/wiki/Kinect

23. Texas Instruments. URL: http://www.ti.com.

24. Ginjupalli, S. A Gestural Human Computer Interface for Smart Health , MS Thesis, University of Missouri – Kansas City, 2013.

25. Ganesan, S., and Anthony, L. Using the kinect to encourage older adults to exercise: a prototype. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*. ACM (2012): 2297-2302.

26. Morency, L. P., Quattoni, A., and Darrell, T. Latent-dynamic discriminative models for continuous gesture recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on,* IEEE (2007): 1-8.

27. Kollmann, A., Riedl, M., Kastner, P., Schreier, G., and Ludvik, B. Feasibility of a mobile phone–based data service for functional insulin treatment of type 1 diabetes mellitus patients. *Journal of Medical Internet Research 9.5* (2007): 1-9.

28. WiiGee: A Java Based Gesture Recognition Library for the Wii Remote. URL: http://www.wiigee.org/.

29. Consolvo, S., Everitt, K., Smith, I., and Landay, J. A. Design requirements for technologies that encourage physical activity. *In Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM (2006): 457-466.

30. Lange, B., Chang, C. Y., Suma, E., Newman, B., Rizzo, A. S., and Bolas, M. Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE (2011): 1831-1834.

31. DePriest, D., and Barilovits, K. LIVE: Xbox Kinect© s Virtual Realities to Learning Games. In *16th ANNUAL TCC Worldwide Online Conference, Hawa* (2011): 48-54.

32. Staiano, A. E., and Calvert, S. L. The promise of exergames as tools to measure physical health. *Entertainment Computing*, *2*.1 (2011): 7-21.

33. Haskell, W. L., et al. Physical activity and public health: updated recommendation for adults from the American College of Sports Medicine and the American Heart Association. *Medicine and Science in Sports and Exercise* 39.8 (2007): 1423-1434.

34. Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. The action workflow approach to workflow management technology. In *Proceedings of the 1992 ACM Conference on ComputerSsupported Cooperative Work*. ACM (1992): 281-288.

35. Jovanov, E., Milenkovic, A., Otto, C., and De Groen, P. C. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, *2*.1 (2005): 1-6.

36. Tsai, C. C., Lee, G., Raab, F., Norman, G. J., Sohn, T., Griswold, W. G., and Patrick, K. Usability and feasibility of PmEB: A mobile phone application for monitoring real time caloric balance. *Mobile Networks and Applications, 12.2-3* (2007): 173-184.

37. Wii Remote Controller. URL: http://en.wikipedia.org/wiki/Wii_Remote.

38. F-Measure: Information about Calculation Precision, recall and F-measure URL: http://www.creighton.edu/fileadmin/user/HSL/docs/ref/Searching_-_Recall_Precision.pdf

39. 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*, IEEE (2006): 507-513.

40. Chronos Watch. URL: http://processors.wiki.ti.com/index.php/EZ430Chronos?DCMP=Chronos&HQS=Other+OT+chronoswiki
41. Wii Controller. URL: http://mentalfloss.com/article/19178/4-ways-unleash-power-your-wiimote-controller
42. DarwiinRemote Application. URL: http://en.wikipedia.org/wiki/DarwiinRemote
43. Android Application. URL: http://www.your-android.de/ueber-uns/mach-mit/
44. Android Operating System. URL: https://en.wikipedia.org/wiki/Android_(operating_system)
45. Beurer Heart Rate Monitor. URL: http://www.beurer.com/web/en/product/heart_rate_monitors/heart_rate_monitors

VITA

Pratima Gorla was born on November 05, 1988, in Anantapur, Andhra Pradesh, India. She completed her schooling in Hyderabad and graduated from high school in 2006. She then completed her Bachelor's degree in Electronics and Communication Engineering from Jawaharlal Nehru Technological University in Hyderabad in 2010. Upon the completion of her Bachelor's, she was placed in Infosys as a Software Engineer.

In December 2010, Ms. Pratima 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. Pratima worked as an intern at Saepio Technologies. During fall 2012, she rendered her services to Ericsson as a Network Engineer Intern. Upon completion of her requirements for the Master's Program, Ms. Pratima plans to work for Cerner.