A PERVASIVE MIDDLEWARE FOR ACTIVITY RECOGNITION WITH SMARTPHONES

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
PRAKASH REDDY VAKA

B.E, Andhra University College of Engineering, 2011

Kansas City, Missouri
2015

A PERVASIVE MIDDLEWARE FOR ACTIVITY RECOGNITION WITH SMARTPHONES

Prakash Reddy Vaka, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2015

ABSTRACT

Activity Recognition (AR) is an important research topic in pervasive computing. With the rapid increase in the use of pervasive devices, huge sensor data is generated from diverse devices on a daily basis. Analysis of the sensor data is a significant area of research for AR. There are several devices and techniques available for AR, but the increasing number of sensor devices and data demands new approaches for adaptive, lightweight and accurate AR. We propose a new middleware called the Pervasive Middleware for Activity Recognition (PEMAR) to address these problems. We implemented PEMAR on a Big Data platform incorporating machine-learning techniques to make it adaptive and accurate for the AR of sensor data. The middleware is composed of the following: (1) Filtering and Segmentation to detect different activities; (2) A human centered adaptive approach to create accurate personal models, leveraging on the existing impersonal models; (3) An activity library to serve different mobile applications; and (4) Activity Recognition services to accurately perform AR. We evaluated recognition accuracy of PEMAR using a generated dataset (15 activities, 50 subjects) and USC-Human Activity Dataset (12 activities, 14 subjects) and observed a better accuracy for personal trained AR compared to impersonal trained AR. We tested the applicability and adaptivity of PEMAR by using several motion based applications.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled “A Pervasive Middleware for Activity Recognition with Smartphones,” presented by Prakash Reddy Vaka, candidate for the Master of Science degree, and certify that in their opinion, it is worthy of acceptance.

Supervisory Committee

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

Deendayal Dinakarpandian, M.D, Ph.D.
School of Computing and Engineering

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

TABLE OF CONTENTS

ILLUSTRATIONS

Figure Page

TABLES

Table Page

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to Dr. Lee for her continuous support throughout my thesis. Also I would never have been able to finish my dissertation without the guidance of my committee members, help from friends and family. I would like to acknowledge the students of Big Data Analytics & Apps for their support. This work was in part supported by a grant from the University of Missouri Research Board.

CHAPTER 1

1. INTRODUCTION

1.1 Motivation

There is an increase in the use of wearable sensors during the past few years and this increase is predicted to grow seven folds by the year 2019 [1]. These wearable sensors are becoming part of human life; integrated into various accessories and garments that are used in our day-to-day life such as hats, wristbands, socks, shoes, eyeglasses, wristwatches, headphones and smartphones. With the advancements of cloud-computing, many wearable sensor systems can now be easily connected to a cloud service, which makes it easier and cheaper to maintain the data generated from these sensors. There are many applications of wearable sensors in health and wellness monitoring, safety monitoring, home rehabilitation, treatment efficacy assessment and early detection of disorders. Below are some of the factors driving the sensors and wearable markets are the affordability of the sensors, fusion of sensors into consumer-end devices, decreasing size of physiological sensors, increased health and fitness awareness, rise in home and remote patient monitoring, reduced digital health costs, increasing mobile and smartphone penetration into day to day life and increasing patient/physician acceptance.

In coming years, data generated by the sensors is expected to increase by 42% of all data by 2020, up from 11% in 2005. Recognition of human activities from the data generated from wearable sensors is an interesting area of research, but there is a demand for a system/framework that could use the massive data generated from these sensors and also be adaptive in nature for used on mobile platforms to achieve pervasiveness.

1.2 Problem Statement

The growing wearable sensors demand an adaptive approach for pervasive platforms that can support diverse sensors and activity recognition on diverse devices. Some of the factors inhibiting the activity recognition of wearable sensors are reusability and portability. Pervasive approach of activity recognition also requires a framework that could reduce the computation on integrating device/mobile platforms. Activity Recognition system should also support modularity where the activity model generation could be separated from recognition module for reusability and portability. To support adaptive activity recognition and mobile applications, there is a very strong need for adaptive, real-time platforms for mobile devices that can handle diverse stream data generated from various sensing devices. The major obstacles involved in handling real-time stream data on a mobile platform are due to the lack of the capabilities that dynamic activity recognition applications demand, such as support for performance, adaptability, real-time, and applicability. Traditional activity recognition systems either perform activity recognition on the server (offer recognition) or on a mobile platform (online recognition) [2], while there is a need to combine both to address above issues.

CHAPTER 2

2. RELATED WORK

In this section, we introduce activity recognition and discuss some of the existing human activity recognition systems that rely on wearable sensors. Human activity recognition is an important area of computer vision research and applications. The goal of the activity recognition is an automated analysis (or interpretation) of ongoing events and their context from sensor data. Its applications include surveillance systems, patient monitoring systems, and a variety of systems that involve interactions between persons and electronic devices such as human-computer interfaces. Most of these applications require recognition of high-level activities, often composed of multiple simple (or atomic) actions of persons. We discuss different research works and techniques for human activity recognition. We talk about both the methodologies developed for simple individual-level activities and compare different approaches with PEMAR. We describe some of the wearable sensors used for human activity recognition. A variety of devices and techniques are used for gesture and activity recognition, such as wired gloves, cameras and accelerometers.

2.1 Activity Recognition Technology

Human activity recognition has been addressed in two different ways, namely using external sensors and wearable sensors. In the case of external sensors, the devices are fixed in foreordained points of interest, so the deduction of activities entirely depends on the voluntary interaction of the users with the sensors. In the case of wearable sensors, the devices are attached to the user. We discuss each of the approaches in detail in the later sections.

External Sensor (Video Tracking) for Activity Recognition:

Video Signals from cameras are used as input for the recognition of gestures. Hand and body gestures are detected using an edge detection algorithm [3]. These algorithms are computationally expensive and dedicated hardware is required to detect the gestures or body movement. Recent research carried at Cambridge on digits, utilizes a camera for the gesture control offered by Kinect [4]. The camera can be worn on a wrist with the help of a wrist strap and it tracks 2D movement of the hand rather than 3D. It could also recognize finger movement that is used to control the software. This approach can be used in applications such as controlling the games with hand movements and translation of a sign language into a written text.

A company called Leap Motion uses a new type of motion controller for controlling gestures [5]. Leap consist of a simple motion controller, which can be plugged into a USB port to transfer data to the PC. Leap software has to be installed and once the motion controller is plugged in, it turns the 8 cubic feet of air in front of it into 3D interaction space. All the user gestures which are performed in that space are tracked by the leap. The size of the camera is about the size of a business-card holder, and can discern all 10 fingers individually. Such a detail in scanning and recognition allows us to do track actions like pinch-to-zoom, or zero in which can be used in application like maps and to view images in an operation theater.

Wearable Sensors for Activity Recognition:

The major part of the previous work on gesture recognition is based on human detached recognition or computer vision techniques. Computer vision techniques will not the need the user to wear any external sensor, but lighting condition and camera angle/field of vision are important constraints of computer vision based approaches. In cases of poor lighting and object obstructions, it is difficult to recognize gestures using a camera-based system. Additionally, it is not practically possible to have to face the camera all the time.

Accelerometer-based gesture recognition is a technique that is compatible with almost all physical and computing environments. The accelerometer sensors are small enough and are able be to fit in small devices, and communicate wirelessly, hence easily wearable for interacting with a wide range of applications. Examples of some of the accelerometer based gesture recognition devices include Nintendo Wii-mote, Chronos watch from Texas Instruments [6], Fitbit and smart phones. The Nintendo Wii-mote contains an integrated 3-axis acceleration sensor, and connects via the Bluetooth human interface device protocol for transmitting data [7].

The HAR systems are broadly classified as online recognition systems and offline recognition systems. Online recognition systems are pervasive in nature and provide real time response for activity recognition. Offline recognition systems perform HAR on a backend server where the response times are not real time, but are scalable enough for multiple activity recognitions. In this thesis we evaluate on the common HAR wearable sensors, accelerometers. Figure 1 shows a typical accelerometer sensor, typical accelerometer is a device that is used to measure proper acceleration (g-force). Proper acceleration is different from coordinate acceleration; i.e. rate of change of velocity. For example, an accelerometer at rest on the surface of the Earth will measure an acceleration g= 9.81 m/s2 straight upwards. By contrast, accelerometers in free fall orbiting and accelerating due to the gravity of Earth will measure zero.

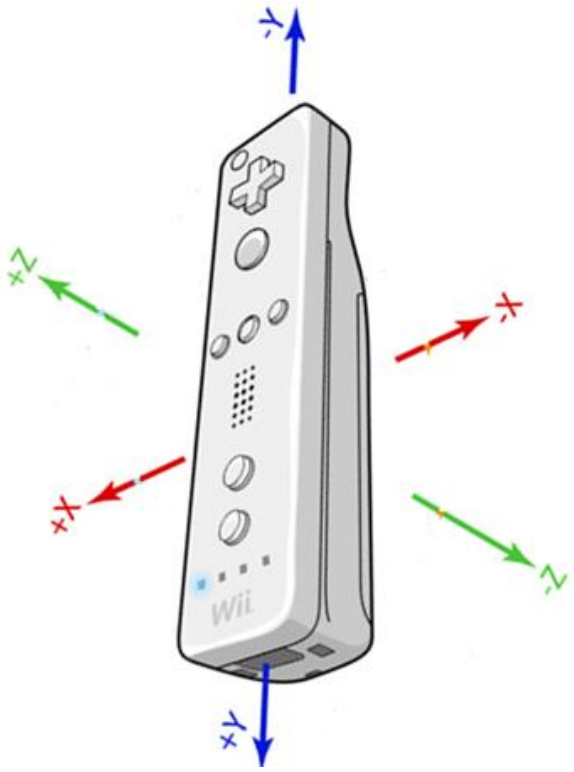

Figure 1: Presentation of Acceleration Experienced by an Accelerometer

Different HAR systems have very different purposes and associated challenges, so they are evaluated separately. General Human activity system dependent on the following factors.

- Different Recognized activities
- Type of sensors and the measured attributes
- Integration device
- Level of obtrusiveness (low, medium, or high).
- Type of data collection protocol
- Level of energy consumption (low, medium, or high).
- Classifier flexibility level, either user-dependent or user independent.
- Feature extraction method(s)
- Learning algorithm(s)
- Overall accuracy for all activities

2.2 Online Recognition

Applications, which use online activity recognition systems, can be more responsive and interactive, compared to offline recognition. In the field of healthcare, monitoring patients' physical or mental pathologies continuously is crucial for their safety, and quick recovery. Similarly, activities and games can enhance user experience in interactive games or simulators. We describe some of the most recognized works on online HAR in detail.

eWatch [8]

eWatch is an online activity recognition system, which embeds a microcontroller and sensors within a device [8]. This can be can be worn as a sport watch. There are four sensors included in the device

- accelerometer
- light sensor
- thermometer
- microphone

The sensors are passive, hence there is no external communication necessary, thus making the eWatch energy efficient. A decision tree C4.5 approach and time-domain feature extraction are extracted for activity recognition. There a total of six ambulant activities and the overall accuracy is 92.5% and less than 70% for activities such as descending and ascending. The detection time (feature extraction and classification) for the eWatch are 0.3 ms, which is very responsive. The data collection was under a supervision of researchers, which makes the dataset not ideal for real case scenario. Figure 2 shows some of the wearable sensors used today.

Figure 2: Wearable Sensors

Vigilante [9]

Vigilante is a mobile application developed for real-time human activity recognition [9] and it uses android platform. A hardware is a chest sensor strap, which can measure acceleration and physiological characteristics. Three ambulation activities with C4.5 decision tree classifier, gave an overall accuracy of 92.6%. The response time of the new gesture classification is 8% of the window length e. Evaluation results were with different users, without the need of training for each user. System uses permanent Bluetooth to communicate between the sensor and the mobile device, which makes it moderately energy efficient.

Tapia et al [10]

Tapia system can recognize 30 activities, which include activities like lifting weight, rowing, push-ups, etc. Evaluation data consists of user dependent and user independent from 21 subjects.

Reported accuracy is at an average is 94.6% for user-dependent analysis and 56% for user-independent. If the activities are not considered based on the intensities, then the user-independent accuracy is reported at 80.6%. The system consists of five accelerometers place on the subject's arm, wrist, hip, ankle and thigh. It also includes a heartrate monitor attached to the chest. Since the sensors communicate wirelessly via Bluetooth protocol, the system consumes high energy. The integration device being a laptop restricts the subject free movement.

ActiServ [10]

ActiServ is an activity recognition service for mobile phones [10]. They make use of only the accelerometer sensor from the mobile phone. Using fuzzy inference system, they classify ambulation and phone activities. ActiServ is energy efficient system and portable, there is no use of external sensor involved. User dependent training accuracy is 90% and user-independent lies in between 71% and 97%. For higher accuracy, user-independent data is processed for days on the mobile platform, for real-time response the accuracy drops to 71%. The drawback from this system, is the orientation of the device should be proper to classify activities. Sitting and standing get confused if the orientation of the device is not proper.

Table 1: Comparing Different Online Gesture Recognition Systems

Online HAR system	**Sensors**	**Learning Algorithm**	**Accuracy**
eWatch [7]	ACC,EMV(wrist)	C4.5,NB	94%
Vigilante [8]	ACC,VS	C4.5	92.6%
Tapia [9]	ACC, HRM	C4.5,NB	86%
ActiServ [10]	ACC	RFIS	71%-98%
Ermes [12]	ACC	DT	94%
Brezmes [11]	ACC	KNN	80%

Other Approaches

Brezmes et al. [11] proposed a system for HAR, featuring a mobile application. They have used the k-nearest neighbors' classifier. This is computationally expensive and is not suitable to scale in smart phones, since it need the complete data set, which can be huge. Besides the system requires each user to provide his own data for better accuracy. Ermes et al. [12] developed an online system for HAR which reaches 94% overall average accuracy. However, the evaluation carried is user-dependent. The data collected is from three users and suggests this is not applicable and adaptive for new users.

2.3 Offline Recognition

Applications, which use offline recognition system, can be less responsive and interactive, compared to online recognition. Application requiring slow response time for HAR use offline systems, such as application that track patients diet habits, and calculating the calories burnt during an exercise routine. We describe some of the most recognized works on offline HAR in detail.

Parkka [13]

Parkka introduced a system that classifies seven activities, such as rowing, riding a bike, standing, walking, running and Nordic walk. Twenty two signals were analyzed. This requires a number of sensors on the individual's chest, wrist, finger, forehead, shoulder, upper back, and armpit. A compact computer is the integration device, which weighs 5kg. Speech recognizer is applied to the speech signal to extract the time- and frequency-domain. This demands a high processing requirement and also privacy related issues reading subject's continuous speech data. The paper reports an accuracy of 86%.

Bao [14]

Bao introduced a system that classifies twenty daily activities and ambulation, such as watching TV, vacuuming and scrubbing. A total of 5 accelerometers were used to collect the

acceleration data, located at subject's arm, knee, ankle and hip. A 5% reduce accuracy is reported with only two accelerometers. Activities are classified with a C4.5 decision tree classifier, the overall accuracy reported was 84%. There is confusion between activities such as moving on elevator or an escalator, adding location data has increased the accuracy significantly up to 95%.

Khan [15]

Khan introduced a system that classifies ambulation, activities and transitions between such as watching TV, vacuuming and scrubbing. A single accelerometer is placed on the subject's chest and the data is transferred to the computer via Bluetooth protocol. The sampling rate of the accelerometer is 20Hz. An artificial neural network is used for classification, and an accuracy of 97.76% is reported of user independent activities in the paper.

Zhu [16]

Zhu and Sheng [16] proposed a system, which used Hidden Markov Models (HMM) to recognize activities. Two accelerometers are used to collect data, which are placed on user's wrist and waist. The raw data is sent to a computer via Bluetooth protocol to process the data. This device placement and configuration makes it uncomfortable, since the subject has to wear wired links, which can interfere with his movements. The data is collected from a single subject, hence the evaluation and performance is not comparable with other systems.

Table 2: Comparing Different Offline Gesture Recognition Systems

Offline HAR system	**Sensors**	**Learning Algorithm**	**Features**	**Accuracy**
Parkka [13]	ACC,ENV	KNN	TD,FD	86%
Bao [14]	ACC (wrist, ankle, thigh, elbow, hip)	C4.5,KNN,NB	TD,FD	84%
Khan [15]	ACC (chest)	C4.5,NB	AR,SMA,TA,LDA [3]	97.9%
Zhu [16]	ACC (waist, wrist)	HMM	AV,3DD	90%

2.4 Machine Learning Algorithms used for Activity Recognition

In this section, we discuss some of the machine learning approaches for activity recognition used in the related works. Some of the approaches are lightweight to be used on an online recognition system for real time response, where some are heavy and cannot be run an online platform.

Hidden Markov Model

A hidden Markov model [17] is one in which you do not know the sequence of states the model went through to generate the emissions, but you observe a sequence of emissions. Hidden Markov Model is used in many fields, such as Pattern recognition, Gesture Recognition and Speech Recognition. HMMs are stochastic models, which are used for data that is serial or temporal. The word "hidden" in the Hidden Markov Model refers to the hidden states that are mapped to the data. HMM model is typically used for modeling sequences of events. HMM model is particularly useful when the data is noisy and incomplete. This is based on efficient algorithms for learning and recognition such as Baum-Welch and Viterbi algorithms and estimating probability distributions

Support Vector Machine

This technique is also used extensively for gesture recognition [17]. It is a supervised linear classification method with a property of maximizing margins between classes. It also has nonlinear extensions with an appropriate choice of kernel functions. An SVM model represents the examples as points in space, mapped such that the examples of the different categories will have a clear gap between them, which is as wide as possible. Represent new examples/data, on the map in the same space and predict on which side of the gap they fall on. Such a representation of new data could classify new gesture data. Z-normalization is used to make all the dimensions equal. During the recognition phase, the incoming movement data is transformed and normalized. This is similar to the training samples before SVM classifies the gesture.

Decision Tree Algorithm

A decision tree [18] is a flowchart-like structure in which each internal node represents a "test" on an each branch represents the outcome of the test and each leaf node represents a class label. The paths from root to leaf represents classification rules. In these decision tree structures, each leave represents a class label. Branches represent conjunctions of features that lead to those class labels. Decision tree approach is used for activity recognition with high accuracy [18]. Decision tree algorithm is less computational expensive compared to others such as SVM, Naïve Bayes and HMM. Extracted features from the accelerometer data are used to build the decision tree, which can then classify new acceleration data.

Random Forest Algorithm

Random Forests [19] grows many single classification trees, similar to a tree in a forest. Creating of Individual trees is similar to the decision tree approach. Each tree in the random forest will extend to the largest possible limit and without pruning. Building of individual tree takes place with sampling from the original training data at random, but with replacement. To classify any new data (input vector), the vector will pass down each free in the forest. Each tree gives the classification, and votes “yes” for the class. The forest chooses the result or classification of the input vector by picking the most voted class. Random forest is used in application of activity recognition to achieve high accuracy detection.

Naïve Bayes

Naive Bayes classifiers is a set of simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between the features. Naive Bayes classifiers algorithms are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem.

Table 3 shows some of the features used by several related works in the field of human activity recognition. These features are selected depending on the related algorithm being implemented and the computational complexity for each of the technique.

Table 3: Features for Activity Recognition

Group	Features
Time Domain	Mean, standard deviation, variance, interquartile range (IQR), mean absolute deviation (MAD), correlation between axes, entropy, and kurtosis
Frequency Domain	Fourier Transform (FT) and Discrete Cosine Transform (DCT)
Others	Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Autoregressive Model (AR), and HAAR filters

2.5 Accelerometer Devices and Applications

2.5.1 Chronos Watch

eZ430-Chronos is a wearable wireless development system[6]. It can be used as a wirelsess sensor to receive and collect data. It consists of a three-axis accelerometer that can detect the movement of the device in 3 dimensional space.

Acceleration mode – RF

This mode on the chronos watch would activate the accelerometers on the watch and would stream the data. Collection of the stream data is possible with the Chronos control center PC software. The take the watch to acceleration mode, “#” key on the on the watch has to be long pressed until ACC" is shown on the LCD. Once the watch is “Acc” mode a continuous 3D acceleration data is transmitted to the PC using TI's SimpliciTI protocol stack.

eZ430-Chronos RF Access Point

RF access point is a hardware component which could communicate with the Chronos watch wirelessly using RF and also to receive data from it. The Control center can read data from the chronos watch only if the RF access point is plugged into the PC when the transmission is taking place.

Transmission of Acceleration Data

Acceleration data, which is being generated on the watch when it is ACC mode, can be seen in the PC (Control Center) when the RF access point is connected. A “start access point button” is available in this interface, the data from the watch is collected to the PC and 3D acceleration values are transmitted.

The Chronos Flying Mouse is an application developed in integration with chronos watch, which could mimic the behavior of an actual mouse connected to a PC [6]. It is designed to be highly accurate and customizable depending on user requirements. Since the device is wireless and can be customized to multiple inputs to mimic the mouse input, this has many application to control power point presentations or play interactive games. The primary mode of the application allows user to control the mouse movement. We have used chronos flying mouse to collect data for analysis.

This flying mouse application has many joystick modes that allow gaming using a wide variety of PC games. This joystick mode allows any existing video game to use chronos without special programs and allows the Chronos to interact with a wide variety of applications.

2.5.2 Wired Gloves

Wired gloves or data gloves is an input device used for human-computer interaction. These devices are worn like normal gloves and consist of various sensors and technologies which can detect delicate movements. These are used to detect the 3D orientation of the hand and also the movement of the fingers.

There are multiple application for such devices, these are used in areas such as 3D simulations, virtual reality interactions, multimedia education, help physically challenged people and even in for rehabilitation programs to train people.

Electronic transducers and strain gauge sensing devices determine the position and orientation of the hand wearing the data glove. The wired gloves also contain small lightweight sensors, which can connect using a connection port.

A 3 space and 3 dimensional position and orientation sensor tracks and records the hand gestures [22]. A motion tracker available in the gloves, such as a magnetic tracking device tracks data rotation and position of the glove. Software that comes with device handles the recognition of the device movement and orientation. Sign language or other symbolic functions can be categorized using the useful information derived from the software. Data glove is a part of hap-tic science. Hap-tic science is the field of science, which is concerned with applying tactile sensation to human interaction with computers [23]. A wired glove can act as a hap-tic device because it simulates physical contact between the human and computer. The cost of wired gloves is huge, with the tracking device and the finger bend sensors have to be bought separately.

2.5.3 Sensor Tag

Sensor tag is a sensor device developed by Texas Instruments and which has a collection of inbuilt sensors. Below are some of the sensors that are available in the sensor tag.

- KXTJ9 accelerometer
- MAG3110 magnetometer
- IMU-3000 MEMS gyroscope
- C953H barometric pressure sensor
- TMP006temperature sensor
- SHT21 digital humidity sensor

The device has a Bluetooth low energy protocol to transmit the data to any computing device, which has Bluetooth low energy compatibility. Since the device can communicate over wirelessly, it has many applications as that of sensor tag and even more with the availability of other sensors. This device reduced the development time of Bluetooth enabled application in mobile platforms from months to hours.

Sensor Tag Android Application

Texas instruments has released an android application (Simplelink SensorTag), which can connect to the sensor tag using bluetooth low energy protocol and read data from multiple sensors on the device at the same time. The data read from the device is customizable; i.e the application has an option to set up the frequency of the receiving data. With the increase in the frequency of the data, the power consumption of the device increases.

The sensor, which was used, for this thesis work is the 3-axis accelerometer; the frequency of the accelerometer is customizable. The highest frequency of acceleration data received using the sensor tag is 10Hz.

Table 4: Comparing Different Platforms

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

CHAPTER 3

3. MIDDLEWARE FOR GESTURE RECOGNITION

3.1 Overview

This thesis work presents a middleware, called the Pervasive Middleware for Activity Recognition (PEMAR) that aims to turn the above challenges into an opportunity for increasing the adaptability and applicability of activity recognition with a middleware for active games/applications on mobile computing devices. The PEMAR middleware is based on two-layer architecture: Activity Modeling layer and Activity Recognition layer. In this work, we have proposed an intelligent and adaptive model for recognizing gestures when motions are involved with the orientation, speed, and resulting accuracy that are significantly dependent on the complexity of gesture models, overlapping gesture models, and variations in gestures between individuals.

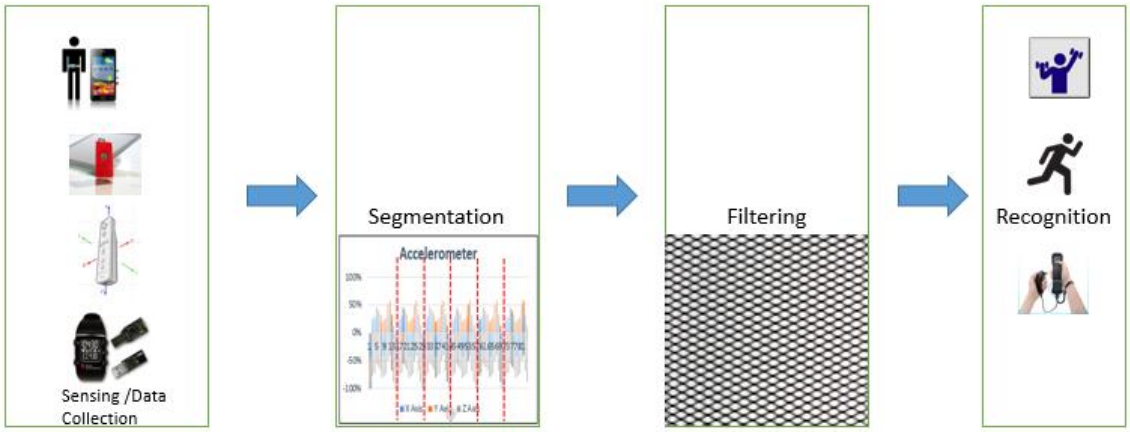

Figure 3: Flow Diagram of the Recognition System

Figure 3 shows the steps involved in human activity recognition with accelerometer data. The first step includes the data collection, where the data is collected from different sensing devices via data transfer protocols. These data transfer protocols include Bluetooth low energy, ZigBee and Wireless networks IEEE 802.11. The second stage is the data processing layer, where the data is segmented and filtered depending of the configuration for each of the device and activity type. The

third module is the human activity recognition module where the sensor data is labelled into one of the existing activity classes. We will discuss each of the above modules in detail in the later sections.

3.2 Data Collection

With the wide range of wearable sensors, the data collected from these devices need to be transferred to an integrating platform. Different wearable sensors use different network protocols to communicate with the wearables sensors. Table 5 shows some of the existing data transfer protocols and wearable sensors.

Table 5: Data Collection Device and Network Protocols

Device	Network Protocol	Maximum Frequency	Energy Consumption
Chronos Watch	Radio Frequency	915MHz	<35 mA
Sensor Tag	Bluetooth Low Energy	2.4 GHz	<15mA
Wii Remote	Bluetooth Classic	2.4 GHz	<30mA

Each of the above protocols has a maximum limit for data transfer that can be achieved using each device. The energy consumption of each of the devices also depends on the operating configuration of the sensor, mainly the frequency of the operating accelerometer sensor and the network protocol. Figure 4 shows the different devices discusses above.

Figure 4: Different Wearable Accelerometer Sensors

In this thesis section, we tested the PEMAR framework with Sensor tag and Android Smart Phone sensor, we found that the frequency on both the devices is different and the signature for the same activity is different on each device. The frequency on the Sensor Tag is found to be 10Hz and the one on the Android Smart Phone gave close to 200Hz, when the data was read not in a UI thread in the smart phone. To test the accuracy of the application, we used a benchmark dataset (Human Activity Dataset) released by University of Southern California [17]. The data collected for the benchmark dataset is from a Motion Node inertial sensing device with a frequency of 100Hz. Magnitude of the accelerometer sensor used in the benchmark data is +-6g. Figure 5 shows the graphical representation of the accelerometer data from the benchmark dataset. In the next section, we discuss about the data processing part of activity recognition.

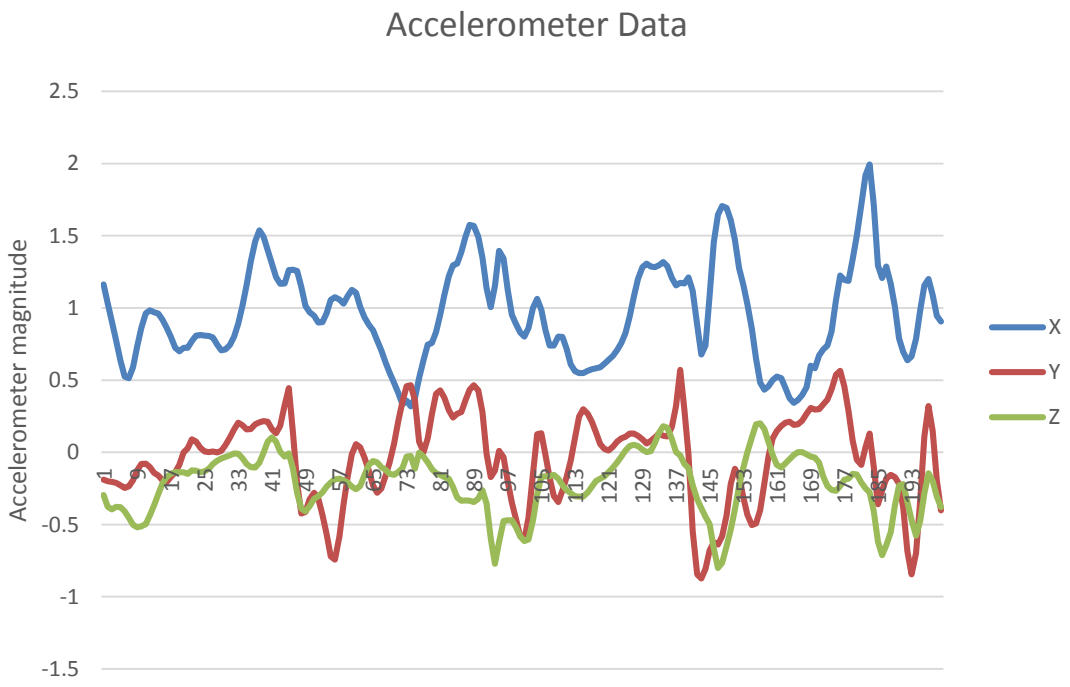

Figure 5: Graphical Representation of Raw Data from an Accelerometer

3.3 Data Processing

The human activities can be represented as a set of vectors data generated from the sensors device. Different activities have an individual footprint unique to each other. But the data generated or collected during the data collection phase from the inertial accelerometer sensors tend to have noise associated with it. Hence, we need a data processing module which would remove the noise from the data and prepare the data for any activity recognition approach. We perform filtering and segmentation on the sensor data in the processing phase. In the next sections we discuss in detail each of the steps involved. For data processing we categorize activities broadly into three groups (i) Short activities, (ii) Long activities and (iii) Short gestures. Short activities are performed for a short duration and tend to have an end of activity after a short period. Long activities are carried out for a long duration of time and do not have an immediate end. Short gestures are used during an interactive game. Table 6 shows examples for each of them.

Table 6: Different Activities Classification

Activity Type	**Activities**
Short Activities	Push Up, Pull Up, Crunch, Jumping, Stomp, Dumbbell Curl, Toe touches
Long Activities	Running, Walking, Cycling, Sleeping, Walking Down Stairs, Walking Up Stairs, Elevator Ride
Short Gestures	Circle, Square, Triangle, Z, Move left , Move Right, Move Up, Move Down, Move forward, Move backward

Segmentation

The activity vector generated for each of the above categories of activities is to be detected with accuracy. The body position and acceleration of players are expressed in X, Y, and Z coordinates in motion space. The X, Y, and Z axes are the body axes of the accelerometer**.** The start point for an activity and end of a particular activity has to be determined depending on the activity category. We propose an activity aware segmentation framework, which handles the segmentation depending on the activity category.

Below are the three segmentation techniques supported by PEMAR. We will discuss each of the below techniques in details.

(i) Dynamic Segmentation

(ii) Windows-based Segmentation

(iii) User triggered Segmentation

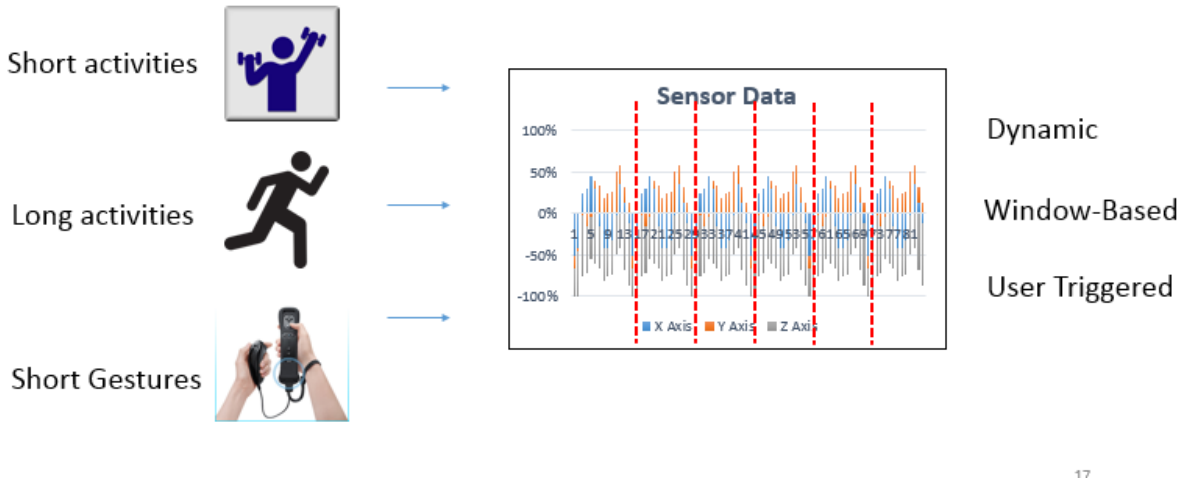

Figure 6: Different Activities and Segmentation Techniques

Dynamic Segmentation

This segmentation is used for recognizing the start and end of short activities, where the system automatically detects the activity vector. In the case of an accelerometer sensor, the start and end are determined by the change in the accelerometer values. The start is determined if the change in acceleration value from the previous instance is greater than an already determined threshold depending on the sensor details. Similarly, the end of each gestures is calculated if the change in acceleration vector from the previous vectors is less than an already set threshold.

$$d = \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2}$$

x, y, z – acceleration data at time t

x_1, y_1, z_1– acceleration data at time $t + \alpha$ (next instance)

The start of gesture is triggered when $d > \Delta$ and the end is determined when $d < \mu$. These values of μ and Δ are constants and need to be configured depending on the sensors settings. Based on the heuristic approach we found the best set of μ and Δ to be 0.2 and 0.6 respectively. Figure 7 shows the dynamic segmentation of accelerometer data.

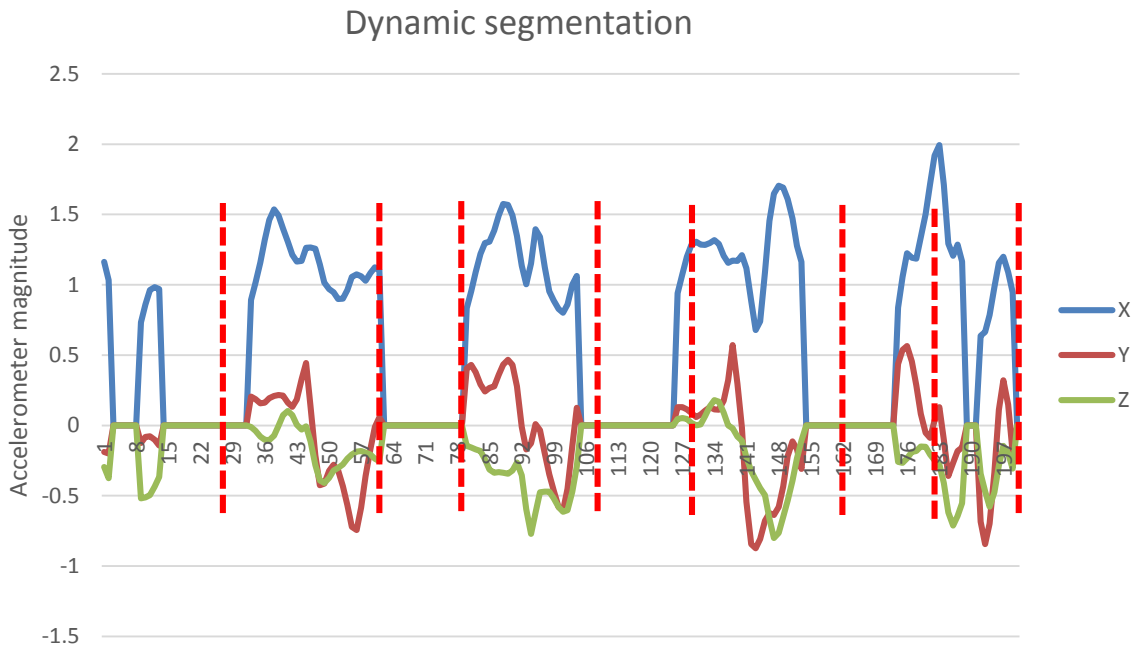

Figure 7: Dynamic Segmentation for Short Activities

Windows-based Segmentation

The windows based segmentation is used in scenario where the start and end of any activity cannot be determined, since the activity is long running. Window-based segmentation uses a time frame based approach to segment the sensor data. The generated sensor data is segmented using a period of t seconds and considering an overlap of 50%. The overlap is considered to take into account the transition data and not to lose any sensor data for activity recognition.

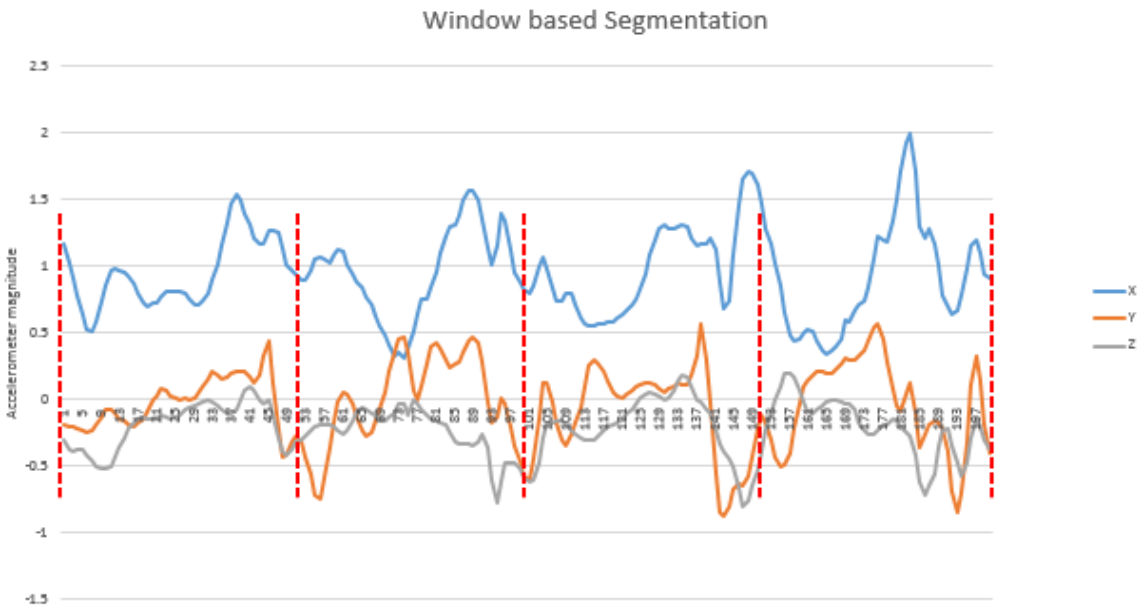

Figure 8: Raw Data Windows based Segmentation

The window size t is customizable depending on the sensors configuration. Windows based segmentation is best suitable for long running activities. The response time for activity recognition using this approach is not real time and fails to detect short activities. The t value of 2 seconds is determined to give good accuracy for ambulation. Figure 8 shows the windows-based segmentation of the accelerometer data.

User Triggered Segmentation

The segmentation using this approach gives the best results, since the start and end of each gesture or user, mostly using an input trigger such as a button, which triggers the segmentation. This approach can be used for detecting short gestures, since the user performs these gestures during an interactive system, which generally asks for user to specify the start and end of the gesture. Activity recognition using this segmentation is carried out in real time, since the user triggers the end of a gesture and the recognition kicks off immediately after the user trigger.

Short activities require the response time of recognition to be short, which allows the short activities or gestures to be used to interactively use the applications. The response time for an interactive system is considered instantaneous if it response time is less than 0.1 sec. Such quick responses are not achieved using an offline approach, hence online approach are required. Table 7 summarizes different segmentation techniques and the activities suitable for each of them.

Table 7: Segmentation Techniques are Suitable Activities

Segmentation Technique	Suitable Activity Group
Dynamic Segmentation	Short Activities (Push Up, Pull Up, Crunch, Jumping, Stomp, Dumbbell Curl, Toe touches)
Windows-based Segmentation	Long Activities (Running, Walking, Cycling, Sleeping, Walking Down Stairs, Walking Up Stairs, Elevator Ride)
User Triggered Segmentation	Short Activities/Gestures (Circle, Square, Triangle, Z, Move left , Move Right, Move Up, Move Down, Move forward, Move backward)

Filtering

The sensor data collected for activity recognition contains a lot of noise, because of the nature of the wearable sensors are built. Hence, we have to clean the sensors data before processing it. Each gesture has some essential components that constitute the signature of an activity, but also include actions at the beginning and end of the gesture that confound its interpretation. We implement a sensors aware filtering approach, where different filtering techniques are implemented and can be used to clean the data as follows:

(i) Low pass filter removes fast and sudden gestures (Very High Jerk)

(ii) High pass filter removes short and low magnitude sensor data

(iii) Directional Equivalence Filter

Low Pass Filter

This filter removes noise sensor data that is very high in magnitude that is caused by any sudden jerks in the sensor movement. It is used in scenarios where the set of gestures/activities being recognized and the sensor configuration never produce any high magnitude jerk data. Figure 9 shows the raw accelerometer data. Figure 10 shows the result after the low pass filter is applied the raw data.

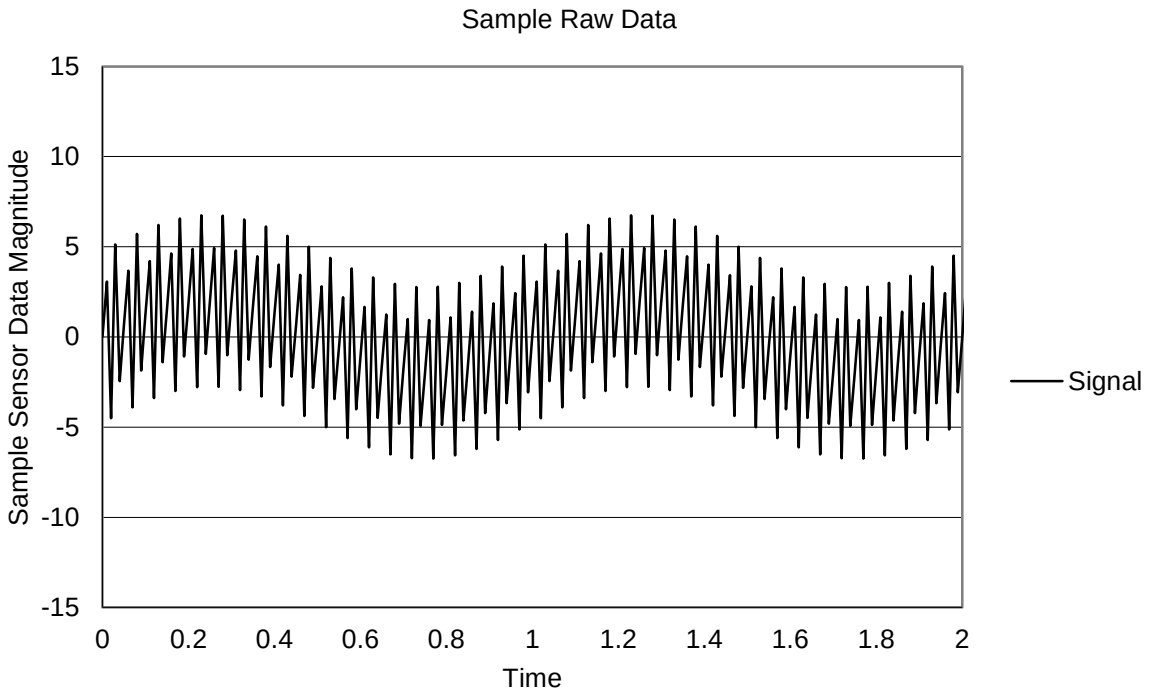

Figure 9: Sample Raw Data from Sensor Devices

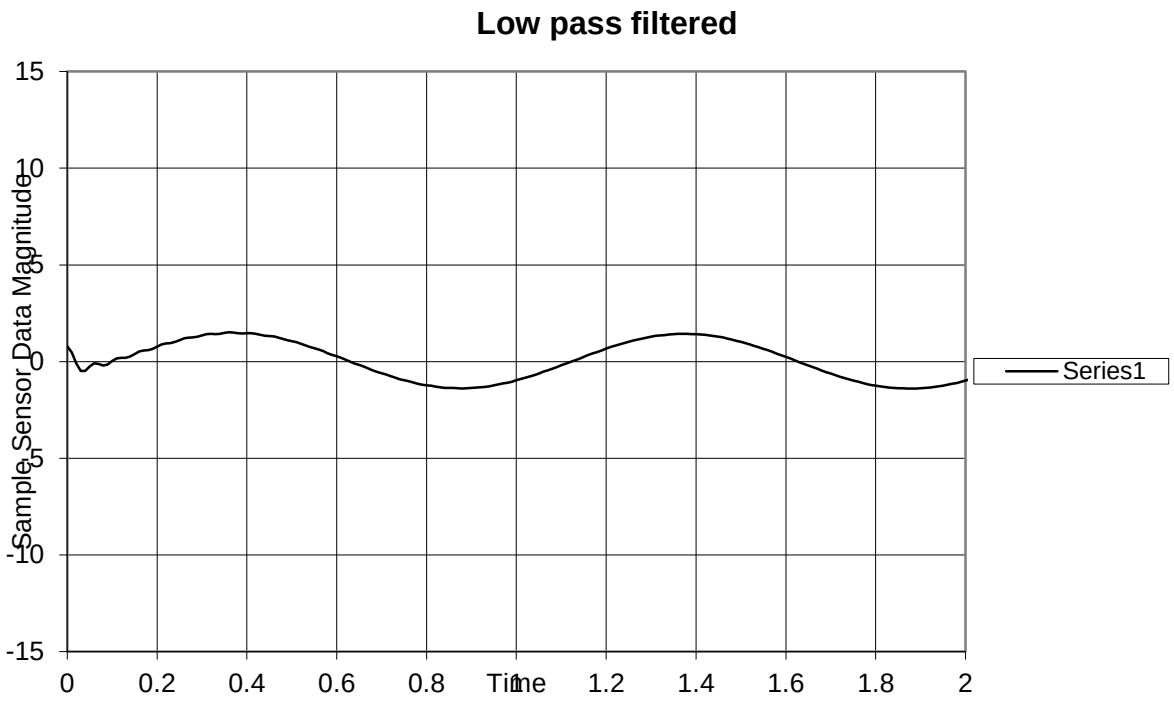

Figure 10: Low Pass Filter Applied on Raw Data

High Pass Filter

High pass filter is used to attenuate sensor data that is not significant and not contributing to the footprint of the activity. We implement a high pass filter by keeping track of the magnitude of sensor data and if magnitude of the sensor data $|\vec{a}| < \alpha$, we attenuate these sensor readings. Figure 11 shows the raw accelerometer sensor data. Figure 12 shows the high pass filter output.

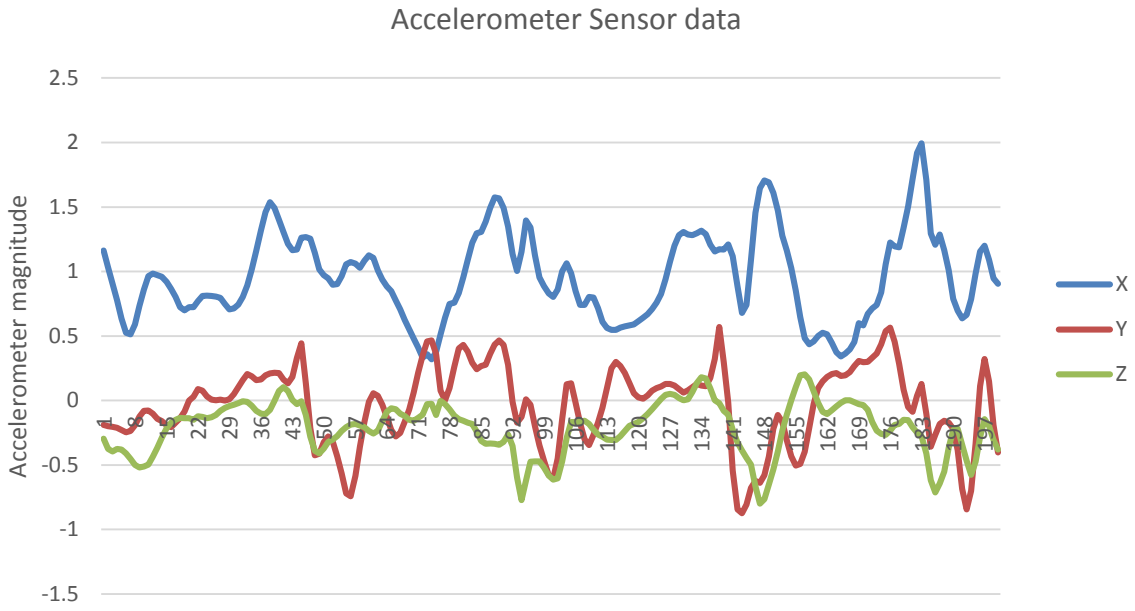

Figure 11: Raw Sensor Data

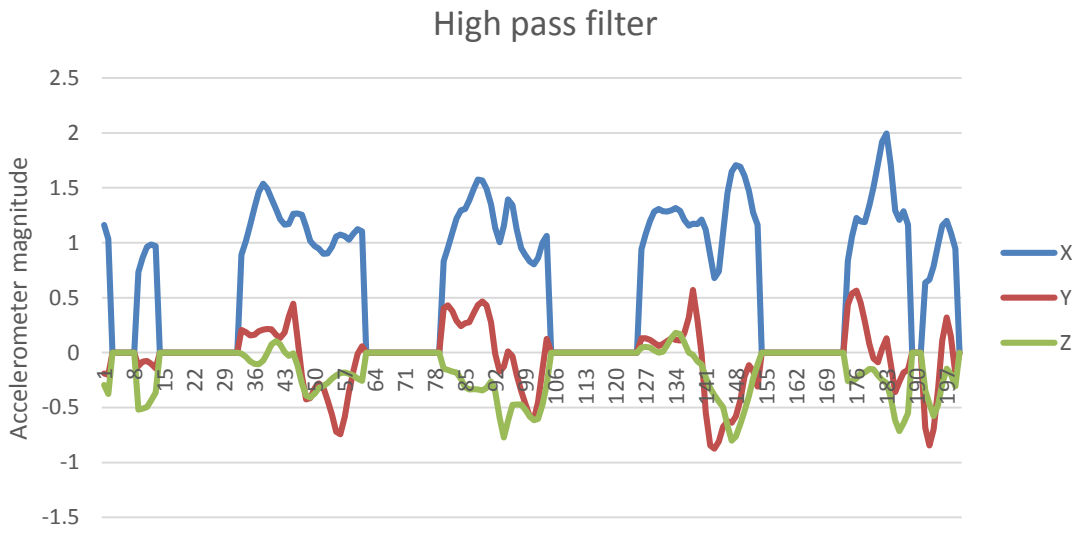

Figure 12: Sensor Data after High Pass Filter

Dynamic Equivalence Filter

Directional equivalence filter is used to reduce redundant sensor data from the activity recognition flow; this would reduce the computation required for activity recognition on pervasive devices. Figure 13 shows the accelerometer data output from the high pass filter. Figure 14 shows the directional equivalence filter result.

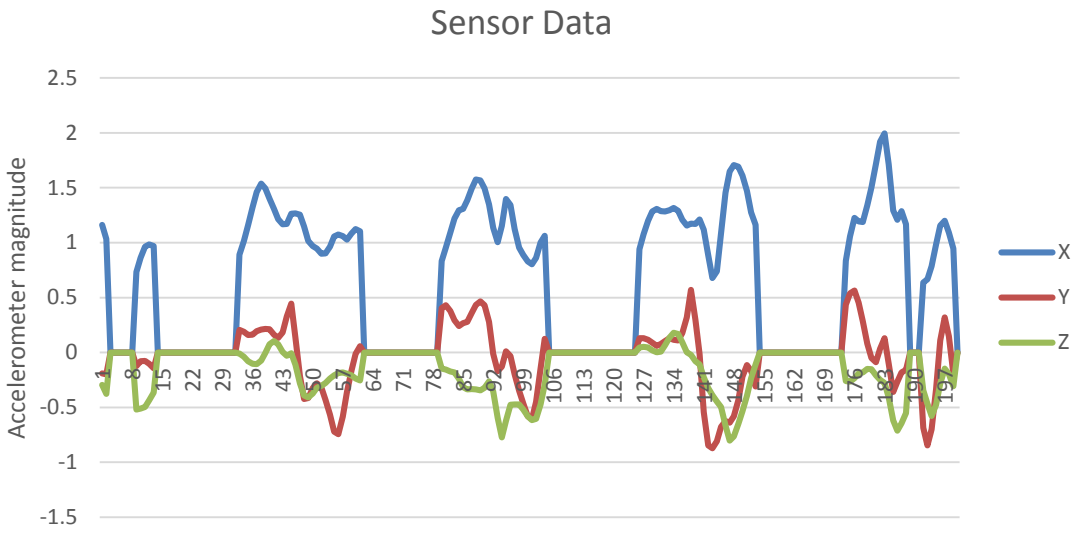

Figure 13: Sensor Data Input for Directional Equivalence Filter

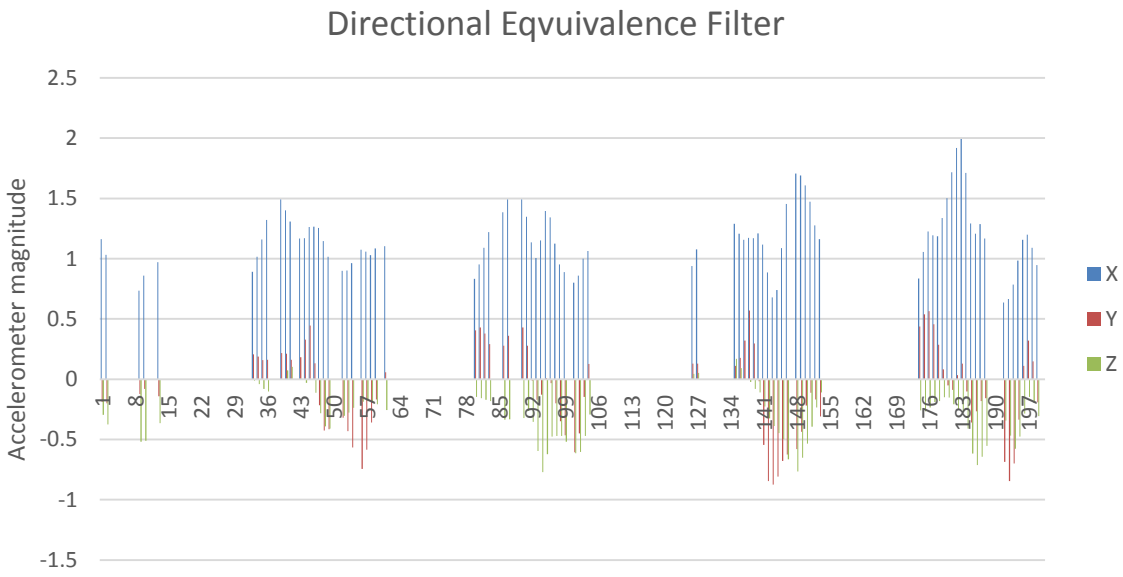

Figure 14: Sensor Data Output from Directional Equivalence Filter

3.4 Model Generation and Training

Model Generation

Human activity recognition, similar to other machine learning approaches contain two stages, training and testing. Measured attributes from users activity data is used for training stage. This requires a dataset of user performing certain activity. Using one of the above mentioned segmentation in section 3.3, time series data is split into windows to apply feature extractions. After this required features are filtered from the raw sensor data. With the extracted feature data learning techniques are used to generate an gestures/activity model. Similarly, for recognizing activities, data is collected using a segmentation technique, and features are extracted from the window. Extracted feature set is evaluated with the prior trained activity/gestures model to generate a label for the tested gesture.

PEMAR framework provides a plug and play feature for the above model generation flow. The model generation using training data is separated with the testing/evaluation modules, does providing portability and reducing the training data generation time for each user. It uses an offline approach for feature extraction and model generation. The sensor data generated from the user is segmented and filtered on a pervasive platform (Smart Phone), this processed data is transferred over the network to offline model generation system. Offline system would then extract the features from the segmented data and use this for generating a model for that particular activity. These models are updated to a database (HBase). PEMAR supports multiple machine learning approaches for activity recognition. As discussed before the data is diverse and there is need of an adaptive framework to support such a change and the sensor data is expected to grow with the increase of wearable sensors. Hence, to address both the above issues we implemented PEMAR on a big data platform, by using the distributed computing paradigm Map Reduce.

For the evaluation of PEMAR we have only considered some of the time domain features, which will be explained in detail in the implementation of PEMAR. PEMAR model generation and training was carried out using Hidden Markov Model (HMM) algorithm, but the evaluation was carried on with other algorithms considering some of the time domain features. The sequence of the vectors generated for an activity is split into different states for the HMM algorithm. And the transition of the acceleration vector between the states is observed with a training data for each of the activities. These trained models with the training data are saved to database (Activity Library).

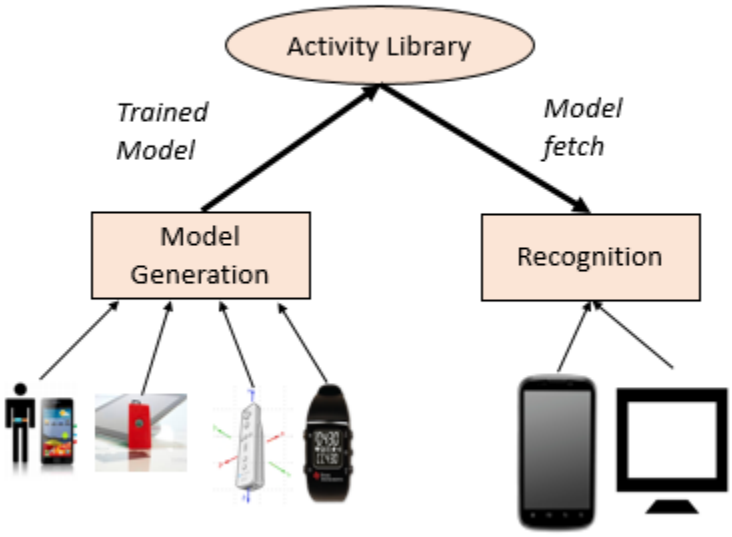

Figure 15: PEMAR Model Handling

PEMAR also introduce an approach for model updating using user personal data. The user first fetches a user independent trained activity model from the database, and as the user continues to use the application, PEMAR collects the user personal activity data. Collected user personal data is used to generated personal activity model. This personal activity model is downloaded the next time the user tried to use the recognition application. The frequency of data from each device is different

and the acceleration magnitude range, hence a separate activity model is generated for each of these devices in PEMAR. The frequency reported in the one received on an integrating device/application such as smartphone or desktop.

3.5 Activity Recognition

The proposed activity recognition model is composed of four major steps for recognition:

(i) Segmentation of activities based on the type of activity

(ii) Filtering of data to remove noise

(iii) Feature Extraction for machine learning approaches and model training

(iv) Activity Labelling for activity recognition

PEMAR framework provides an activity library that contains trained activity models. These models can be fetched onto a pervasive platform where the recognition is carried out directly using the already trained activity models. This approach reduces the time for model generation on a mobile platform and provides the reusability of the activity model and also adaptive in nature for new activity recognitions.

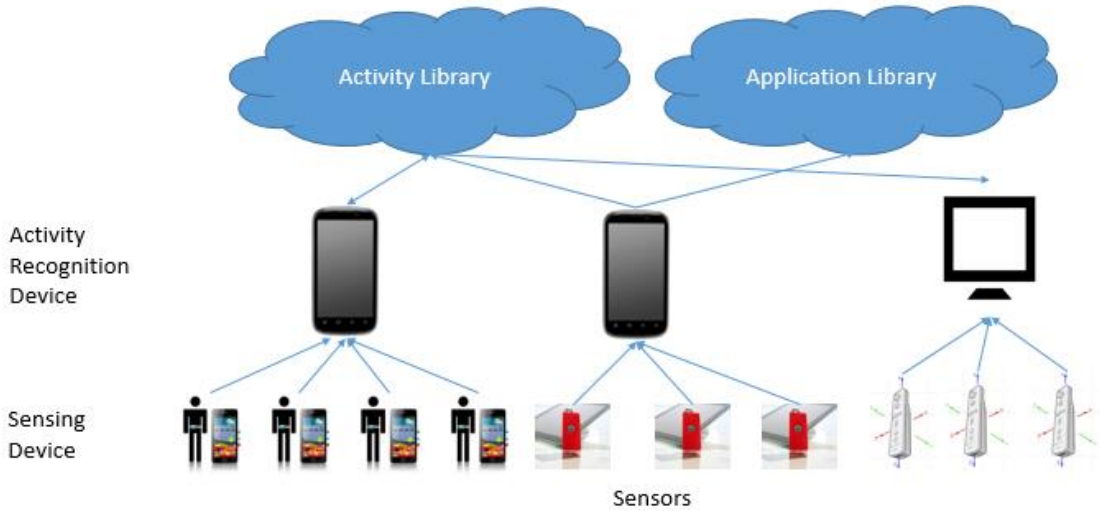

Figure 16: PEMAR Gesture Recognition Framework

PEMAR provides an activity library and an application library; these can be used to detect different activities on same/different applications. The training models generated are with different machine learning applications; the recognition can use different machine learning model and the respective features for the recognition. PEMAR provides modularity for different machine learning applications for activity recognition. Figure 16 shows different sensing devices and integrating device collecting data from them and using the activity library for human activity recognition.

For the Active Mobile layer, motion based games that involve the whole body, especially hand and leg movements captured by diverse devices such as SensorTag, TI Chronos Watch, remote Wii systems, and smart phones can be downloaded from the activity library, and then they can be dynamically configured with the users' customized gestures. In this middleware, gesture models developed and stored in the activity library were utilized by different gaming applications.

Current implementation of PEMAR on mobile platform doesn't involve any feature extraction techniques, to keep the activity recognition as light weight as possible. We implemented mobile recognition of PEMAR using a lightweight version of JAVA HMM library. The communication of the sensing device with the mobile device is carried over Bluetooth Low Energy and the data is then transferred to the offline system over the network.

CHAPTER 4

4. IMPLEMENTATION AND APPLICATIONS

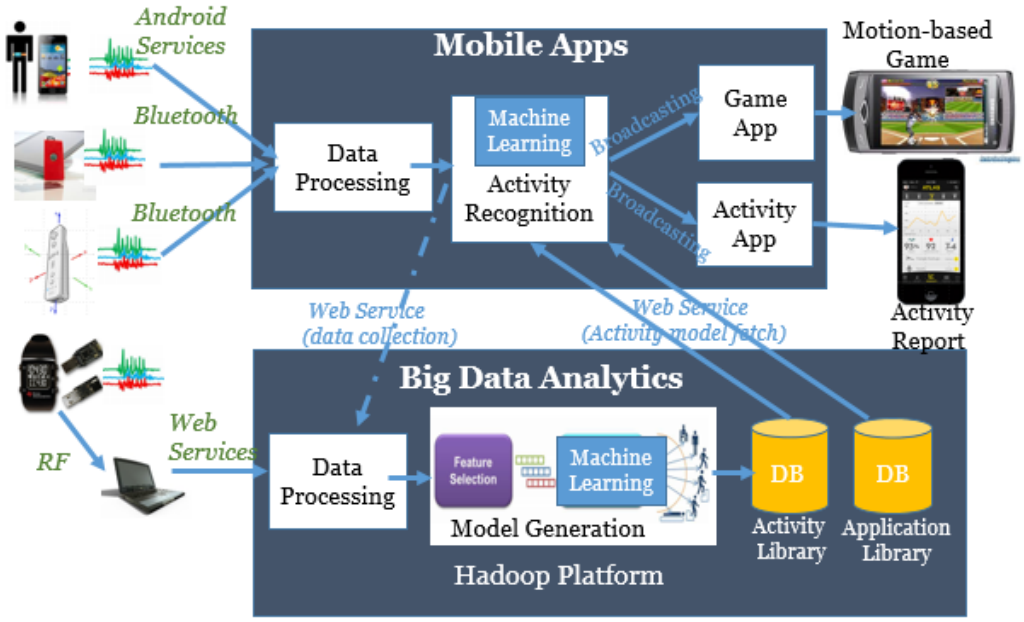

Figure 17: PEMAR Framework

4.1 PEMAR Architecture

Previous chapters focus on explaining the architecture and different methodologies involved. In this section we discuss how the PEMAR system has been implemented. In this section we discuss the implementation of individual module in detail in PEMAR. For human actviity recognitinon subjects' can use an exisiting recognition modal to start their activity recognition, they can also traing their own specific gesture modal for recognition. We have written custom program job to run in batch that can generate user specific modals for specific activities taking into consideration of device.

PEMAR implementation of data collection was done for two sensing devices in detail, the Sensor Tag from Texas Instruments and the Android inbuilt sensor. There are multiple sensors available in the Sensor Tag, but we have only activated and used the accelerometer sensors of the device. Each sensor on the Sensor Tag has a unique UUID assosciated with it. Below are the UUIDs of the accelerometer sensor in the Sensor Tag.

1. Accelerometer Service : f000aa10-0451-4000-b000-000000000000
2. Data Char : f000aa11-0451-4000-b000-000000000000
3. Config Char : f000aa12-0451-4000-b000-000000000000
4. Period Char : f000aa13-0451-4000-b000-000000000000

The config char UUID of the accelerometer is used to set the frquency of the data for the accelerometer. We have seen a maximum of 10Hz with the version of Sensor Tag (CC2541) we used. Since the Sensor Tag operated with Bluetooth Low energy protocol for communication of the device also depends on the integrating device to support BLE. Android version 4.3 and above supports BLE and hence the Sensor Tag as well. We observed that the version of BLE API on Android 4.3 is not stable enough to give reliable commincation every time. Since we wanted the data to be collected in the backend for activity recognition, we made a service class to interact with the sensing device. This service class has no UI thread assisicated hence the data read is the highest possible for communcation.

We also used the android smart phone built-in accelerometer sensor, the API is availabe from Google Android to use the built-in sensor. We observed that the built-in android sensor has better frequency than the Sensor Tag, hence the footprint of a particular human activity could be tracked precisely. We also save the data collected from the sensing device to a file which will be transffered over the network to offline big data platform.

Once the data is collected from the sensing device we use one of the segmentation and filtering technique to remove noise from the data and do the pre processing. The segmentation technique is decided based on the activity type being recognized. We implemented different apporaches for sigmentations and made code snippets available for use on mobile platform for each of them.

Code Snipper for Dynamic Segmentation

```
/*** Start*****/
d= Math.sqrt( Math.pow((x-x1),2 )  + Math.pow((y-y1),2 ) + Math.pow((z-z1),2 ));
     if(d>=0.3 && !trigger){
            Log.i("start","start");
            trigger=true;
     }else if(d<=0.1 && trigger){
            Log.i("end", "end");
            trigger=false;
            try {
/*** End*****/
```

The collected train data for the first time is used to train an activity model, we used HMM for model generation building and recognition.

Code snippet for the Model Generation.

Map function

Map function mark all the data for each user and activity respectively, the format of the key for the map input would be <userId_activity Id>.

```
public void map(LongWritable key, Text value, Context context
                ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString(),","); // line to string token


    word.set(itr.nextToken());
        if(itr.hasMoreTokens())
    values.set(itr.nextToken()+",");
```

```
    context.write(word,values);
   }
```

Reduce function

Reduce function input is the list of all the gestures which are performed by that user in the data. Since the key is <userId_ActivityId> all the gestures with this key will be returned to the same reduce function. HMM alogirthm is used to train the modals for each of the activities performed by individual subject. Existing java based HMM library JAHMM is used for the implementation.

```
public void reduce(Text key, Iterable<Text> values,Context context
                ) throws IOException, InterruptedException {
          ArrayList<String> punchdata=new ArrayList<String>();
          int sum =0;
          //get the count of seq data
     for (Text val : values) {
            sum+=1;
            punchdata.add(val.toString().trim());
     }
  reducerOutValue = HMMTraining(punchdata,key.toString());
   //HMM coding
   context.write(key, reducerOutValue); // create a pair <keyword, number of occurences>
 }
```

Algorithm HMM MapReduce Training Algorithm

Input: String P which contains <Gesture name G + "token" + sequence S>

Output: Gesture name G and sequence concatenate C Map M2<G, C>

```
1.  map
2.      define Gesture and sequence Map M1<G,S>
3.              Iteration<- P StringTokenizer with "token"
4.              if Iteration hasMoreTokens
5.                 M1 save pair (G, S)
6.       end if
7.  end map
8.  reduce
9.    for each e ∈ M1
10.        var a = M1 get value with e
11.             if M2 has key e
12.           var b = M2 get value with e
13.           M2 save pair (e, a+":"+b)
14.      end if
15.      else
16.       M2 save pair (e, a)
17.     end else
18.   end for
19.   HMMTraining (M2)
20.  end reduce
```

Sample Accelerometer Data

Key : <user + walking + nexus7>

Data:

1.16278	-0.18929	-0.29578
1.032296	-0.20014	-0.3763
0.905245	-0.20376	-0.3946
0.778195	-0.211	-0.3763
0.633975	-0.2291	-0.37996
0.524094	-0.24719	-0.40925
0.513792	-0.23634	-0.45317
0.589336	-0.19291	-0.50075
0.733555	-0.12414	-0.51905
0.860606	-0.08071	-0.51173
0.96362	-0.07709	-0.49709
0.984223	-0.10243	-0.43487
0.970487	-0.14224	-0.36532
0.960186	-0.16033	-0.28846
0.91898	-0.19291	-0.21526
0.86404	-0.20738	-0.17134
0.802231	-0.17119	-0.14937
0.723254	-0.13862	-0.13473
0.699217	-0.08433	-0.13839
0.723254	-0.00109	-0.13839
0.723254	0.027861	-0.14937
0.771327	0.089386	-0.12375

Model Retrieval

Generated models are available in Hbase, a RESTful Web service is available to download the required HBase gesture model.

Below method returns the HMM serialized object model in the string format, which can be de-serialized.

```
@GET
@Path("/name/{i}")
```

```
public String getModel(@PathParam("i") String gesture) throws IOException{

        Configuration config = HBaseConfiguration.create();
        config.clear();
        config.set("hbase.zookeeper.quorum", "localhost");
        config.set("hbase.zookeeper.property.clientPort","2181");
        config.set("hbase.master", "localhost:60010");
        HTable table = new HTable(config, "gestures");
        Get g = new Get(Bytes.toBytes(gesture));
        g.addColumn(Bytes.toBytes("device"), Bytes.toBytes("chronos") );
        Result r = table.get(g);
        String s1="";
        for(KeyValue kv: r.list()){
                s1 = Bytes.toString(kv.getValue());
        }
        return s1;
}
```

The PEMAR system is implemented and evaluated on a Cloudera CentOS VMware virtual machine with Intel Xeon processor and 12GB memory. The system and evaluation model are implemented in Java programming language. We used Eclipse as IDE to run the Java code. In addition, we used Hadoop 2.0.1, HBase 0.93, and JAHMM library as Java dependencies to compile the whole project. In order to demonstrate the applicability of the PEMAR middleware, we developed many sensor-based motion games controlled by a wide variety of the gesture models available from the Activity library. These apps were implemented by modifying existing open source game projects for a

transformation of their touch-based control model into a motionbased control through some devices such as SensorTag andChronos watch. These devices were used to generate our datasets including Acceleration, Temperature, Humidity GPS, and luminous intensity. The frequency of the data collected varied based on the sensor. The acceleration data from the device had a frequency up to 10 Hz and was transferred to the Action library using a web service. The MazeMan motion sensor app (Fig. 3) is a single player game, and it can recognize five gestures like left, right, up, down, restart. By utilizing the gesture models we built, we were able to capture each and every move, which we made, and we reported the total number of moves made by the player. We also showed the number of unscored moves made by the player, and this might be useful to understand how many moves were wasted without scoring. Max response time is the maximum time taken by the player in order to make a move. We estimated the playing style of the user by using the data. This app explicitly compares two motion profiles and highlights the differences with training consisting of iterative attempts to reduce/eliminate discrepancies. The activity analysis report will be shown to a user with his/her activity evaluation for each day/month. The path of the user's movement during exercise is displayed on Google maps with the markers for path boundaries such as green for start and pink for end.

4.2 PEMAR Implementation and Applications

An android application is developed, to track the long running activities. As described in section 3.2, the application collects the subjects' acceleration data and analyses using an online system. Along with the acceleration data, users' location data is also collected.

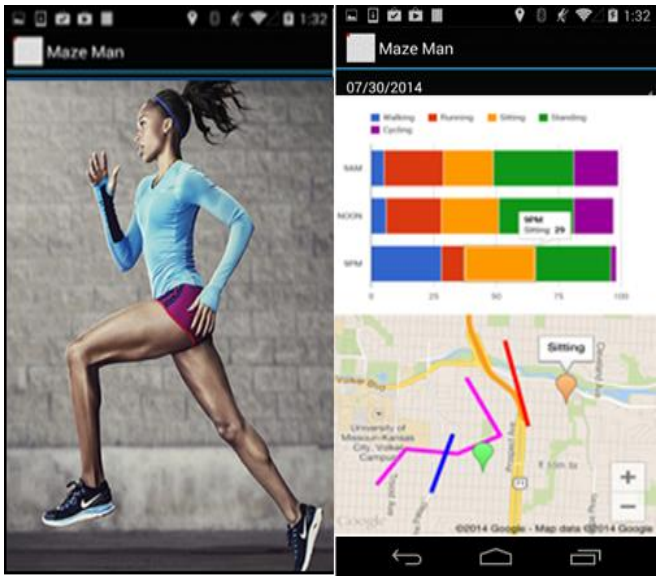

Figure 18: Maze Runner and Long Running Application

PEMAR provides an API to modify any existing android application or game to work with gestures input from any one of the compatible sensing device. We provide a ConnectionService class which would run in the backend thread to connect to sensing device (Sensor Tag) and would take care of the activity recognition. The user also has to change the onTouch trigger of the original application to work with the gesture input. Once the ConnectionService labels any new gesture/activity it would broadcast a signal across the current application running. The application should be changed to read the broadcasted data and trigger the respective onTouch events.

We have also built an Android application which has the meta information of all the existing android games modified to work with gesture input and existing trained model for activity recognition. These models can be changed depending on the application being used. The Meta application allows the user to select a particular game and then depending on the game give the user an option to select a set of activities to choose. Figure 19 shows the Meta application, which contains the list of all the

existing modified applications that work with PEMAR. Figure 20 below shows list of all activities that work with PEMAR.

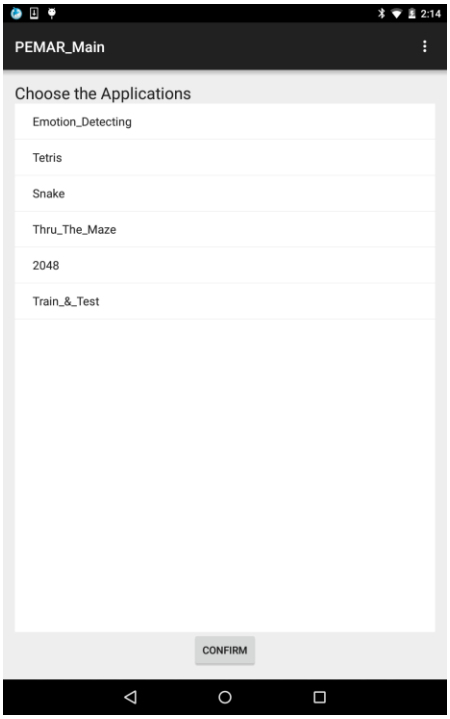

Figure 19 : Meta App List of Applications

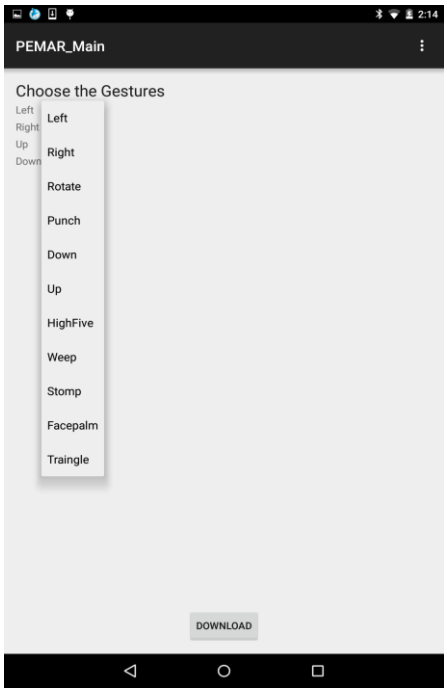

Figure 20 : List of activities that work with PEMAR

Below are some of the applications modified to work with gestures input and using the PEMAR framework.

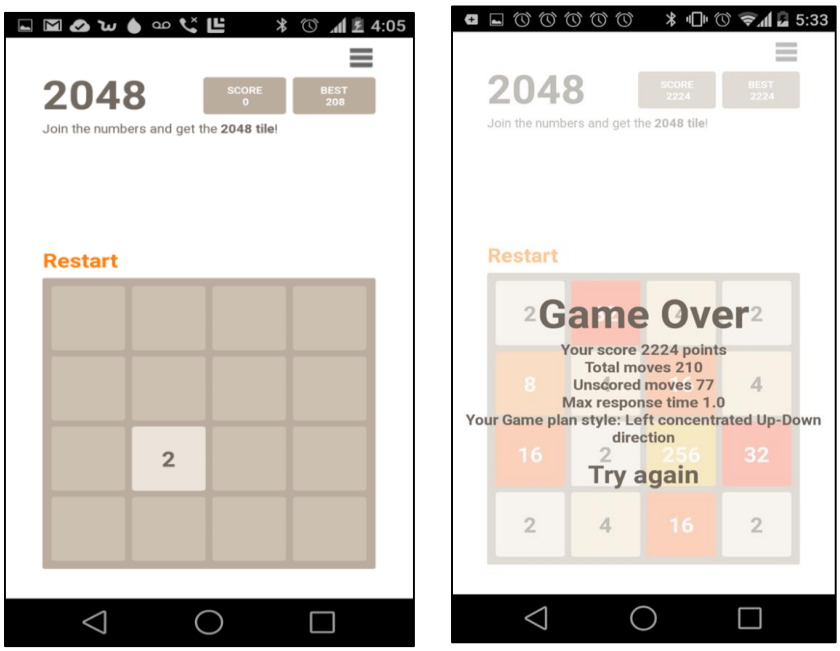

Figure 21: 2048 - Gesture based Gaming App

Figure 21 shows a 2048 gaming application on android platform. The 2048 open source gaming application has been modified to take gesture as input and perform the touch actions. The total number of gestures considered for playing are five, which include Left, right, up, down ,punch. 2048 is a single player puzzle game, which requires the user to slide the number titles on a square 4*4 grid, and upon combining them a tile would be created which has a number 2048. The sensor device used for the data collection and game play is Sensor Tag and the segmentation technique used is dynamic segmentation.

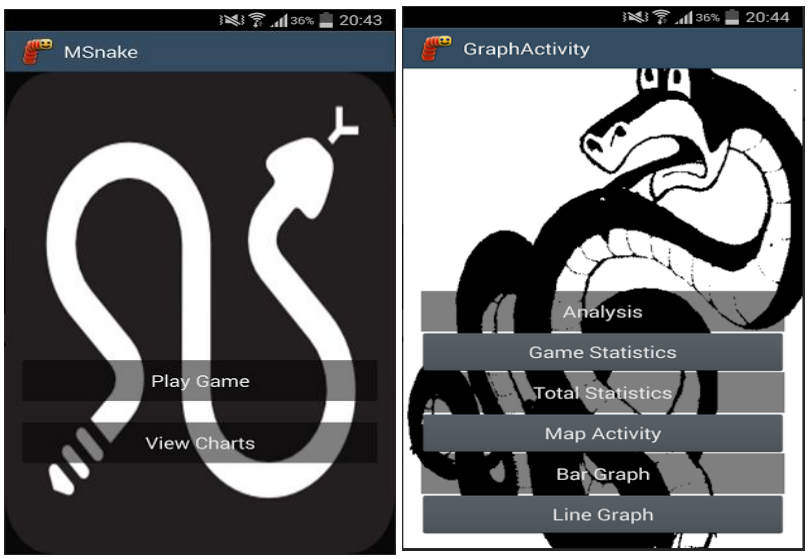

Figure 22: MSnake - Motion based Gaming App

Figure 22 shows, MSnake a gaming application on android platform. The Snake game consists of four movements, Left, Right, Up and Down. The Snake has to eat the apples on the screen to grow in size and gain points; snake should not touch the walls of the screen. The more the apples the snake eats, more the score.

Below table shows the list of all the applications built using PEMAR.

1. 2048
2. BigDataAnalytics_FinalProject_EmotionSense
3. Gesture Alert
4. GITRON
5. Maze_Man
6. MSnake
7. MSnake2
8. Number 2048
9. Sensor Write
10. Tetris
11. WhatToDo

CHAPTER 5

5. RESULTS AND EVALUATION

5.1 Experimental Setup

Different experimental setups are used to evaluate the PEMAR framework. For evaluating the recognition performance we used a mobile platform, and used Nexus 7 2012 model configuration. Below is the configuration of the mobile platform device.

NETWORK	Technology	No cellular connectivity
LAUNCH	Announced	2012, June
	Status	Available. Released 2012, July
BODY	Dimensions	198.5 x 120 x 10.5 mm (7.81 x 4.72 x 0.41 in)
	Weight	340 g (11.99 oz)
	SIM	No
DISPLAY	Type	LED-backlit IPS LCD capacitive touchscreen, 16M colors
	Size	7.0 inches (~59.6% screen-to-body ratio)
	Resolution	800 x 1280 pixels (~216 ppi pixel density)
	Multitouch	Yes, up to 10 fingers
	Protection	Corning Gorilla Glass
PLATFORM	OS	Android OS, v4.1 (Jelly Bean), upgradable to v4.4.2 (KitKat), planned upgrade to v5.0 (Lollipop)
	Chipset	Nvidia Tegra 3
	CPU	Quad-core 1.2 GHz Cortex-A9
	GPU	ULP GeForce
MEMORY	Card slot	No
	Internal	8/16/32 GB, 1 GB RAM
CAMERA	Primary	1.2 MP, 1280 x 960 pixels
	Features	Video-calling
	Video	720p
	Secondary	No

SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes, with stereo speakers
	3.5mm jack	Yes
COMMS	WLAN	Wi-Fi 802.11 b/g/n
	Bluetooth	v3.0
	GPS	Yes
	NFC	Yes
	Radio	No
	USB	microUSB v2.0
FEATURES	Sensors	Accelerometer, gyro, proximity, compass
	Messaging	Email, Push Email, IM
	Browser	HTML
	Java	No
		- MP4/H.264 player - MP3/WAV/eAAC+/WMA player - Organizer - Photo/video editor - Document viewer - Voice memo - Predictive text input (Swype)
BATTERY		Non-removable Li-Ion 4325 mAh battery (16 Wh)
	Stand-by	
	Talk time	Up to 10 h (multimedia)
MISC	Colors	Black
	SAR EU	1.39 W/kg (body)

Source: www.gsmarena.com

We evaluated different machine learning algorithms accuracy using weka tool kit. Below is the configuration of the machine.

System Configuration

CPU 2.10-GHz Intel Core i5-3310M
Operating System MS Windows 7 Professional (64-bit)
RAM 8GB
Hard Drive Size 1TB
Hard Drive Speed 7,200rpm
Hard Drive Type SATA
Hard Drive Display Size 15.6
Native Resolution 1366x768
Optical Drive DVD /-RW
Optical Drive Speed 8X
Graphics Card Intel HD Graphics 3000
Video Memory 1.6GB
Wi-Fi 802.11b/g/n
Wi-Fi Model Intel Centrino Wireless-N 1030
Bluetooth Bluetooth 3.0 + EDR

5.2 Dataset

The datasets used for the evaluation are from University of Southern California [17]– Human activity dataset and generated dataset. This is a benchmark dataset for human activity recognition. Dataset consists of data collected from 14 subjects performing 12 activities. Each subject repeated each activity five times, that is 60 files generated for each subject, based on his activity, and 840 files in total. The activities performed are both outdoor and indoor on different days. The sampling rate on the device used for collecting the data is 100Hz, and the acceleration range of the device is +-6g. The device is mounted on the waist of the user in a mobile pouch.

The activities listed in the dataset are :

1. Walking Forward
2. Walking Left
3. Walking Right

4. Walking Upstairs
5. Walking Downstairs
6. Running Forward
7. Jumping Up
8. Sitting
9. Standing
10. Sleeping
11. Elevator Up
12. Elevator Down

The dataset contains other useful information as well, such as

- Age
- Height
- Weight
- Activity name
- Activity number
- Trial number

The benchmark dataset contains only long running activities and cannot be tested for short gestures, hence we took the help of the students in the course Big Data Analytics and Applications to generated a short activity dataset for evaluation. We collected data from 50 students and collected a total of 15 gestures among them. The students used Sensor Tag to collect the data. Based on both the datasets, we evaluate the PEMAR framework.

5.3 Accuracy

Accuracy of multiple machine learning algorithms is compared using the above data set (Benchmark USC-HAD). Tests are carried for user dependent recognition and user independent recognition. It is observed that user dependent training has higher accuracy when compared to user independent training. The segmentation technique used for the USC-HAD is window-based segmentation. Accuracy evaluation is carried using a weka tool kit. The raw benchmark data set is segmented and processed to be used with weka.

User Dependent

From the user dependent accuracy evaluation the better performing algorithm is Random Forest followed by HMM in the most cases followed by Decision Tree, SVM and Naïve Bayes respectively. The data for each user consists of the 12 gestures, hence the accuracy is to recognize a gesture among the 12 choices. The features used for the recognition algorithms are time domain features :

(i) Mean

(ii) Standard Deviation

(iii) Correlation between XY

(iv) Correlation between YZ

(v) Correlation between XZ

(vi) Root Mean Square (RMS)

Table 8: User Dependent Activity Recognition Accuracy

Algorithm	Naive Bayes	SVM	Decision Tree	Random Forest	HMM
User1	83.3676 %	82.9569 %	87.0637 %	92.0945 %	84.3429 %
User2	79.3946 %	79.1036 %	87.7183 %	91.1525 %	83.6438 %
User3	89.0037 %	88.0443 %	86.7897 %	91.0701 %	87.6753 %
User4	80.7571 %	75.7729 %	81.6404 %	86.8139 %	82.2082 %
User5	79.7559 %	78.5355 %	85.5707 %	90.4523 %	85.6425 %
User6	82.8698 %	85.9039 %	88.4324 %	93.1732 %	87.6738 %
User7	89.56 %	90.6786 %	90.7532 %	93.1394 %	89.3363 %
User8	86.6699 %	86.573 %	84.6825 %	89.7722 %	85.4096 %
User9	87.5528 %	85.0557 %	87.1687 %	89.9731 %	87.6681 %
User10	84.1238 %	83.963 %	82.8376 %	88.6656 %	84.4051 %
User11	90.1752 %	92.0411 %	92.2696 %	93.8309 %	91.6984 %
User12	92.1131 %	91.4063 %	93.4524 %	**94.6057 %**	92.0015 %
User13	86.2109 %	84.5313 %	87.3438 %	91.6797 %	88.2422 %
User14	77.0406 %	74.6539 %	77.4702 %	83.8186 %	77.5656 %

Table 6 shows the respective overall accuracy of each of the user with user specific training data with 10 fold cross validation. We observed the maximum accuracy of activity recognition with Random Forest algorithm and closely followed by decision tree algorithm. When compared to other algorithms Random forest gave better accuracy, but the time taken to build a model with random forest is the longest when compared to other algorithms. Hence this is not the best suitable approach for activity recognition on a mobile platform.

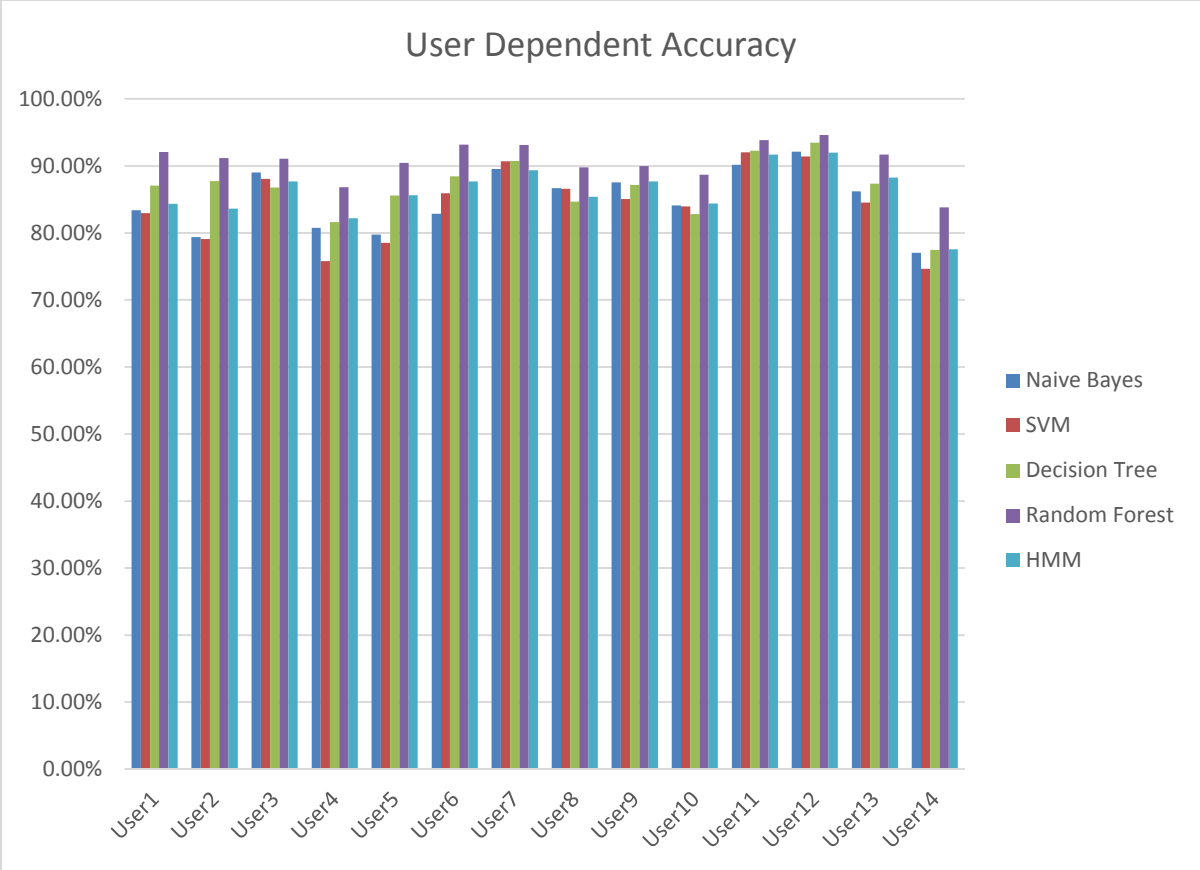

Figure 23: Accuracy User Specific

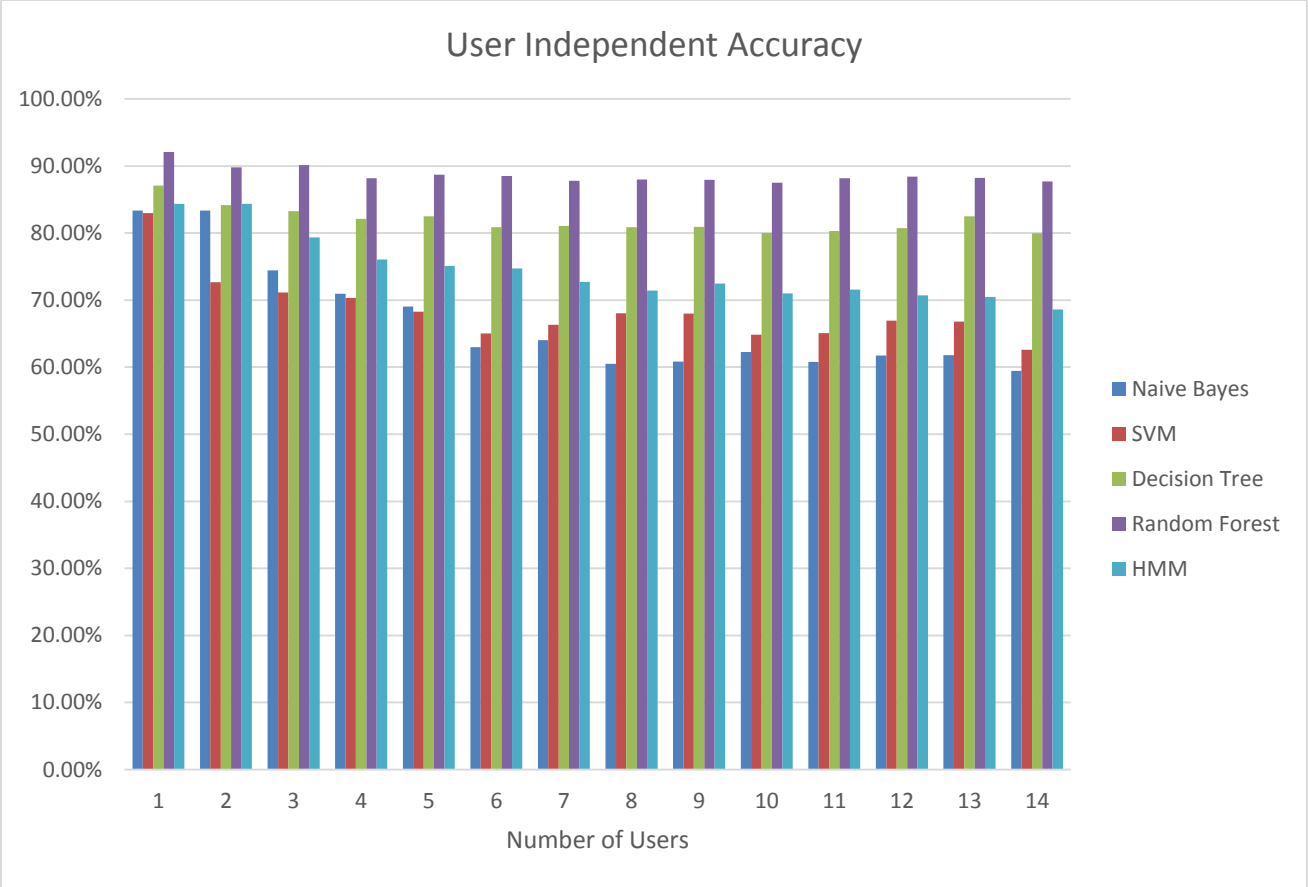

Figure 24: Accuracy User Independent

User Independent

The accuracy of machine learning algorithms reduce with the increase in user independent training data, the random forest algorithm and decision tree algorithms perform better compared to other algorithms. We also observe that with the increase in the number of users the recognition accuracies goes down.

Table 9: User Independent Algorithms Accuracy

Algorithm	Naive Bayes	SVM	Decision Tree	Random Forest	HMM
User1	83.3676 %	82.9569 %	87.0637 %	92.0945 %	84.3429 %
User2	83.3676 %	72.6678 %	84.1517 %	89.7709 %	84.3429 %
User3	74.4473 %	71.1412 %	83.2703 %	90.1414 %	79.3467 %
User4	70.9507 %	70.3451 %	82.1223 %	88.1926 %	76.0672 %
User5	69.0461 %	68.296 %	82.4978 %	88.6986 %	75.1219 %
User6	62.9997 %	65.0245 %	80.8579 %	88.5189 %	74.7312 %
User7	64.0267 %	66.334 %	81.0383 %	87.8044 %	72.7339 %
User8	60.4775 %	68.0148 %	80.8702 %	87.9938 %	71.4209 %
User9	60.8545 %	67.9823 %	80.9084 %	87.9459 %	72.4724 %
User10	62.2759 %	64.8211 %	80.0232 %	87.4806 %	70.9947 %
User11	60.7961 %	65.0785 %	80.3014 %	88.1606 %	71.5728 %
User12	61.723 %	66.9569 %	80.7482 %	88.4181 %	70.6883 %
User13	61.7958 %	66.7788 %	82.4978 %	88.2004 %	70.4624 %
User14	59.4259 %	62.5851 %	79.9786 %	87.6841 %	68.6326 %

5.4 Run Time Performance

In the experiment, we analyzed the training/testing runtime performance of different algorithms and the PEMAR runtime for each individual gesture. We conducted test cases for this and measured the average time taken to run the training in both environments, that is online and offline. The run time evaluation is carried out for the online systems. Systems in which the recognition response should be instantaneous. We also evaluate the time taken to build a training model in online system.

Table 10: Time Taken to Build Model

Device	Training Repetition for each gesture	Time Taken to build model (sec)	Time to download model (sec)
Nexus 7	60	64	0.7406
Nexus 7	120	123	0.8106
Nexus 7	180	183	0.7814
Nexus 7	240	255	0.7105

Table 10 shows the comparison of model building time on a mobile platform and the time it takes to fetch and use an existing model from the database. We also evaluate the accuracy of recognition with the increase in the choices of recognition models. We observed that with an increase in the choices of the activity models the overall accuracy of the recognition system goes down.

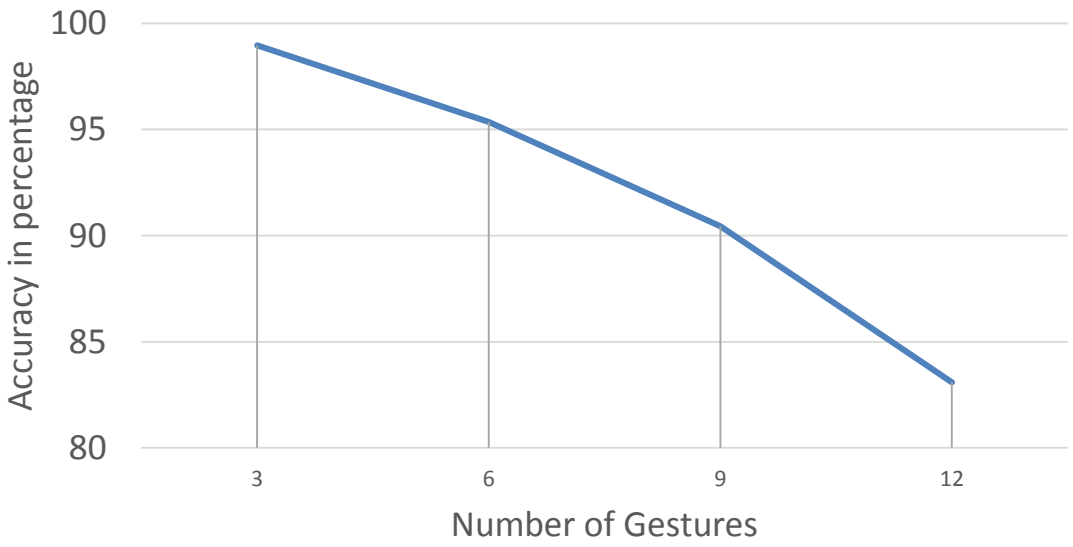

Figure 25: Percentage Accuracy with Number of Models

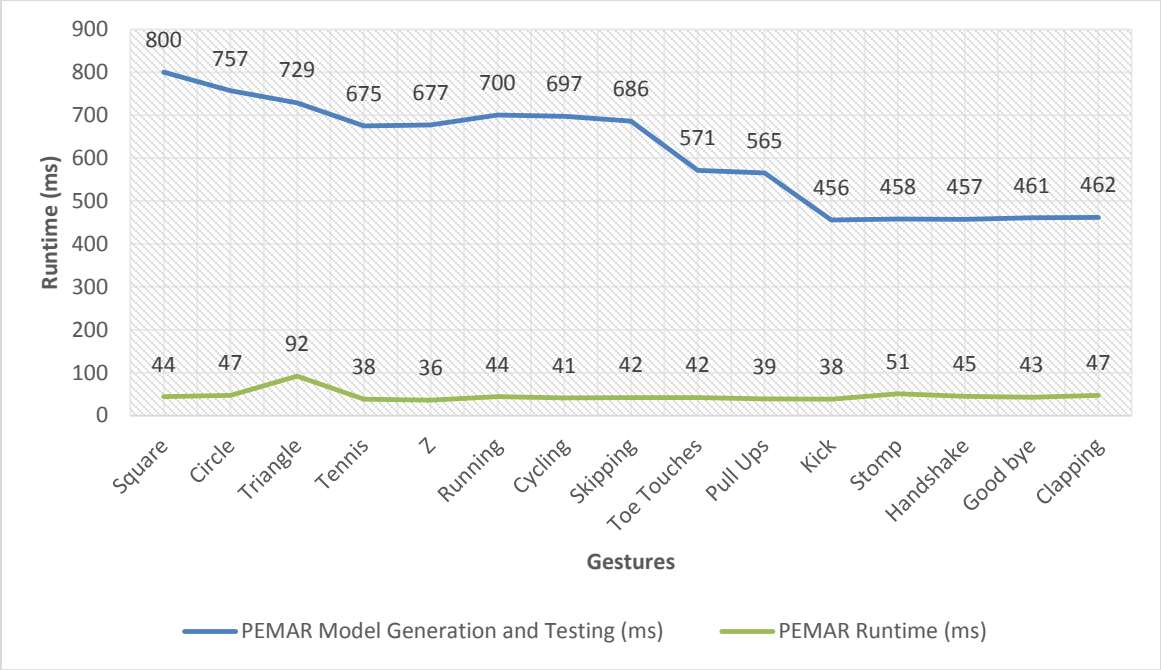

Figure 26: Performance Evaluation of PEMAR for Individual Activity Models

For the performance evaluation of the system, we evaluated the PEMAR model generation and download time with model generation time on a mobile platform. Figure 24 shows the evaluation results, where the type of gesture is represented on the x-axis and runtimes are represented on the y-axis. We found that the runtime performance of PEMAR, i.e model download time is better compared to the model generation time on a mobile platform. Figure 27 shows the Precision, Recall and F-measure of different activities performed and reported by the students of the Couse Big Data Analytics and Applications.

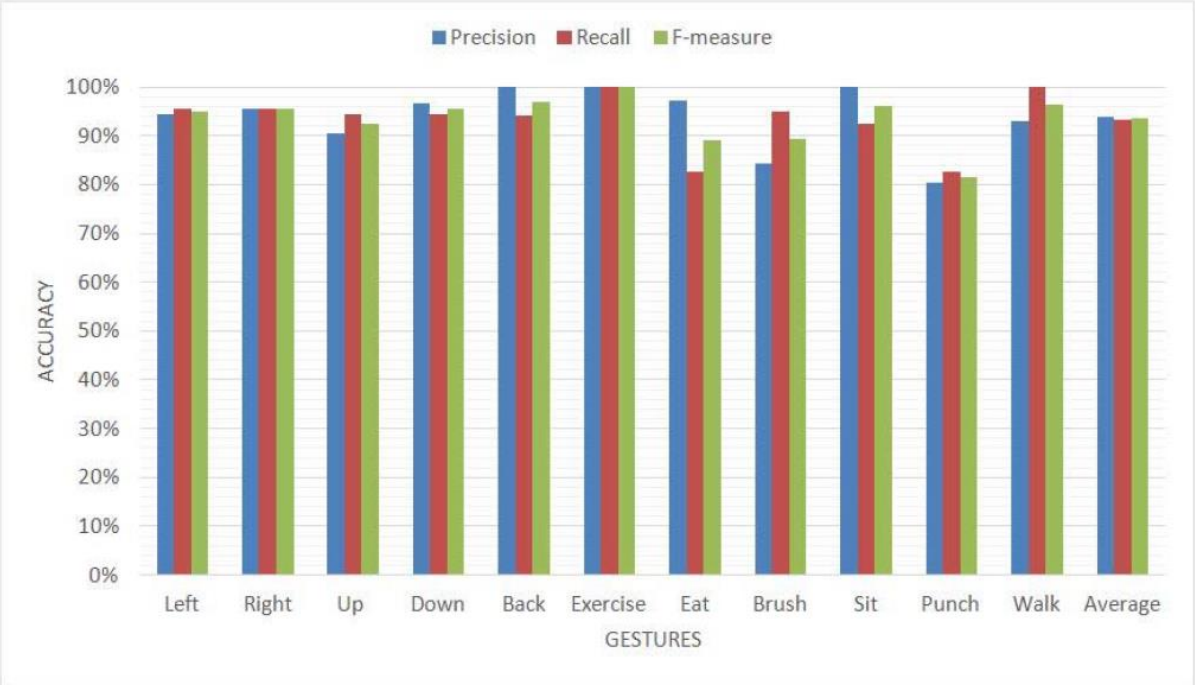

Figure 27 : Individual Gesture Recognition Accuracy

6. CONCUSION AND FUTURE WORK

6.1 Conclusion

We proposed the middleware for activity recognition whose potential is to be able to support both online and offline recognition systems to achieve better performance and reduce development time. We have proposed an architecture for an adaptive middleware system, which can handle diverse data from multiple users using the system at the same time. We developed multiple android applications that use the existing system for long running activities and short running activities as well. User dependent models are created dynamically without the user explicitly training for the model, this makes save a lot of time and initial set up time and offer better accuracy with increased usage of the application. An implementation of recognition system with the HMM model is created, which uses a filter to remove the bad and not so significant data and capture the sequence data. We have showed that such a middleware would increase the accuracy of the recognition and the performance of the system.

6.2 Future Work

There are some of the limitations of the project and scope for enhancement. PEMAR application requires a smart device for the carrying the computation, since the data from the wearable sensors would require a computational system. The proposed system is limited and cannot handle multiple sensors at the same time on a human body for activity recognition. The current algorithm model generation is in a batch mode, this increase the wait time and adds delay for model updating. Different gaming applications can be implemented with multiple mobile platforms. We can also develop applications with multiple devices like Wiimote on the same platform.

REFERENCES

1. Information Handling Services. URL: www.ihs.com
2. Lara, O.D., and Labrador, M.A. A Survey on Human Activity Recognition using Wearable Sensors. Communications Surveys & Tutorials. IEEE, Vol.15, no.3, p. 1192-1209, Third Quarter 2013
3. 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, Vol.23, p. 18-33, 1999.
4. 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, p. 11-14, 2008.
5. 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. IEEE, Vol.2, p. 1521-1527, 2006.
6. Gavrila, D. M. The visual analysis of human movement: A survey. Computer Vision and Image Understanding. Vol.73.1, p. 82-98, 1999.
7. 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. IEEE, Vol.3, p. 2596-2601, 2001.
8. Maurer, U., Smailagic, A., Siewiorek, D.P., and Deisher, M. Activity recognition and monitoring using multiple sensors on different body positions. In International Workshop on Wearable and Implantable Body Sensor Networks. IEEE Computer Society, p. 4-10, Washington, DC, USA, 2006.
9. Tapia, E. M., Intille, S. S., Haskell, W., Larson, K., Wright, J., King, A., and Friedman, R. Real-time recognition of physical activities and their intensities using wireless accelerometers and

a heart rate monitor. Wearable Computers. 11th IEEE International Symposium. IEEE, p.37-40, 2007.

10. Zephyr Bioharness BT. http://www.zephyr-technology.com/bioharness-bt.html.

11. Berchtold, M, et al. An extensible modular recognition concept that makes activity recognition practical. In the book, In KI: Advances in Artificial Intelligence. Published by Springer Berlin Heidelberg, p. 400-409, 2010.

12. Brezmes, T., Juan-Luis, G., and Josep, C. Activity recognition from accelerometer data on a mobile phone. In the book, In Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living. Published by Springer Berlin Heidelberg, p. 796-799, 2009.

13. Ermes, M., Parkka, J., and Cluitmans, L. Advancing from offline to online activity recognition with wearable sensors. In Engineering in Medicine and Biology Society. 30th Annual International Conference of the IEEE. p. 4451-4454, 2008.

14. Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J., and Korhonen, I. Activity classification using realistic data from wearable sensors. Information Technology in Biomedicine. IEEE Transactions, Vol.10 (1), p. 119-128, 2006.

15. Bao, L., and Intille, S. S. Activity recognition from user-annotated acceleration data. In the book, Pervasive computing. Published by Springer Berlin Heidelberg, p. 1-17, 2004.

16. Khan, A. M., Lee, Y. K., and Lee, S. Y. Accelerometer's position free human activity recognition using a hierarchical recognition model. In e-Health Networking Applications and Services (Healthcom). 12th IEEE International Conference, p. 296-301, 2010.

17. Zhu, C., and Sheng, W. Human daily activity recognition in robot-assisted living using multi-sensor fusion. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference, p. 2154-2159, 2009.

18. Zhang, M., and Sawchuk, A. A. A feature selection-based framework for human activity recognition using wearable multimodal sensors. In Proceedings of the 6th International Conference on Body Area Networks. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 92-98, 2011.

19. Hu, J., Kong, Y., and Fu, Y. Activity recognition by learning structural and pairwise mid-level features using random forest. In Automatic Face and Gesture Recognition (FG). 10th IEEE International Conference and Workshops, p. 1-6, 2013.

20. Lester, J., Choudhury, T., and Borriello, G. A practical approach to recognizing physical activities. In the book, Pervasive Computing. Published by Springer Berlin Heidelberg, p. 1-16, 2006.

21. Yang, J., and Xu Y. Hidden markov model for gesture recognition. Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, No. Cmu-Ri-Tr-94-10, 1994.

22. Arteaga, S. M., Kudeki, M., and Woodworth, A. Combating obesity trends in teenagers through persuasive mobile technology. ACM SIGACCESS Accessibility and Computing, Vol.94, p. 17-25, 2009.

23. Dempster, W. T., and Gaughran, G. R. Properties of body segments based on size and weight. American Journal of Anatomy 120.1. p. 33-54, 1967.

24. Physical Activity: A Journal about the Physical Activity and Factors Motivating Physical Activity. URL: https://www.presidentschallenge.org/informed/digest/docs/200009digest.pdf

25. 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, Vol.23, p. 18-33, 1999.

26. 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, Vol.2, p.1521-1527, 2006.

27. Gavrila, D. M. The visual analysis of human movement: A survey. Computer Vision and Image Understanding, Vol.73.1, p.82-98, 1999.

28. 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, Vol.3, p.2596-2601, 2001.

29. 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, Vol.1, p.281-284, 2000.

30. Liu, J., Zhong, L., Wickramasuriya, J., and Vasudevan, V. uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing, Vol.5.6, p.657-675, 2009.

31. Pylvänäinen, T. Accelerometer based gesture recognition using continuous HMMs. In the book, Pattern Recognition and Image Analysis. Published by Springer Berlin Heidelberg, p.639-646, 2005.

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

33. Statistics about Obesity in US: Centers for Disease Control and Prevention. URL: http://www.cdc.gov/healthyyouth/obesity/facts.htm.

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

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

36. 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, Vol.2.2, p.1-11, 2009.

37. Wu, Y., and Huang, T. S. Vision-based gesture recognition: A review. Urbana, Vol.51, p.103-115 1999.

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

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

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

41. 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, p.2297-2302, 2012.

42. 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, p.1-8, 2007.

43. 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, Vol.9.5, p.1-9, 2007.

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

45. 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, p.457-466, 2006.

46. 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, p.1831-1834, 2011.

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

48. Staiano, A. E., and Calvert, S. L. The promise of exergames as tools to measure physical health. Entertainment Computing. Vol.2.1, p.7-21, 2011.

49. 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. Vol.39.8, p.1423-1434, 2007.

50. 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, p.281-288, 1992.

51. 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. Vol.2.1, p.1-6, 2005.

52. 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. Vol.12.2-3, p. 173-184, 2007.

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

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

55. 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, p. 507-513, IEEE, 2006.

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

VITA

Prakash Reddy Vaka was born on December 14, 1989, in Guntur, Andhra Pradesh, India. He completed his Bachelor's degree in Electrical and Electronics Engineering from Andhra University College of Engineering in Visakhapatnam in 2011. Upon the completion of his Bachelor's, he worked with Infosys Limited as a Systems Engineer for two years.

In December 2014, Mr. Prakash came to the United States to pursue Computer Science at the University of Missouri-Kansas City (UMKC), specializing in Software Engineering. During January 2014 to May 2015, Mr. Prakash worked as a Graduate Research Assistant at University of Missouri – Kansas City, under the guidance of Dr. Yugyung Lee. After graduation, he plans to join as an Application Developer at Kansas State University.